Quinta lista de exercícios de Programação Orientada a Objetos

Questões teóricas

1. Diferencie herança, agregação e composição.
2. O que são classes abstratas e qual o seu uso mais comum?
3. Um classe abstrata pode herdar de outra classe abstrata? Por que?

Questões práticas

4. Copie as 4 classes fornecidas abaixo (Veiculo, VeiculoTerrestre, Carro e a Driver Class). Então, responda às questões subsequentes:

```
public abstract class Veiculo {
    protected String modelo;
    public void liga(){
        System.out.println(modelo + " ligado! (Veiculo)");
    }
    public abstract void desliga();
}

public abstract class VeiculoTerrestre extends Veiculo{
    @Override
    public void desliga(){
        System.out.println(modelo + " desligado! (Veiculo Terrestre)");
    }
}
```

```
public class Carro extends VeiculoTerrestre{
  private class Motor{
    String modelo;
    public void liga(){}
  }
  private Motor meuMotor;
  String modelo;
  public Carro(String modelo){
      this.modelo = modelo;
      meuMotor = new Motor();
  }
  @Override
  public void liga(){
    meuMotor.liga();
    System.out.println(modelo + " ligado! (Carro)");
  }
}
```

```
import java.util.ArrayList;
public class Teste {
    public static void main(String[] args) {
        Carro c1 = new Carro("VW");
        Carro c2 = new Carro("GM");
        Carro c3 = new Carro("Honda");
        ArrayList<Carro> lc = new ArrayList<Carro>();
        lc.add(c1);
        lc.add(c2);
        lc.add(c3);
        lc.getFirst().liga();
        //lc = null
        c1.liga();
    }
}
```

5. Execute o programa. Quantas mensagens são exibidas? A(s) mensagem(ns) é(são) de qual(is) carro(s)?
6. Descomente a penúltima linha da driver class (lc = null) e também acrescente ao código da driver uma última linha onde você invoca novamente o método "lc.getFirst().liga();" O que aconteceu?
7. Remova o último comando "lc.getFirst().liga();" mas mantenha "lc = null". A lista de carros foi apagada. Verifique se os carros ainda existem após esse comando!
8. Agora remova também a instrução "lc = null". Crie a classe Onibus, com **exatamente a mesma implementação da classe Carro**, mas neste caso, **não implemente o método liga()**. Agora, inclua ao final da driver class as 3 linhas de código abaixo:
   Onibus o1 = new Onibus("MeuOnibus");
   o1.liga();
   o1.desliga();
9. Execute o programa. Ao ser ligado, as mensagens produzidas pelo ônibus falam em Veiculo, Veiculo Terrestre ou Ônibus? Por que?

10. As mensagens produzidas pelo ônibus contém um erro: Fala-se em “null” no lugar do modelo. Corrija a falha para que a mensagem torne-se correta **pela simples remoção de uma única linha de código em alguma das classes**!
11. Crie mensagens personalizadas para a classe Onibus, ao ser ligado e desligado.

AGREGAÇÃO x COMPOSIÇÃO

12. Crie um código exemplo de agregação: com as classes Disciplina, Aluno e também a driver class. Permita que cada disciplina possa ter vários alunos e cada aluno possa estar em mais de uma disciplina.
13. Crie um código exemplo de composição: Crie a classe Residencia e a classe Cômodo, e faça com que cada novo objeto do tipo Residencia criado tenha necessariamente ao menos um cômodo, mas possivelmente mais de um. Igualmente, não permita que um cômodo possa existir sem estar atrelado a um objeto de Residencia. Verifique a consistência de visibilidade das suas classes e atributos!