Questão 2) Reflita e Responda com suas palavras as seguintes perguntas:

1. O que é herança em Programação Orientada a Objetos (POO) e qual seu principal objetivo?

R= Herança trata-se de um mecanismo da POO que permite que uma classe (subclasse/classe filha) herde atributos e métodos de outra classe (superclasse/classe mãe) de forma a garantir o reaproveitamento de código e estabelecer uma relação hierárquica entre classes, facilitando a manutenção e a organização do sistema como um todo.

2. Qual é a diferença entre uma superclasse e uma subclasse? Dê um exemplo de cada.

R= A primeira trata-se da classe mais genérica, a qual contém atributos e comportamentos comuns a suas derivadas. A segunda é uma classe mais específica que herda da primeira seus atributos e comportamentos, além de possuir atributos e comportamentos particulares.
Exemplo:

```
//Superclasse

public class Pessoa{

    protected String nome;

    protected int idade;

    public void apresentar(){

        System.out.println("Olá, meu nome é "+nome+" e tenho " +idade+" anos.");

    }

}

//Subclasse

public class Estudante extends Pessoa{

    private String curso;

    public void apresEstudante(){

        super.apresentar();

    }

}
```

3. Dada a seguinte hierarquia UML: Pessoa → Estudante, o que significa afirmar que "todo Estudante é uma Pessoa, mas nem toda Pessoa é um Estudante"?

R= Tratando-se de uma relação de herança do tipo especialização, significa que Estudante (classe filha) herda de Pessoa (classe mãe), então todo objeto do tipo Estudante possui as características (métodos e atributos) de uma Pessoa, sendo o inverso falso.

4. Quais são as vantagens do uso da herança no desenvolvimento de software orientado a objetos? Por que uma subclasse não consegue acessar diretamente atributos declarados como private na superclasse? Como isso pode ser resolvido?

R= A capacidade de se reutilizar partes do código, a facilidade de manutenção advinda dessa reutilização, a possibilidade de se hierarquizar e, portanto, visualizar com maior facilidade a estrutura do código, além da capacidade de se aplicar polimorfismo. Quanto aos atributos private: a subclasse não consegue acessar diretamente os atributos private da superclasse em razão deles estarem encapsulados, o que pode ser solucionado a partir do uso de métodos get e set ou da mudança do modificador dos atributos de private para protected.

5. Qual o símbolo e a direção da seta usada para representar herança em diagramas de classes UML?

R= O símbolo utilizado é uma seta com ponta triangular vazada que parte da subclasse para a superclasse.

6. Para que serve a palavra-chave super em Java? Cite dois contextos diferentes em que ela pode ser usada.

R= A palavra-chave super é usada para acessar elementos da superclasse (métodos e atributos) que se encontrem protegidos, como no caso de uma chamada de método usando super.nomedometodo() ou quando se declara atributos herdados da superclasse no construtor da classe filha com super(atributo x).

7. Um sistema define as classes Professor, ProfHorista e ProfDE. Por que é mais vantajoso centralizar atributos comuns na classe Professor ao invés de declará-los separadamente nas subclasses?

R= Porque evita duplicação de código e garante sua reutilização e a manutenção, além de permitir a aplicação de polimorfismo, podendo alterações em características presentes na superclasse serem feitas apenas uma vez e ecoarem nas classes derivadas sem dificuldade.

8. Suponha que você esteja modelando um sistema de transporte. Como você organizaria uma hierarquia de classes para representar Transporte, TransporteTerrestre, TransporteAereo, Carro, Avião e Helicóptero? Qual o papel da herança nessa modelagem?

R= Transporte – TranporteTerrestre -> Carro

|

Transporte Aereo -> Avião, Helicoptero

No problema há: uma relação de especialização de Transporte com TransporteAereo e com TransporteTerrestre; relação de especialização de TransporteAereo com Aviao e Helicoptero, e de TransporteTerrestre com Carro. Quanto ao papel da herança, ela permite definir comportamentos e atributos generalizados em Transporte e especializá-los nas subclasses, tendo também as classes filhas métodos e atributos particulares.

9. O que acontece se, em uma subclasse, você não chamar explicitamente o construtor da superclasse usando super()? Em que situação isso pode gerar erro?

R= Quando uma subclasse é definida e seu construtor não chama explicitamente o construtor da superclasse utilizando super(), o compilador insere automaticamente uma chamada ao construtor sem argumentos da superclasse. No caso de a superclasse possuir um construtor sem parâmetros, o código compila e funciona normalmente (mesmo sem a chamada explícita a super()). O(s) erro(s) ocorre(m) em situações onde a superclasse define apenas construtores com parâmetros e não declara um construtor sem argumentos, pois nesse caso se a subclasse omitir a chamada explícita a super() com os argumentos necessários, o compilador não saberá identificar qual construtor da superclasse deve ser chamado.