Universidade da Amazônia

Curso: Análise e Des. de Sistemas / Ciência da Computação

Turma: ALC XXX 01 03 NXA

Disciplina: Programação Orientada a Objetos

Professor: Alan Souza

LISTA DE EXERCÍCIO 1

1) Leia os itens abaixo que falam sobre características de linguagens de programação:

I. Quando o código-fonte de um projeto é compilado, gera-se o bytecode que, por sua vez, é interpretado por uma máquina virtual.

II. O código-fonte é criado pelo programador e, quando compilado, cria-se um arquivo executável (.exe) que é processado diretamente pelo processador.

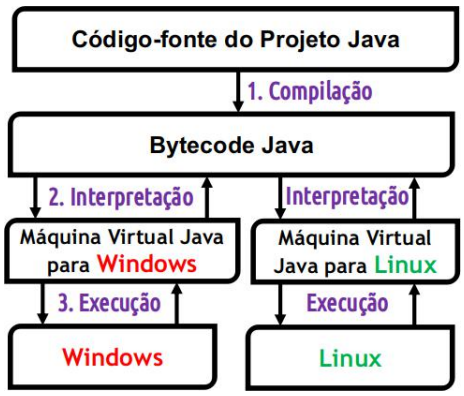
III. As variáveis precisam ser desalocadas da memória e quem programa esse processo é o programador.

IV. As variáveis são automaticamente desalocadas da memória principal pelo "coletor de lixo".

Qual dos itens acima referem-se a linguagem de programação Java?

a) I, II, III e IV.
b) I e IV apenas.
c) II e III apenas.
d) I apenas.
e) II e IV apenas.

2) O esquema mostrado na figura abaixo representa o funcionamento de um programa desenvolvido em Java:



Os passos 1. Compilação, 2. Interpretação, 3. Execução são processados, respectivamente, pelos seguintes componentes:

a) JDK, JVM, JRE.
b) JVM, JDK, JRE.
c) JRE, JVM, JDK.
d) JVM, JDK, JDK.
e) JDK, JRE, JVM.

3)

Os programadores da linguagem Java concentram-se em criar seus próprios tipos definidos pelo usuário. Esses tipos são conhecidos como:

a) métodos.
b) objetos.
c) bytecodes.
d) classes.
e) variáveis.

4)
Considere o seguinte trecho de código na linguagem de programação Java.

```
1   public class Pessoa {
2       private String nome;
3       private int idade;
4       public void setNome (String nome) {
5           this.nome = nome;
6       }
7       public void getNome ( ) {
8           return this.nome;
9       }
10      public void setIdade (int idade) {
11          this.idade = idade;
12      }
13      public void getIdade ( ) {
14          return this.idade;
15      }
16  }
```

A utilização dos métodos getters e setters, à maneira dos declarados nas linhas 4, 7, 10 e 13 do código apresentado, é uma estratégia para aplicar um importante conceito de orientação a objetos chamado

a) generalização.
b) herança.
c) encapsulamento.
d) sobrecarga.
e) polimorfismo.

5)

Na linguagem de programação Java, quando o método de uma classe não possui um modificador de acesso explicitamente declarado significa que esse método pode ser acessado

a) por todas as classes do mesmo pacote em que foi declarado.
b) por qualquer outra classe, além daquela a que pertence.
c) pela classe a que pertence, de forma exclusiva.
d) pela classe em que foi declarado e suas subclasses, e por membros de outras classes no mesmo pacote.
e) pela classe filha, caso haja relacionamento de herança.

6)

Considere a classe Java abaixo.

```
public class Processo {
        private String numeroProcesso;
}
```

Um Técnico especializado em TI afirma, corretamente, que:

a) para incluir um valor no atributo numeroProcesso através de um objeto dessa classe será necessário criar um método privado getNumeroProcesso.
b) poderão ser incluídos nessa classe um construtor que não recebe parâmetros e um construtor que recebe como parâmetro o número do processo.
c) a instrução Processo p = new Processo("10453"); instancia um objeto dessa classe utilizando o construtor padrão vazio.
d) para permitir encapsulamento, os novos atributos e métodos a serem incluídos nessa classe terão que ser privados.
e) não será possível instanciar um objeto dessa classe, pois ela não tem construtor.

7)

Em uma aplicação Java orientada a objetos que usa relações de herança, uma

a) subclasse não pode ter mais que um construtor, mesmo que receba parâmetros diferentes.
b) superclasse não pode ter métodos sobrecarregados ou sobrescritos.
c) subclasse não pode sobrescrever um método da superclasse, mas o contrário é permitido.
d) subclasse herda somente os atributos e os métodos privados da superclasse.
e) subclasse normalmente usa a anotação @Override para indicar que um método da superclasse foi sobrescrito.

8)

Se uma classe na linguagem Java é declarada com o modificador abstract, então essa classe:

a) não pode ser referenciada;
b) não pode ser estendida;
c) não pode ser instanciada;
d) pode ser instanciada apenas uma vez;
e) não pode possuir métodos estáticos.

9)

Para responder, considere a classe abaixo que faz parte de uma aplicação Java.


```
public class Empregado extends Pessoa
{
private int numeroCtps;
private double renda;
   public Empregado(int numeroCtps, double renda, short id, String nome)
   {
      super(id, nome);
      this.numeroCtps = numeroCtps;
      this.renda = renda;
   }
   public Empregado()
   {
   }
   public int getNumeroCtps()
   {
      return numeroCtps;
   }
   public void setNumeroCtps(int numeroCtps)
   {
      this.numeroCtps = numeroCtps;
   }
   public double getRenda()
   {
      return renda;
   }
   public void setRenda(double renda)
   {
      this.renda = renda;
   }
}
```

De dentro do método main de uma classe da mesma aplicação, pode-se instanciar um objeto da classe Empregado utilizando-se a instrução Java:

a) Empregado emp = new Empregado(3471,2345.00,1,'Ana Paula');
b) Empregado e = new Empregado(68345,1380,56,1,"Pedro");
c) empregado = new Empregado(123,1380.0,1.0, “Pedro”);
d) Empregado emp = new Empregado(184,5678.56);
e) Empregado e = new Empregado( );

10)
Considere as classes a seguir, presentes em uma aplicação Java orientada a objetos:


```
public class Funcionario {
private int id;
private String nome;
private double valorBase;
    public Funcionario() {
    }
    public Funcionario(int id, String nome, double valorBase) {
        this.id = id;
        this.nome = nome;
        this.valorBase=valorBase;
    }
    public double getValorBase() {
        return valorBase;
    }
    public double calcularSalario(){
        return valorBase;
    }
}
public class Mensalista extends Funcionario{
private double descontos;
    public Mensalista(double descontos, int id, String nome, double
    valorBase) {
        super(id, nome, valorBase);
        this.descontos = descontos;
    }
@Override
    public double calcularSalario(){
        return super.getValorBase() - descontos;
    }
}
public class Diarista extends Funcionario {
    private int diasPorSemana;
    public Diarista( int diasPorSemana, int id, String nome, double
    valorBase) {
        super(id, nome, valorBase);
        this.diasPorSemana = diasPorSemana;
    }
    @Override
    public double calcularSalario(){
        return super.getValorBase() * diasPorSemana;
    }
}
```

Em uma classe principal foram digitadas, no interior do método main, as seguintes linhas:

```
double s;
Funcionario f;
f = new Diarista(3, 10456, "Ana Maria", 90);
s = f.calcularSalario( );
System.out.println(s);
f = new Mensalista(298.56, 10457, "Pedro Henrique", 877.56);
s = f.calcularSalario( );
System.out.println(s);
```

As linhas que contêm a instrução s = f.calcularSalario( ); demonstram um conceito da orientação a objetos conhecido como

a) encapsulamento.
b) sobrecarga de métodos.
c) polimorfismo de classes.
d) sobrescrita de construtores.
e) métodos abstratos.

DISCURSIVAS:

11)

No contexto da Programação Orientada a Objetos (POO), responda as seguintes perguntas:

a) O que são classes e do que elas são compostas?

b) O que são atributos de uma classe?

c) O que são métodos de uma classe?

d) O que são os objetos de uma classe?

12)

Analise o cenário da figura desta questão. Trata-se de uma imagem de um jogo de vídeo game de futebol. Em seguida, proponha uma classe com três atributos e dois métodos que pode fazer parte desse jogo.

Modelo:

public class _____ {

}

13)

Daniel desenvolveu uma classe chamada Equacao2Grau e, dentro dela, programou um método para calcular o delta, dado os coeficientes, chamado calcDelta. Entretanto, quando ele executa o comando abaixo, ocorre um erro na segunda linha:

1. Equacao2Grau eq1 = new Equacao2Grau();
2. int d = eq.calcDelta(1, -1, -12)
3. System.out.println("Delta = " + d);

Explique o que pode estar ocasionando o(s) erro(s).