Pergunta 1

Em Java, uma **coleção** é uma estrutura de dados que permite armazenar vários objetos. A própria coleção também é um objeto, na qual é possível realizar operações como adição e remoção de elementos.

Analise o código do programa a seguir, identificando como a coleção de objetos foi implementada e utilizada por meio da classe ***ArrayList***.

```java
public class Livro {
	private String nome;
	private String autor;
	private int ano;
	public Livro(String nome, String autor, int ano) {
		this.nome = nome;
		this.autor = autor;
		this.ano = ano;
	}
	public void imprime() {
		System.out.println("Nome : " + nome);
		System.out.println("Autor: " + autor);
		System.out.println("Ano  : " + ano);
	}
}
```

```java
import java.util.ArrayList;

public class Biblioteca {
	private String nome;
	private ArrayList<Livro> livros;

	public Biblioteca(String nome) {
		this.nome = nome;
		this.livros = new ArrayList<Livro>();
	}
	public static void main(String[] args) {
		Biblioteca bib = new Biblioteca ("Minha Estante");
		bib.livros.add(new Livro("1984", "George Orwell", 1949));
		bib.livros.add(new Livro("Hamlet", "Shakespeare", 1603));
		bib.livros.add(new Livro("Anna Karenina", "Tolstói", 1877));
		bib.livros.remove(2);
		System.out.println("-- " + bib.nome);
		for(Livro x : bib.livros)
			x.imprime();
		System.out.println(bib.livros.size());
		bib.livros.clear();
		System.out.println(bib.livros.size());
	}
}
```

Em relação ao código Java apresentado, determine a(s) afirmativa(s) correta(s) a seguir.

- A **linha 5** do código da classe **biblioteca** declara, mas não instancia o atributo **livros**.
- A **linha 9** do código da classe **biblioteca** instancia, mas não declara o atributo **livros**.
- A **linha 17** do código da classe **biblioteca** imprime na tela de *console* a seguinte informação:

```
--
livros
```

- Após a execução do iterador das **linhas 18 e 19** do código da classe **biblioteca**, será impressa na tela de *console* a seguinte informação:

```
Nome :
1984
Autor:
George
Orwell
Ano  :
1949
Nome :
Anna
Karenina
Autor:
Liev
Tolstói
Ano  :
1877
```

- A **linha 22** do código da classe **biblioteca** imprime na tela de *console* a seguinte informação:

```
0
```

A. III e IV.

B. II e III.

C. IV.

D. V.

E. I, III e V.

Pergunta 2

Na **Programação Orientada a Objetos**, ou **POO**, a construção do *software* se baseia na interação de unidade denominadas de objetos, os quais estão definidos e estruturados em classes. Dentre todos os recursos oferecidos pela POO, um dos mais importantes é o **encapsulamento**, que permite proteger os membros de uma classe (atributos e métodos), de forma que o acesso a eles seja controlado.

Em relação à POO, determine as afirmativas corretas a seguir.

- Em Java, quando utilizamos o modificador de acesso ***private*** em um atributo, entendemos que esse atributo apenas será acessível por todas as classes que estiverem no mesmo pacote da classe em que foi definido.
- Os métodos do tipo ***getter*** e ***setter*** devem ser declarados com o modificador de acesso do tipo ***protected***.
- Em Java, quando utilizamos o modificador de acesso ***protected*** em um atributo, entendemos que esse atributo com certeza será acessível por objetos das classes que estão no mesmo pacote da classe do atributo em questão.
- Em Java, quando utilizamos o modificador de acesso ***public*** em um atributo, entendemos que esse atributo será acessível por objetos de quaisquer outras classes.
- Os métodos do tipo ***getter*** e ***setter*** são recomendados para permitir o acesso controlado aos atributos privados da classe, provendo assim encapsulamento.

A. II e III.

B. III e V.

C. I, II e IV.

D. I e II.

E. III, IV e V.

Pergunta 3

O **encapsulamento**, na orientação a objetos, é utilizado para proteger os membros de uma classe (atributos e métodos), de forma que o acesso a eles seja controlado.

Verifique o encapsulamento nas classes em Java a seguir, procurando entender se os modificadores de acesso utilizados permitem a execução desse programa.

```java
package P2;

public class A {
    public     int a;
    protected  int b;
    private    int c;

    protected void calcula() {
        c = a + b;
    }

    public void imprime () {
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

```java
package P2;

public class Execucao {
    public static void main(String[] a)
    {
        A atividade = new A();
        atividade.a = 3;
        atividade.b = 4;
        atividade.calcula();
        atividade.imprime ();
    }
}
```

De acordo com o código apresentado, determine a(s) afirmativa(s) correta(s) a seguir.

- O programa possui duas classes, em um mesmo pacote, mas não executa, pois apresenta erro devido à falta de permissão de acesso ao atributo ou método.
- Ocorre erro de permissão de acesso quando a classe **execucao** tenta realizar a **linha 8**, na qual existe a tentativa de atribuição de valor: `atividade.a = 3;`.
- Ocorre erro de permissão de acesso quando a classe **execucao** tenta realizar a **linha 10**, na qual existe a tentativa de invocação de método: `atividade.imprime();`.
- O programa executa e exibe os valores de cada atributo da classe **A** na tela de *console*: **a = 3; b = 4; c = 7**.
- Se mudarmos o modificador de acesso do atributo **A.c** para ***protected***, o programa continua executando sem alterações.

A. I, II e III.

B. IV e V.

C. V.

D. IV.

E. I e III.

Pergunta 4

O comando ***switch case*** é usado quando temos várias opções (ou escolhas) e precisamos realizar tarefas diferentes para cada uma delas. O trecho de código em Java a seguir usa o switch case para apresentar diferentes saídas na tela de console. Verifique o comportamento do código, de acordo com o valor da variável **opt**.

```
1    int opt = ... ;  // opt pode ser qualquer valor inteiro;
2
3    switch (opt) {
4      case 1:
5          System.out.println("primeiro");
6      case 2:
7          System.out.println("segundo");
8      case 3:
9          System.out.println("terceiro");
10     default:
11         System.out.println("qualquer posição");
12   }
```

A respeito do **código do programa apresentado**, **assinale a alternativa correta** a seguir.

A. Se **opt = 5**, nada será impresso na tela de *console*.

B. Se **opt = 2**, será impresso na tela de *console*:

segundo.

C. Se opt = 3, será impresso na tela de *console*:

terceiro

qualquer posição.

D. Se **opt = 0**, será impresso na tela de *console*:

`primeiro`.

E. O código está incorreto, pois não possui o comando *break*.

Pergunta 5

Ao trabalharmos com expressões aritméticas em qualquer linguagem de programação, verificamos que a precedência (ou a ordem de realização) das operações matemáticas é obedecida integralmente.

No programa Java a seguir, observe como é o resultado das operações matemáticas utilizadas na classe **teste**. Verifique também como essa classe é instanciada e como sua instância é utilizada.

```
1    public class Teste {
2         private int   exp1;
3         private int   exp2;
4
5         protected void calcular1(int x) {
6                this.exp1 = x + 8*3+2-18/3+3*2;
7         }
8         protected void calcular2(int x) {
9                this.exp2 = x + 8*(3+2)-18/(3+3)*2;
10        }
11        public static void main(String[] args) {
12               Teste t = new Teste();
13               t.calcular1(0);
14               t.calcular2(4);
15               System.out.println(t.exp1);
16               System.out.println(t.exp2);
17        }
18   }
```

De acordo com o código apresentado, analise as afirmativas corretas a seguir.

- A classe **teste** é instanciada apenas uma vez.
- A classe **teste** declara modificadores de acesso nos seus atributos.
- A classe **teste** declara um método construtor.
- A linha 15 do programa exibe na tela de *console* o número **26**.

- A linha 16 do programa exibe na tela de *console* o número **32**.

A. IV e V.

B. I e II.

C. II e V.

D. I e V.

E. I, II e IV.

Pergunta 6

Classe é uma estrutura modelo utilizada para representar, em uma linguagem de programação, objetos do mundo real. Em uma classe, declaramos atributos e métodos, que representam as características desse objeto. Verifique a declaração da classe a seguir, identificando seus elementos.

```
1  public class Calculadora {
2          static int contAdicao;
3          static int contSubtracao;
4
5          static int soma (int a, int b) {
6                  contAdicao ++;
7                  return a + b;
8          }
9          static int subtrai (int a, int b) {
10                 contSubtracao += 1;
11                 return a - b;
12         }
13 }
```

De acordo com o código apresentado, determine a(s) afirmativa(s) correta(s) a seguir.

- A classe **Calculadora** define seu **comportamento** com dois atributos e seu **estado** com dois métodos, do tipo ***static***.
- O método **soma** deve usar a palavra-chave ***return***, pois foi declarado como ***static***.

- A **linha 6** do código incrementa a variável local **contAdicao** em uma unidade.
- O **contSubtracao** armazena quantas vezes o método **subtrai** foi invocado.
- A linha **10** do código decrementa o atributo **contSubtracao** em uma unidade.

A. I e IV.

B. I, II e V.

C. IV.

D. Todas as afirmativas estão incorretas.

E. II e V.

Pergunta 7

Em Java, **coleção** é uma estrutura de dados que permite armazenar vários objetos. A própria coleção também é um objeto, na qual é possível realizar operações como adição e remoção de elementos.

Verifique o código do programa a seguir, identificando como a **coleção de objetos** foi implementada e utilizada por meio da classe ***ArrayList***.

```java
public class Selo {
	private String nome;
	private double valor;
	private int ano;

	public Selo(String nome, double valor, int ano) {
		this.nome = nome;
		this.valor = valor;
		this.ano = ano;
	}
	public void imprime() {
		System.out.println("Nome Selo: " + nome);
		System.out.println("Valor    : R$ " + valor);
	}
}
```

```java
  import java.util.ArrayList;

public class Colecao {
	private String nomeColecao;
	private ArrayList<Selo> meusSelos;
	public Colecao(String nomeColecao) {
		this.nomeColecao = nomeColecao;
		meusSelos = new ArrayList<Selo>();
	}
	private void imprime() {
		System.out.println("-- Coleção: " + this.nomeColecao + " --");
		for(Selo s : this.meusSelos) {
			s.imprime();
		}
	}
	public static void main(String[] args) {
		Colecao minhaColecao = new Colecao("Selos do Brasil");
		Selo selo1 = new Selo("Carnaval", 0.5, 1998);
		Selo selo2 = new Selo("Independência", 0.25, 2010);
		Selo selo3 = new Selo("Finados", 0.15, 1980);
		minhaColecao.meusSelos.add(selo3);
		minhaColecao.meusSelos.add(selo2);
		minhaColecao.imprime();
		System.out.println("Total = " + minhaColecao.meusSelos.size());
	}
}

```

I. A **linha 6** do código da classe **selo** declara o início do método construtor da classe.

II. A **linha 8** do código da classe **colecao** instancia a classe ***ArrayList***, para que ela possa ser utilizada para manter os objetos da classe **selo**.

III. A **linha 23** do código da classe **colecao** imprime na tela de console a seguinte informação:

```
-- Coleção: Selos
do Brasil --
Nome Selo:
Carnaval
Valor    : R$ 0.5
Nome Selo:
Independência
Valor  : R$ 0.25
Nome Selo:
Finados
Valor  : R$ 0.15
```

IV. A **linha 24** do código da classe **colecao** imprime na tela de console a seguinte informação:

```
** Total de Selos
= 2
```

V. As **linhas 12, 13 e 14** do código da classe **colecao** percorrem toda a coleção de objetos, invocando o método construtor dos objetos da classe **selo**.

A. I, III e V.

B. I, II e IV.

C. I, II e III.

D. II e III.

E. II, III e V.

Pergunta 8

Na **Programação Orientada a Objetos (POO)**, a construção de um *software* se baseia na interação de unidades denominadas de objetos, os quais, por sua vez, são definidos e estruturados em classes.

Programar com **classes** e **objetos** traz a grande vantagem de ser mais adequado ao processo mental natural de agrupamento e mais perto da nossa experiência do mundo real. Por exemplo, uma classe **micro-ondas** teria o método **cozinhar**, o objeto **celular** poderia ter o método

enviarSMS, exemplos bem próximos ao nosso dia a dia, facilitando sua representação como classes e objetos na programação.

Em relação à POO e linguagem Java, determine a(s) afirmativa(s) correta(s) a seguir.

- Em **Java**, quando utilizamos a palavra-chave ***void*** antes do nome de um método, devemos usar o comando ***return*** no corpo do código desse método.
- Em **Java**, quando utilizamos o comando ***new*** antes da chamada de um método qualquer de uma classe, significa que queremos instanciar um objeto da classe em que o método foi declarado.
- Em uma classe **Java**, quando utilizamos a palavra reservada ***this*** antes da referência a um **atributo**, ele deve ter sido declarado como ***static***.
- Em **Java**, uma classe que não possua o método ***main(String[] args)*** declarado com o modificador ***public static void***, não pode ser executada.
- Em **Java**, quando, por exemplo, declaramos a importação de uma classe **limpeza** do pacote **servicos** (código **import Servicos.Limpeza**;) e utilizamos o método **varrer** da classe **limpeza** da seguinte forma: **Limpeza.varrer** (“sala”);, significa que o método **varrer** foi declarado como ***static*** na classe **limpeza**.

A. III.

B. I e V.

C. II e III.

D. II, III e IV.

E. IV e V.

Pergunta 9

Os **comandos condicionais**, ou de desvio, permitem alterar o fluxo de execução de um código, dependendo do resultado de uma **condição**, que pode ser verdadeiro ou falso. Verifique como é utilizado o condicional do código Java a seguir.

```java
int x = -4;
int y = -3;
int z;

if (x != y) {
    z = (x > y ? 2*x : -2*y);
}else {
    z = x + y;
}
System.out.println("z = " + z);
```

O que aparece impresso após a execução? Assinale a alternativa correta a seguir.

A. z = -8.

B. z = -7.

C. z = 6.

D. z = 6.

E. z = 8.

Pergunta 10

No Java, existem comandos alternativos para usar um **laço de repetição** (*loop*). O comando do tipo ***while*** é usado para **repetir** (**iterar**) uma parte do programa várias vezes. Já o comando do tipo ***do-while*** é usado quando o número de iterações não é fixo e é preciso executar o *loop* pelo menos uma vez.

Ainda é possível controlar as repetições em um *loop* com os comandos ***break*** e ***continue***. Quando encontramos um ***break***, o *loop* imediatamente é encerrado e o controle do programa irá para a primeira instrução após o *loop*. Já a instrução ***continue*** é usada dentro de um *loop* quando é preciso pular (saltar) para a próxima iteração do *loop*, sem necessariamente interrompê-lo.

Considere o programa Java a seguir, que utiliza os dois tipos de *loop*: ***while*** e ***do-while***, assim como os comandos ***break*** e o ***continue***. Analise o comportamento do programa.

```
1     int i = 10;
2     do {
3         i --;
4         if (i == 6)
5                 break;
6         System.out.println(i);
7     }while(i > 1);
8     System.out.println("i = " + i);
9
10    int k = 5;
11    while (k > 1) {
12        k--;
13        if (k == 2)
14                continue;
15        System.out.println(k);
16    }
17    System.out.println("k = " + k);
```

O que é apresentado na tela de *console* quando a execução do programa atinge, respectivamente, as linhas **8** e **17** do código? Assinale a alternativa correta a seguir.

A. i = 2 e k = 1.

B. i = 1 e k = 1.

C. i = 2 e k = 2.

D. i = 6 e k = 2.

E. i = 6 e k = 1.