UNIVERSIDADE VILA VELHA
ESPÍRITO SANTO

Curso de Ciência da Computação			
Disciplina: Programação Orientada a Objetos I		**Nota:** 1,5	**Rubrica Coordenador**
Professor: Cássio Capucho Peçanha			
Aluno: Lucas Carrijo Ferrari			
Turma: CC2M	**Semestre:** 1º	**Valor:** 7,0 pts	
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?

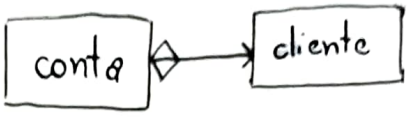
chave

- 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(double saldo_criacao, double limite_criacao){
        this.saldo = saldo_criacao;
        this.limite = limite_criacao;
    }
    public void Sacar(double valor_saque){
        saldo=saldo-valor_saque;
    }
    public void Depositar(double valor_deposito){
        saldo=saldo+valor_deposito;
    }
}
public class Avaliacao {
    public static void main(String[] args) {
        Cliente cliente = new Cliente();
        cliente.nome = "Cássio Caoucho Peçanha";
        cliente.cpf = "124.456.369-76";
        cliente.endereco = "Rua X, N.8";
        Conta conta = new Conta(0.0, 500);
        conta.Dono = cliente;
        System.out.println("Hello World!");
    }
}
```

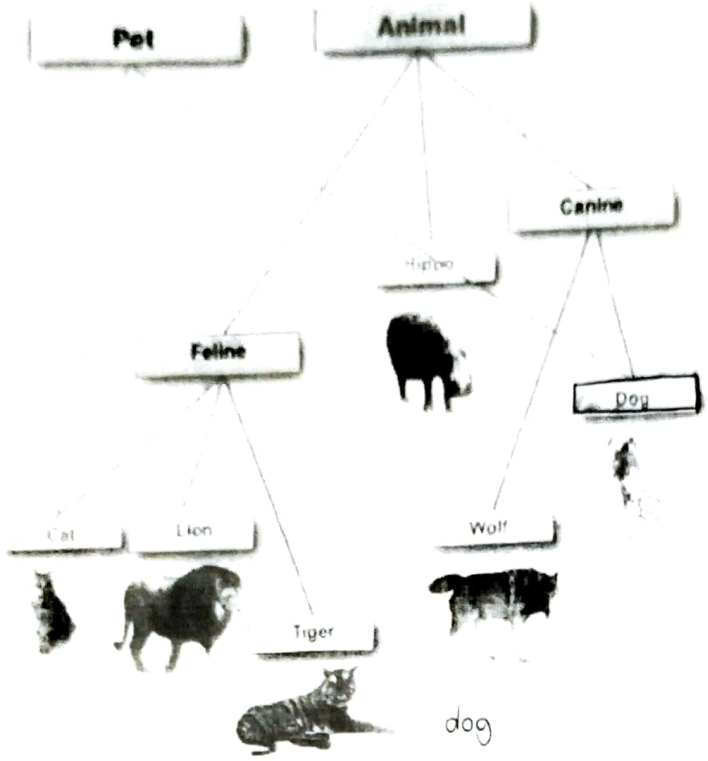


Avaliação

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



dog

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~~

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

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

```
public interface iAnimal {
    abstract void barulho ();
}

public class Cachorro implements iAnimal {
    public String nome;

    public void barulho () {
        System.out.println ("Au Au Au!");
    }
}
```

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.

```
public abstract class Pessoa {
    public String nome;
    public int idade;

    public int aumentar Idade ( [redacted] ) {
        [redacted]
        this.idade = this.idade + 1;
        return this.idade;
    }

    abstract void correr ();
}

public class Aluno extends Pessoa {
    public String nome;
    public int idade;

    public int aumentar Idade ();

    public void correr () {
        System.out.println ( this.nome + " está correndo muito, cuidado!");
    }
}
```

Boa prova!