PROCERGS
CENTRO DE SOLUÇÕES EM GOVERNO ELETRÔNICO

Cia. de Processamento de Dados do Estado do Rio Grande do Sul

CONCURSO PÚBLICO NÍVEL SUPERIOR E MÉDIO

C.01/09
TÉCNICO EM COMPUTAÇÃO COM ÊNFASE NA TECNOLOGIA JAVA

CADERNO	**1**
GABARITO	**1**
APLICAÇÃO	**MANHÃ**

Aplicação: 07/março

LEIA COM ATENÇÃO AS INSTRUÇÕES

1 - A duração da prova é de 4 horas, já incluído o tempo de preenchimento do cartão de respostas.
2 - O candidato que, na primeira hora de prova, se ausentar da sala e a ela não retornar, será eliminado do concurso público.
3 - Os três últimos candidatos a terminar a prova deverão permanecer na sala e somente poderão sair juntos do recinto, após aposição em ata de suas respectivas assinaturas.
4 - Você poderá levar o seu caderno de questões faltando 1 hora para o término da Prova.

INSTRUÇÕES - PROVA OBJETIVA

1 - Confira atentamente se este caderno de perguntas, que contém **60** questões objetivas, está completo.
2 - Confira se seus dados e o ***cargo*** escolhido, indicados no ***cartão de respostas***, estão corretos. Se notar qualquer divergência, notifique imediatamente o Fiscal/Chefe Local. Terminada a conferência, você deve assinar o cartão de respostas no espaço apropriado.
3 - Verifique se o número do Gabarito e do Caderno de Perguntas é o mesmo.
4 - Cuide de seu ***cartão de respostas***. Ele não pode ser rasurado, amassado, dobrado nem manchado.
5 - Para cada questão objetiva são apresentadas cinco alternativas de respostas, apenas uma das quais está correta. Você deve assinalar essa alternativa de modo contínuo e denso.
6 - Se você marcar mais de uma alternativa, sua resposta será considerada errada mesmo que uma das alternativas indicadas seja a correta.

AGENDA

- 16/03/2010, divulgação do gabarito da Prova objetiva: http://concursos.biorio.org.br

- 17 a 19/03/2010, recursos contra o gabarito da Prova Objetiva na Internet: http://concursos.biorio.org.br até as 17h

- Após a transmissão imprimir e entregar no Protocolo da PROCERGS de 9h às 17h

- 07/04/2010, divulgação do resultado da análise dos recursos da Prova Objetiva.

- 08/04/2010, divulgação do Resultado Final da Prova Objetiva

- Informações:
Tel: 21 3525-2480 das 9 às 17h;
Internet:
http://concursos.biorio.org.br
E-mail:
procergs2009@biorio.org.br

Posto de Atendimento:
Av. Ipiranga nº 1.090 (Colégio Estadual Protásio Alves)
9h às 12h e das 13h30min às 17h

BIO RIO Concursos

CONHECIMENTOS ESPECIFÍCOS

01 - São características da linguagem de programação ***Java***, EXCETO:

(A) é multiplataforma;
(B) todo programa deve implementar pelo menos uma classe;
(C) é *case-sensitive* (ou seja, faz distinção entre caracteres maiúsculos e minúsculos);
(D) é impossível definir um método com o mesmo nome de um atributo definido anteriormente;
(E) é possível invocar métodos de uma classe *C* sem que seja necessária a instanciação de objetos de *C*.

02 - Dos trechos de códigos Java apresentados a seguir, quatro exibem a mesma saída ao usuário. Marque a alternativa que **NÃO** apresenta essa saída ao usuário:

(A)
```
int x[] = {1,5,4,3,2};
for (int i=0; i<x.length;i++) {
    System.out.println (x[i]);
}
```

(B)
```
int x[] = {1,5,4,3,2};
for (int i : x) {
    System.out.println (i);
}
```

(C)
```
int x[] = {1,5,4,3,2}, i=-1;
while (i<x.length) {
    System.out.println (x[++i]);
}
```

(D)
```
int x[] = {1,5,4,3,2}, i=0;
do {
    System.out.println (x[i]);
}
```

(E)
```
int x[] = {1,5,4,3,2}, i=0;
for (;i<x.length;System.out.println(x[i++]));
```

03 - Todo programa, independente da linguagem na qual é expresso, precisa ser traduzido para que o *hardware* possa compreendê-lo e, consequentemente, executá-lo. Considerando o processo de tradução de programas Java, avalie as seguintes afirmativas:

I - Um programa fonte Java, para ser executado, passa pelos processos de compilação e interpretação;
II - O programa executável (*.exe*) é obtido ao final do processo de interpretação;
III - O arquivo *.class* é formado por *bytecodes*.

Assinale a alternativa correta:

(A) apenas as afirmativas I e II estão corretas;
(B) apenas as afirmativas I e III estão corretas;
(C) apenas as afirmativas II e III estão corretas;
(D) nenhuma das alternativas está correta;
(E) todas as afirmativas estão corretas.

04 - Sobre a **JVM (*Java Virtual Machine*)**, são feitas as seguintes afirmativas:

I - É capaz de traduzir *bytecodes* para a linguagem da máquina na qual executa;
II - É responsável pela compilação do programa Java, gerando assim o arquivo *.class*;
III - Gera *bytecodes* em conseqüência do processo de interpretação;
IV - Simula uma máquina real, permitindo que um mesmo programa Java seja executado em diferentes plataformas de *hardware* e *software*.

Assinale a alternativa correta:

(A) apenas as afirmativas I e II estão corretas;
(B) apenas as afirmativas I e IV estão corretas;
(C) apenas as afirmativas II e III estão corretas;
(D) apenas as afirmativas III e IV estão corretas;
(E) apenas a afirmativa IV está correta.

05 - A figura a seguir indica o emprego da estrutura *if* em *Java*.

```
if ( n == 1 )
   System.out.println( "O numero e 1" );
else  if ( n == 2 )
      System.out.println( "O numero e 2" );
else
      System.out.println( "O numero não e 1 ou 2" );
```

A sintaxe que emprega a estrutura ***seleção / caso*** correspondente está indicada na seguinte alternativa:

(A)
```
Switch ( n ) {
   case 1:
        System.out.println( "O numero e 1" );
        break;
   case 2:
        System.out.println( "O numero e 2" );
        break;
   default:
        System.out.println( "O numero não e 1 ou 2" );
        break;
}
```

(B)
```
Case ( n ) {
   switch 1:
         System.out.println( "O numero e 1" );
         exit;
   switch 2:
         System.out.println( "O numero e 2" );
         exit;
      else:
         System.out.println( "O numero não e 1 ou 2" );
         exit;
}
```

(C)
```
Switch ( n ) {
     1:  System.out.println( "O numero e 1" );
         break;
     2:  System.out.println( "O numero e 2" );
         break;
   else:  System.out.println( "O numero não e 1 ou 2" );
         break;
}
```

(D)
```
Case ( n ) {
     1 :  System.out.println( "O numero e 1" );
         exit;
     2:  System.out.println( "O numero e 2" );
         exit;
default:  System.out.println( "O numero não e 1 ou 2" );
         exit;
}
```

(E)
```
Switch {
     n = 1:  System.out.println( "O numero e 1" );
         exit;
     n = 2:  System.out.println( "O numero e 2" );
         exit;
      else:  System.out.println( "O numero não e 1 ou 2" );
         exit;
}
```

06 - Atualmente, a linguagem de programação ***Java*** tem sido utilizada em diferentes áreas da computação. Dentre os métodos mostrados, aquele que contém erro em sua construção é:

(A)
```
int g( ) {
    system.out.print( "interior do metodo g" );
     int h( ) {
    system.out.print( "interior do metodo h" );
    }
}
```

(B)
```
public long mtd ( long n ) {
    if ( n < 2 )
       return n ;
    else
       return mtd( n – 1 ) + mtd( n – 2 );
}
```

(C)
```
void f ( float a ) {
      system.out.println( a );
}
```

(D)
```
int soma ( int n ) {
    if ( n == 0 )
       return 0;
    else
       return n + soma( n – 1 );
}
```

(E)
```
int sum ( int x, int y ) {
    int resultado;
     resultado = x + y;
     return resultado;
}
```

07 - A seguir, são apresentadas algumas definições para o conceito de *classe*:

I - Um conjunto de objetos instanciados a partir do mesmo tipo;
II - Um molde que define a estrutura de objetos que compartilham as mesmas características, assim como as operações que podem realizar;
III - Um conjunto de atributos cujos valores são compartilhados por diversos objetos.

Assinale a alternativa correta:

(A) apenas a definição I é correta;
(B) apenas a definição II é correta;
(C) apenas a definição III é correta;
(D) apenas as definições I e II são corretas;
(E) as definições I, II e III são corretas.

08 - Analise o código em ***Java*** a seguir.

```
Public class CLS22 {
        public static void main( String args[] )
        {
           int  x = 11,
                y = 0,
                w = 6;
           System.out.println( x++ );
           System.out.println( x );
           System.out.println( ++x );
           y = x % w;
           System.out.println( ++y );
           System.exit( 0 );
        }
}
```

Após a execução, os valores de saída em sequência, serão respectivamente:

(A) 11, 12, 12 e 1;
(B) 12, 12, 12 e 1;
(C) 11, 12, 13 e 1;
(D) 12, 12, 13 e 2;
(E) 11, 12, 13 e 2.

09 - Considerando a sobrecarga de métodos permitida pela linguagem ***Java***, é correto afirmar que:

(A) o número de parâmetros deve ser diferente em cada versão do método;
(B) se o número e tipo dos parâmetros forem os mesmos em todas as versões, os tipos de retorno devem ser distintos;
(C) o tipo de retorno e os tipos dos parâmetros devem ser diferentes em cada versão, independente do número de parâmetros;
(D) se o número de parâmetros for o mesmo em todas as versões, os tipos destes devem ser diferentes;
(E) inexiste restrição quanto à quantidade e tipos dos parâmetros e ao tipo de retorno.

10 - Analise o programa abaixo em ***Java***.

```
01 public class ClasseX {
02   public static void main (String[] args) {
03     String x[] = {"CASA", "BOLA", "PORTA"};
04     int p = 3;
05     x[--p] = x[1];
06     x[p--] = "ARVORE";
07     for (int i=1;i<x.length;i++) {
08       System.out.println (x[i]);
09     }
10   }
11 }
```

Nesse caso é correto afirmar que:

(A) ocorre um erro de compilação na linha 03;
(B) ocorre uma exceção na linha 05;
(C) ocorre um erro de compilação na linha 06;
(D) a saída do programa inclui o número 7;
(E) a saída do programa inclui a palavra *BOLA*.

11 - No contexto ***Java***, em relação ao conceito de método construtor, **NÃO** é correto afirmar que:

(A) a ausência de sua implementação em uma classe implica em erro de compilação;
(B) é chamado durante a instanciação de objetos da classe na qual está definido;
(C) deve possuir o mesmo nome da classe na qual está implementado;
(D) pode ser sobrecarregado;
(E) não é herdado.

12 - No contexto ***Java***, em relação aos métodos de classe e métodos de instância é correto afirmar que:

(A) a operação realizada por um método de classe não afeta obrigatoriamente os atributos dos objetos desta classe;
(B) métodos de instância não têm acesso aos atributos de classe definidos na classe na qual se encontra;
(C) ambos exigem que a classe na qual estão definidos seja instanciada para que sejam executados;
(D) dentro de uma classe, um método de instância não pode invocar um método de classe;
(E) um método de classe não pode ser sobreescrito.

13 - A linguagem ***Java*** permite que a palavra reservada *final* seja utilizada como modificador de atributos, métodos e até mesmo classes. Sobre seu uso, avalie as seguintes afirmativas:

I - Um atributo *final* comporta-se como uma constante e, por este motivo, deve ter seu valor definido no momento de sua declaração.
II - Um método abstrato não pode ser declarado como *final*.
III - Na prática, todos os métodos de uma classe *final* são também caracterizados como *final*, mesmo que não sejam definidos explicitamente como tal.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e III estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

14 - Analise o código a seguir, referente às Classe1 e Classe2, em ***Java***.

```
public class Classe1 {
    static int a = 5;
    int b = 10;
    String str1 = "Ana";
    String str2 = "Maria";

    public static void main(String[] args){
        Classe1 c1 = new Classe1();
        Classe2 c2 = new Classe2();
        c1.retorna(c2, a, c1.b, c1.str1);
        System.out.print(c2.getC() + ", " + c2.d + ",
" + c2.e + ", ");
        System.out.print(a + ", " + c1.b + ", ");
        System.out.print(c1.str1 + ", " + c1.str2 + ", "
+ c2.str1);
    }

    public void retorna(Classe2 p1, int p2, int p3,
String p4){
        p1.setC(p1.getC()+8); p1.d ++; p1.e = 5;
p1.str1 = "Ana";
        p2 = p2 + 3; p3--; p4 = "Bob"; str2 =
        "Pedro";
    }
}

public class Classe2 {
    private int c = 0;
    int d = 1;
    static int e = 3;
    String str1 = "Helena";
    public int getC(){ return c; }
    public void setC(int c){ this.c = c; }
```

Após a execução do método *main()*, o resultado obtido na saída padrão é:

(A) 8, 1, 3, 8, 9, Bob, Pedro, Ana;
(B) 8, 2, 5, 5, 10, Ana, Pedro, Ana;
(C) 8, 2, 3, 8, 10, Ana, Maria, Helena;
(D) 0, 1, 5, 5, 10, Ana, Pedro, Helena;
(E) 0, 2, 5, 8, 9, Bob, Maria, Helena.

15 - Considere, em ***Java***, uma classe *C1* que herde de uma classe *C2*. Os objetos *a* e *b* (da classe *C1*) e *c* e *d* (da classe *C2*) são instanciados. Nesse caso é correto afirmar que:

(A) a classe *C1* só pode herdar de *C2* se *C2* possuir pelo menos um de seus atributos classificado como *public* ou *protected*;
(B) o objeto *a* tem acesso direto a todos os atributos e métodos definidos em *C*1 e em *C2*;
(C) a atribuição do objeto *a* ao objeto *c* (*c = a;*) é válida;
(D) os métodos *public* definidos e implementados em *C2* são acessíveis por qualquer objeto da classe *C1*; por este motivo, não é possível sobrescrevê-los em *C1*;
(E) *a* e *b* possuem o mesmo conjunto de atributos, armazenando os mesmos valores.

16 - O seguinte trecho de código em ***Java*** foi implementado considerando que:

I - As classes *E1* e *E2* herdam da classe *E3* que, por sua vez, herda de *Exception.*
II - A classe *C* possui um método *public m()*, que pode gerar exceções do tipo *E1*, *E2* ou *E3*.

```
01 try {
02    C obj = new C();
03    C.m();
04    System.out.print ("ABC");
05 }
06 catch (E1 e) {
07    System.out.print ("DEF");
08 }
09 catch (E2 e) {
10    System.out.print ("GHI");
11 }
12 catch (E3 e) {
13    System.out.print ("JKL");
14 }
15 finally {
16    System.out.print ("MNO");
17 }
18 System.out.print ("PQR");
```

Assinale a alternativa que fornece o que será exibido ao usuário caso ocorra uma exceção *E1* na linha 03 do código:

(A) *DEFPQR;*
(B) *ABCDEFPQR;*
(C) *DEFMNOPQR;*
(D) *DEFJKLMNOPQR;*
(E) *ABCDEFMNOPQR.*.

17- Em relação aos modificadores de acesso a membros de uma classe, avalie as seguintes afirmativas:

I - Métodos *public* só podem acessar atributos que também sejam *public*.
II - Definir atributos como *private* é recomendado pela Orientação a Objetos, uma vez que restringe o acesso apenas à própria classe e suas subclasses.
III - Os métodos *setters* e *getters* devem ser declarados como *private* para que o encapsulamento seja mantido.

Assinale a alternativa correta:

(A) apenas as afirmativas I e II estão corretas;
(B) apenas as afirmativas I e III estão corretas;
(C) apenas as afirmativas II e III estão corretas;
(D) nenhuma das alternativas está correta;
(E) todas as afirmativas estão corretas.

18 - A linguagem ***Java*** permite que classes, interfaces e afins sejam reunidos em módulos denominados de pacotes (*packages*). Sobre esse recurso, NÃO é correto afirmar que:

(A) o comando *import* possibilita que a classe que está sendo definida tenha acesso a recursos definidos em outros pacotes;
(B) os atributos definidos em determinada classe *C* são acessíveis diretamente por todas as demais classes que se encontram no pacote onde *C* foi declarada;
(C) por padrão, o pacote *java.lang* é importado; ou seja, mesmo que a aplicação não especifique este pacote na cláusula *import*, ela terá acesso aos seus componentes públicos;
(D) ao declarar uma nova classe, pode-se adicioná-la a determinado pacote inserindo o comando *package* ***nome****;* como o primeiro comando do arquivo fonte, onde ***nome*** especifica o pacote desejado;
(E) em Java, o modificador de acesso padrão de um membro de uma classe é o *package*, indicando que é acessível diretamente por qualquer classe (ou interface ou afim) dentro do pacote no qual foi definido.

19 - Considere o programa a seguir, representado pelas seguintes classes ***Java***.

```
01 public class C1 {
02   private int x = 10;
03   public int m () {
04     x++;
05     return x;
06   }
07 }
08 public class C2 extends C1 {
09   private int x = 20;
10   public int m () {
11     return ++x;
12   }
13 }
14 public class Teste {
15   public static void main (String[] args) {
16     C1 obj1 = new C1();
17     C2 obj2 = new C2();
18     int S = obj1.m();
19     obj1 = obj2;
20     S += obj1.m();
21     System.out.println (S);
22   }
23 }
```

O resultado da tentativa de execução dessa aplicação é:

(A) ocorre um erro de compilação em *C2.java*, por redefinição do atributo *x*;
(B) ocorre um erro de compilação em *Teste.java* na linha 19 (atribuição inválida);
(C) o valor 12 é exibido;
(D) o valor 22 é exibido;
(E) o valor 32 é exibido.

20 - Em *Java*, classes e métodos podem ser declarados como abstratos, através do uso do modificador *abstract*. Acerca desse tópico, são feitas as seguintes afirmativas:

I - Uma classe abstrata não pode ser utilizada para a instanciação de objetos;
II - Um método abstrato não pode ser declarado como *final*;
III - Em uma classe abstrata, todos os métodos são abstratos, mesmo que não sejam declarados explicitamente com o modificador *abstract*.

Assinale a alternativa correta:

(A) apenas as afirmativas I e II estão corretas;
(B) apenas as afirmativas I e III estão corretas;
(C) apenas as afirmativas II e III estão corretas;
(D) nenhuma das afirmativas está correta;
(E) todas as afirmativas estão corretas.

21 - A linguagem ***Java*** permite o desenvolvimento de aplicações concorrentes a partir da implementação de *threads*. Marque a única alternativa que **NÃO** representa uma característica das *threads* Java:

(A) a interface *Runnable* fornece um único método, *run*, que deve conter o código a ser executado pela *thread*;
(B) a interface *Lock* provê métodos que permitem que duas ou mais *threads* executem de maneira sincronizada;
(C) uma forma de criar uma aplicação concorrente consiste em definir uma classe que herde de *Thread* e, em seguida, sobreescrever o método *run*;
(D) uma das implementações do método *sleep* da classe *Thread* especifica um tempo, em milissegundos, durante o qual a *thread* terá sua execução suspensa;
(E) o método *newFixedThreadPool* da classe *Executors* permite a execução de um número fixo de *threads* e se esse número for inferior ao número de *threads* disparadas pela aplicação, uma exceção é gerada.

22 - *Applets Java* consistem em programas que podem ser executados por um *browser* (navegador). Assinale a alternativa que **NÃO** representa uma característica de tais programas:

(A) o método *applet* estende a classe *Applet*;
(B) o método *stop* conclui a execução da *applet*;
(C) o método *destroy* faz com que os recursos alocados à *applet* sejam liberados;
(D) o método *init* é chamado sempre antes da primeira chamada ao método *start*;
(E) o método *start* é chamado sempre que a página *web* que contém a *applet* é visitada.

23 - A API JDBC (*Java Database Connectivity*) fornece diversas classes e interfaces que possibilitam a aplicações Java se conectarem com bases de dados relacionais. A seguir, são apresentadas descrições de uso dessa API:

I - O método *getConnection*, da classe *DriverManager*, permite a conexão da aplicação com a base de dados;
II - A conexão com o banco deve ser encerrada quando não mais for necessária. Para isto, recomenda-se que exceções sejam tratadas e que a conexão seja desfeita no bloco *finally*;
III - O comando *executeQuery*, da interface *Statement*, retorna um objeto *ResultSet* contendo os registros da base de dados que atendem ao comando especificado.

Assinale a alternativa correta:

(A) apenas as afirmativas I e II estão corretas;
(B) apenas as afirmativas I e III estão corretas;
(C) apenas as afirmativas II e III estão corretas;
(D) nenhuma das afirmativas está correta;
(E) todas as afirmativas estão corretas.

24 - Os recursos fundamentais de redes em Java são declarados pelas classes e interfaces do pacote *java.net*. Sobre o desenvolvimento de aplicações dessa natureza **NÃO** é correto afirmar que:

(A) a classe *ServerSocket* implementa um *socket* que servidores podem utilizar para ouvir e aceitar conexões solicitadas por clientes;
(B) um objeto da classe *Socket* pode ser criado pela aplicação cliente para que a conexão com o servidor seja estabelecida;
(C) assim que a execução do servidor é encerrada, todos os clientes conectados têm suas execuções concluídas, automaticamente;
(D) o *socket*, após criado, comporta-se como um arquivo de entrada e saída de dados; isto é, os dados a serem trocados pelo cliente e pelo servidor são disponibilizados através de operações de escrita e leitura no *socket*;
(E) a classe *DatagramSocket* representa um *socket* para envio e recebimento de datagramas, com uso do protocolo UDP.

25 - A linguagem ***Java*** possui dois tipos de programas que são denominados nas bibliografias de aplicativo e *applet*. Nesse sentido, analise as afirmações que seguem sobre o conceito básico de *applet*.

I - É implementada e interpretada exatamente igual a uma aplicação.
II - É interpretada pelo *browser* e deve se encontrar no código HTML.
III - Conforme uma aplicação, é necessário que a *applet* seja compilada e interpretada.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

26 - Analise o código ***Java*** que segue em que são aplicados os operadores de pré-decremento e pós-decremento:

```
public class Decrementa {
    public static void main ( string args {} )
    {
            int x, y = 100;
            x = --y;
            x = y--;
            system.out.println ( x );
            system.out.println ( y );
    }
}
```

Ao exibir os conteúdos das variáveis x e y declaradas com int tem-se os seguintes valores, respectivamente para estas varáveis x e y:

(A) 99 e 98;
(B) 98 e 99;
(C) 99 e 99;
(D) 98 e 98;
(E) 97 e 97.

27 - A linguagem ***Java*** possui várias estruturas de controles de repetição de códigos. Dentre estas estruturas existe a que é denominada de *do*/*while* que é caracterizada pela palavra reservada *do*, as instruções que formam o algoritmo desejado, entre chaves para caracterizar um bloco, e a palavra reservada *while*, seguida de parênteses e a condição desejada (com ponto-e-vírgula após fechar os parênteses). De acordo com estes conceitos, analise as seguintes afirmativas:

I - A condição é testada no início da execução.
II - Se o resultado do teste é VERDADEIRO, permanece a execução das instruções contidas no bloco de repetição.
III - Se o resultado do teste é FALSO, a execução é finalizada com saída da estrutura de repetição.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

28 - É possível desenvolver aplicações em ***Java*** através de vários ambientes de desenvolvimento integrado (IDE's). Dentre as opções mais utilizadas pode-se destacar:

(A) SmallTalk, BlueJ e WebSphere;
(B) WebSphere, Eclipse e Phyton;
(C) BlueJ, SmallTalk e NetBeans;
(D) Eclipse, NetBeans e Jbuilder;
(E) Jbuilder, Phyton e BlueJ.

29 - Em ***Java***, as instruções ***break*** e ***continue*** alteram o fluxo de execução das instruções em um algoritmo. A instrução ***break***, quando executada em uma estrutura ***while***, ***for***, ***do/while*** ou ***switch,*** e a estrutura ***continue***, quando executada em uma estrutura ***while***, ***for*** ou ***do/while***, têm por finalidade, respectivamente:

(A) executar a instrução que verifica a condição de verdadeiro ou falso da estrutura de repetição; e pular as instruções que se encontram no bloco da instrução (continue), prosseguindo com a verificação da condição se é falso ou verdadeiro;
(B) pular as instruções que se encontram no bloco da instrução (break), prosseguindo com a verificação da condição da estrutura de repetição se é falso ou verdadeiro; e executar as instruções após a estrutura em que se encontra a instrução (continue);
(C) executar as instruções após a estrutura em que se encontra a instrução (break); e pular as instruções que se encontram no bloco da instrução (continue), prosseguindo com a verificação da condição da estrutura de repetição se é falso ou verdadeiro;
(D) pular as instruções que se encontram no bloco da instrução (break), prosseguindo com a verificação da condição se é falso ou verdadeiro; e executar a instrução que verifica a condição de verdadeiro ou falso da estrutura de repetição;
(E) executar a instrução que verifica a condição de verdadeiro ou falso da estrutura de loop; e pular as instruções que se encontram no bloco da instrução (continue) para verificar a condição de verdadeiro ou falso da estrutura de repetição.

30 - A compilação de um programa fonte na linguagem ***Java*** tem como resultado um arquivo denominado de *bytecode*. Analise as afirmativas que seguem sobre as características desse arquivo.

I - É um arquivo dependente da plataforma onde ocorreu a compilação.
II - É um arquivo independente da plataforma onde ocorreu a compilação.
III - É um arquivo que deve ser interpretado na plataforma onde se deseja executá-lo.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

31 - A linguagem ***Java*** possui uma palavra reservada denominada de *final* que tem por objetivo declarar uma variável de instância cujo conteúdo não pode ser alterado. Caso um programa em Java tenha em seu código uma instrução que altera uma variável (de instância) declarada com esta palavra reservada então é correto afirmar que:

(A) um erro ocorre em tempo de execução;
(B) um erro ocorre em tempo de compilação;
(C) somente um erro é armazenado no arquivo *bytecode*;
(D) é apresentado um erro se for executado em um *browse;*
(E) a variável é inicializada para seu valor default (zerada).

32 - No que tange ao *Desenvolvimento WEB JAVA*, analise as afirmativas a seguir, a respeito da tecnologia ***JSP ("JavaServer Page")***:

I - Utiliza páginas JSP, com extensão *.jsp* ou *.jspx*, criadas pelo desenvolvedor da web e que incluem especificações *JSP* e tags customizadas, em combinação com outras tags estáticas, *HTML* ou *XML*.
II - Disponibiliza uma tecnologia simples e rápida para criar páginas que exibem conteúdo gerado dinamicamente, define a interação entre o servidor e a página JSP, e descreve o formato e sintaxe da página.
III - Emprega *servlets* - programas escritos na linguagem Java e executados no cliente, em oposição aos *applets*, executados no *browser* do servidor.

Assinale a alternativa correta:

(A) somente a afirmativa I está correta;
(B) somente a afirmativa II está correta;
(C) somente as afirmativas I e II estão corretas;
(D) somente as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

33 - A palavra reservada *final* da questão anterior é inicializada no programa ***Java*** no seguinte contexto:

(A) somente no construtor da classe;
(B) somente no momento da sua declaração;
(C) no construtor da classe ou na declaração da variável (de instância);
(D) no momento da sua declaração ou ao ter seu conteúdo alterado;
(E) no construtor da classe ou ao ter seu conteúdo alterado.

34 - A linguagem ***Java*** permite que o nome de uma variável de instância tenha o mesmo nome de uma variável local de um método. Ao processar esta variável no método, a variável de instância é ocultada. Analise as afirmativas que seguem no intuito da variável instância ser processada no método.

I - A variável de instância jamais poderá ser acessada em um método.
II - A variável de instância é acessada em um método com a palavra reservada *this*.
III - A variável local é acessada em um método com a palavra reservada *this*.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

35 - Analise o código das três classes (Alfa, Beta e Gama) ***java*** a seguir:

```
class Alfa {
 protected Beta cl2;
}
class Gama extends Alfa {   }
class Beta {   }
```

Nesse caso é correto afirmar que:

(A) Alfa é uma Gama e tem uma Beta;
(B) Gama é uma Beta e tem uma Alfa;
(C) Beta tem uma Alfa e Alfa é uma Gama;
(D) Beta tem uma Gama e Gama é uma Alfa;
(E) Gama tem uma Beta e Gama é uma Alfa.

36 - A respeito das tecnologias Java/J2EE, analise as afirmativa a seguir.

I - EJB é uma arquitetura de componentes multi-plataforma para o desenvolvimento de aplicações Java Enterprise Edition (Java EE), multicamadas, distribuídas, escaláveis e orientadas a objetos.
II - JTA é uma API da linguagem Java que permite a componentes baseados em Java/J2EE criar, enviar, receber e ler mensagens.
III - JSF é uma especificação feita pela Sun que visa padronizar o uso de transações distribuídas feitas por aplicativos Java.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

37 - Observe as instruções a seguir, que indicam a utilização de ***Applet Java*** inserido em código ***HTML***.

```
<html>
<head>
<head>
<body>
<table>
<tr><td><applet code="Marquee" align="baseline"
            width="400"  height="40">
   <param name="back_color" value="200 100 150">
   <param name="delay" value="7">
   <param name="font_bold" value="yes">
   <param name="font_italic" value="yes">
   <param name="font_size" value="16">
   <param name="marquee" value="* PROCERGS *">
   <param name="shift" value="3">
   <param name="text_color" value="255 0 255">
</applet></td></tr>
</table>
</body>
</html>
```

Os parâmetros indicados interagem com o arquivo “Marquee”, que possui a seguinte extensão no nome:

(A) css;
(B) exe;
(C) applet;
(D) java;
(E) Class.

38 - Cada objeto de uma classe tem sua própria cópia de todas as variáveis de instância da classe. No entanto, há na linguagem ***Java*** uma palavra reservada denomina *static* e a variável declarada com esta palavra reservada é denominada variável de classe. Analise as afirmativas que seguem que especificam algumas das aplicações desta palavra reservada.

I - Quando é necessário uma cópia de uma única variável compartilhada para todos os objetos da classe.
II - Quando é necessário uma variável que tenha o mesmo conceito e acesso de uma variável global.
III - Quando é necessário uma variável de instância independente de uma classe.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

39 - De modo semelhante a uma variável de classe, há na linguagem ***Java*** o método de classe que contém na sua sintaxe a palavra reservada *static*. Esse método é acessado:

(A) exclusivamente na classe que declarou o método;
(B) por uma variável de classe, declarada com *static*;
(C) pelo próprio nome da classe seguido de um ponto e o nome do método;
(D) após a criação de um objeto externo da classe em que o método pertence;
(E) através de um outro método de escopo *public* pertencente a classe onde o método foi declarado.

40 - Em relação ao construtor de uma classe, **NÃO** é correto afirmar que:

(A) é um método com o nome exatamente o mesmo da classe;
(B) a classe pode conter construtores sobrecarregados para inicializar os objetos da classe de modos diferentes;
(C) se nenhum construtor é definido para uma classe, o compilador cria um construtor *default* que não recebe nenhum argumento ou parâmetro;
(D) todos os construtores de subclasses (classe especializada) devem chamar um dos construtores da superclasse (classe generalizada) direta explícita ou implicitamente;
(E) a subclasse (classe especializada) ao chamar o construtor de uma superclasse (classe generalizada) explicitamente é sempre do construtor *default* (da superclasse).

41 - Observe a estrutura a seguir, em ***Java***.

```
While <condição>
      {
            // Bloco de instruções no loop
       }
```

Com a finalidade de forçar o término da execução e a saída incondicional do **while**, utiliza-se no bloco de instruções no "loop" a seguinte instrução:

(A) *end;*
(B) *halt;*
(C) *exit;*
(D) *quit;*
(E) *break.*

42 - Em ***Java***, no que tange à herança de uma subclasse (classe especializada) de uma superclasse (classe generalizada), é herdado (pela classe especializada), é correto afirmar que:

(A) todos os membros não-*private*, exceto os construtores;
(B) todos os membros *private*, exceto os construtores;
(C) todos os membros não-*private*, e os construtores;
(D) todos os membros *private*, e os construtores;
(E) todos os membros e os construtores.

43 - Em relação à palavra reservada da linguagem ***Java*** *super*, **NÃO** é correto afirmar que:

(A) a chamada em uma subclasse (classe especializada) do construtor da sua super-classe (classe generalizada) com a palavra reservada *super* deve ser a primeira instrução no construtor da subclasse;
(B) a chamada em uma subclasse (classe especializada) do construtor da sua super-classe (classe generalizada) com a palavra reservada *super* deve corresponder aos parâmetros especificados em um dos construtores da superclasse;
(C) os construtores de uma superclasse (classe generalizada) são herdados por suas subclasses (classe especializada) e, portanto, não necessariamente devem ser chamados explicitamente com a palavra reservada *super* quando um objeto da subclasse é instanciado e o construtor da superclasse possui parâmetros;
(D) a chamada explícita em uma subclasse (classe especializada) do construtor da sua super-classe (classe generalizada) necessariamente deve ser com a palavra reservada *super*, fato que não ocorre com a chamada implícita;
(E) quando um objeto de uma subclasse (classe especializada) é instanciado, o construtor da superclasse (classe generalizada) deve ser chamado, explicitamente com a palavra reservada *super* ou implicitamente, para inicializar as variáveis de instância da superclasse do objeto de subclasse criado.

44 - A classe ***JApplet*** possui vários métodos que são chamados para a execução de um *applet*. De acordo com esse conceito, analise as seguintes afirmativas:

I - O método *start* é chamado após o método *init* ter completado sua execução.
II - O método *start* é chamado antes de o método *init* ter completado sua execução.
III - O método *start* é chamado durante a execução do método *init* para que este último possa melhor iniciar os *threads*.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

45 – Na linguagem ***Java***, em relação ao tratamento de exceção **NÃO** é correto afirmar que:

(A) um bloco *try* possui 0 (zero) ou mais blocos *catch*;
(B) o bloco *finally* é sempre executado independente se ocorreu uma exceção;
(C) quando ocorre uma exceção, o controle do programa abandona o bloco *try* e continua a execução no primeiro bloco *catch*;
(D) o bloco *finally* não é executado caso o bloco *try* seja encerrado com uma instrução *return*, *break* ou *continue*;
(E) o bloco *finally* é indicado para liberar recursos que foram alocados no seu bloco *try* correspondente.

46 - O ***J2EE (Java 2 Enterprise Edition)*** é considerada uma plataforma ***Java*** padrão, direcionada para aplicações multi-camadas, baseadas em componentes que são executados em um servidor de aplicações, voltada para redes, internet e intranets e que contém bibliotecas especialmente desenvolvidas para o acesso a servidores, a sistemas de e-mail e a banco de dados. A plataforma J2EE contém componentes nativos, cada um com funcionalidades distintas, sendo dois definidos a seguir:

I - permite o desenvolvimento de páginas com conteúdo dinâmico que executam no servidor;
II - possibilita o desenvolvimento de aplicações empresariais associadas diretamente ao negócio do cliente e que executam no servidor.

Esses componentes são denominados, respectivamente:

(A) PHP (Hipertext Preprocessor) e EJB (Enterprise Java Beans);
(B) ASP (Active Server Pages) e EJB (Enterprise Java Beans);
(C) ASP (Active Server Pages) e JTA(Java Transaction API);
(D) JSP (Java Server Pages) e EJB (Enterprise Java Beans);
(E) JSP (Java Server Pages) e JTA(Java Transaction API).

47 - A linguagem ***Java*** permite a implementação de sobrecarga de métodos. Analise as afirmativas que seguem sobre este conceito.

I - os métodos sobrecarregados possuem o mesmo nome.
II - os métodos sobrecarregados diferem entre si pelo número de parâmetros, ou pelos tipos de dados dos parâmetros, ou pela ordem destes parâmetros.
III - os métodos sobrecarregados têm a vantagem de criar vários métodos que implementam tarefas semelhantes, com tipos de dados, geralmente, diferentes, tornando os programas mais legíveis e compreensíveis.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

48 - Sobre os membros *private*, *public* e *protected* de uma classe ***Java*** é correto afirmar que:

(A) os membros *private* de uma superclasse são acessados de qualquer lugar do programa que tenha uma referência para o tipo desta superclasse. Os membros *publice* de uma superclasse são acessados apenas em métodos dessa superclasse. Os membros *protected* de uma superclasse podem ser acessados por métodos da superclasse, por métodos da subclasse e por métodos de outras classes pertencentes ao mesmo pacote;
(B) os membros *public* de uma superclasse são acessados de qualquer lugar do programa que tenha uma referência para o tipo desta superclasse. Os membros *protected* de uma superclasse são acessados apenas em métodos dessa superclasse. Os membros *private* de uma superclasse podem ser acessados por métodos da superclasse, por métodos da subclasse e por métodos de outras classes pertencentes ao mesmo pacote;
(C) os membros *public* de uma superclasse são acessados de qualquer lugar do programa que tenha uma referência para o tipo desta superclasse. Os membros *private* de uma superclasse são acessados apenas em métodos dessa superclasse. Os membros *protected* de uma superclasse podem ser acessados por métodos da superclasse, por métodos da subclasse e por métodos de outras classes pertencentes ao mesmo pacote;
(D) os membros *protected* de uma superclasse são acessados de qualquer lugar do programa que tenha uma referência para o tipo desta superclasse. Os membros *private* de uma superclasse são acessados apenas em métodos dessa superclasse. Os membros *public* de uma superclasse podem ser acessados por métodos da superclasse, por métodos da subclasse e por métodos de outras classes pertencentes ao mesmo pacote;
(E) os membros *protected* de uma superclasse são acessados de qualquer lugar do programa que tenha uma referência para o tipo desta superclasse. Os membros *public* de uma superclasse são acessados apenas em métodos dessa superclasse. Os membros *private* de uma superclasse podem ser acessados por métodos da superclasse, por métodos da subclasse e por métodos de outras classes pertencentes ao mesmo pacote.

49 - A linguagem ***Java*** permite a implementação do conceito de vinculação dinâmica de métodos. Analise as afirmativas que seguem sobre o significado correto deste conceito.

I - Várias subclasses (classes especializadas) contendo o mesmo método sobrecarregado da sua superclasse (classe generalizada) é executado dinamicamente, ou seja, em tempo de execução seleciona o método correto de acordo com a subclasse do objeto instanciado.
II - Uma única subclasse (classe especializada) que herda de várias superclasses (classes generalizadas), sendo assim, a subclasse possui membros de todas as superclasses.
III - Várias subclasses (classes especializadas) contendo variáveis de instância herdadas da sua superclasse (classe generalizada) que são executadas dinamicamente.

Assinale a alternativa correta:

(A) apenas a afirmativa I está correta;
(B) apenas a afirmativa II está correta;
(C) apenas as afirmativas I e II estão corretas;
(D) apenas as afirmativas II e III estão corretas;
(E) todas as afirmativas estão corretas.

50 - Em ***Java***, sobre o conceito de classe abstrata **NÃO** é correto afirmar que:

(A) os objetos desta classe não podem ser instanciados;
(B) estas classes são importantes para a implementação de herança múltipla;
(C) estas classes são utilizadas em situações de herança, onde elas passam a ser a superclasse;
(D) estas classes podem e devem ser referenciadas quando forem aplicadas no código para processamento das suas subclasses;
(E) estas classes têm a finalidade de fornecer uma superclasse apropriada da qual outras classes podem herdar interface e/ou implementação.

51 - Suponha que um modelo orientado a objetos em ***Java*** possui uma superclasse denominada de veículo com duas subclasses: carro e bicicleta. A opção que corresponde à forma INCORRETA de atribuição em Java é:

(A) Bicicleta b1 = new Bicicleta();
(B) Car c1 = new Car();
(C) Car c1 = new Veiculo();
(D) Veiculo v1 = Veiculo();
(E) Veiculo v1 = new Car().

52 - Um analista deseja utilizar dois operadores em ***JavaScript***, o primeiro que retorne o resto da divisão de um número por outro e o segundo que corresponda ao operador E, que permita a comparação entre duas expressões, retornando um valor lógico verdadeiro, desde essas sejam verdadeiras. Os símbolos utilizados para esses operadores são, respectivamente:

(A) % e AND
(B) %% e &
(C) % e &&
(D) MOD e AND
(E) MOD e &&

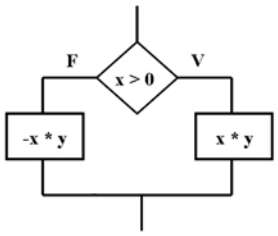
53 - Um princípio da ***Orientação a Objetos*** propõe ocultar determinados elementos de uma classe das demais classes. Ao colocar uma proteção ao redor dos atributos e criar métodos para prover o acesso a estes métodos, o objetivo é prevenir contra os efeitos colaterais indesejados que podem afetá-los ao ter essas propriedades modificadas de forma inesperada. Nessas condições, analise o exemplo a seguir.

```
1.  import java.util.*;
2.  import javax.swing.*;
3.  class Student {
4.      public static void main(String[] args) {
5.          Function obj = new Function();
6.          obj.setNome("Teste");
7.          JOptionPane.showMessageDialog(null, obj.getNome());
8.          System.gc();
9.      }
10. }
11.
12. class Function {
13.     private String nome="";
14.
15.     //Método Set (Muda os dados)
16.     public void setNome(String nome) {this.nome = nome;}
17.     //Método get (Retorna os dados)
18.     public String getNome() {return nome;}
19. }
```

Esse princípio é denominado:

(A) herança;
(B) acoplamento;
(C) polimorfismo;
(D) modularidade;
(E) encapsulamento.

54 - Observe o fluxograma e o código abaixo em *JavaScript*.




```
<script>
document.write("<h2>Operador
condicional</h2>");
x = 2;
y = 3;
INSTRUÇÃO
document.write(z);
</script>
```

A codificação do fluxograma que utiliza operador condicional e que substitui a palavra INSTRUÇÃO no código é:

(A) z = (x>0 : x*y % -x*y);
(B) z = (x>0 ? x*y $ -x*y);
(C) z = (x>0 : x*y $ -x*y);
(D) z = (x>0 ? x*y : -x*y);
(E) z = (x>0 : x*y : -x*y);

55 - Observe o código abaixo, em ***JavaScript***.

```
var count = 0;
while (count < 10) {
        document.write(count + "<br">);
        count++;
}
```

A sintaxe equivalente que emprega a estrutura de controle ***"repetir ... até que ..."*** é:

(A)
```
var count = 0;
repeat {
        document.write(count + "<br">);
        count++;
}
until (count = 10);
```

(B)
```
var count = 0;
do {
        document.write(count + "<br">);
        count++;
}
until (count = 10);
```

(C)
```
var count = 0;
repeat {
        document.write(count + "<br">);
        count++;
}
while (count = 10);
```

(D)
```
var count = 0;
do {
        document.write(count + "<br">);
        count++;
}
while (count = 10);
```

(E)
```
var count = 0;
do while {
        document.write(count + "<br">);
        count++;
}
until (count = 10);
```

56 - Analise o código abaixo, em ***JavaScript***.

```
<script>
document.write("<h2>Números</h2>");
for (i=0, j=1, k=0, w=0; i<5; i++, w=j+k, j=k, k=w){
   document.write("Saída (" + i + ") = " + w);
   document.write("<br>");
}
</script>
```

Após a execução em um browser, serão mostrados na tela do monitor, os seguintes valores:

(A) 1, 1, 2, 3 e 4;
(B) 0, 1, 1, 2 e 3;
(C) 0, 1, 2, 3 e 4;
(D) 0, 1, 3, 4 e 5;
(E) 1, 2, 3, 4 e 5.

57 - Analise as afirmativas a seguir, referentes às características das estruturas de controle utilizadas em programação.

I - Na estrutura ***repetir...até_que...*** a saída ocorre quando a condição é falsa, enquanto que na ***enquanto...faça...*** a saída ocorre quando a condição é verdadeira.
II - Na estrutura ***repetir...até_que...*** o teste da condição ocorre no final, enquanto que na ***enquanto...faça...*** o teste da condição ocorre no início.
III - Na estrutura ***repetir...até_que...*** o bloco de instruções no *loop* é executado pelo menos uma vez, enquanto que na ***enquanto...faça...*** o bloco de instruções no *loop* só é executado quando a condição testada é verdadeira.

Assinale a alternativa correta:

(A) somente a afirmativa III estiver correta;
(B) somente as afirmativas I e II estiverem corretas;
(C) somente as afirmativas I e III estiverem corretas;
(D) somente as afirmativas II e III estiverem corretas;
(E) todas as afirmativas estiverem corretas.

58 - Nos sistemas *"time sharing"*, na condição em que se tem mais usuários que memória suficiente para armazenar todos os processos, foi necessária a criação de um esquema para gerenciamento de memória, de forma a manter os excedentes em disco rígido. Para rodar, os processos são trazidos à memória. O movimento de processos entre a memória e o disco e vice-versa é denominado:

(A) multiprogramação com "swapping";
(B) monoprogramação com "hot- swap";
(C) multiprocessamento com "plug-and-play";
(D) multiprogramação com sincronização "pipeline";
(E) multiprocessamento com escalonamento "round robin".

59 - O ***File Transfer Protocol – FTP*** é o protocolo padrão da arquitetura *TCP/IP*, utilizado para copiar arquivos de um *host* para outro. O FTP estabelece duas conexões TCP, por meio de duas portas conhecidas, uma usada para a conexão de dados e a outra para conexão de controle. Essas portas são identificadas, respectivamente, pelos números:

(A) 21 e 20;
(B) 20 e 21;
(C) 53 e 80;
(D) 66 e 67;
(E) 67 e 66.

60 - Uma variação importante do comando ***SELECT*** é dada por uma cláusula ***SQL***, que remove as entradas duplicadas do conjunto de resultados. Esta cláusula é mostrada no seguinte comando:

(A) select only CARGO from PROCERGS;
(B) select unique CARGO from PROCERGS;
(C) select distinct CARGO from PROCERGS;
(D) select distinct CARGO to PROCERGS;
(E) select unique CARGO to PROCERGS.

BIO-RIO *Concursos*

Av. Carlos Chagas Filho, 791 - Cidade Universitária - Ilha do Fundão – RJ

Central de Atendimento: (21) 3525-2480

Internet: http://concursos.biorio.org.br

E-mail: concursos@biorio.org.br

procergs2009@biorio.org.br