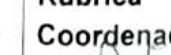




UNIVERSIDADE  
**VILA VELHA**  
ESPIRITO SANTO

Curso de Ciência da Computação			
Disciplina: Programação Orientada a Objetos I		Nota: 6,5	Rubrica Coordenador
Professor: Cássio Capucho Peçanha			
Aluno: [REDACTED]		Valor: 7,0 p <sup>tos</sup>	
Turma: CC2M	Semestre: 1º.		
Data: 21/06/2023	Avaliação: Bimestral		

### INSTRUÇÕES DA PROVA

- Leia atentamente as questões antes de respondê-las;
- Todas as questões deverão ser respondidas com CANETA azul ou preta;
- Prova a lápis não tem direito à revisão;
- As questões objetivas rasuradas serão consideradas nulas;
- Desligue o celular, não consulte material, colegas ou fontes de qualquer outra natureza. Evite que sua prova seja recolhida pelo professor por atitudes indevidas.
- PROVA SEM CONSULTA E INDIVIDUAL.

**1ª. questão.** (1,0 ponto). Considere que uma empresa está desenvolvendo um novo canal de atendimento ao consumidor. Para que possa realizar o atendimento de forma organizada as solicitações de atendimentos serão alocadas em uma coleção de dados em que, os atendimentos deverão ser realizados em ordem de registro. Cada registro possui uma chave gerada ao solicitar atendimento. As chaves são sequenciais. Qual seria a melhor estrutura para o projeto?

- ☒ a) Map.
- ☐ b) Array
- ☐ c) List.
- ☐ d) String.
- ☐ e) Set.



1,0

2ª questão. (1,0 ponto) Dados os seguintes trechos de código, sobre a relação de Cliente e Conta podemos afirmar que

```

public class Cliente {
    public String nome;
    public String cpf;
    public String endereco;
}

public class Conta {
    private double saldo;
    private double limite;
    public int numero;
    public Cliente dono;
    public Conta() {
        this.saldo = saldo_criacao;
        this.limite = limite_criacao;
    }
    public void sacar(double valor) {
        saldo -= valor;
    }
    public void depositar(double valor) {
        saldo += valor;
    }
}

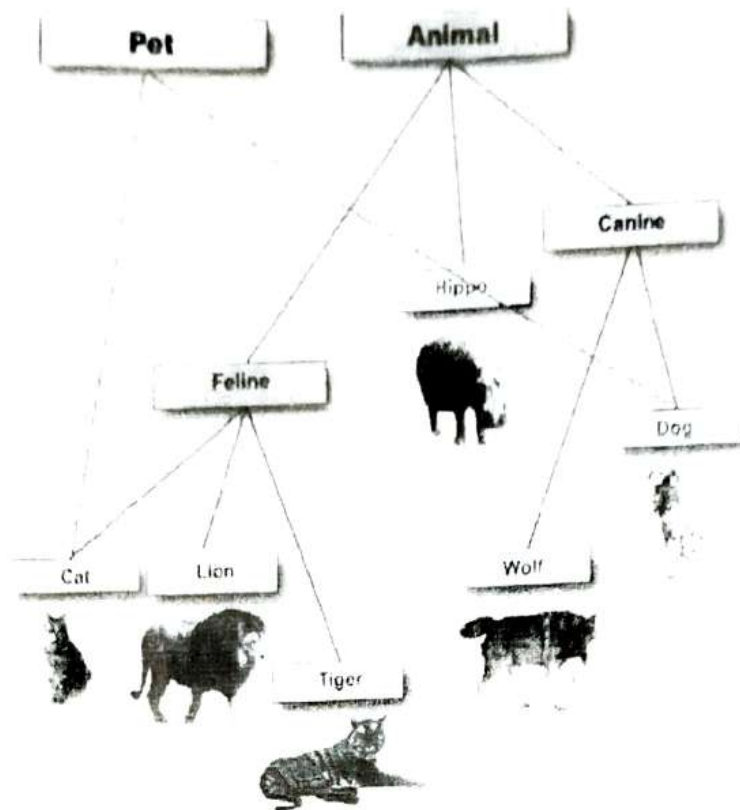
public class Main {
    public static void main(String[] args) {
        Cliente cliente = new Cliente();
        cliente.nome = "Carlos Eduardo Pereira";
        cliente.cpf = "123.456.789-00";
        cliente.endereco = "Rua X, 123";
        Conta conta = new Conta();
        conta.dono = cliente;
        System.out.println("Hello World!");
    }
}

```

É correto afirmar que:

- a) A relação de Cliente e Conta pode ser definida como Associação Simples
- b) A relação Cliente e Conta pode ser definida como Composição
- c) A relação Cliente e Conta pode ser definida como Agregação
- d) Cliente e Conta não possuem relação direta
- e) Um cliente não pode existir sem uma conta

3ª. questão. (1,0 ponto) A partir do modelo representado na imagem abaixo



Considerando o padrão estabelecido em C#. Qual seria a declaração que melhor se adequa a classe Dog?

- a) Class Dog extends Canine implements IPet
- b) Class Dog extends Animal implements ICanine
- c) Class Dog extends Canine implements IAnimal
- d) Class Dog extends Pet extends Canine
- e) Class Dog extends Canine extends Pet

✓ 10

4ª. questão. (1,0 ponto). Referente aos conceitos de herança e classes abstratas. Analise as seguintes afirmações:

A superclasse pode ser uma classe abstrata. Verdadeiro

EM DECORRÊNCIA DISSO  
Ao herdar uma classe abstrata, precisamos sobrescrever todos os seus métodos declarados. Verdadeiro

✓ 05

5ª. questão. (1,5 ponto) Desenvolva um método que atenda ao conceito de polimorfismo da programação orientada a objetos.

```
import java.util.*;
```

```
public class Animal {
```

```
    public String nome;
```

```
    public int peso;
```

```
    public void comer() {
```

```
        System.out.println("O animal comeu");
```

```
    }
```

```
}
```

```
public class Cachorro extends Animal {
```

```
    public String raca;
```

```
    public Cachorro (String n, int p, String r) {
```

```
        this.nome = n;
```

```
        this.peso = p;
```

```
        this.raca = r;
```

```
    }
```

```
    public void comer() { // Sobrescrita
```

```
        System.out.println("O cachorro comeu.");
```

```
    }
```

```
    public void comer (String comida) { //sobrecarga
```

```
        System.out.println("O cachorro comeu " + comida);
```

```
    }
```

```
}
```

6ª. questão. (1,5 ponto) Desenvolva duas classes (uma abstrata e uma concreta) onde a segunda deverá herdar os métodos da primeira. A classe abstrata deve ter um método abstrato e um método concreto.

```
import java.util.*;

public class abstract Animal {
    public String nome;
    public int peso;
    public abstract void emitirSom();
    public void comer() {
        System.out.println("O animal come.");
    }
}

public class Cachorro extends Animal {
    public String raca;

    public Cachorro(String n, int p, String r) {
        this.nome = n;
        this.peso = p;
        this.raca = r;
    }

    public void emitirSom() {
        System.out.println("O cachorro late.");
    }

    public void comer() {
        System.out.println("O cachorro come carne.");
    }
}
```

**Boa prova!**