Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP1 2° semestre de 2017.

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) (4.0 pontos)

Você foi chamado pelo Comitê Olímpico Internacional para desenvolver uma biblioteca de classes Java que ajude a gerenciar times, esportes e quadro de medalhas das próximas Olimpíadas. A especificação que lhe deram foi a seguinte:

(a) **(1.0 ponto)** A classe Time implementa a ideia concreta de um time que participa dos jogos olímpicos. Um time deve possuir um nome e uma nacionalidade obrigatórios e imutáveis. Logo, os estados desses dois atributos devem ser informados via um construtor de inicialização e não poderão ser alterados após a construção do objeto. São exemplos de times o time masculino de basquetebol e o time feminino de tênis de mesa. Para fins de simplificação, não foi modelada a existência explícita de atletas nos times. Logo, a ideia de atleta não precisa ser implementada. A classe Time deve conter métodos que permitam a consulta de seu nome e nacionalidade.

(b) **(1.0 ponto)** A classe Esporte define a ideia abstrata de um dos 42 esportes olímpicos. O item **(c)** especifica alguns desses esportes. O que todo esporte tem em comum é um nome obrigatório e imutável, informado via construtor de inicialização, e a referência para os times que ficaram, respectivamente, em primeiro, segundo e terceiro lugar na competição. Como pré-requisito, a colocação dos times deve ser armazenada em um Array de Arrays, sendo o mais externo de tamanho 3 (uma posição por colocação) e os mais internos de tamanho n, específico por esporte concreto. Por exemplo, existe a disputa de basquetebol masculino e feminino. Logo, é preciso armazenar dois primeiros colocados, dois segundos colocados e dois terceiros colocados ($n=2$ nesse caso) no basquetebol.

Tênis de mesa, por sua vez, é disputada em três modalidades, tanto no masculino quanto no feminino, totalizando $n=6$ times em cada colocação. A ideia abstrata de esporte não consegue definir o valor de n *a priori*. Por isso, n deve ser informado no construtor de inicialização da classe abstrata e não poderá ser alterado após a construção. A classe Esporte deve conter métodos que permitam tanto a atribuição quanto a consulta de times em cada colocação. Esses métodos não poderão receber nem retornar Arrays. A classe também deverá disponibilizar um método de consulta ao nome do esporte.

(c) **(1.0 ponto)** Dos 42 esportes presentes nos jogos olímpicos, você deverá implementar classes concretas para apenas dois deles: basquetebol e tênis de mesa. Ambas as classes deverão conter apenas o construtor padrão, pois todos os dados requeridos pela ideia abstrata de esporte são conhecidos na construção do contexto especializado. Os valores podem ser deduzidos dos exemplos apresentados no item **(b)**.

(d) **(1.0 ponto)** A classe QuadroDeMedalhas mantém a coleção de 42 esportes presentes nas olimpíadas. A coleção é inicializada no construtor padrão da classe e não pode ser alterada durante a vida de um objeto de QuadroDeMedalhas. Por questão de tempo, reserve espaço para os 42 esportes mas inclua na coleção apenas instâncias de basquetebol e tênis de mesa. A classe deve fornecer um método de consulta que retorna a instância de um esporte a partir de seu nome e um método de consulta que retorne um valor inteiro que indica quantas medalhas de ouro um determinado país conquistou.

Não é necessária/permitida a comunicação com o usuário via entrada e saída padrão.

Não é permitido o uso de classes como ArraysList, LinkedList ou similares, disponíveis na Java API.

```
//LETRA (a)
public class Time {
   private final String nome;
   private final String nacionalidade;

   public Time(String nome, String nacionalidade) {
      this.nome = nome;
      this.nacionalidade = nacionalidade;
   }

   public final String getNome() { return nome; }
   public final String getNacionalidade() { return nacionalidade; }
}


//LETRA (b)
public abstract class Esporte {
```

```
    private final String nome;
    private final Time[][] times;

    public Esporte(String nome, int n) {
        this.nome = nome;
        this.times = new Time[3][n];
    }

    public final String getNome() { return nome; }
    public final int getN() { return this.times[0].length; }
    public Time getTimeOuro(int id) { return this.times[0][id]; }
    public Time getTimePrata(int id) { return this.times[1][id]; }
    public Time getTimeBronze(int id) { return this.times[2][id]; }

    public void setTimeOuro(Time time, int id) {
        this.times[0][id] = time;
    }

    public void setTimePrata(Time time, int id) {
        this.times[1][id] = time;
    }

    public void setTimeBronze(Time time, int id) {
        this.times[2][id] = time;
    }
}


//LETRA (c)
public class Basquetebol extends Esporte {
    public Basquetebol() { super("Basquetebol", 2); }
}

public class TenisDeMesa extends Esporte {
    public TenisDeMesa() { super("TenisDeMesa", 6); }
}


//LETRA (d)
public class QuadroDeMedalhas {
    private final Esporte[] esportes;

    public QuadroDeMedalhas() {
```

```
        this.esportes = new Esporte[42];
        this.esportes[0] = new Basquetebol();
        this.esportes[1] = new TenisDeMesa();
    }

    public Esporte getEsporte(String nome) {
        for (Esporte e : this.esportes) {
            if (e.getNome().equals(nome)) { return e; }
        }
        return null;
    }

    public int getMedalhas(String pais) {
      int ouro = 0;
      for (Esporte e : this.esportes) {
        for (int id = 0; id < e.getN(); ++id) {
          Time t = e.getTimeOuro(id);
          if (t != null && t.getNacionalidade().equals(pais)) {
            ouro++;
          }
        }
      }
      return ouro;
    }
}
```

Questão 2) (3.0 pontos)
Considere o trecho de programa abaixo que simula a preparação de uma receita de arroz num programa Java:

```
class Ingrediente {
        String nome;
        int quantidade;
        String unidade; // litro, gramas, colher de chá, etc
}
class Receita {
        Ingrediente ingredientes[];
        int numIngredientes;

        public void aquecer () {...}
        public void fritar () {...}
        public void ferver () {...}
        public void mexer () {...}
        public void encerrar (boolean seco) {...}
}
public class AP1_2017_2_Q2 {
        public static void main(String[] args) {
                Ingrediente oleo = new Ingrediente("Oleo", 50, "ml");
        }
}
```

a) Forneça construtores para a classe Ingrediente e Receita. Forneça ainda um segundo construtor para a classe Ingrediente, o qual permite que apenas o nome seja informado. A quantidade pode ser obtida chamando o método estático random() da classe Math. Como o método retorna um valor na faixa [0..1], multiplique o valor por 100 para obter um valor mais real. O valor da unidade pode ser "qualquer".
b) Na classe Receita crie um método chamado adicionar(Ingrediente), o qual adiciona um ingrediente à receita.
c) Considerando o exemplo de criação do ingrediente óleo na main(), continue a codificação da receita abaixo. Não é necessário codificar os métodos de preparo, nem criar nenhum novo método.

Ingredientes:
50 ml de Óleo
Cebola e alho a gosto
Sal a gosto
2 xícaras de arroz
4 xícaras de água

Modo de Preparo:
Aquecer óleo (azeite ou manteiga)
Fritar cebola e alho
Colocar 4 xícaras de água , sal a gosto e deixar ferver
Adicionar 2 xícaras de arroz, mexer bem
Quando secar, desligar o fogo, deixar descansar e em seguida soltar com um garfo

RESPOSTA:

```
class Ingrediente {
	String nome;
	int quantidade;
	String unidade; // litro, gramas, colher de chá, etc

	public Ingrediente(String nome, int quantidade, String unidade) {
		this.nome = nome;
		this.quantidade = quantidade;
		this.unidade = unidade;
	}

	public Ingrediente(String nome) {
		this(nome, (int)(Math.random()*100), "qualquer");
	}
}

class Receita {
	Ingrediente ingredientes[];
	int numIngredientes;

	public Receita(int num) {
		ingredientes = new Ingrediente[num];
		numIngredientes = 0;
```

```
	}

	public void adicionar (Ingrediente i) {
		ingredientes[numIngredientes] = i;
		numIngredientes++;
	}

	public void aquecer () {}
	public void fritar () {}
	public void ferver () {}
	public void mexer () {}
	public void encerrar (boolean seco) {}
}

public class AP1_2017_2_Q2 {
	public static void main(String[] args) {
		Ingrediente oleo = new Ingrediente("Oleo", 50, "ml");
		Ingrediente cebola = new Ingrediente("Cebola");
		Ingrediente alho = new Ingrediente("Alho");
		Ingrediente sal = new Ingrediente("Sal");
		Ingrediente arroz = new Ingrediente("Arroz", 2, "xícara");
		Ingrediente agua = new Ingrediente("Agua", 4, "xícara");

		Receita arrozSimples = new Receita (6);
		arrozSimples.adicionar(oleo);
		arrozSimples.aquecer();
		arrozSimples.adicionar(alho);
		arrozSimples.adicionar(cebola);
		arrozSimples.fritar();
		arrozSimples.adicionar(agua);
		arrozSimples.adicionar(sal);
		arrozSimples.ferver();
		arrozSimples.adicionar(arroz);
		arrozSimples.mexer();
		arrozSimples.encerrar(true);
	}
}
```

Questão 3) (3.0 pontos)
Dada a classe abaixo, a qual representa um ponto em 2 dimensões:

```
class Ponto {
	private double x, y;

	public Ponto(double x, double y) {
		this.x = x;
		this.y = y;
	}
}
```

(a) Defina uma classe Ponto3D que permite a criação de um ponto em 3 dimensões.
(b) Dado um outro ponto como argumento (um outro objeto Ponto3D ou as coordenadas x, y e z deste outro ponto), retorne o objeto Ponto3D referente à diferença entre as coordenadas.

(c) Calcule a distância entre 2 pontos definindo um método de instância (método não estático). Supondo ponto P com dimensões px, py, pz, Q com dimensões qx, qy e qz, a distância é calculada com a seguinte fórmula:

$$distancia = raiz\_quadrada\ (\ (px - qx)^2 + (py - qy)^2 + (pz - qz)^2\ )$$

Obs.: 1) Utilize os conceitos de OO vistos sempre que possível; 2) A raiz quadrada pode ser calculada com o método Math.sqrt

RESPOSTA:

```
class Ponto {
	private double x, y;
	public Ponto(double x, double y) {
		this.x = x;
		this.y = y;
	}
	// Necessário para acessar os campos na classe Ponto3D
	public double getX() { return x; };
	public double getY() { return y; };
}
// item a)
class Ponto3D extends Ponto {
	private double z;

	public Ponto3D(double x, double y, double z) {
		super(x, y);
		this.z = z;
	}
	public double getZ() { return z; };

	// item b) OU ...
	public Ponto3D diferenca (Ponto3D p) {
		return new Ponto3D (p.getX() - this.getX(), p.getY() -
this.getY(), p.getZ() - this.getZ());
	}
	// ... item b)
	public Ponto3D diferenca (double x, double y, double z) {
		return new Ponto3D (x - this.getX(), y - this.getY(), z -
this.getZ());
	}
	// item c)
	public double distancia (Ponto3D p) {
		return Math.sqrt(Math.pow(p.getX() - this.getX(), 2) +
				Math.pow(p.getY() - this.getY(), 2) +
				Math.pow(p.getZ() - this.getZ(), 2));
	}
	// Era pedido método de instância. Entretanto, esta versão poderá ser
aceita, desde declarada de forma correta
	public static double distancia (Ponto3D p, Ponto3D q) {
		return Math.sqrt(Math.pow(p.getX() - q.getX(), 2) +
				Math.pow(p.getY() - q.getY(), 2) +
				Math.pow(p.getZ() - q.getZ(), 2));
	}
}
```