**GOVERNO DO ESTADO DO RIO DE JANEIRO**
SECRETARIA DE ESTADO DE CIÊNCIA E TECNOLOGIA
FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA - FAETEC
FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO
FERNANDO MOTA

| Professor(a): MÁRCIO BELO | Disciplina 3P013 | Turno ( ) Manhã (X) Noite | Avaliação: ( )AV1 (X)AV2 ( )AVS ( )AVF | Data: 03/07/23 |
|---|---|---|---|---|
| Nome do Aluno(a) em letra de forma: | | | Matrícula: | |
| Nota: | Nota por extenso | Visto Prof(a) | Nota Revista | Nota por extenso | Visto Prof(a) |

Considere o código a seguir:

```
abstract class Conta {
  public double saldo=0.0;
  public void depositar(double valor) {
      saldo += valor - valor*0.02;
  }
}
class ContaCorrente extends Conta {
    private int codigo;
    public int getCodigo() { return codigo; }
    public ContaCorrente(int codigo) {
        this.codigo=codigo;
    }
}
```

1.[0,5] Sobre o código Conta anterior, são verdadeiras:
I. Uma instância da classe ContaCorrente pode acessar o método depositar.
II. Uma instância da classe ContaCorrente permite o acesso à variável saldo.
III. Uma instância da classe ContaCorrente, acessada de um código externo à própria classe, NÃO pode acessar diretamente a variável codigo.
A. I e II B. Somente III C. II e III D. Somente I E. Nenhuma F. Somente II G. Todas H. I e III

2.[0,5] Sobre o código anterior, são verdadeiras:
I. O código Conta c1=new Conta(); funcionará e criará uma instância de Conta com saldo zerado.
II. O código ContaCorrente cc=new ContaCorrente(); funcionará e criará uma instância de ContaCorrente com saldo zerado.
III. O código ContaCorrente cc=new ContaCorrente(123); funcionará e criará uma instância de ContaCorrente com saldo zerado.
A. Somente I B. Todas C. I e II D. Nenhuma E. Somente III F. Somente II G. I e III H. II e III

-0,5

3.[0,5] Sobre coleções ordenadas em Java, considere as seguintes assertivas:
I. Para que um objeto seja colocado numa coleção ordenada, a classe a qual ele pertence deve obrigatoriamente implementar a interface Comparable.
II. Um HashSet exige dois tipos de objetos ordenáveis, enquanto um TreeSet não.
III. Um ArrayList só é ordenado por demanda, através do uso do método de classe sort da classe Collections.
São verdadeiras: A. Somente I B. I e III C. Somente II D. Somente III E. I e II

-0,5

4.[0,5] Considere o código Java a seguir:

```
01:public class CA {
02:    int a=1;
03:    CA() { }
04:    CA(int v) {
05:          this.a=v;
06:    }
07:    int getA() { return a + a; }
08:    public static void main(Strings args[]) {
09:          int valor;
10:          CA x = new CA();
11:          CA y = new CA( 3 );
12:          CA z = new CA( x.getA() );
13:          valor = x.getA() + y.getA() + z.getA();
14:          System.out.println( valor );
15:    }
16:}
```

Quais métodos são os construtores da classe CA mostrado no código anterior?
A. CA() e CA(int v)
B. CA() e getA()

C. CA(int v) e main(String args[])
D. getA()
E. CA(int v) e getA()
F. CA(), CA(int v) e getA()
G. CA()
H. CA(int v)

5.[1,0] Considere o seguinte código em Java:

```
// imports omitidos
class Operacao {
        private double valor;
        public double getValor() {
                return this.valor;
        }
        // ... código omitido
}
public class Caixa {
        private String nomeOperador;
        private List<Operacao> operacoes = new ArrayList<>();
        // ... código omitido
        public double obterTotal() {
                // o código reposta entrará aqui !!!!
        }
}
```

Complete o trecho de código completa o método obterTotal, de forma que ele retorne o valor acumulado de operações.

6.[0,5] Considere as afirmações acerca de sobrecarga de métodos:
I. Permite implementar numa classe dois métodos que tenham o mesmo nome;
II. As assinaturas dos métodos sobrecarregados devem ser iguais;
III. O compilador java decide qual método sobrecarregado a utilizar baseado no tipo dos argumentos passados ao método;
São verdadeiras: A. I e II B. Todas C. Somente III D. II e III E. Nenhuma F. I e III G. Somente II H. Somente I

7.[1,0]

```
public class 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");
  }
}
```

Cite e explique o resultado do código anterior.

8.[1,5] Considere o código abaixo como se cada classe estivesse em seu próprio arquivo:

```
public interface Comunicavel {
        public abstract String falar();
}
public class Normal implements Comunicavel {
        @Override
        public String falar() {
                return "biro biro";
        }
}
public class Maluco implements Comunicavel {
        @Override
        public String falar() {
                return "lero lero";
        }
}
```

```
import java.util.ArrayList;
import java.util.List;
public class Auditorio {
	private List<Comunicavel> faladores;
	public Auditorio() {
		// complete aqui
	}
	public void palestrar() {
		for(Comunicavel comunicavel : faladores) {
			System.out.println(comunicavel.falar());
		}
	}
	public static void main(String[] args) {
		new Auditorio().palestrar();
	}
}
```

Ele resultou na seguinte saída no console:

lero lero
biro biro
lero lero
lero lero
lero lero

Complete o código da classe Auditorio, no ponto indicado, para gerar o resultado esperado.

9.[4,0] Com base no teste a seguir, desenvolva a classe Conta para passar em todos os testes do novo software MolhaMao da empresa Odebritch:

```
public class ContaTest {

    @Test
    public void testRepartePixuleco() {

        Conta c = new Conta("FeioMonte", "DeluvioSoares");
        assertEquals("FeioMonte:DeluvioSoares=100", c.listarPercentuais());
        c.adicionarPropineiro( "ZeDorceu", 30);
        assertEquals("FeioMonte:DeluvioSoares=70,ZeDorceu=30", c.listarPercentuais());
        c.adicionarPropineiro( "MarcosValeria", 20);
        assertEquals("FeioMonte:DeluvioSoares=50,ZeDorceu=30,MarcosValeria=20",
c.listarPercentuais());

        try {
            c.adicionarPropineiro( "JoaoVaquinhaNeto", 60);
            fail("Olho grande!");
        }
        catch(PixulecoException pe) {
            assertEquals("Excede o pixuleco de 50 do DeluvioSoares", pe.getMessage());
            assertEquals("FeioMonte:DeluvioSoares=50,ZeDorceu=30,MarcosValeria=20",
c.listarPercentuais());
        }
        String listaPixuleco = c.distribuirPixuleco( 1000000.8);
        assertEquals("FeioMonte:DeluvioSoares=500000.4,ZeDorceu=300000.24,MarcosValeria=200000.16",
listaPixuleco);
    }
}
```

*** BOA PROVA ***