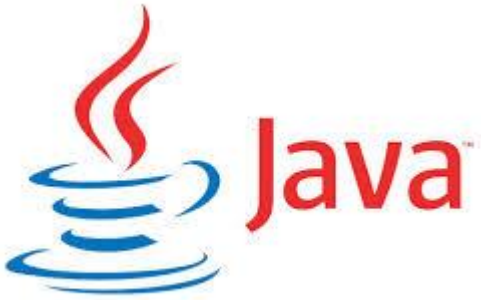
ITnerante

TIMASTERS

Linguagem JAVA

Curso de Exercícios
CESGRANRIO



PROF. VINICIUS REIS - professor.vreis@gmail.com

Apresentação Pessoal

Vinicius Reis

- Formação Acadêmica
  - Graduação em Tecnologia de Sistemas de Informação – Faculdade São José
  - Pós Graduação em Análise e Projeto de Sistemas – Universidade Gama Filho
- Atuação
  - Analista de Sistemas – SERPRO
- Aprovações
  - SERPRO, DATAPREV, MPU, INPI, Petrobras Distribuidora, entre outras.

Motivação

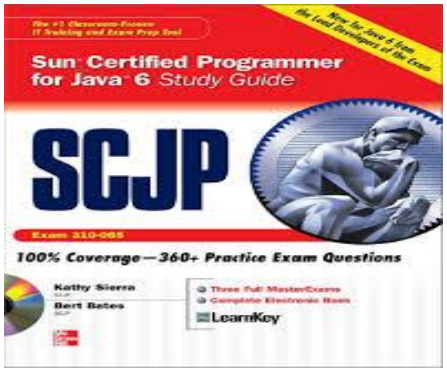
- **PETROBRAS 2012**
  - 50 questões específicas
  - 8 questões relacionadas à JAVA(**16%**)

- **BNDES 2013**
  - 40 questões específicas
  - 8 relacionadas à JAVA (**20%**)

- **FINEP 2014**
  - 25 questões específicas
  - 4 relacionadas à JAVA (**16%**)

Bibliografia Recomendada

Livro: Use a Cabeça! Java
Autor: Sierra, Kathy
Editora: Alta Books



Livro: Sun Certified Programmer for Java 6
Autores: Sierra, Kathy; Bates, Bert
Editora:Mcgraw Hill

Livro: Java – Como Programar – 8ª Ed. 2010
Autor: Deitel
Editora: Prentice Hall - BR

Ainda está com dúvidas? Recorra à especificação:
http://docs.oracle.com/javase/specs/

TIMASTERS

Ementa

- **Módulo 1**

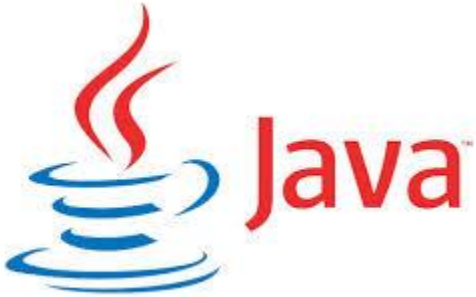
- Características da linguagem Java
- Modificadores de Acesso
- Operadores Aritméticos
- Operadores de Acréscimo e Decréscimo
- Estruturas Condicionais e de Repetição
- Passagem de Parâmetros
- Comparação de Wrappers
- Estruturas de Dados

Ementa

- **Módulo 1**

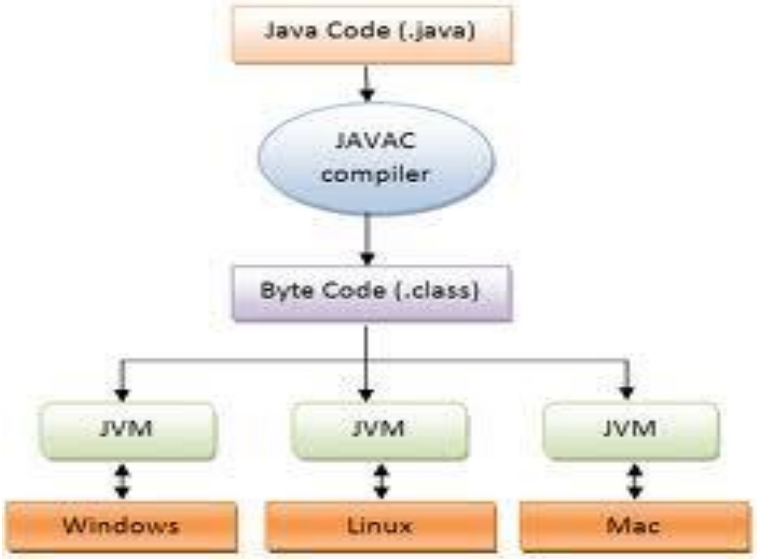
- Exceptions
- Collections
- Classes Abstratas e Interfaces (Introdução)
- Herança (Introdução)
- Classe Math
- Recursividade
- Operadores Lógicos
- Classe String
- Pilha e Heap
- Complexidade Ciclomática
- Complexidade de Algoritmos

Questões



Revisão Teórica – Características da Linguagem JAVA

- **Multiplataforma** – Qualquer dispositivo que possua uma máquina virtual JAVA (JVM) poderá executar os bytecodes.



01 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q56

Muito utilizada para desenvolvimento de aplicativos Web, a tecnologia Java tem como principal característica gerar aplicações que rodam em qualquer dispositivo que tenha acesso a Internet, utilizando, entre outros recursos, o software

(A) JBC (Java Bytecode Console)

(B) JDB (Java Developer Builder)

(C) JMS (Java Management Server)

(D) JAC (Java Application Controler)

(E) JVM (Java Virtual Machine)

01 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q56

Muito utilizada para desenvolvimento de aplicativos Web, a tecnologia Java tem como principal característica gerar aplicações que rodam em qualquer dispositivo que tenha acesso a Internet, utilizando, entre outros recursos, o software

(A) JBC (Java Bytecode Console)
(B) JDB (Java Developer Builder)
(C) JMS (Java Management Server)
(D) JAC (Java Application Controler)
(E) JVM (Java Virtual Machine)

Revisão Teórica – Modificadores de Acesso

Modificadores de acesso para atributos de classe em JAVA

- **Public** – Qualquer objeto pode manipular o atributo, independente do pacote.
- **Protected** – Apenas objetos da mesma classe, descendentes ou classes de mesmo pacote podem manipular o atributo.
- **Package** (default) – Apenas objetos do mesmo pacote podem manipular o atributo. Tal modificador é assumido quando a visibilidade é omitida.
- **Private** – Apenas objetos da mesma classe podem manipular o atributo.

Obs: Atentar para a diferença entre a abrangência do modificador de acesso **protected** em JAVA (objetos de mesma classe, descendentes ou classes de mesmo pacote) e o modificador **protected** da UML (apenas objetos de mesma classe ou descendentes).

Revisão Teórica – Modificadores Não-Referentes a Acesso

- **Final** – Pode ser utilizado em classes, atributos ou variáveis. Seu objetivo é impedir uma modificação posterior.

- **Abstract** – Pode ser utilizado em métodos ou em classes.
  - Método Abstract: A implementação do método será postergada para sua subclasse. Se houver uma "cadeia de heranças", a classe concreta deverá implementá-lo.
  - Classe Abstract: Nunca poderá ser instanciada. Seu único propósito é ser estendida (subclassificada).

02 – BNDES 01/2012 – Analista de Sistemas - Desenvolvimento – Q63

Esta questão foi suprimida / retirada do material.
Mantivemos no slide apenas fala não prejudicar a sequência correta de numeração das questões.

02 – BNDES 01/2012 – Analista de Sistemas - Desenvolvimento – Q63

Esta questão foi suprimida / retirada do material.
Mantivemos no slide apenas fala não prejudicar a sequência correta de numeração das questões.

Revisão Teórica – Operadores Aritméticos

- **Adição (+)**
- **Subtração (-)**
- **Multiplicação (*)**
- **Divisão (/)**
- **Resto (%) –** Divide o operando esquerdo pelo direito com o resultado sendo o resto.

  Ex: 3 % 2 produz como resultado o valor 1.

Revisão Teórica – Operadores de Acréscimo e Decréscimo

- **Acréscimo (++)**
- **Decréscimo (--)**

OBS: Os operadores de acréscimo e decréscimo podem ser inseridos antes ou depois de uma operação. Esta diferença de posicionamento do operador poderá produzir resultados distintos.

Ex:

```
public class Teste{
    public static void main(String[] args) {
        int a = 0;
        System.out.println(a++); → Será impresso 0
        int b = 0;
        System.out.println(++b); → Será impresso 1
    }
}
```

Revisão Teórica – Iterações

- **Loop For**
  - **Estrutura Básica**

```
for (int  x = 0;   x<10;     x++) {
```

inicialização condição incremento/decremento

```
    sequência de instruções
}
```

OBS: Nenhuma das três seções do loop for são necessárias!

03 – CHESF 01/2012 – Analista de Sistemas – Q48

Considere o trecho de código que corresponde ao método principal de uma classe em linguagem Java.

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {
        if (n % 2 != 0) {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

Qual o resultado produzido por esse método quando ele é corretamente executado?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 5

03 – CHESF 01/2012 – Analista de Sistemas – Q48

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {
        if (n % 2 != 0) {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

n = 1
x = 0
i

03 – CHESF 01/2012 – Analista de Sistemas – Q48

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {      executa 10 vezes o while
        if (n % 2 != 0) {   n é impar?      entrará na condição de n ímpar para n = {1,3,5,7,9)
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

n = 1
x = 0
i

03 – CHESF 01/2012 – Analista de Sistemas – Q48

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {
        if (n % 2 != 0) {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

n é impar?

entrará na condição de n ímpar para n = {1,3,5,7,9)

só entrará no for para n = 9

só entrará no if para n = 9

i só será menor que n para n = 5,7 e 9

03 – CHESF 01/2012 – Analista de Sistemas – Q48

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {
        if (n % 2 != 0) {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

x será acrescido de 1 por 3 vezes. Como inicializou em 0. Seu valor final será 3.

03 – CHESF 01/2012 – Analista de Sistemas – Q48

Considere o trecho de código que corresponde ao método principal de uma classe em linguagem Java.

```
public static void main(String[] args) {
    int n = 1;
    int x = 0;
    int i;
    while (n < 10) {
        if (n % 2 != 0) {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0)
                    break;
            }
            if (i < n) {
                x++;
            }
        }
        n++;
    }
    System.out.println(x);
}
```

Qual o resultado produzido por esse método quando ele é corretamente executado?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 5

Revisão Teórica – Método Main

- A assinatura do método main indica que ele pode receber um array de Strings.
  - A assinatura é a seguinte: public static void main(String[] args)
- Os argumentos são passados por linhas de comando e são separados por espaços.

  Ex: java ClasseTeste valor1 valor2 valor3
- Se quisermos passar uma String com espaços em seu interior, devemos passá-la entre aspas duplas.

  Ex: java ClasseTeste valorsemespaço “valor com espaço”

04 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB - Q21

Analise o código de um programa Java a seguir.

```
public class TestaArgs {
    public static void main(String[] args) {
        System.out.println(args[5]);
    }
}
```

Considere o seguinte comando:
java –hotspot TestaArgs um dois três quatro cinco seis sete

O que será impresso pelo programa ao executar esse comando?
(A) dois
(B) três
(C) quatro
(D) cinco
(E) seis

04 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB - Q21

Analise o código de um programa Java a seguir.

```
public class TestaArgs {
    public static void main(String[] args) {
        System.out.println(args[5]);
    }
}
```

Considere o seguinte comando:
java –hotspot TestaArgs um dois três quatro cinco seis sete

O que será impresso pelo programa ao executar esse comando?
(A) dois
(B) três
(C) quatro
(D) cinco
(E) seis

Revisão Teórica – Método Equals

- **Características**

  - Método da classe Object.
  - Utilizado na comparação entre objetos.
  - Cada classe pode sobrescrevê-lo se achar conveniente.
  - Determina se duas classes são “significativamente equivalentes”.

Revisão Teórica – Comparação de Wrappers utilizando Equals

- Um wrapper “empacota” valores primitivos em objetos.
  Ex: Integer, Character, Float, entre outros.

- **Regra Geral**: Para todas as classes wrappers, dois objetos são iguais se forem do mesmo tipo e tiverem o mesmo valor.

Ex:

```
Integer valor1 = 200;
Integer valor2 = 200;
if (valor1.equals(valor2)){
    System.out.println(“Objetos iguais”);
 }
```

A execução do trecho de código acima imprimirá “Objetos iguais”, devido à regra geral dos wrappers explicada anteriormente.

Revisão Teórica – Comparação de Wrappers Utilizando "=="

- De uma forma geral, o sinal de "==" compara as referências dos objetos, ou seja, se elas apontam para o mesmo endereço de memória.

- Na comparação de **alguns tipos de wrappers** utilizando "==", duas instâncias serão iguais quando seus valores primitivos forem os mesmos. Os tipos são os seguintes: Boolean, Byte, Character de \u00000 até \u007f (7f é 127 em decimal), Short e Integer de -128 até 127.

Ex:

```
Integer valor1 = 200;
Integer valor2 = 200;
if (valor1==valor2){
   System.out.println("Objetos iguais");
}
```

A execução do trecho de código acima imprimirá "Objetos iguais", pois a classe Integer está incluída na lista acima.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

Nas linguagens orientadas a objeto, existe uma diferença entre a referência a um objeto e o valor do objeto. Em java, o operador = = e o método equals, este definido para a classe Object, apresentam comportamento específico que tem relação
com essa característica.
Considere o exemplo de um código Java 6 a seguir.

Executando-se esse código, em que é possível testar como o comportamento exemplificado foi implementado para a classe
Integer e o tipo **int**, a resposta impressa será
(A) sssss (B) ssnns (C) snnns (D) nnsss (E) nssss

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```


a e b são Integer, logo o equals irá comparar o conteúdo.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

r1 = ‘s’

a e c são Integer, logo o equals irá comparar o conteúdo.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

r1 = ‘s’
r2 = ‘s’

a e d são Integer, logo o == irá comparar os valores primitivos.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

r1 = ‘s’
r2 = ‘s’
r3 = ‘s’

a e e são Integer, logo o == irá comparar os valores primitivos.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

r1 = ‘s’
r2 = ‘s’
r3 = ‘s’
r4 = ‘s’

a é Integer e f é int. Neste caso, a comparação será pelos valores primitivos.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

```
public class Questao {
  public static void main(String[] args) {
    Integer a,b,c,d,e;
    int f;
    char r1,r2,r3,r4,r5;
    String s ;
    a=1; b=1; c=a;
    d=b; f=1; e=f;
    r1=(a.equals(b)? 's' : 'n');
    r2=(a.equals(c)? 's' : 'n');
    r3=(a==d? 's' : 'n');
    r4=(a==e? 's' : 'n');
    r5=(a.equals(f)? 's' : 'n');
    s = ""+r1+r2+r3+r4+r5 ;
    System.out.println(s);
  }
}
```

r1 = ‘s’
r2 = ‘s’
r3 = ‘s’
r4 = ‘s’
r5 = ‘s’

→ Será impresso sssss.

05 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q36

Nas linguagens orientadas a objeto, existe uma diferença entre a referência a um objeto e o valor do objeto. Em java, o operador = = e o método equals, este definido para a classe Object, apresentam comportamento específico que tem relação com essa característica.
Considere o exemplo de um código Java 6 a seguir.

Executando-se esse código, em que é possível testar como o comportamento exemplificado foi implementado para a classe **Integer** e o tipo **int**, a resposta impressa será

(A) sssss
(B) ssnns
(C) snnns
(D) nnsss
(E) nssss

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 01/02

Considere o seguinte trecho de código implementado em Java:

```
class ArraySort{
   private double [] a;
   private int nElems;
   // outras funcoes omitidas
   // …
   public void Sort(){
      int in, out;
      for (out = nElems-1; out > 1; out--)
         for (in = 0; in < out; in++)
            if (a[in] > a[in+1])
               swap(in, in+1);
   } // fim da Sort
   private void swap (int x, int y){
      double temp = a[x];
      a[x] = a[y];
      a[y] = temp;
   }
} // fim da classe ArraySort
```

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 02/02

O código apresenta a implentação de uma classe ArraySort, que contém um método denominado Sort, cuja finalidade é implementar a

(A) arrumação dos nós na forma adequada em uma árvore binária.

(B) arrumação dos nós na forma adequada em uma árvore B.

(C) ordenação em uma lista encadeada utilizando o método do quicksort.

(D) ordenação em um vetor utilizando o método do quicksort.

(E) ordenação em um vetor utilizando o método da bolha (bubble sort).

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 01/02

```
class ArraySort{
   private double [] a;
   private int nElems;
   // outras funcoes omitidas
   // …
   public void Sort(){
      int in, out;
      for (out = nElems-1; out > 1; out--)
         for (in = 0; in < out; in++)
            if (a[in] > a[in+1])
               swap(in, in+1);
   } // fim da Sort
   private void swap (int x, int y){
      double temp = a[x];
      a[x] = a[y];
      a[y] = temp;
   }
} // fim da classe ArraySort
```


Loop externo da penúltima posição até a primeira em ordem decrescente

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 01/02

```
class ArraySort{
   private double [] a;
   private int nElems;
   // outras funcoes omitidas
   // …
   public void Sort(){
      int in, out;
      for (out = nElems-1; out > 1; out--)
         for (in = 0; in < out; in++)
            if (a[in] > a[in+1])
               swap(in, in+1);
   } // fim da Sort
   private void swap (int x, int y){
      double temp = a[x];
      a[x] = a[y];
      a[y] = temp;
   }
} // fim da classe ArraySort
```

Loop interno da primeira posição até a ultima em ordem crescente

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 01/02

```
class ArraySort{
   private double [] a;
   private int nElems;
   // outras funcoes omitidas
   // …
   public void Sort(){
      int in, out;
      for (out = nElems-1; out > 1; out--)
         for (in = 0; in < out; in++)
            if (a[in] > a[in+1])
               swap(in, in+1);
   } // fim da Sort
   private void swap (int x, int y){
      double temp = a[x];
      a[x] = a[y];
      a[y] = temp;
   }
} // fim da classe ArraySort
```

Compara se o valor da posição atual do loop interno é maior que o valor de sua próxima posição

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 01/02

```
class ArraySort{
   private double [] a;
   private int nElems;
   // outras funcoes omitidas
   // …
   public void Sort(){
      int in, out;
      for (out = nElems-1; out > 1; out--)
         for (in = 0; in < out; in++)
            if (a[in] > a[in+1])
               swap(in, in+1);
   } // fim da Sort
   private void swap (int x, int y){
      double temp = a[x];
      a[x] = a[y];
      a[y] = temp;
   }
} // fim da classe ArraySort
```

Se for maior, realiza a troca

06 – Liquigas Distribuidora 02/2012 – Tecnologia da Informação – Des. de Aplicações – Q33 – 02/02

O código apresenta a implentação de uma classe ArraySort, que contém um método denominado Sort, cuja finalidade é implementar a

(A) arrumação dos nós na forma adequada em uma árvore binária.

(B) arrumação dos nós na forma adequada em uma árvore B.

(C) ordenação em uma lista encadeada utilizando o método do quicksort.

(D) ordenação em um vetor utilizando o método do quicksort.

(E) ordenação em um vetor utilizando o método da bolha (bubble sort).

Revisão Teórica – Características da Linguagem JAVA

- **Passagem de Parâmetros**

✓ **Por Valor**: Uma cópia dos bits da variável será passada como parâmetro.
✓ **Por Referência**: Em JAVA, isto não ocorre.

"O método chamado não pode alterar a variável do chamador, embora no que diz respeito a variáveis de referência de objeto, o método pode alterar o objeto que a variável referencia". (Sierra, Kathy)

Obs: O que existe em JAVA é a passagem de endereços de memória, que nunca poderá ser confundida com passagem de parâmetros por referência.

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software - MAR/2010 – Q17 – 01/06

```
public class SomaMisteriosa {
    private static void somaTres(int x[]) {
        x[0] += 3;
    }
    private static void somaDois(int x) {
        x += 2;
    }
    public static void main(String args[]) {
        int x = 0;
        int y[] = { 0 };
        somaDois(x);
        somaTres(y);
        somaDois(y[0]);
        System.out.print(x + " " + y[0]);
    }
}
```

Após a execução do trecho acima, será impresso

(A) 2 5

(B) 1 5

(C) 0 5

(D) 0 3

(E) 0 0

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software - MAR/2010 – Q17 – 02/06

```
public static void main(String args[]) {
    int x = 0;
    int y[] = { 0 };
    somaDois(x);
    somaTres(y);
    somaDois(y[0]);
    System.out.print(x + " " + y[0]);
}
 private static void somaTres(int x[]) {
     x[0] += 3;
 }
 private static void somaDois(int x) {
     x += 2;
 }
```

➡ **É passado o conteúdo da variável x**

Escopo do método main
X = 0
Y[] = {0}

➡ **x é modificado apenas no escopo do método somaDois**

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software - MAR/2010 – Q17 – 03/06

```
public static void main(String args[]) {
    int x = 0;
    int y[] = { 0 };
    somaDois(x);
    somaTres(y);
    somaDois(y[0]);
    System.out.print(x + " " + y[0]);
}
private static void somaTres(int x[]) {
    x[0] += 3;
}
private static void somaDois(int x) {
    x += 2;
}
```

⇒ **É passado o endereço de memória de y**

Escopo do método main
X = 0
Y[0] = 3

⇒ **y[0] é modificado no endereço de memória informado**

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 - Q17 – 04/06

```
public static void main(String args[]) {
    int x = 0;
    int y[] = { 0 };
    somaDois(x);
    somaTres(y);
    somaDois(y[0]);
    System.out.print(x + " " + y[0]);
}

private static void somaTres(int x[]) {
    x[0] += 3;
}
private static void somaDois(int x) {
    x += 2;
}
```

⇒ É passado o conteúdo de y[0]

⇒ y[0] é modificado no escopo do método somaDois

Escopo do método main
X = 0
Y[0] = 3

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 - Q17 – 05/06

```
public static void main(String args[]) {
    int x = 0;
    int y[] = { 0 };
    somaDois(x);
    somaTres(y);
    somaDois(y[0]);
    System.out.print(x + " " + y[0]);
}

private static void somaTres(int x[]) {
    x[0] += 3;
}
private static void somaDois(int x) {
    x += 2;
}
```

⇒ É impresso o valor 0 3

Escopo do método main
X = 0
Y[0] = 3

07 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 - Q17 – 06/06

```
public class SomaMisteriosa {
   private static void somaTres(int x[]) {
      x[0] += 3;
   }
   private static void somaDois(int x) {
      x += 2;
   }
   public static void main(String args[]) {
      int x = 0;
      int y[] = { 0 };
      somaDois(x);
      somaTres(y);
      somaDois(y[0]);
      System.out.print(x + " " + y[0]);
   }
}
```

Após a execução do trecho acima, será impresso

(A) 2 5

(B) 1 5

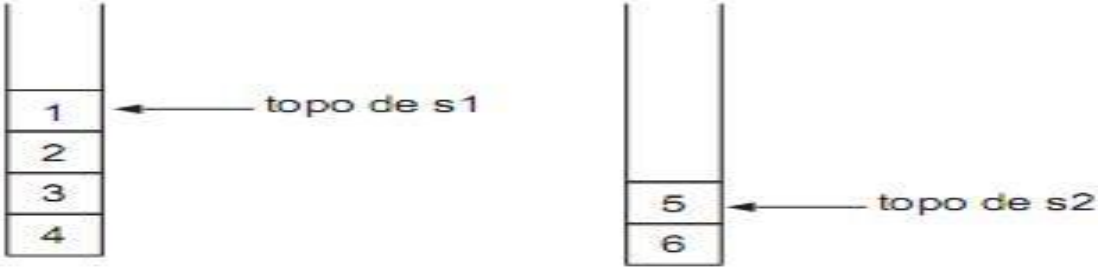
(C) 0 5

(D) 0 3

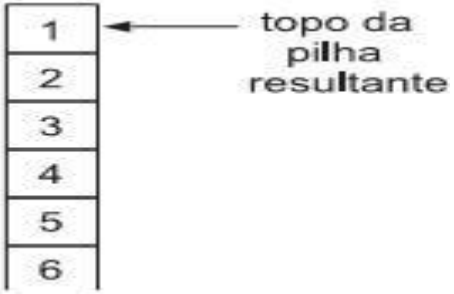
(E) 0 0

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

O método concat(s1,s2) recebe duas pilhas como parâmetros e retorna a concatenação de s1 com s2. Por exemplo, suponha que as pilhas abaixo sejam passadas para concat():



O método concat() irá produzir uma pilha na qual o elemento que estará no seu topo será o topo da pilha s1. Além disso, o elemento no topo de s2 ficará imediatamente abaixo da base de s1. A Figura a seguir exibe a pilha produzida pelo método concat() a partir das pilhas s1 e s2:



Qual implementação do método concat() produz o resultado descrito acima?

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

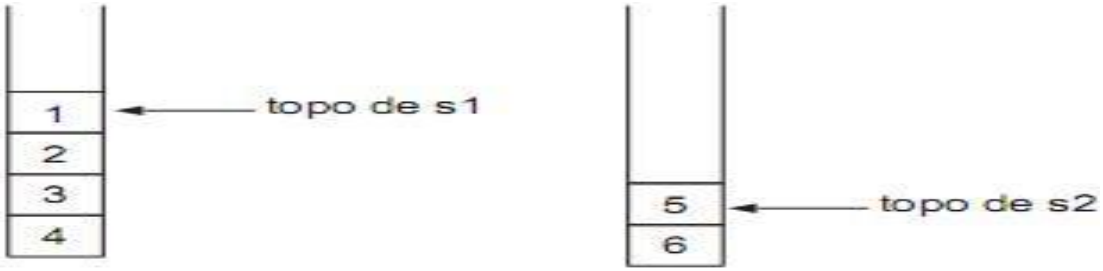
(A)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Stack<Integer> nova=new Stack<Integer>();
   for( ;!s1.empty();nova.push(s1.pop()));
   for( ;!s2.empty();nova.push(s2.pop()));
   return nova;
}
```

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(A)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Stack<Integer> nova=new Stack<Integer>();
   for( ;!s1.empty();nova.push(s1.pop()));
   for( ;!s2.empty();nova.push(s2.pop()));
   return nova;
}
```



Erro - A nova lista será criada na seguinte ordem: 6-5-4-3-2-1

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(B)

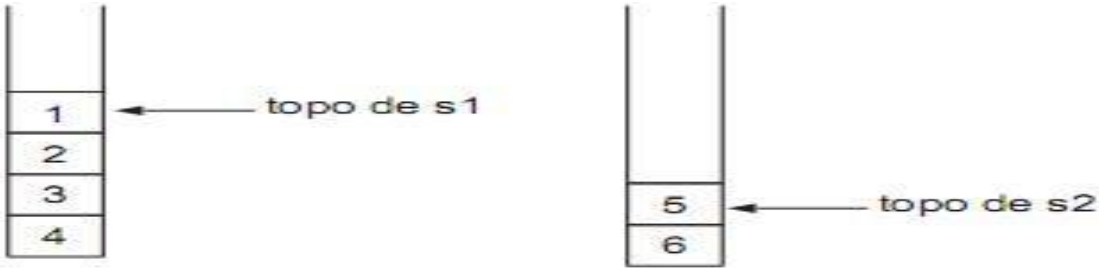
```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   for( ;!s1.empty();s2.push(s1.pop()));
   return s2;
}
```

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(B)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   for( ;!s1.empty();s2.push(s1.pop()));
   return s2;
}
```

Erro - A lista s2 foi reaproveitada e os itens de s1 foram retirados. A lista final ficou com: 4,3,2,1,5,6



08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(C)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Stack<Integer> nova=new Stack<Integer>();
   ArrayList<Integer> aux=new ArrayList<Integer>();
   for( ;!s1.empty();aux.add(s1.pop()));
      for( ;!s2.empty();aux.add(s2.pop()));
         for(int i=0;i<aux.size();i++)
            nova.push(aux.get(i));
   return nova;
}
```

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(C)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Stack<Integer> nova=new Stack<Integer>();
   ArrayList<Integer> aux=new ArrayList<Integer>();
   for( ;!s1.empty();aux.add(s1.pop()));
   for( ;!s2.empty();aux.add(s2.pop()));
   for(int i=0;i<aux.size();i++)
   nova.push(aux.get(i));
   return nova;
}
```

Erro – A lista aux estava com a ordem correta, mas ao incluírmos os elementos na nova pilha, a ordem foi invertida.
Obs: Reparem o detalhe dos loops que não estão aninhados.

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(E)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Integer i=null,j=null;
   Stack<Integer> nova=new Stack<Integer>();
   if(s1.empty() && s2.empty())
      return s2;
   if(!s1.empty())
      i=s1.pop();
   if(!s2.empty())
      j=s2.pop();
   concat(s1,s2);
   if(i!=null)
      nova.push(i);
   if(j!=null)
      nova.push(j);
   return nova;
}
```

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(E)

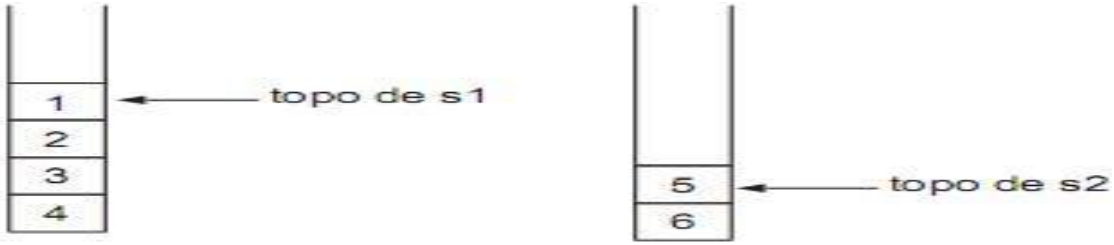
```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Integer i=null,j=null;
   Stack<Integer> nova=new Stack<Integer>();
   if(s1.empty() && s2.empty())
      return s2;
   if(!s1.empty())
      i=s1.pop();
   if(!s2.empty())
      j=s2.pop();
   concat(s1,s2);
   if(i!=null)
      nova.push(i);
   if(j!=null)
      nova.push(j);
   return nova;
}
```

condição de parada da função recursiva

remove o elemento do topo de s1

remove o elemento do topo de s2

chamada recursiva



A pilha retornada terá os valores 5,1

TIMASTERS

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(D)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
    Integer i;
    if(s1.empty())
        return s2;
    i=s1.pop();
    concat(s1,s2);
    s2.push(i);
    return s2;
}
```

08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(D)

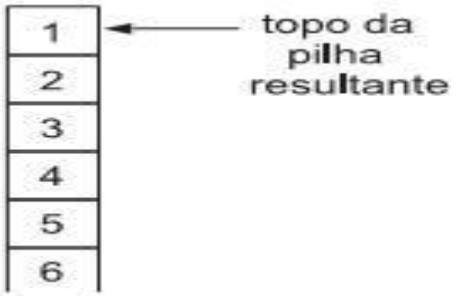
```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
   Integer i;
   if(s1.empty())
      return s2;
   i=s1.pop();
   concat(s1,s2);
   s2.push(i);
   return s2;
}
```

condição de parada da função recursiva

remove os elementos do topo de s1 (1,2,3,4)

chamada recursiva

Quando os valores de i são "desempilhados", eles são inseridos no topo de s2



08 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q62

(D)

```
public Stack<Integer> concat(Stack<Integer> s1,Stack<Integer> s2){
    Integer i;
    if(s1.empty())
        return s2;
    i=s1.pop();
    concat(s1,s2);
    s2.push(i);
    return s2;
}
```

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

As classes Java a seguir representam, respectivamente, uma fila e seus nós.

```
public class Fila {
   No ini=null; // referência para o primeiro elemento da fila
   No fin=null; // referência para o último elemento da fila
   public No insere(No n)
   {
   }
}
public class No {
   No prox;
   int info;
   public No(int i)
   {
      info=i;
   }
}
```

Qual implementação do método insere() permite inserir corretamente um novo elemento na fila, preservando a sua semântica?

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

(A)

```
public No insere(No n)
{
   fin=n;
   fin.prox=n;
   if(ini==null)
      ini=fin;
   return n;
}
```

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

```
(A)
public No insere(No n)
{
   fin=n;
   fin.prox=n;
   if(ini==null)
      ini=fin;
   return n;
}
```

Explicação simplificada:
1) inserirmos um nó com info = 1
fin = 1
1.prox = 1
ini ==null → VERDADE
ini = 1
return 1

2) inserimos um nó com info = 2
fin=2
2.prox=2 → ERRO DA QUESTÃO
Ini==null → FALSO
Return 2

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

```
(B)
public No insere(No n)
{
   No ant=null,cur=ini;
   for(;cur!=null;cur=cur.prox)
      ant=cur;
   n.prox=ini;
   ini=n;
   if(fin==null)
      fin=ini;
   return n;
}
```

Erro: A lista ficará assim:
2,1

Explicação simplificada:
1) inserirmos um nó com info = 1
ant = null; cur=null;
não entra no for
n.prox=null
ini = 1
Fin==null → VERDADE
Fin=1
Return 1

2) inserimos um nó com info = 2
ant = null; cur=1;
sai do for com ant=1
n.prox=1
ini = 2
fin==null → FALSO
return 2

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

```
(D)
public No insere(No n)
{
   No ant=null,cur=ini;
   for(;cur!=null&&n.info>cur.info;cur=cur.prox)
      ant=cur;
   if(ant==null)
   {
      n.prox=ini;
      ini=n;
   }
   else
   {
       n.prox=ant.prox;
       ant.prox=n;
   }
   if(n.prox==null)
      fin=n;
    return n;
}
```

A lista não está "fechada"

Explicação simplificada:
1) inserirmos um nó com info = 1
ant = null; cur=null;
não entra no for
ant==null → VERDADE
n.prox = null
ini = 1
n.prox==null → VERDADE
fin=1
return 1
2) inserimos um nó com info = 2
ant = null; cur=1;
sai do for com ant=1
entra no else
n.prox= null e ant.prox = 2
n.prox==null → VERDADE
fin=2
return 2

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

(E)

```
public No insere(No n)
{
   n.prox=fin;
   fin.prox=n;
   if(ini==null)
      ini=fin;
   return n;
}
```

Explicação simplificada:
1) inserirmos um nó com info = 1
n.prox=null
fin.prox=1 → NullPointerException

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

```
(C)
public No insere(No n)
{
   No ant=null,cur=ini;
   for(;cur!=null;cur=cur.prox)
      ant=cur;
   fin=n;
   if(ini==null)
      ini=fin;
   else
      ant.prox=n;
   return n;
}
```

← **Detalhe importante**

Aqui a lista está "fechada"

Explicação simplificada:
1) inserirmos um nó com info = 1
ant = null; cur=null;
não entra no for
fin=1
ini==null → VERDADE
ini=1
return 1

2) inserimos um nó com info = 2
ant = null; cur=1;
sai do for com ant=1
fin=2
entra no else
ant.prox=2
return 2

09 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q35

(C)

```
public No insere(No n)
{
   No ant=null,cur=ini;
   for(;cur!=null;cur=cur.prox)
      ant=cur;
   fin=n;
   if(ini==null)
      ini=fin;
   else
      ant.prox=n;
   return n;
}
```

10 – Liquigas 02/2012 – Profissional Jr. Desenvolvimento de Aplicações – Q24

Considere o seguinte trecho de código em linguagem Java.

```
class S {
    private int Tamanho;
    private int[] sArray;
    private int top;

public S(int s) // constructor
{
    Tamanho = s;
    sArray = new long[Tamanho]; // create array
    top = -1;
}

 public void coloca(int j)
 {
     sArray[++top] = j;
 }

public long tira()
{
    return sArray[top--];
}
} // end class S
```

10 – Liquigas 02/2012 – Profissional Jr. Desenvolvimento de Aplicações – Q24

```
public void coloca(int j)
{
   sArray[++top] = j;
}
public long tira()
{
   return sArray[top--];
}
```

Esse trecho implementa uma classe que corresponde a
uma
(A) fila de inteiros
(B) pilha de inteiros
(C) árvore binária com valores inteiros nos nós
(D) lista encadeada de inteiros
(E) grafo com custos inteiros nas arestas

10 – Liquigas 02/2012 – Profissional Jr. Desenvolvimento de Aplicações – Q24

Esse trecho implementa uma classe que corresponde a
uma
(A) fila de inteiros
(B) pilha de inteiros
(C) árvore binária com valores inteiros nos nós
(D) lista encadeada de inteiros
(E) grafo com custos inteiros nas arestas

11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47 – 01/04

As classes Java a seguir ocupam arquivos distintos. Elas são usadas para implementar árvores binárias nas quais os nós armazenam valores inteiros.

```
package estruturas;
class ArvNo {
   int info;
   ArvNo esq=null,dir=null;
}
```

```
package estruturas;
public class Arv {
   private ArvNo raiz;
   public void exibe(int val){
      percorre(raiz,val);
}
```

```
private void percorre(ArvNo r,int val) {
   if(r==null)
      return;
   percorre(r.dir,r.info);
   percorre(r.esq,r.info);
   if(r.info>val)
      System.out.print(r.info+" ");
   }
}
```

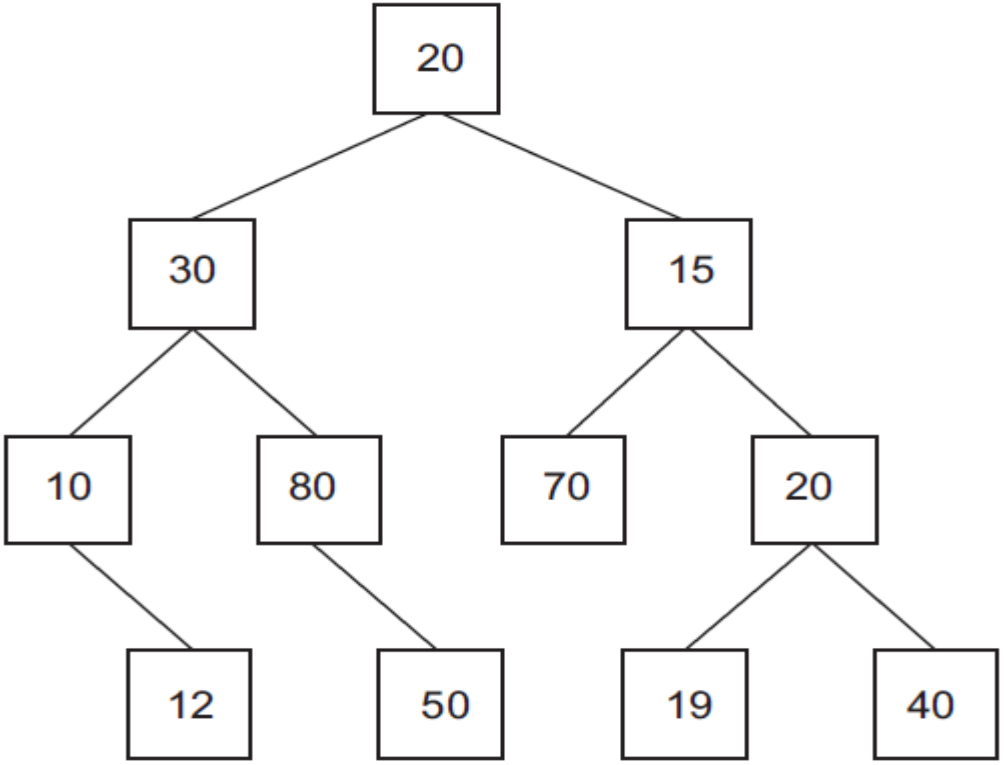
11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47 – 02/04

A criação dos nós de uma árvore é realizada pelo construtor da classe Arv. Esse construtor, entretanto, não é exibido por ser irrelevante para a questão. É necessário saber apenas que, após a execução do construtor, a variável de instância raiz irá referenciar o nó raiz da árvore criada. A classe Main a seguir foi elaborada para utilizar as classes descritas acima.

```
import estruturas.*;
public class Main {
   public static void main(String[] args) {
      Arv a=new Arv();
      a.exibe(0);
   }
}
```

11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47 – 03/04

Seja a seguinte árvore binária:



11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47 – 04/04

Supondo que uma árvore como essa seja criada após a instanciação de um objeto da classe Arv, o que será exibido no console quando o método main() acima for executado?

(A) 40 20 70 20 80 30 12

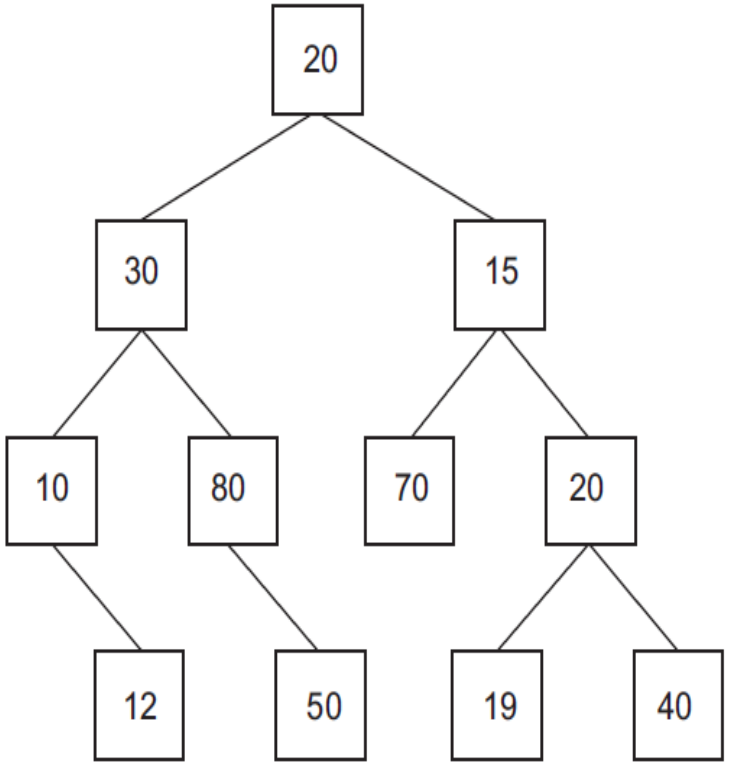
(B) 40 20 70 80 12 30 20

(C) 20 20 40 70 30 80 12

(D) 20 30 12 80 70 20 40

(E) 12 80 30 70 40 20 20

11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47




```
package estruturas;
public class Arv {
   private ArvNo raiz;
   public void exibe(int val){
      percorre(raiz,val);
}

private void percorre(ArvNo r,int val) {
   if(r==null) <-- condição de parada
      return;
   percorre(r.dir,r.info);
   percorre(r.esq,r.info);
   if(r.info>val) <-- compara o "filho" com o "pai"
      System.out.print(r.info+" ");
   }
}
```


TIMASTERS

11 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q47

Supondo que uma árvore como essa seja criada após a instanciação de um objeto da classe Arv, o que será exibido no console quando o método main() acima for executado?

(A) 40 20 70 20 80 30 12

(B) 40 20 70 80 12 30 20

(C) 20 20 40 70 30 80 12

(D) 20 30 12 80 70 20 40

(E) 12 80 30 70 40 20 20

Revisão Teórica – Exceptions

```
try{
//“região protegida”
}
```

```
catch(PrimeiraExcecao){
//Código de tratamento da PrimeiraExcecao
}

catch (SegundaExcecao){
//Código de tratamento da SegundaExcecao
}
```

```
finally{
//Bloco que sempre será executado, ocorrendo ou não exceções
}
```

12 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q21 – 01/02

Sejam as seguintes classes Java:

```
public class CA {
   int val=0;
   public void op1(int x){
      val+=x;
   }
   public void op2(int x,int y){
      val-=x+y;
   }
   int getVal(){
      return val;
   }
}
```

```
public class CB extends CA{
   public void op1(int x){
      val-=x;
   }
   public void op2(int x,int y){
      try{
         val+=x/y;
      }
      catch(Exception e){
         val=10;
      }
      finally{
         val++;
      }
   }
}
```

12 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q21 – 02/02

```
public class Main {
   public static void main(String[] args) {
      CA obj=new CB();
      obj.op1(10);
      obj.op2(5,8);
      System.out.printf("%d\n",obj.getVal());
   }
}
```

O que será exibido no console quando for executado o método main()?

(A) −10
(B) −9
(C) −3
(D) 10
(E) 11

12 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q21 – 02/02

```
public class Main {
   public static void main(String[] args) {
      CA obj=new CB();
      obj.op1(10);
      obj.op2(5,8);
      System.out.printf("%d\n",obj.getVal());
   }
}
```

instância de CB

val= -10

```
public class CA {
   int val=0;
   public void op1(int x){
      val+=x;
   }
   public void op2(int x,int y){
      val-=x+y;
   }
   //o restante foi omitido
```

```
public class CB extends CA{
   public void op1(int x){
      val-=x;
   }
   public void op2(int x,int y){
      try{
         val+=x/y;
      }
      catch(Exception e){
         val=10;
      }
      finally{
         val++;
      }
   }
}
```

12 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q21 – 02/02

```
public class Main {
   public static void main(String[] args) {
      CA obj=new CB();
      obj.op1(10);
      obj.op2(5,8);
      System.out.printf("%d\n",obj.getVal());
   }
}
```

instância de CB

val= -10

-10+ = 5/8 = -10+0 = -10

```
public class CA {
   int val=0;
   public void op1(int x){
      val+=x;
   }
   public void op2(int x,int y){
      val-=x+y;
   }
   //o restante foi omitido
```

```
public class CB extends CA{
   public void op1(int x){
      val-=x;
   }
   public void op2(int x,int y){
      try{
         val+=x/y;
      }
      catch(Exception e){
         val=10;
      }
      finally{
         val++;
      }
   }
}
```

--10++ = -9

12 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q21 – 02/02

```
public class Main {
   public static void main(String[] args) {
      CA obj=new CB();
      obj.op1(10);
      obj.op2(5,8);
      System.out.printf("%d\n",obj.getVal());
   }
}
```

O que será exibido no console quando for executado o método main()?

(A) −10

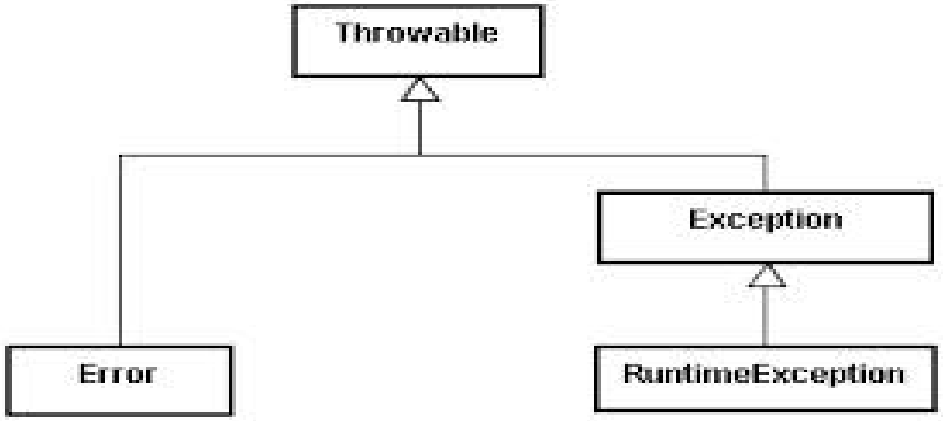
(B) −9

(C) −3

(D) 10

(E) 11

Revisão Teórica – Exceptions (continuação)




```
catch(ExceçãoEspecífica){
//Código de tratamento da ExceçãoEspecifica
}

catch (ExceçãoGenérica){
//Código de tratamento da ExcecaoGenerica
}
```

A exceção mais específica dentro da hierarquia deverá vir antes.

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

Como resultado da execução desse programa, é impressa a cadeia de caracteres

(A) UXYZ
(B) UXZ
(C) XY
(D) XYZ
(E) XZ

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

É “lançada” uma exception do tipo IllegalArgumentException

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

← A exceção específica é capturada e é impresso “X”

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

Não entrará aqui, pois o tratamento foi feito na Exception mais específica

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

← Como o finally sempre é verificado, é impresso “Z”

13 – Petrobras Distribuidora 01/2011 – Analista de Sistemas – Ênfase em JAVA - CRM e WEB – Q44

Considere a classe em Java apresentada a seguir.

```
public class Questao {
   public static void main(String[] args) {
      try {
         op(0);
      } catch (IllegalArgumentException e) {
         System.out.print ("X");
      } catch (Exception e) {
         System.out.print ("Y");
      } finally {
         System.out.print ("Z");
      }
   }
   private static void op(int v) throws IllegalArgumentException {
      if (v == 0)
         throw new IllegalArgumentException("U");
   }
}
```

Como resultado da execução desse programa, é impressa a cadeia de caracteres

(A) UXYZ
(B) UXZ
(C) XY
(D) XYZ

(E) XZ

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46 – 01/02

```
1  class EX {
2
3      public static void f() {
4          throw new RuntimeException("Não implementada");
5      }
6
7      public static void main(String args[]) {
8          System.out.println("INICIO");
9          f();
10         System.out.println("FIM");
11     }
12 }
13
```

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46 – 02/02

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa
(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

(C) compila e, ao executar, imprime 3 mensagens na saída padrão: INICIO, Não implementada e FIM.

(D) compila e, ao executar, imprime, na saída padrão, INICIO, Não implemetada e, em seguida, o programa é abortado.

(E) compila e, ao executar, imprime, na saída padrão, INICIO e, em seguida, é abortado, imprimindo, na saída de erro, o rastro da pilha, incluindo a mensagem "Não implementada".

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, “ throws RuntimeException”, indicando que a função f pode lançar exceções.

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

(C) compila e, ao executar, imprime 3 mensagens na saída padrão: INICIO, Não implementada e FIM.

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

(C) compila e, ao executar, imprime 3 mensagens na saída padrão: INICIO, Não implementada e FIM.

(D) compila e, ao executar, imprime, na saída padrão, INICIO, Não implemetada e, em seguida, o programa é abortado.

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

(C) compila e, ao executar, imprime 3 mensagens na saída padrão: INICIO, Não implementada e FIM.

(D) compila e, ao executar, imprime, na saída padrão, INICIO, Não implemetada e, em seguida, o programa é abortado.

(E) compila e, ao executar, imprime, na saída padrão, INICIO e, em seguida, é abortado, imprimindo, na saída de erro, o rastro da pilha, incluindo a mensagem "Não implementada".

14 – TRANSPETRO 03-2011 – Analista de Sistemas Jr. Eng. de Software – Q46

Analisando-se o código acima, escrito na linguagem java, conclui-se, quanto à compilação e à execução, que o programa

(A) não compila e não executa, pois falta, na linha 3, " throws RuntimeException", indicando que a função f pode lançar exceções.

(B) não compila e não executa, pois a linha 9 deveria estar envolvida por uma construção try/catch, uma vez que a função f pode lançar uma exceção do tipo RuntimeException.

(C) compila e, ao executar, imprime 3 mensagens na saída padrão: INICIO, Não implementada e FIM.

(D) compila e, ao executar, imprime, na saída padrão, INICIO, Não implemetada e, em seguida, o programa é abortado.

(E) compila e, ao executar, imprime, na saída padrão, INICIO e, em seguida, é abortado, imprimindo, na saída de erro, o rastro da pilha, incluindo a mensagem "Não implementada".

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26 – 01/02

Sejam as seguintes classes Java:

```
public class Teste {
   private int x;
   public Teste() {
      x=10;
   }
   public Teste(int c,int d) {
      x=c+d;
   }
   public int getX() {
      return x;
   }
```

```
public void m1(int a) {
   int p;
   try {
      p=x%a;
   if(p<4)
      throw new Exc01();
   }
   catch(Exc01 e) {
      x+=5;
      return;
   }
   catch(Exception e) {
      x+=7;
      return;
   }
   finally {
      x+=9;
   }
   return;
}
```

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26 – 02/02

```
public class Q01 {
    public static void main(String[] args){
        Teste t=new Teste(1,2);
         t.m1(5);
         System.out.println(t.getX());
    }
}
```

O que será exibido no console quando da execução da função main() acima?

(A) 19
(B) 17
(C) 14
(D) 10
(E) 8

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26

```
public class Q01 {
   public static void main(String[] args){
      Teste t=new Teste(1,2);
       t.m1(5);
       System.out.println(t.getX());
   }
}
 public class Teste {
    private int x;
    public Teste() {
       x=10;
    }
    public Teste(int c,int d) {
       x=c+d;
    }
    public int getX() {
       return x;
    }
```

← chama o construtor da classe Teste passando dois inteiros

← x = 1+2 → x = 3

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26

```
public class Q01 {
   public static void main(String[] args){
      Teste t=new Teste(1,2);
      t.m1(5);
      System.out.println(t.getX());
   }
}
```

chama o método m1 da classe Teste

valores até o momento
x = 3

```
public void m1(int a) {
   int p;
   try {
      p=x%a;
   if(p<4)
      throw new Exc01();
   }
   catch(Exc01 e) {
      x+=5;
      return;
   }
   catch(Exception e) {
      x+=7;
      return;
   }
   finally {
      x+=9;
   }
   return;
}
```

p=3%5=3

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26

```
public void m1(int a) {
   int p;
   try {
      p=x%a;
   if(p<4)
      throw new Exc01();
   }
   catch(Exc01 e) {
      x+=5;
      return;
   }
   catch(Exception e) {
      x+=7;
      return;
   }
   finally {
      x+=9;
   }
   return;
}
```

lança a exceção Exc01

x=3+5=8

valores até o momento
x = 3
p=3

15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26

```
public void m1(int a) {
   int p;
   try {
      p=x%a;
   if(p<4)
      throw new Exc01();
   }
   catch(Exc01 e) {
      x+=5;
      return;
   }
   catch(Exception e) {
      x+=7;
      return;
   }
   finally {
      x+=9;    ←  x=8+9=17
   }
   return;
}
```

valores até o momento
x = 8
p=3

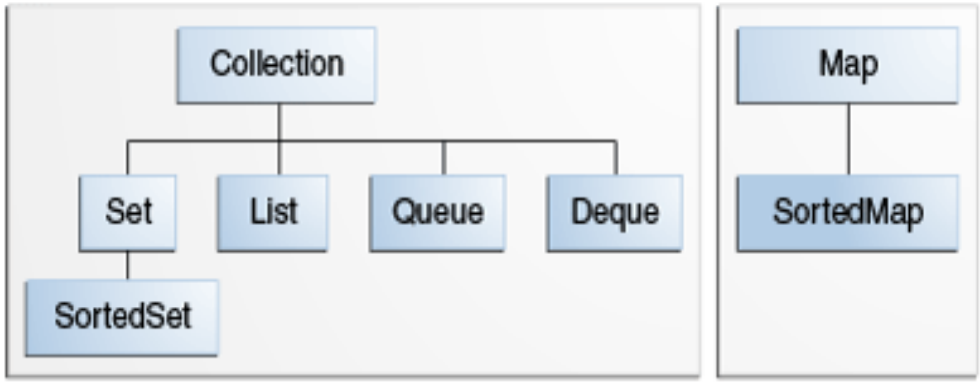
15 – FINEP 01/2011 – Analista - Desenvolvimento de Sistemas – Q26 – 02/02

```
public class Q01 {
    public static void main(String[] args){
        Teste t=new Teste(1,2);
         t.m1(5);
         System.out.println(t.getX());
    }
}
```

O que será exibido no console quando da execução da função main() acima?

(A) 19
(B) 17
(C) 14
(D) 10
(E) 8

Revisão Teórica – Collections



- Um detalhe importante é que nem todas as interfaces herdam de Collection, embora Map e SortedMap também sejam consideradas conjuntos.
- “Collection” é o nome dado à interface java.util.Collection, da qual Set, List, Queue e Deque são derivadas.
- “Collections” é a classe java.util.Collections a qual armazena vários métodos estáticos para serem usados com os conjuntos.
- Conjuntos se dividem em: **listas**, **conjuntos**, **mapas** e **queues**.

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();
(B) Set<Integer> b=new TreeSet<Integer>();
(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();
(D) SortedSet<Integer> d=new TreeSet<Integer>();
(E) Deque<Integer> e=new LinkedList<Integer>();

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>(); ←

(B) Set<Integer> b=new TreeSet<Integer>();

(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();

(D) SortedSet<Integer> d=new TreeSet<Integer>();

(E) Deque<Integer> e=new LinkedList<Integer>();

TreeSet implementa SortedSet, que por sua vez implementa Set, que por sua vez implementa Collection

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();

(B) Set<Integer> b=new TreeSet<Integer>(); ←

(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();

(D) SortedSet<Integer> d=new TreeSet<Integer>();

(E) Deque<Integer> e=new LinkedList<Integer>();

TreeSet implementa SortedSet, que por sua vez implementa Set

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();
(B) Set<Integer> b=new TreeSet<Integer>();
(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();
(D) SortedSet<Integer> d=new TreeSet<Integer>(); ← TreeSet implementa SortedSet
(E) Deque<Integer> e=new LinkedList<Integer>();

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();
(B) Set<Integer> b=new TreeSet<Integer>();
(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();
(D) SortedSet<Integer> d=new TreeSet<Integer>();
(E) Deque<Integer> e=new LinkedList<Integer>(); ← Deque “conhece” LinkedList

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();
(B) Set<Integer> b=new TreeSet<Integer>();
(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values(); ← HashMap e Set não são compatíveis
(D) SortedSet<Integer> d=new TreeSet<Integer>();
(E) Deque<Integer> e=new LinkedList<Integer>();

16 – FINEP 01/2013 – Analista - Desenvolvimento de Sistemas – Q42

Seja o seguinte programa Java:

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Collection<Integer> a=new TreeSet<Integer>();
      Set<Integer> b=new TreeSet<Integer>();
      Set<Integer> c=(new HashMap<Integer,Integer>()).values();
      SortedSet<Integer> d=new TreeSet<Integer>();
      Deque<Integer> e=new LinkedList<Integer>();
   }
}
```

Qual comando produz um erro de compilação?

(A) Collection<Integer> a=new TreeSet<Integer>();

(B) Set<Integer> b=new TreeSet<Integer>();

(C) Set<Integer> c=(new HashMap<Integer,Integer>()).values();

(D) SortedSet<Integer> d=new TreeSet<Integer>();

(E) Deque<Integer> e=new LinkedList<Integer>();

17 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q26

Uma aplicação Java precisa manter na memória principal do computador uma coleção de objetos com as seguintes características:

- poderá conter dezenas de milhares de objetos;
- seus objetos não estarão ordenados;
- um número considerável de objetos poderá ser inserido em tempo de execução;
- a operação mais executada será o percurso sequencial na ordem inversa de inserção dos objetos na coleção.

Diante dessas características, qual das classes irá proporcionar à aplicação a melhor *performance* em relação à manipulação
dessa coleção?

(A) LinkedList<E>
(B) ArrayList<E>
(C) HashSet<E>
(D) HashMap<K,V>
(E) TreeSet<E>

17 – Petrobras 01/2012 – Analista de Sistemas Jr. Engenharia de Software – Q26

Uma aplicação Java precisa manter na memória principal do computador uma coleção de objetos com as seguintes características:

- poderá conter dezenas de milhares de objetos;
- seus objetos não estarão ordenados;
- um número considerável de objetos poderá ser inserido em tempo de execução;
- a operação mais executada será o percurso sequencial na ordem inversa de inserção dos objetos na coleção.

Diante dessas características, qual das classes irá proporcionar à aplicação a melhor *performance* em relação à manipulação
dessa coleção?

(A) LinkedList<E>
(B) ArrayList<E>
(C) HashSet<E>
(D) HashMap<K,V>
(E) TreeSet<E>

18 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q65 – 01/02

Sejam as seguintes classes Java, que ocupam arquivos distintos:

---------- arquivo ExcecaoA.java ----------

```
public class ExcecaoA extends Exception {
}
```

---------- arquivo ExcecaoAB.java ----------

```
public class ExcecaoAB extends ExcecaoA {
}
```

---------- arquivo ClasseA.java -------

```
public abstract class ClasseA {
   private int x=1;
   int y=2;
   public ClasseA(int p){
      x=p;
   }
public int mt_a(int a,int b){
   try {
      if(a%2==1)
         throw new ExcecaoAB();
      return a+b;
   }
   catch(ExcecaoAB e) {
   return a*x+b*y;
   }
   catch(Exception e) {
      return a*x-b*y;
   }
}
```

18 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q65 – 02/02

```
-------------------- arquivo ClasseB.java ----------------
public class ClasseB extends ClasseA {
   int x=2;
   int y=4;
   public ClasseB(){
      super(0);
   }
   public int mt_a(int a,int b){
   try {
      if(b%2==1) throw new ExcecaoA();
      return a-b;
   }
   catch(ExcecaoAB e) { return 4*y;}
   catch(Exception e) { return 5*x;}
   finally {
      x++;
      y++;
   }
}
```

```
-------------------- arquivo Q07.java ----------------
public class Q07 {
   public static void main(String[] args) {
      ClasseA c=new ClasseB();
      System.out.println(c.mt_a(2,3));
   }
}
```

O que será exibido no console quando o método main()
for executado?

(A) 10
(B) –1
(C) 5
(D) 0
(E) 14

18 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q65

```
public class Q07 {
   public static void main(String[] args) {
      ClasseA c=new ClasseB();
      System.out.println(c.mt_a(2,3));
   }
}

public class ClasseB extends ClasseA {
   int x=2;
   int y=4;
   public ClasseB(){
      super(0);
   }
   public int mt_a(int a,int b){
   try {
      if(b%2==1) throw new ExcecaoA();
      return a-b;
   }
   catch(ExcecaoAB e) { return 4*y;}
   catch(Exception e) { return 5*x;}
   finally {
      x++;
      y++;
}
```

polimorfismo

método da ClasseB

O construtor de A será chamado

```
---------- arquivo ClasseA.java -------
public abstract class ClasseA {
   private int x=1;
   int y=2;
   public ClasseA(int p){
      x=p;
   }
public int mt_a(int a,int b){
   try {
      if(a%2==1)
         throw new ExcecaoAB();
      return a+b;
   }
   catch(ExcecaoAB e) {
   return a*x+b*y;
   }
   catch(Exception e) {
      return a*x-b*y;
   }
}
```

18 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q65 – 02/02

---------- arquivo ClasseB.java ----------

```
public class ClasseB extends ClasseA {
   int x=2;
   int y=4;
   public ClasseB(){
      super(0);
   }
   public int mt_a(int a,int b){
   try {
      if(b%2==1) throw new ExcecaoA();
      return a-b;
   }
   catch(ExcecaoAB e) { return 4*y;}
   catch(Exception e) { return 5*x;}
   finally {
      x++;
      y++;
   }
}
```

---------- arquivo Q07.java ----------

```
public class Q07 {
   public static void main(String[] args) {
      ClasseA c=new ClasseB();
      System.out.println(c.mt_a(2,3));
   }
}
```

O que será exibido no console quando o método main()
for executado?

(A) 10
(B) –1
(C) 5
(D) 0
(E) 14

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

Qual o resultado (saída) do programa acima?

(A) 1
(B) 3
(C) 15
(D) 153
(E) 1530

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=153

antes do while
total=0.0
n=153
qtdDigitos=0
temp=153

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=153

1ª execução do while
total=0.0
n=153
qtdDigitos=1
temp=15

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=15

2ª execução do while
total=0.0
temp=15
n=153
qtdDigitos=2
temp=1

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=1

3ª execução do while
total=0.0
temp=15
n=153
qtdDigitos=3
temp=0

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=153

antes do 2º while
total=0.0
temp=153
n=153
qtdDigitos=3
temp=0

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=153

1ª execução do while
total=27
temp=15
n=153
qtdDigitos=3
digito=3

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=15

2ª execução do while
total=27+125=152
temp=1
n=153
qtdDigitos=3
digito=5

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

temp=1

total=153

3ª execução do while
total=152+1=153
temp=0
n=153
qtdDigitos=3
digito=1

19 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q69

Considere o trecho de código fonte a seguir, escrito em linguagem Java.

```
public class Main {
   public static void main(String[] args) {
      double total = 0.0;
      int temp, n;
      temp = n = 153;
      int qtdDigitos = 0;
      while (temp > 0) {
         qtdDigitos++;
         temp = temp / 10;
      }
      temp = n;
      while (temp > 0) {
         int digito = temp % 10;
         total += Math.pow(digito, qtdDigitos);
         temp = temp / 10;
      }
      System.out.println((int)total);
   }
}
```

Qual o resultado (saída) do programa acima?

(A) 1
(B) 3
(C) 15
(D) 153
(E) 1530

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 01/04

Um programa Java armazena informações sobre notas fiscais relativas a serviços prestados por uma empresa. As notas fiscais são divididas em notas emitidas para pessoas físicas (PF) e para pessoas jurídicas (PJ). As classes a seguir são utilizadas para representar tais informações:

```
package vendas;
   public abstract class NotaFiscal {
      int data;
      String codCliente;
      double valor;
      public double getValor(){
         return valor;
      }
   public abstract double getValImposto();
}
```

```
package vendas;
public class NotaPJ extends NotaFiscal {
   public double getValImposto(){
   // método irrelevante para o problema em questão
   }
}
```

```
package vendas;
public class NotaPF extends NotaFiscal {
   public double getValImposto(){
   // você tem que implementar este método
   }
}
```

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 02/04

As informações sobre as notas fiscais estão organizadas em uma árvore binária de busca, **ordenadas segundo a data de emissão da nota**. A classe a seguir representa um nó dessa árvore binária de busca:

```
package estruturas;
   import vendas.*;
   class ArvNo {
      int data;
      ArvNo esq=null,dir=null;
      NotaFiscal ln[];
   }
```

As variáveis esq e dir são usadas, respectivamente, para referenciar as subárvores à esquerda e à direita de um nó. A variável data armazena a data de emissão da nota, no formato AAMMDD (por exemplo, 22/04/2012 será armazenada como o inteiro 120422). A variável ln referencia um array contendo todas as notas fiscais emitidas na data em questão (variável data).

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 03/04

A classe Arv é utilizada para implementar uma árvore binária de busca constituída por nós do tipo ArvNo. Sua estrutura é a seguinte:

```
package estruturas;
import vendas.*;
public class Arv {
   private ArvNo raiz=null;
   public double totImpostoPF(int dini,int dfim){
      return calcTotImpostoPF(raiz,dini,dfim);
   }
   private double calcTotImpostoPF(ArvNo r,int dini,int dfim){
      // você tem que implementar este método
   }
}
```

O método calcTotImpostoPF() tem por objetivo calcular o somatório do imposto sobre serviços (ISS) recolhido sobre cada nota fiscal emitida para uma pessoa física entre duas datas passadas como parâmetros (dini <= ArvNo.data <= dfim).
Por ser um método privado, ele é acionado a partir do método público totImpostoPF().

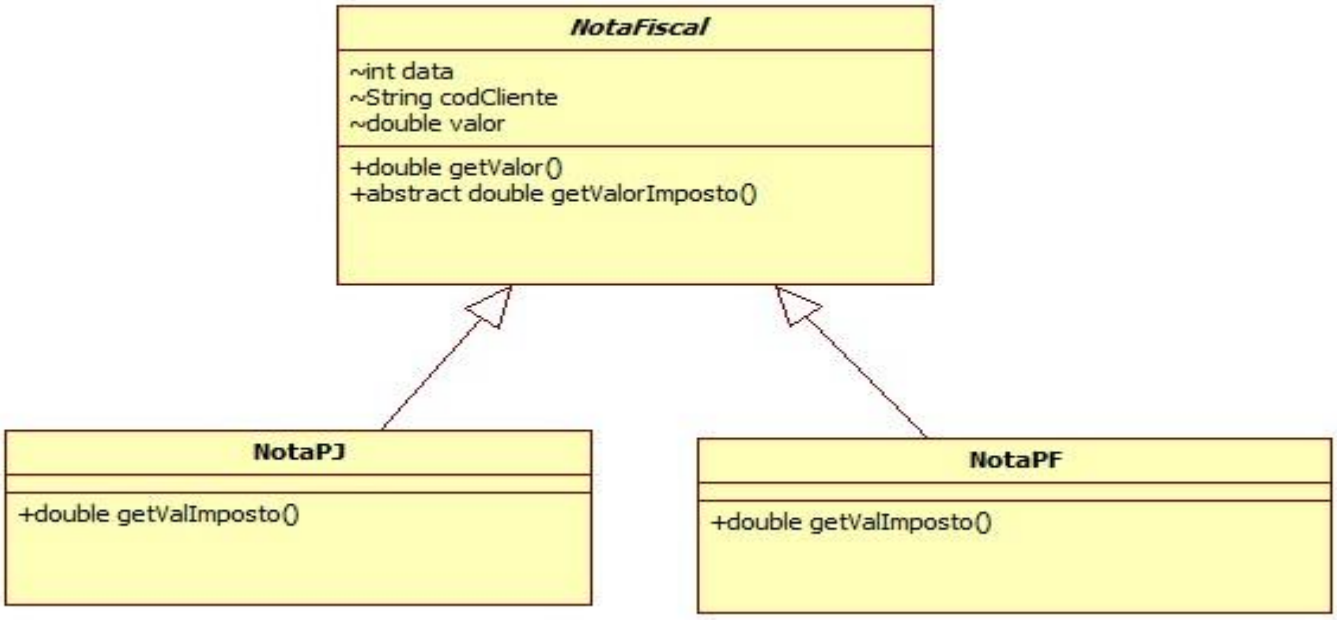
20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

Escreva um algoritmo **recursivo** para o método calcTotImpostoPF() que percorra a árvore binária de busca e calcule o somatório do ISS recolhido sobre cada nota fiscal emitida para uma pessoa física entre as datas fornecidas. A sua solução deve levar em conta o seguinte:

- Apenas as notas referentes a pessoas físicas devem ser consideradas.
- O ISS cobrado sobre notas fiscais emitidas para pessoas físicas é de 10% do valor da nota.
- Assuma que a variável Arv.raiz referencia o nó raiz de uma árvore (possivelmente vazia) que foi carregada por um método não exibido no enunciado desta questão.
- Nenhuma nova propriedade (método, variável de instância ou variável estática) pode ser inserida nas classes apresentadas no enunciado.
- Nenhuma modificação pode ser feita no código apresentado no enunciado, com exceção do corpo dos métodos que façam parte da solução do problema.
- O algoritmo deve levar obrigatoriamente em conta as características de uma árvore binária de busca.
- Apenas as implementações **recursivas** do método calcTotImpostoPF() serão consideradas.

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

1ª Parte – Entendimento da questão



20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

1ª Parte – Entendimento da questão

```
package estruturas;
   import vendas.*;
   class ArvNo {
      int data;
      ArvNo esq=null,dir=null;
      NotaFiscal ln[];
   }
```

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

1ª Parte – Entendimento da questão

```
package estruturas;
import vendas.*;
public class Arv {
   private ArvNo raiz=null;
   public double totImpostoPF(int dini,int dfim){
      return calcTotImpostoPF(raiz,dini,dfim);
   }
   private double calcTotImpostoPF(ArvNo r,int dini,int dfim){
      // você tem que implementar este método
   }
}
```

Deverá calcular o **somatório do imposto sobre serviços (ISS) recolhido sobre cada nota fiscal** emitida para uma pessoa física entre duas datas passadas como parâmetros.

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

1ª Parte – Entendimento da questão

- Apenas as notas referentes a pessoas físicas devem ser consideradas.
- O ISS cobrado sobre notas fiscais emitidas para pessoas físicas é de 10% do valor da nota.
- Assuma que a variável Arv.raiz referencia o nó raiz de uma árvore (possivelmente vazia) que foi carregada por um método não exibido no enunciado desta questão.
- Nenhuma nova propriedade (método, variável de instância ou variável estática) pode ser inserida nas classes apresentadas no enunciado.
- O algoritmo deve levar obrigatoriamente em conta as características de uma árvore binária de busca.
- Apenas as implementações **recursivas** do método calcTotImpostoPF() serão consideradas.

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

2ª Parte – Implementação do método getValImposto

```
public double getValImposto(){
   return getValor()*0.1;
}
```

```
package vendas;
public class NotaPF extends NotaFiscal {
   public double getValImposto(){
   // você tem que implementar este método
   }
}
```

```
package vendas;
   public abstract class NotaFiscal {
      int data;
      String codCliente;
      double valor;
      public double getValor(){
         return valor;
      }
   public abstract double getValImposto();
}
```

Dica: O ISS cobrado sobre notas fiscais emitidas para pessoas físicas é de <u>10% do valor da nota</u>.

20 – BNDES 01/2012 – Análise de Sistemas – Desenvolvimento – Q04 (Discursiva) – 04/04

3ª Parte – Implementação do método calcTotImpostoPF

```
private double calcTotImpostoPF(ArvNo r,int dini,int dfim){
   if(r==null)
      return 0;
   if(r.data>=dini && r.data<=dfim){      <-- a data deverá estar no intervalo
      double soma=0.0;
      for(NotaFiscal n:r.ln)      <-- percorre as notas fiscais
         if(n instanceof NotaPF)      <-- condição essencial
            soma+=n.getValImposto();
         return soma+calcTotImpostoPF(r.esq,dini,dfim)+ calcTotImpostoPF(r.dir,dini,dfim);
   }
   if(r.data>dfim)      <-- condição de parada
      return calcTotImpostoPF(r.esq,dini,dfim);
   return calcTotImpostoPF(r.dir,dini,dfim);
}
```

chamada recursiva

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

Sendo a, b, c e d variáveis do tipo *boolean*, qual dos comandos abaixo **NÃO** é equivalente aos demais?

(A) if (b && !(!a && !c)) d = true;

(B) if (! (!a || !b) || (b && c)) d = b;

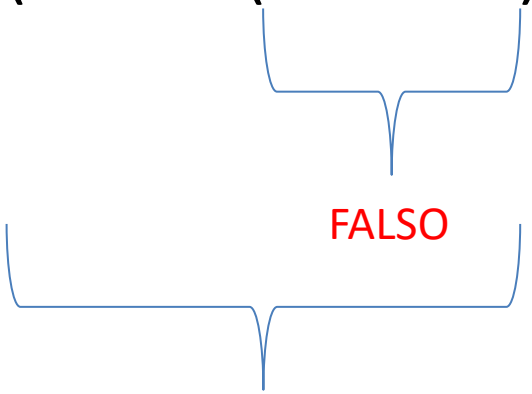
(C) if ((a && b) || (!a && b && c)) d = b;

(D) if ((a && b && !c) || (!a && b && c)) d = true;

(E) if ((a || c) && b) d = a || c;

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

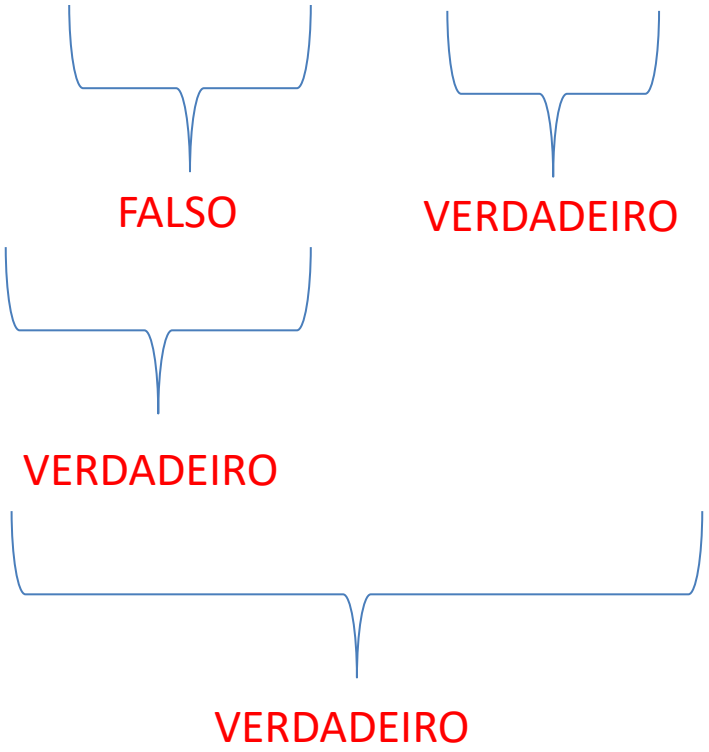
(A)if (b && !(!a && !c)) d = true;



D=VERDADEIRO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

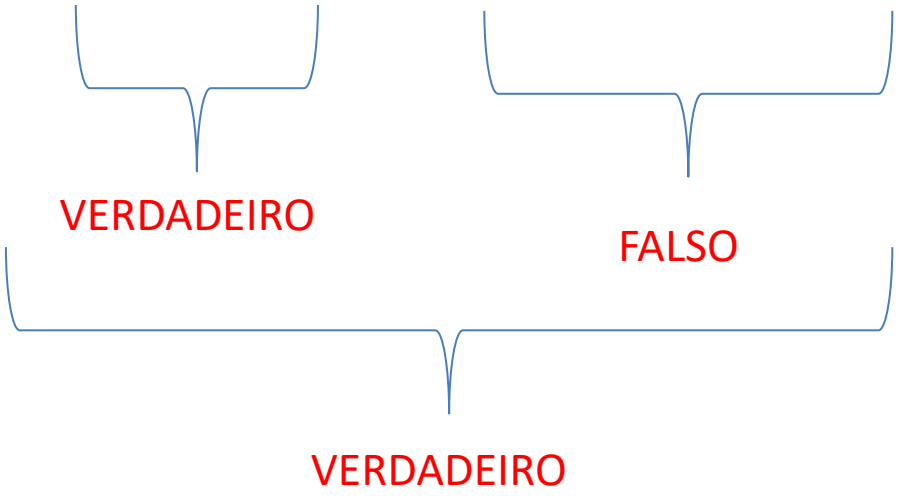
(B) if (! (!a || !b) || (b && c)) d = b;



D=VERDADEIRO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

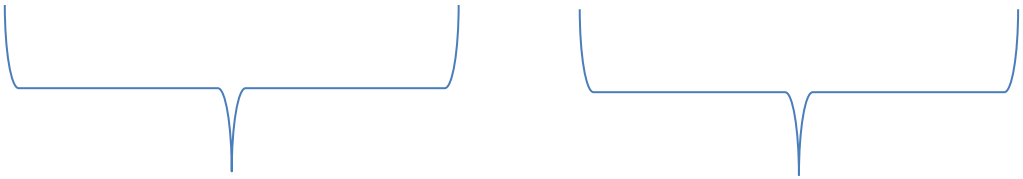
(C) if ((a && b) || (!a && b && c)) d = b;



D=VERDADEIRO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

(D) if ((a && b && !c) || (!a && b && c)) d = true;



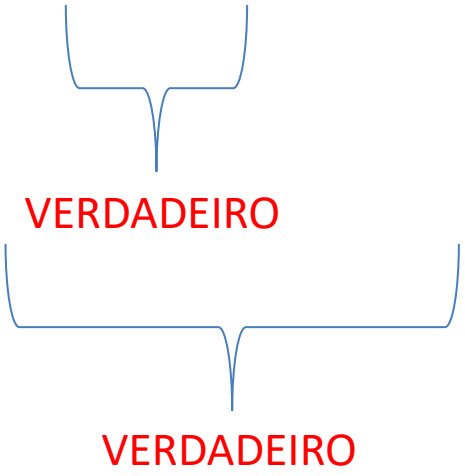
FALSO

FALSO

D=FALSO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

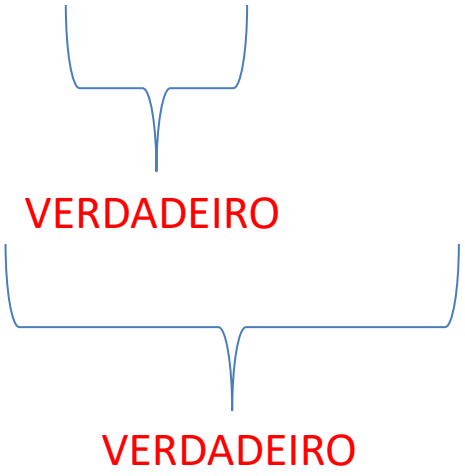
(E) if ((a || c) && b) d = a || c;



D=VERDADEIRO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

(E) if ((a || c) && b) d = a || c;



D=VERDADEIRO

21 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – MAR/2010 – Q46

Sendo a, b, c e d variáveis do tipo *boolean*, qual dos comandos abaixo **NÃO** é equivalente aos demais?

(A) if (b && !(!a && !c)) d = true;

(B) if (! (!a || !b) || (b && c)) d = b;

(C) if ((a && b) || (!a && b && c)) d = b;

(D) if ((a && b && !c) || (!a && b && c)) d = true;

(E) if ((a || c) && b) d = a || c;

Revisão Teórica – Características da Linguagem JAVA (Continuação)

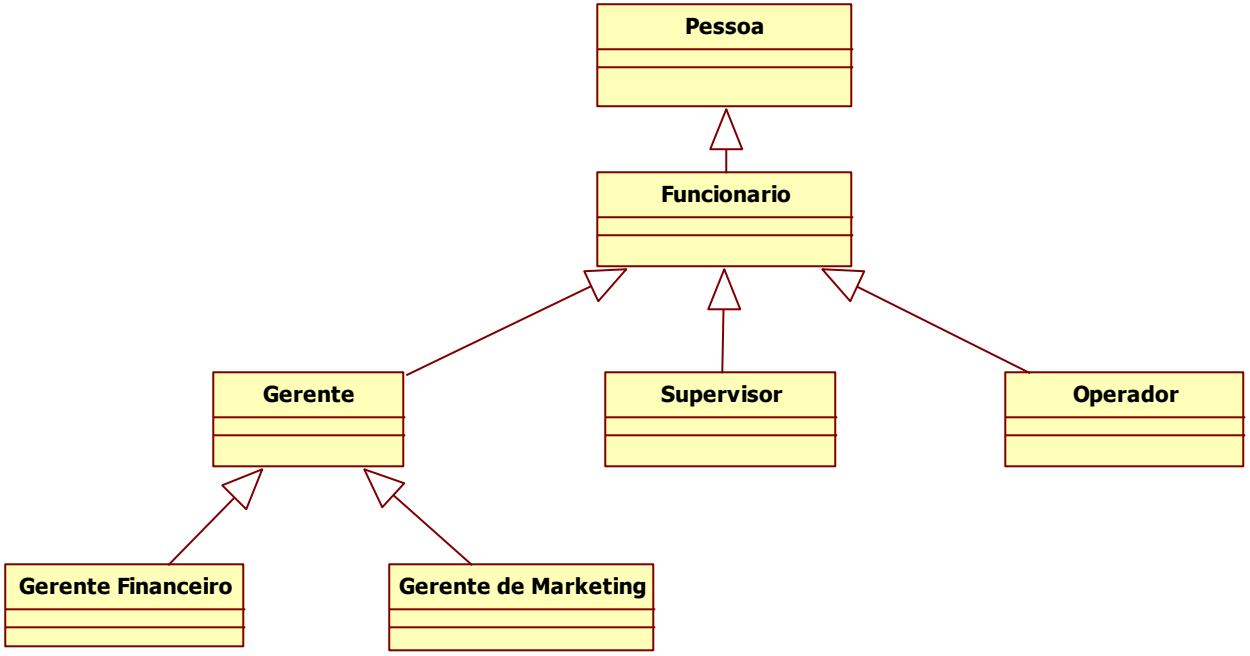
- **Herança** – “Forma de reutilização de software na qual uma nova classe é criada, absorvendo membros de uma classe existente e aprimorada com capacidades novas ou modificadas.” (Deitel)

  - Em java, a hierarquia de classes se inicia na classe Object (pacote java.lang).

  - Os relacionamentos podem ser divididos em: “**é um**” e “**tem um**”.

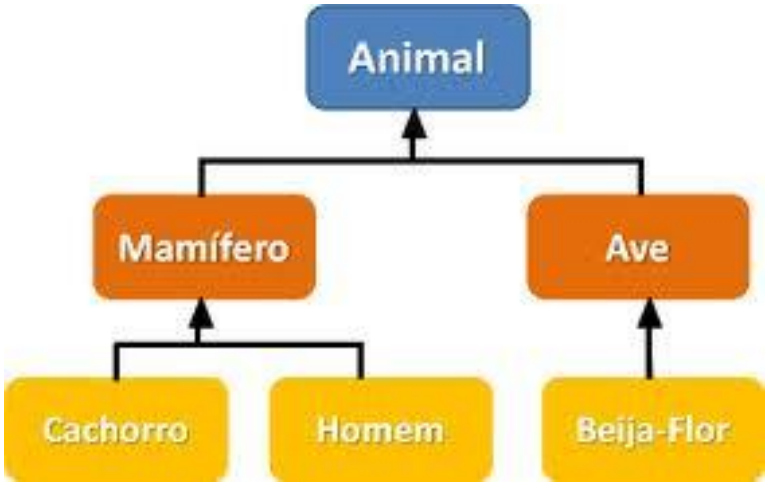
Revisão Teórica – Características da Linguagem JAVA (Continuação)

- **Hierarquia de Herança**



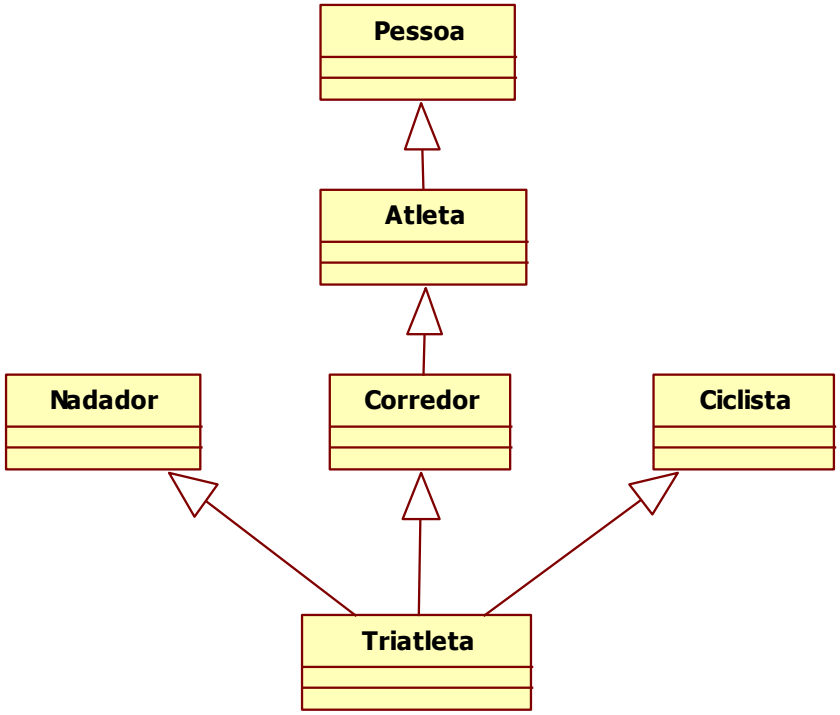
Revisão Teórica – Características da Linguagem JAVA (Continuação)

- **Herança Simples** – Uma subclasse possui apenas uma superclasse.



Revisão Teórica – Características da Linguagem JAVA (Continuação)

- **Herança Múltipla** – Uma subclasse possui mais de uma superclasse.



OBS: A herança múltipla em JAVA poderá ser substituída pela implementação de interfaces.

Revisão Teórica – Utilização de Strings em JAVA

Características da classe String:

- São Imutáveis
- Residem em um local especial da memória chamado pool de strings
- Seu uso contínuo poderá gerar diversas Strings temporárias
- Seu uso contínuo acarreta em perda de performance

22 – BNDES 01/2009 – Análise de Sistemas – Desenvolvimento – Q65

Qual das afirmações a seguir faz uma apreciação correta a respeito da linguagem de programação Java?

(A) O conceito de herança múltipla é implementado nativamente.

(B) Uma classe pode implementar somente uma interface ao mesmo tempo.

(C) Uma classe pode implementar uma interface ou ser subclasse de outra classe qualquer, mas não ambos simultaneamente.

(D) A construção de um método que pode levantar uma exceção, cuja instância é uma subclasse de java.lang.RuntimeException, não exige tratamento obrigatório por parte do programador dentro daquele método.

(E) Objetos da classe java.lang.String têm comportamento otimizado para permitir que seu valor seja alterado sempre que necessário, liberando imediatamente a memória usada pelo conteúdo anterior.

22 – BNDES 01/2009 – Análise de Sistemas – Desenvolvimento – Q65

Qual das afirmações a seguir faz uma apreciação correta a respeito da linguagem de programação Java?

(A) O conceito de herança múltipla é implementado nativamente.

(B) Uma classe pode implementar somente uma interface ao mesmo tempo.

(C) Uma classe pode implementar uma interface ou ser subclasse de outra classe qualquer, mas não ambos simultaneamente.

(D) A construção de um método que pode levantar uma exceção, cuja instância é uma subclasse de java.lang.RuntimeException, não exige tratamento obrigatório por parte do programador dentro daquele método.

(E) Objetos da classe java.lang.String têm comportamento otimizado para permitir que seu valor seja alterado sempre que necessário, liberando imediatamente a memória usada pelo conteúdo anterior.

23 – BNDES 2008 – Análise de Sistemas – Desenvolvimento – Q40

Observe o seguinte programa JAVA:

```
package p;
public class exemplo {
   public exemplo() {
}
```

```
public static void main(String[] args) {
try {
     System.out.println(1/0);
     System.out.println("M");
    }
catch (ArithmeticException ex2) {
      System.out.print("X");
    }
catch (Exception ex3) {
      System.out.print("Y");
    }
finally {
      System.out.print("Z");
     }
System.out.print("F");
  }
}
```

A saída desse programa é

a) MXYZF
b) XYZ
c) XF
d) XZF
e) XYZF

23 – BNDES 2008 – Análise de Sistemas – Desenvolvimento – Q40

Observe o seguinte programa JAVA:

```
package p;
public class exemplo {
   public exemplo() {
}
```

```
public static void main(String[] args) {
try {
     System.out.println(1/0);
     System.out.println("M");
    }
catch (ArithmeticException ex2) {
      System.out.print("X");
    }
catch (Exception ex3) {
      System.out.print("Y");
    }
finally {
      System.out.print("Z");
     }
System.out.print("F");
  }
}
```

A saída desse programa é

a) MXYZF
b) XYZ
c) XF
d) XZF
e) XYZF

Revisão Teórica – Pilha e Heap

- Variáveis de instância e objetos residirão na heap.

- Variáveis locais residirão na pilha.

  - ✓ Heap e Pilha são locais da memória.

Estes conceitos são importantes?

Revisão Teórica – Pilha e Heap

```
class Collar{}

class Dog{
    Collar c; //variável de instância
    String name; //variável de instância

    public static void main (String[] args){
        Dog d; //variável local
        d = new Dog();
        d.go(d);
    }
    void go(Dog dog){ //variável local
        c=new Collar();
        dog.setName("Tex");
    }
    void setName(String dogName){ //variável local
        name = dogName;
    }
}
```

main() é colocado na **pilha**

d é criada na **pilha**, mas ainda não existe um objeto Dog

um novo objeto Dog é criado, e atribuído à variável de referência d

uma cópia da variável de referência d é passada para o método go()

o método go() é colocado na **pilha**

um objeto collar é criado na **heap** e atribuído à variável de instância

setName é adicionado à **pilha**

A variável de instância name agora também se refere ao objeto dogName

24 – CAPES 2008 – Análise de Sistemas – Analista em Ciência e Tecnologia Jr.

Em que porção da JVM (Java Virtual Machine) são armazenados objetos instanciados em um programa JAVA ?

a) Heap

b) Gunit

c) Stack Pool

d) Dump Buffer

e) Text Segment

24 – CAPES 2008 – Análise de Sistemas – Analista em Ciência e Tecnologia Jr.

Em que porção da JVM (Java Virtual Machine) são armazenados objetos instanciados em um programa JAVA ?

a) Heap

b) Gunit

c) Stack Pool

d) Dump Buffer

e) Text Segment

25 – Petrobras 2008 – Analista de Sistemas Jr. Eng. de Software – Q64

Considere o trecho de código a seguir.

```
if (x != x + 0) {
   System.out.println("Condição satisfeita.");
}
```

Se x for da classe String e tiver sido inicializado, esse trecho de código Java

(A) imprimirá a mensagem, apenas se x não for "0".

(B) imprimirá a mensagem, apenas se x não tiver sido inicializado com null.

(C) imprimirá a mensagem, independente do valor de x.

(D) gerará um erro de compilação.

(E) compilará, mas nunca imprimirá a mensagem.

25 – Petrobras 2008 – Analista de Sistemas Jr. Eng. de Software – Q64

Considere o trecho de código a seguir.

```
if (x != x + 0) {
   System.out.println("Condição satisfeita.");
}
```

Se x for da classe String e tiver sido inicializado, esse trecho de código Java

(A) imprimirá a mensagem, apenas se x não for "0".

(B) imprimirá a mensagem, apenas se x não tiver sido inicializado com null.

(C) imprimirá a mensagem, independente do valor de x.

(D) gerará um erro de compilação.

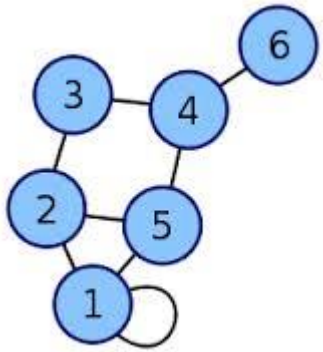
(E) compilará, mas nunca imprimirá a mensagem.

Revisão Teórica – Complexidade Ciclomática

- Também chamada de Complexidade Condicional.
- Indica a complexidade de um programa, método ou trecho de programa.
- Mede o número de caminhos linearmente independentes.

A medição poderá ser feita de duas formas:

- **Grafo de Fluxo**
- **Contando o número de estruturas de repetição e seleção**



Revisão Teórica – Complexidade Ciclomática

A complexidade ciclomática em um método será calculada da seguinte forma:

Para n igual ao número de estruturas de seleção e repetição no programa, então a complexidade ciclomática do programa é n+1.

Operação	Quantidade de Pontos
IF-THEN	1 ponto
IF-THEN-ELSE	1 ponto
CASE	1 ponto para cada opção, exceto OTHERWHISE
FOR	1 ponto
REPEAT	1 ponto
OR ou AND	1 ponto para cada
NOT	não conta
Chamada recursiva	não conta

Revisão Teórica – Complexidade Ciclomática

```
if (num==0){
   fib=0;
}
else{
   if (num==1){
      fib=1;
   }
   else{
      ultimoFib=1;
      cont=1;
      do {
         penultimoFib=ultimoFib;
         ultimoFib=fib;
         fib=penultimoFib+ultimoFib;
         cont++;
      } while (cont==num);
   }
}
```

No exemplo à esquerda, qual é a complexidade ciclomática?

Revisão Teórica – Complexidade Ciclomática

```
if (num==0){        <-- será contado
   fib=0;
}
else{               <-- não será contado
   if (num==1){     <-- será contado
      fib=1;
   }
   else{            <-- não será contado
      ultimoFib=1;
      cont=1;
      do {
         penultimoFib=ultimoFib;
         ultimoFib=fib;
         fib=penultimoFib+ultimoFib;
         cont++;
      } while (cont==num);   <-- será contado
   }
}
```

Revisão Teórica – Complexidade Ciclomática

```
if (num==0){
   fib=0;
}
else{
   if (num==1){
      fib=1;
   }
   else{
      ultimoFib=1;
      cont=1;
      do {
         penultimoFib=ultimoFib;
         ultimoFib=fib;
         fib=penultimoFib+ultimoFib;
         cont++;
      } while (cont==num);
   }
}
```

A complexidade ciclomática do trecho de código é 4.

26 – Petrobras 2008 – Análise de Sistemas Jr. – Processos de Negócios

Considere o seguinte código de um método de uma classe Java:

```
public boolean primo( int x ) {
   if (x == 1 || x == 2) {
      return true;
   }
   int raiz = (int) Math.sqrt((double)x);
   for (int i = 2; i <= raiz; i++) {
      if (x % i == 0) {
         return false;
      }
   }
   return true;
   }
```

Qual a complexidade ciclomática do método?

a) 2
b) 3
c) 4
d) 5
e) 6

26 – Petrobras 2008 – Análise de Sistemas Jr. – Processos de Negócios

Considere o seguinte código de um método de uma classe Java:

```
public boolean primo( int x ) {
   if (x == 1 || x == 2) {
      return true;
   }
   int raiz = (int) Math.sqrt((double)x);
   for (int i = 2; i <= raiz; i++) {
      if (x % i == 0) {
         return false;
      }
   }
   return true;
   }
```

Qual a complexidade ciclomática do método?

a) 2
b) 3
c) 4
d) 5
e) 6

27 – TJ-RO 2008 – Analista Judiciário – Tecnologia da Informação

O que imprimirá a linha de código em Java a seguir?

```
System.out.println("1+1+1="+1+1+'1');
```

a) 1+1+1=21

b) 3=21

c) 1+1+1=111

d) 3=111

e) 111=111

27 – TJ-RO 2008 – Analista Judiciário – Tecnologia da Informação

O que imprimirá a linha de código em Java a seguir?

```
System.out.println("1+1+1="+1+1+'1');
```

a) 1+1+1=21

b) 3=21

c) 1+1+1=111

d) 3=111

e) 111=111

Curiosidades

```
System.out.println(1+1);
```

```
System.out.println(1+'1');
```

```
System.out.println('a'+'1');
```

```
System.out.println("a"+'1');
```

```
System.out.println('a'+1);
```

Curiosidades

System.out.println(1+1); ← 2

System.out.println(1+'1'); ← 50

System.out.println('a'+'1'); ← 146

System.out.println("a"+'1'); ← a1

System.out.println('a'+1); ← 98

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 01/02

Analise as seguintes classes escritas em JAVA:

```
package classes;
public class Main {
   static public abstract class Operacao{
      public abstract int executar(int pa, int pb);
   }
   static public class classeA extends Operacao{
      public classeA(String s) {
         System.out.println(s);
      }
      private void metodoX(){
         System.out.println("Método X");
      }
      public int executar(int pa, int pb){
         return pa*pb;
   }
}
```

```
static public class classeB extends Operacao {
   public int executar(int pa, int pb){
      return pa+pb;
   }
}
static public class classeC extends classeA{
   public classeC(String s){
      super(s);
   }
   public static void processar(Operacao op, int pa, int pb){
      System.out.println(op.executar(pa, pb));
   }
 }
   public static void main(String[] args) {
     classeC.processar(new classeB(), 2, 3);
   }
```

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla.

b) método metodoX não está disponível a objetos criados para a classeC.

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla. 

b) método metodoX não está disponível a objetos criados para a classeC.

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42

Analise as seguintes classes escritas em JAVA:

```
package classes;
public class Main {
   static public abstract class Operacao{
      public abstract int executar(int pa, int pb);
   }
   static public class classeA extends Operacao{
      public classeA(String s) {
         System.out.println(s);
      }
      private void metodoX(){
         System.out.println("Método X");
      }
      public int executar(int pa, int pb){
         return pa*pb;
   }
}
```

```
static public class classeB extends Operacao {
   public int executar(int pa, int pb){
      return pa+pb;
   }
}
static public class classeC extends classeA{
   public classeC(String s){
      super(s);
   }
   public static void processar(Operacao op, int pa, int pb){
      System.out.println(op.executar(pa, pb));
   }
 }
   public static void main(String[] args) {
     classeC.processar(new classeB(), 2, 3);
   }
```

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla. 

b) método metodoX não está disponível a objetos criados para a classeC. 

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42

Analise as seguintes classes escritas em JAVA:

```
package classes;
public class Main {
   static public abstract class Operacao{
      public abstract int executar(int pa, int pb);
   }
   static public class classeA extends Operacao{
      public classeA(String s) {
         System.out.println(s);
      }
      private void metodoX(){
         System.out.println("Método X");
      }
      public int executar(int pa, int pb){
         return pa*pb;
   }
}
```

```
static public class classeB extends Operacao {
   public int executar(int pa, int pb){
      return pa+pb;
   }
}
static public class classeC extends classeA{
   public classeC(String s){
      super(s);
   }
   public static void processar(Operacao op, int pa, int pb){
      System.out.println(op.executar(pa, pb));
   }
 }
   public static void main(String[] args) {
     classeC.processar(new classeB(), 2, 3);
   }
```

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla.

b) método metodoX não está disponível a objetos criados para a classeC.

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42

Analise as seguintes classes escritas em JAVA:

```
package classes;
public class Main {
   static public abstract class Operacao{
      public abstract int executar(int pa, int pb);
   }
   static public class classeA extends Operacao{
      public classeA(String s) {
         System.out.println(s);
      }
      private void metodoX(){
         System.out.println("Método X");
      }
      public int executar(int pa, int pb){
         return pa*pb;
   }
}
```

```
static public class classeB extends Operacao {
   public int executar(int pa, int pb){
      return pa+pb;
   }
}
static public class classeC extends classeA{
   public classeC(String s){
      super(s);
   }
   public static void processar(Operacao op, int pa, int pb){
      System.out.println(op.executar(pa, pb));
   }
 }
   public static void main(String[] args) {
     classeC.processar(new classeB(), 2, 3);
   }
```

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla.

b) método metodoX não está disponível a objetos criados para a classeC.

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42

Analise as seguintes classes escritas em JAVA:

```
package classes;
public class Main {
   static public abstract class Operacao{
      public abstract int executar(int pa, int pb);
   }
   static public class classeA extends Operacao{
      public classeA(String s) {
         System.out.println(s);
      }
      private void metodoX(){
         System.out.println("Método X");
      }
      public int executar(int pa, int pb){
         return pa*pb;
   }
}
```

```
static public class classeB extends Operacao {
   public int executar(int pa, int pb){
      return pa+pb;
   }
}
static public class classeC extends classeA{
   public classeC(String s){
      super(s);
   }
   public static void processar(Operacao op, int pa, int pb){
      System.out.println(op.executar(pa, pb));
   }
 }
   public static void main(String[] args) {
     classeC.processar(new classeB(), 2, 3);
   }
```

28 – IBGE 2010 – Análise de Sistemas – Desenvolvimento de Aplicações - Q42 – 02/02

Tendo como base o código acima e as características da programação orientada a objetos em Java, é **INCORRETO** afirmar que o(a)

a) Java não permite herança múltipla.

b) método metodoX não está disponível a objetos criados para a classeC.

c) código demonstra o uso de polimorfismo.

d) método super(s) na classeC, ao ser executado, cria uma instância da classe super.

e) linha System.out.println(op.executar(pa, pb)) irá exibir o valor 5.

Revisão Teórica – Reflection API

A partir da API Reflection, podemos obter as seguintes informações de uma classe:

- Lista de métodos
- Quais interfaces ela implementa
- Quais classes ela estende
- Modificadores aplicados (pubic, abstract, final, ...)
- Pacote no qual ela está contida
- Anotações utilizadas

Revisão Teórica – Synchronized

A palavra chave Synchronized em Java poderá ser utilizada em duas situações: como modificador de um método ou em um trecho de código.

- Em ambos os casos, o seu objetivo é controlar a concorrência.
- Não permite execuções simultâneas.

29 – BACEN 2010 – Analista do Banco Central – Área 1 – Q25

Uma instituição financeira desenvolverá um novo sistema de informação WEB com base na plataforma Java EE 5.
Os programadores devem ter ciência de que, na linguagem de programação Java e tecnologias relacionadas, NÃO é

(A) possível listar, por meio da Reflection API, que anotações foram colocadas em um parâmetro de um método.
(B) possível obter, por meio da Reflection API, as interfaces que determinada classe implementa.
(C) importante declarar métodos como “final”, sempre que possível, para obter aumento de, pelo menos, 50% no desempenho, na JVM da Sun.
(D) importante, por questões de desempenho, evitar utilizar métodos “synchronized”, quando possível.
(E) necessário fechar, explicitamente, objetos JDBC como ResultSet, sob pena de vazamento de memória (memory leak), na conexão com alguns bancos de dados.

29 – BACEN 2010 – Analista do Banco Central – Área 1 – Q25

Uma instituição financeira desenvolverá um novo sistema de informação WEB com base na plataforma Java EE 5.
Os programadores devem ter ciência de que, na linguagem de programação Java e tecnologias relacionadas, NÃO é

(A) possível listar, por meio da Reflection API, que anotações foram colocadas em um parâmetro de um método.
(B) possível obter, por meio da Reflection API, as interfaces que determinada classe implementa.
(C) importante declarar métodos como “final”, sempre que possível, para obter aumento de, pelo menos, 50% no desempenho, na JVM da Sun.
(D) importante, por questões de desempenho, evitar utilizar métodos “synchronized”, quando possível.
(E) necessário fechar, explicitamente, objetos JDBC como ResultSet, sob pena de vazamento de memória (memory leak), na conexão com alguns bancos de dados.

Revisão Teórica – Identificadores

Os identificadores em Java podem ser:

- Identificadores Legais
- Convenções de Código

Os identificadores Legais devem:

- Ser compostos apenas de caracteres Unicode, números, símbolos de moedas e caracteres de conexão (tais como undescore);
- Devem começar com uma letra, um cifrão ($) ou um caracter de conexão;
- Usar palavras que **não** sejam palavras-chave em Java;

Revisão Teórica – Identificadores

Exemplos de caracteres válidos e inválidos em Java:

```
int :b;

int _a;

int $c;

int 7g;

int e#;

int ____2_w;

int nome_maior_do_que_o_tamanho_maximo_permitido;
```

Revisão Teórica – Identificadores

Exemplos de caracteres válidos e inválidos em Java:

int :b; ⟵ inválido

int _a; ⟵ válido

int $c; ⟵ válido

int 7g; ⟵ inválido

int e#; ⟵ inválido

int ____2_w; ⟵ válido

int nome_maior_do_que_o_tamanho_maximo_permitido; ⟵ válido

Revisão Teórica – Identificadores

Lista de palavras-chave Java:

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	**goto**	if	implements	import	instanceof
int	interface	long	**native**	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	**volatile**	while
assert	Enum				

30 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – Q57 – 01/02

Em linguagens de programação, palavras-chaves são aquelas palavras ou identificadores que têm um significado implícito e relevante para a linguagem de programação. Em muitas linguagens, estas palavras-chaves são também palavras reservadas, isto é, não podem ser usadas em outros contextos, pois são reservadas para usos específicos da gramática da linguagem de programação. A linguagem Java possui um pequeno núcleo de palavras reservadas, incluindo os comandos de controle de fluxo (for, while, etc), identificadores de nível de acesso à classe (public, private, etc). Qual das seguintes descrições representa a correta aplicação do conceito de palavras reservadas em Java?

30 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – Q57 – 02/02

a) Não é possível definir um método println dentro de uma classe, pois ele é reservado para uso nas classes que implementam buffers de saída textual em tela e em arquivo.
b) O comando package while; não causa erro nenhum, pois apesar de while ser uma palavra reservada, o compilador identifica o contexto onde a palavra está sendo usada e reconhece-a como o nome do pacote corrente.
c) A definição de uma palavra-chave como palavra reservada impede o uso desta até mesmo como parte de um identificador (como while2), pois os ambientes de desenvolvimento passam a identificar a palavra-chave e causam um erro de compilação.
d) Os nomes das classes de Java não são palavras reservadas, podendo ser utilizados à vontade em outros pacotes, mesmo nos casos de classes de uso comum como a classe File ou a classe Array.
e) Apesar de possuir palavras reservadas, Java, por ser uma linguagem orientada a objetos, permite que se use uma palavra reservada em outro contexto, desde que ela seja qualificada, como no comando package meupacote.while;.

30 – Petrobras 2010 – Analista de Sistemas Jr. Eng. de Software – Q57 – 02/02

a) Não é possível definir um método println dentro de uma classe, pois ele é reservado para uso nas classes que implementam buffers de saída textual em tela e em arquivo.
b) O comando package while; não causa erro nenhum, pois apesar de while ser uma palavra reservada, o compilador identifica o contexto onde a palavra está sendo usada e reconhece-a como o nome do pacote corrente.
c) A definição de uma palavra-chave como palavra reservada impede o uso desta até mesmo como parte de um identificador (como while2), pois os ambientes de desenvolvimento passam a identificar a palavra-chave e causam um erro de compilação.
d) Os nomes das classes de Java não são palavras reservadas, podendo ser utilizados à vontade em outros pacotes, mesmo nos casos de classes de uso comum como a classe File ou a classe Array.
e) Apesar de possuir palavras reservadas, Java, por ser uma linguagem orientada a objetos, permite que se use uma palavra reservada em outro contexto, desde que ela seja qualificada, como no comando package meupacote.while;.

Revisão Teórica – Complexidade de Algoritmos

- Complexidade Assintótica
  - Comportamento a ser observado em uma função f(n), quando n tende ao infinito.

- Limite Superior e Inferior
  - Notação O (Big O Notation) é usada para estabelecer limites superiores de complexidade (**pior caso**).
  - Para o limite inferior (melhor caso), utiliza-se a notação Ω (ômega) e para o caso médio, utiliza-se a notação Θ (teta).

Revisão Teórica – Complexidade de Algoritmos

As complexidades possíveis estão exibidas da menor para a maior:

O (1) : Complexidade constante
O (log n) : Complexidade logarítmica
O (n) : Complexidade linear
O (n log n) : Divisão em sub-problemas
O (n^k) : Polinomial
O (k^n): Exponencial

Revisão Teórica – Complexidade de Algoritmos

Exemplo de medição de complexidade em código-fonte:

```
for (i=0; i<qtdeExecucoes;i++){
   //Comandos
}
```

No caso de instruções simples, elas serão executadas um número constante de vezes, e a complexidade no pior caso será **O(N)**.

```
for (i=0; i<qtdeExecucoes;i++){
   for (j=0; j<qtdeExecucoesInternas;j++){
      //Comandos
   }
}
```

No caso de loops aninhados, O(N) será executado N vezes, logo a complexidade no pior caso será **$O(N^2)$**.

31 – BNDES 2008 – Análise de Sistemas – Desenvolvimento – Q45

Observe o algoritmo em JAVA.

```
public void algoritmo(int[] v) {
   int m;
   int tmp;
   for (int i=0; i<v.length; i++) {
      m = i;
      for (int j=i+1; j<v.length; j++) {
         if (v[j]<v[m]) {
            m=j;
         }
      }
      if (m!=i) {
         tmp = v[m];
         v[m] = v[i];
         v[i] = tmp;
      }
   }
}
```

A complexidade de tempo desse algoritmo, no pior caso, em que *n* corresponde ao número de elementos do vetor *v*, é

(A) Θ(*n*).
(B) O (n log *n*).
(C) O (*n*2).
(D) Θ (*n* log *n*).
(E) Ω (*n*2 log *n*).

31 – BNDES 2008 – Análise de Sistemas – Desenvolvimento – Q45

Observe o algoritmo em JAVA.

```
public void algoritmo(int[] v) {
   int m;
   int tmp;
   for (int i=0; i<v.length; i++) {
      m = i;
      for (int j=i+1; j<v.length; j++) {
         if (v[j]<v[m]) {
            m=j;
         }
      }
      if (m!=i) {
         tmp = v[m];
         v[m] = v[i];
         v[i] = tmp;
      }
   }
}
```

A complexidade de tempo desse algoritmo, no pior caso, em que *n* corresponde ao número de elementos do vetor *v*, é

(A) Θ(*n*).
(B) O (n log *n*).
(C) O (*n*2).
(D) Θ (*n* log *n*).
(E) Ω (*n*2 log *n*).

32 – Petrobras 2012 – Análise de Sistemas – Infraestrutura – Q68

```
public class Main {
   public static void main(String[] args) {
       int x=0;
       for (int x=4; x<5; x++) {
          System.out.print(x);
       }
  }
}
```

Qual o resultado obtido ao se tentar compilar e executar o código acima?

(A) Um erro de compilação, indicando que houve uma tentativa de redefinir a variável x.

(B) Uma advertência de compilação, indicando que x foi redefi nida e a impressão da sequência 01234.

(C) Uma advertência de compilação, indicando que x foi redefi nida e a impressão do número 4.

(D) Uma compilação sem erros ou advertências e a impressão da sequência 01234.

(E) Uma compilação sem erros ou advertências e a impressão do número 4

32 – Petrobras 2012 – Análise de Sistemas – Infraestrutura – Q68

```
public class Main {
   public static void main(String[] args) {
      int x=0;
      for (int x=4; x<5; x++) {
         System.out.print(x);
      }
   }
}
```

Qual o resultado obtido ao se tentar compilar e executar o código acima?

(A) Um erro de compilação, indicando que houve uma tentativa de redefinir a variável x.

(B) Uma advertência de compilação, indicando que x foi redefi nida e a impressão da sequência 01234.

(C) Uma advertência de compilação, indicando que x foi redefi nida e a impressão do número 4.

(D) Uma compilação sem erros ou advertências e a impressão da sequência 01234.

(E) Uma compilação sem erros ou advertências e a impressão do número 4

33 – BNDES 2011 – Análise de Sistemas – Desenvolvimento – Q65

Determinado grupo de pesquisa de uma universidade, no processo de criação de uma linguagem de programação, estabelece que erros de tipo sempre devem ser detectados.

Essa é uma característica conhecida como

a) acoplamento fraco

b) acoplamento forte

c) tipificação fraca

d) tipificação forte

e) tolerância a falhas

33 – BNDES 2011 – Análise de Sistemas – Desenvolvimento – Q65

Determinado grupo de pesquisa de uma universidade, no processo de criação de uma linguagem de programação, estabelece que erros de tipo sempre devem ser detectados.

Essa é uma característica conhecida como

a) acoplamento fraco

b) acoplamento forte

c) tipificação fraca

d) tipificação forte

e) tolerância a falhas

Gabarito

- 1. B
- 2. **RETIRADA DO MATERIAL**
- 3. B
- 4. E
- 5. A
- 6. E
- 7. D
- 8. D
- 9. C
- 10. B
- 11. B
- 12. B
- 13. E
- 14. E
- 15. B
- 16. C
- 17. A
- 18. A
- 19. D
- 20. Discursiva
- 21. D
- 22. D
- 23. D
- 24. A
- 25. C
- 26. D
- 27. C
- 28. D
- 29. C
- 30. D
- 31. C
- 32. A
- 33. D