Ministério da Educação
Universidade Federal de Ouro Preto (Ufop)
Departamento de Computação e Sistemas (Decsi)

Professor: Rafael Alexandre
rfalexandre@ufop.edu.br
Disciplina: CSI032

Trabalho Prático 06 (TP06)

Instruções:

i - O arquivo deve ser entregue em formato .ZIP ou .RAR seguindo a nomenclatura: "XXXX.KKK" onde XXXX é o número de sua matrículo e KKK a extensão do arquivo.

ii - Cada um dos exercícios deve criado em um diretório com o seguinte nome: Exercicio_XX onde XX é o número da questão solucionada.

iii - Para cada programa desenvolvido deverão ser entregues **SOMENTE** os arquivos de projeto e classes Java em seus respectivos pacotes.

iv - O arquivo deve ser enviado via moodle limitado a data e hora de entrega definida no Plano de Ensino. Não serão aceitos trabalhos enviados por e-mail.

.

Questão 1. Responta as perguntas abaixo:

(a) Explique com suas palavras por que uma classe abstrata não pode ser instanciada.

(b) Explique, com suas palavras, por que interfaces não podem ter construtores.

(c) Explique com suas palavras por que não podemos ter construtores declarados com a palavrachave abstract

Obs: as respostas deverão ser entregues em um arquivo PDF.

Questão 2. Identifique e explique o(s) erro(s) na classe abaixo:

```
1  public class Produto {
2      private String identificacao;
3      private double custoDeFabricacao;
4      Produto(String i, double c) {
5          identificacao = i;
6          custoDeFabricacao = c;
7      }
8      abstract public String toString();
9      abstract public void novoCusto(double nc);
10 }
```

Código 1: Classe Produto

Questão 3. Identifique e explique o(s) erro(s) na classe abaixo:

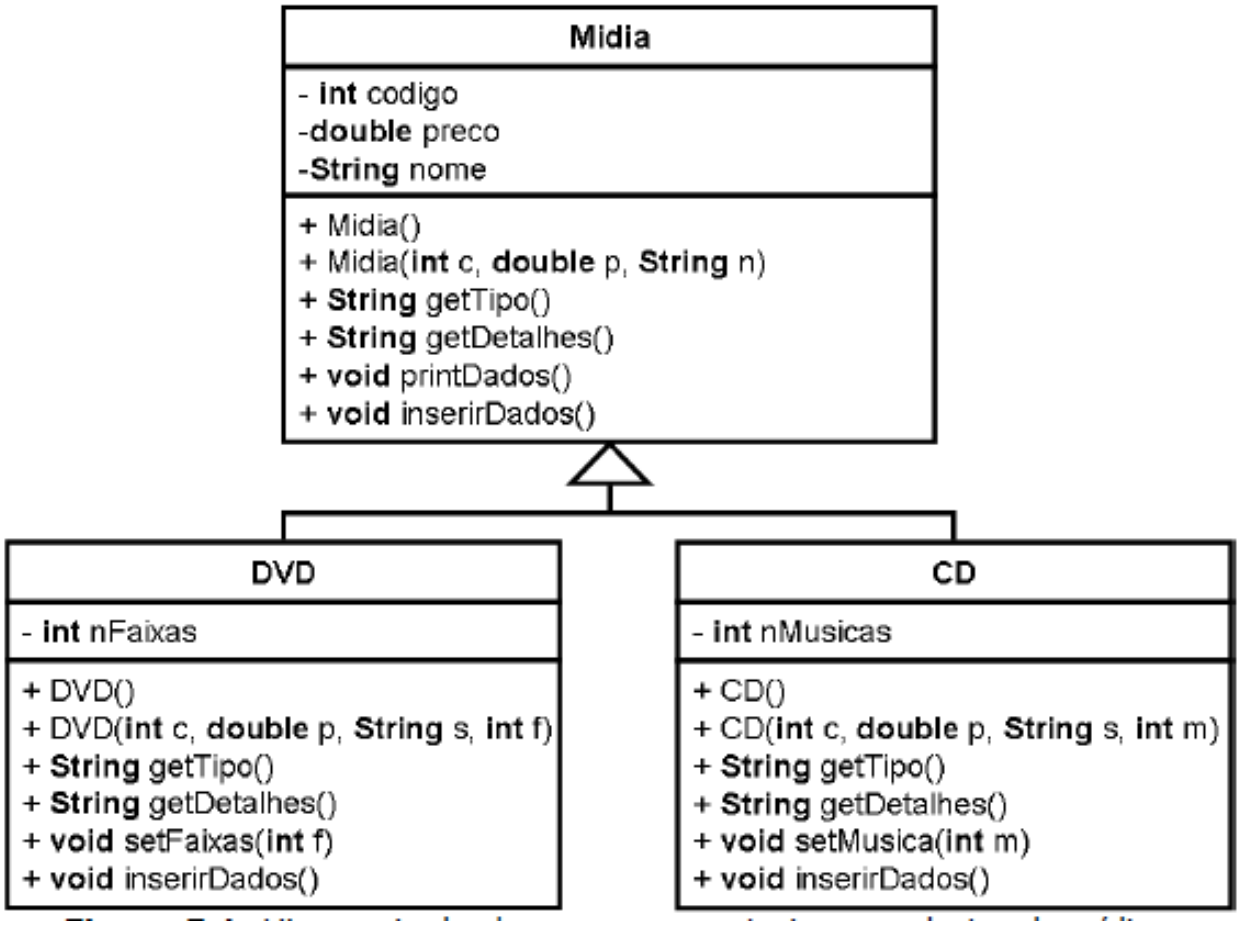
```java
public abstract class Dispositivo {
    private String nome;
    private long capacidadeEmBytes;
    public Dispositivo(String n, long c) {
        nome = n;
        capacidadeEmBytes = c;
    }
    abstract public String toString();
    abstract public double capacidadeEmMegabytes();
}
```

Código 2: Classe Dispositivo

```java
public class DiscoOtico extends Dispositivo {
    public DiscoOtico(long c){
        super("Disco  tico ", 241172480L);
    }
    public String toString() {
        return "Dispositivo:" + nome + " Capacidade:" + c;
    }
}
```

Código 3: Classe DiscoOtico

Questão 4. Considere simular o cadastro de uma loja que vende CD e DVDS e que para tanto, utiliza o diagrama de classes abaixo.



Considere, ainda, o detalhamento abaixo que fornece uma descrição dos métodos que deverão ser elaborados para cada uma das classes:

(a) **getTipo()**: Retorna uma String com o nome da classe.

(b) **getDetalhes()**: Retorna uma String com as informações contidas nos campos.

(c) **printDados()**: Imprime as informações contidas nos campos da classe. Para tanto, usa dois métodos para recuperar estas informações: getTipo() e getDetalhes(). Estas funções por sua vez são polimórficas, ou seja, seu tipo retorno varia de acordo com a classe escolhida, tal que este método é sobreposto nas subclasses.

(d) **inserirDados()**: Insere os dados necessários para se preencher os campos de um objeto de uma dada classe. Seu comportamento é polimórfico.

Além dos métodos descritos acima, deverão ser criados os métodos get e set correspondentes para retorna e modificar o conteúdo dos campos, respectivamente, bem como os construtores com e sem parâmetros de cada classe.

Baseado no que foi apresentado, deseja-se inserir uma nova funcionalidade na qual ao se chamar o método getDetalhes() das classes da Figura **??**. Esta nova funcionalidade consiste em se tocar uma música ao se chamar o método getDetalhes para um objeto da classe CD (por exemplo a primeira música do CD) e mostrar uma imagem ao se chamar getDetalhes para um objeto da classe DVD (por exemplo a capa do DVD). Para tanto, considere a implementação de interfaces necessárias.

Criar um programa que simule o uso de um cadastro de CD e DVDs.

Questão 5. Considere as interfaces abaixo:

```
1 public interface Animal {
2  //Retorna o nome da especie do animal.
3  public String getNomeEspecie();
4  //Retorna o nome do animal.
5  public String getNomeAnimal();
6 }
```

Código 4: Interface Animal

```
1 public interface Ferramentas {
2  public Animal[] filtraEspecie(Animal[] completo, String
      especieFiltrar);
3  public String[] classificaEspecies(Animal[] completo);
4 }
```

Código 5: Interface Ferramentas

Escreva um método que receba dois parâmetros:

(a) um vetor A de objetos que implementam a interface Animal representando diversos animais;

(b) um objeto que implementa a interface Ferramentas.

O método deve contabilizar o número de animais disponíveis em cada uma das espécies e retornar os resultados como um vetor de objetos da classe Resultado (apresentada abaixo). Cada objeto conterá uma espécie e o número de animais da mesma contabilizados. Devem ser consideradas apenas as espécies cujos animais estão presentes no vetor.

```
1     public class Resultado {
2      private String nomeEspecie; // nome da especie
3      private int quantidade; // quantidade de animais da especie
```

```
4       public Resultado(String nomeEspecie, int quantidade) {
5       this.nomeEspecie = nomeEspecie;
6       this.quantidade = quantidade;
7       }
8       public String getNomeEspecie() {
9       return nomeEspecie;
10      }
11      public int getQuantidade() {
12      return quantidade;
13      }
14     }
```

Código 6: Especificação da classe Resultado

Questão 6. Crie uma estrutura hierárquica que contenha as seguintes classes: Veiculo (classe abstrata), Bicicleta e Automóvel. Os métodos da classe Veiculo são todos abstratos e possuem a seguinte assinatura:

- public float acelerar(float velocidade);
- public void parar();

Estes métodos são implementados nas subclasses Automóvel e Bicicleta. Acrescentar na classe Automóvel o método public void trocarOleo(float litros).