Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 2° semestre de 2014.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1) (4.5 pontos)
Estamos rodeados de diversos tipos de aparelhos eletrônicos. Dois deles se destacam: televisão e blue-ray player. Analisando televisões, observa-se que elas possuem:

Fabricante (texto)	Preço (real)	Tela plana ou não (lógico)
Sistema de cor (texto)	Tamanho em polegadas (real)	Código único que identifica aparelhos eletrônicos (inteiro)

São observados em blue-rays:

Fabricante (texto)	Preço (real)	Região de codificação (inteiro)
Sistema de cor (texto)	Reprodução de MP3 (lógico)	Código único que identifica aparelhos eletrônicos (inteiro)

Nesta questão:

(a) Utilize herança para especializar a classe de aparelho eletrônico em televisão e blue-ray player. Distribua os atributos entre as classes de modo a evitar reescrita de código.

(b) De modo coerente, defina cada uma das três classes como abstrata ou concreta.

(c) Cada uma das três classes relacionadas a aparelhos eletrônicos deve ter um único construtor de inicialização para seus atributos.

(d) Sobrecarregue o método toString para retornar a descrição completa do objeto. Isto é, o tipo do objeto e o estado de todos os seus atributos.

(e) Atributos numéricos não podem receber valores menores que zero. Em caso de inconsistência, a exceção DadoInvalidoException deve ser lançada. Esta classe de exceção deve ser declarada como subclasse de Exception.

(f) Atributos do tipo texto não podem receber valores nulos ou texto vazio. Em caso de inconsistência, a exceção DadoInvalidoException deve ser lançada.

(g) O sistema de cor não pode receber valores diferentes dos permitidos pelo seu domínio: “NTSC” ou “PAL-M”. Em caso de inconsistência, a exceção DadoInvalidoException deve ser lançada.

(h) A região do blue-ray player não pode receber valores diferentes dos permitidos pelo seu domínio: 1, 2, 3 ou 4. Em caso de inconsistência, a exceção DadoInvalidoException deve ser lançada.

RESPOSTA:

```
public class Q1_AP2_2014_2{

  public static void main (String[] args) throws Exception{

    AparelhoEletronico[] vet = new AparelhoEletronico[4];

    vet[0] = new BlueRayPlayer("FabBRP", "PAL-M", 100F, true, 1);

    vet[1] = new BlueRayPlayer("FabBRP", "NTSC", 200F, false, 2);

    vet[2] = new Televisao("FabTV", "PAL-M", 300F, 43, true);

    vet[3] = new Televisao("FabTV", "NTSC", 40F, 29, false);



    for(int i = 0; i < vet.length; i++)
      System.out.println(vet[i]);

  }

}
```

```
abstract class AparelhoEletronico {

  private final String fabricante;

  private final String sistemaDeCor;

  private final double preco;

  private static int prox_cod = 0;

  private int codigo;



public AparelhoEletronico(String fabricante, String sistemaDeCor,
double preco) {

    if ((fabricante == null) || (fabricante.equals(""))) {

      throw new DadoInvalidoException("Fabricante inválido.");

    }

    this.fabricante = fabricante;



     if ((sistemaDeCor == null) || ((!sistemaDeCor.equals("NTSC")) && (!
sistemaDeCor.equals("PAL-M")))) {

      throw new DadoInvalidoException("Sistema de cor inválido.");

    }

    this.sistemaDeCor = sistemaDeCor;



    if (preco < 0.0) {
```

```
    throw new DadoInvalidoException("Preço inválido.");
  }
  this.preco = preco;

  codigo = ++prox_cod;
}

public final String obterFabricante() {
  return this.fabricante;
}

public final String obterSistemaDeCor() {
  return this.sistemaDeCor;
}

public final double obterPreco() {
  return this.preco;
}

public final int obterCodigo() {
```

```
    return this.codigo;

  }



  public String toString() {

     return "codigo = " + codigo + ", fabricante = " + this.fabricante + ",
sistema de cor = " + this.sistemaDeCor + ", preço = " + this.preco;

  }

}



class BlueRayPlayer extends AparelhoEletronico {

  private final boolean mp3;

  private final int regiao;



   public BlueRayPlayer(String fabricante, String sistemaDeCor, double
preco, boolean mp3, int regiao) {

    super(fabricante, sistemaDeCor, preco);

    this.mp3 = mp3;



    if ((regiao < 1) || (regiao > 4)) {

      throw new DadoInvalidoException("Região inválidoa");

    }

    this.regiao = regiao;
```

```
  }

  public final boolean tocaMP3() {
    return this.mp3;
  }

  private final int obterRegiao() {
    return this.regiao;
  }

  public String toString() {
     return "Blue-ray player: " + super.toString() + ", mp3 = " + this.mp3 +
", regiao = " + this.regiao;
  }
}

class Televisao extends AparelhoEletronico {
  private final double tamanho;
  private final boolean telaPlana;
```

```
  public Televisao(String fabricante, String sistemaDeCor, double preco,
double tamanho, boolean telaPlana) {

    super(fabricante, sistemaDeCor, preco);



    if (tamanho < 0.0) {

      throw new DadoInvalidoException("Tamanho inválido.");

    }

    this.tamanho = tamanho;

    this.telaPlana = telaPlana;

  }



  public final double obterTamanho() {

    return this.tamanho;

  }



  public final boolean ehTelaPlana() {

    return this.telaPlana;

  }



  public String toString() {

        return "Televisão: " + super.toString() + ", tamanho = " +
this.tamanho + ", tela plana = " + this.telaPlana;
```

```
  }
}

class DadoInvalidoException extends RuntimeException {
  public DadoInvalidoException(String msg) {
    super(msg);
  }
}
```

Questão 2) (1.5 pontos)
Associe cada um dos termos enumerados abaixo com sua descrição:

1	Herança	3	Não permite que seja realizada a implementação dos métodos em seu escopo. Serve como contrato onde as classes que a implementam devem obrigatoriamente declarar todos os seus métodos.
2	Classes abstratas	4	É a capacidade de um objeto ser referenciado de diversas formas.
3	Interface	5	Garante a manutenibilidade do código pois cada alteração deverá ser feita em apenas um local.
4	Polimorfismo	2	Permite a declaração de métodos e atributos concretos e abstratos, porém, não pode ser instanciada diretamente.
5	Encapsulamento	1	É a capacidade de uma classe ter todos os métodos e atributos de instância de outra classe sem precisar reescrever o código.

Questão 3) (4.0 pontos)
Suponha que tenhamos que implementar um sistema para controle de gastos de automóveis. Os gastos essenciais são de 2 tipos: abastecimento e manutenção, conforme as classes abaixo.
Altere este programa de forma que estes objetos possam ser manipulados juntos, **numa mesma coleção**, no método main(). As classes *Manutencao* e *Abastecimento* podem ser modificadas, mas devem continuar existindo. Use recursos de OO sempre que possível.

```java
import java.util.Arrays;
import java.util.List;

class Manutencao {
	String peca;
	double custo;
	public Manutencao(String peca, double custo) {
		this.peca = peca;
		this.custo = custo;
	}
	public double getCusto() {
		return custo;
	}
}

class Abastecimento {
	float valor;
	String posto;
	public Abastecimento(float valor, String posto) {
		this.valor = valor;
		this.posto = posto;
	}
	public float getValor() {
		return valor;
	}
}

public class AP2_2014_2_Q3 {
	public static void main(String[] args) {
		double somaReparos = 0, somaManutencao = 0;
		List <Manutencao> reparos = Arrays.asList(new Manutencao("Freio",
150), new Manutencao("Oleo", 200));
		for (Manutencao m : reparos) {
			somaReparos = somaReparos + m.getCusto();
		}
		List <Abastecimento> abastecimentos = Arrays.asList(new
Abastecimento(80, "BR"), new Abastecimento(50, "Shell"));
		for (Abastecimento a : abastecimentos) {
			somaManutencao = somaManutencao + a.getValor();
		}
		System.out.println("A soma dos valores gastos e': " +
(somaManutencao + somaReparos));
	}
}
```

RESPOSTA:

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```

```java
interface Gasto {
	double getCusto();
}

class Manutencao implements Gasto {
	String peca;
	double custo;
	public Manutencao(String peca, double custo) {
		this.peca = peca;
		this.custo = custo;
	}
	public double getCusto() {
		return custo;
	}
}

class Abastecimento implements Gasto {
	float valor;
	String posto;
	public Abastecimento(float valor, String posto) {
		this.valor = valor;
		this.posto = posto;
	}
	public double getCusto() {
		return valor;
	}
}

public class AP2_2014_2_Q3 {
	public static void main(String[] args) {
		double soma = 0;
		List <Manutencao> reparos = Arrays.asList(new Manutencao("Freio",
150), new Manutencao("Oleo", 200));
		List <Abastecimento> abastecimentos = Arrays.asList(new
Abastecimento(80, "BR"), new Abastecimento(50, "Shell"));
		List <Gasto> gastos = new ArrayList<Gasto>();
		gastos.addAll(reparos);
		gastos.addAll(abastecimentos);
		for (Gasto g : gastos) {
			soma = soma + g.getCusto();
		}
		System.out.println("A soma dos valores gastos e': " + soma);
	}
}
```