Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP1 2° semestre de 2007.

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)

Implemente um programa que modele objetos geométricos. Os objetos geométricos são: retângulo, caixa, círculo, cilindro, paralelogramo, cubo e quadrado. Crie classes para estes objetos de forma a aproveitar ao máximo suas características comuns. Nestas classes, crie construtores parametrizados, os quais devem ser utilizados para inicialização dos atributos dos objetos. Para cada objeto, crie métodos para obter seu perímetro (figuras 2D), sua área (externa, no caso das 3D) e volume (figuras 3D).

Retângulo: Área = base*altura, Perímetro = 2*base + 2*altura

Caixa: Volume = base1*base2*altura, Área = 2*(base1*base2 + base1*altura + base2*altura)

Círculo: Área = $3.14*(raio)^2$, Perímetro = 2*3.14*raio

Cilindro: Volume = $3.14*(raio)^2$*altura, Área = $2*3.14*(raio)^2$ + 2*3.14*raio*altura

Paralelogramo: Área = base*altura, Perímetro = 2*(base + altura)

Cubo: Volume = $(lado)^3$, Área = $6*(lado)^2$

Quadrado: Área = $(lado)^2$, Perímetro = 4*lado

Imaginando que objetos geométricos serão fabricados, e que o material utilizado deve ser economizado, defina um método que compare estes objetos por sua área ocupada e outro que compare por suas capacidades volumétricas.

SOLUÇÃO:
O código abaixo visa atender alguns requisitos de engenharia de código, como fácil extensão e reutilização. Com isso, foram utilizadas interfaces, hierarquias para todos os objetos geométricos com reuso sempre que possível, dentre outros aspectos. Entretanto, o que será cobrado é o que foi pedido no exercício: "Crie classes para estes objetos de forma a aproveitar ao máximo suas características comuns".

```
/* Interface Objeto2D
 * Modela os objetos 2D */
interface IObjeto2D {
     float area();
     float perimetro();
}

/* Interface Objeto3D
 * Modela os objetos 3D
 * Obs.: Esta extensão faz com que objetos 3D tenham
 * um método perímetro. Como esta situação não é
 * apresentada no exercício, neste gabarito o
 * perímetro é 0 para objetos 3D. Entretanto, outras
 * soluções podem ser aceitas */
interface IObjeto3D extends IObjeto2D {
     float volume();
}

/* Classe Retangulo */
class Retangulo implements IObjeto2D {
     float lado1, lado2;

     public Retangulo(float l1, float l2) {
          lado1 = l1;
          lado2 = l2;
     }

     public float area() {
          return lado1*lado2;
     }

     public float perimetro() {
          return 2*lado1 + 2*lado2;
```

```
	}

	public boolean maiorArea(IObjeto2D o) {
		if (this.area() > o.area())
			return true;
		return false;
	}
}

/* Classe Caixa
 * Extende a classe Retangulo adicionando altura */
class Caixa extends Retangulo implements IObjeto3D {
	float altura;

	public Caixa(float l1, float l2, float h) {
		super(l1, l2);
		altura = h;
	}

	public float volume() {
		return lado1*lado2*altura;
	}

	public float area() {
		return 2*(lado1*lado2 + lado1*altura +
lado2*altura);
	}

	public float perimetro() {
		return 0;
	}

	public boolean maiorVolume(IObjeto3D o) {
		if (this.volume() > o.volume())
			return true;
		return false;
	}
}

/* Classe Circulo
 * Define um atributo estático, pois é utilizado
 * por diversos métodos */
class Circulo implements IObjeto2D {
	static float PI = 3.14f;
	float raio;

	public Circulo(float r) {
```

```
            raio = r;
        }

        public float area() {
            return PI*raio*raio;
        }

        public float perimetro() {
            return 2*PI*raio;
        }

        public boolean maiorArea(IObjeto2D o) {
            if (this.area() > o.area())
                return true;
            return false;
        }
}

/* Classe Cilindro
 * Extende a classe Circulo adicionando um
 * atributo altura */
class Cilindro extends Circulo implements IObjeto3D {
        float altura;

        public Cilindro(float r, float h) {
            super(r);
            altura = h;
        }

        public float volume() {
            return PI*raio*raio*altura;
        }

        public float area() {
            return 2*PI*raio*(raio + altura);
        }

        public float perimetro() {
            return 0;
        }

        public boolean maiorVolume(IObjeto3D o) {
            if (this.volume() > o.volume())
                return true;
            return false;
        }
}
```

```
/* Classe Paralelograma
 * É uma especialização de Retangulo. Entretanto,
 * para cálculo de área e perímetro basta herdar
 * os métodos de Retangulo */
class Paralelogramo extends Retangulo {
     float angulo_menor;

     public Paralelogramo(float l1, float l2, float omega) {
          super(l1, l2);
          angulo_menor = omega;
     }
}

/* Classe Cubo
 * É um caso particular de caixa (lados iguais) */
class Cubo extends Caixa {
     public Cubo(float l) {
          super(l, l, l);
     }
}

/* Classe Quadrado
 * É um caso particular de retângulo (lados iguais) */
class Quadrado extends Retangulo {
     public Quadrado(float l) {
          super(l, l);
     }
}
```

Questão 2)

Implemente um programa que faça a ordenação de um vetor de inteiros. Você deve passar, como parâmetro de entrada o vetor a ser ordenado. A saída deste programa será o vetor de entrada ordenado.

SOLUÇÃO:

```
public class ordena_bolha {
  public static void main(String[] args) {
    int vet[] = {20, 10, 3, 5, 0, 70, 100};

    Ordena (vet);

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

```
  }

  public static void Ordena (int[] vet){
     int i, j, menor, temp;

     for (i = 0; i < vet.length; i++){
       menor = i;

       for (j = i + 1; j < vet.length; j++)
        if (vet[j] < vet [menor]) menor = j;

      temp = vet[menor];
      vet[menor] = vet[i];
      vet[i] = temp;
     }
  }
}
```

Questão 3)

Quais são as impressões do programa abaixo? Suas respostas só serão validadas se você explicar o porquê.

```
class refs{
  int id;

  refs(){
    id = 0;
  }

  public static void divide (int elem){
    elem = elem / 2;
    System.out.println(elem);
  }

  public static void modifica(refs x){
    x.id = 3;
  }

  public static void main (String[] args){
    int num = 2;
    refs.divide(num);
    System.out.println(num);

    refs obj = new refs();
    refs.modifica(obj);
    System.out.println(obj.id);
  }
```

```
}
```

SOLUÇÃO:

O resultado da 1ª impressão ("refs.divide(num);") é 1. Este resultado é alcançado porque o programa imprime a divisão do parâmentro de entrada (neste caso, o valor 2) por 1.

O resultado da 2ª impressão ("System.out.println(num);") é 2. Este resultado é obtido porque parâmentros de entrada cujos tipos são os primitivos da linguagem (tais como, char, float, double, int, boolean) não são modificados quando são passados a métodos, mesmo quando se faz alterações de seus valores.

O resultado da 3ª impressão ("System.out.println(obj.id);") é 3. Isto acontece porque o objeto "obj" não é de um tipo primitivo da linguagem. Toda vez que se altera uma variável que não é do tipo primitivo, esta variável sofre alteração.

Questão 4)

Protozoários são seres simples, unicelulares. Cada protozoário tem um número de nascimento seqüencial que o identifica. Este número nunca é alterado e nenhum outro protozoário pode ter a mesma identidade.

O genoma de um protozoário consiste de 30 nucleotídeos. Um nucleotídeo pode ter o valor "A", "C", "G" ou "T". Protozoários se reproduzem por divisão ou emparelhamento com um outro protozoário. Um protozoário que nasceu por divisão ou clone é uma cópia integral de seu pai com exceção da identidade. Um protozoário que nasceu como resultado de pareamento herda uma combinação do genoma do pai, de maneira que cada nucleotídeo tem uma chance de 50% de ter o mesmo valor do nucleotídeo do pai. Às vezes uma mutação ocorre no genoma de um protozoário, o que altera o valor de um nucleotídeo num genoma.

Implemente a classe Protozoário, a qual deve conter os seguintes métodos:

- public Protozoario() constrói um Protozoário com um genoma aleatório
- public Protozoario (char[] genoma) constrói um Protozoário com o dado genoma
- public void mutacao() alteração aleatória de um nucleotídeo do genoma.
- public Protozoario clone() retorna um novo protozoário, o qual nasce por divisão.
- public Protozoario parear(Protozoario outro) constrói um Protozoário por pareamento.
- public String toString() cria uma representação em String de um Protozoário.

Dica: Para geração de valores aleatórios utilize o método de classe *Math.random()*, o qual gera um *double* entre 0 e 1.

SOLUÇÃO:

Abaixo temos uma possível implementação para a classe Protozoário. A classe não precisa ser estritamente esta apresentada, mas precisa ter o conteúdo descrito na questão (estrutura e métodos). Na solução abaixo, alguns métodos são desnecessários, mas deixam a classe mais completa. Estes métodos estão na penúltima e última páginas deste gabarito.

```
class Protozoario {

//--------Variáveis de classe-----------------------------

     private static int contador=0;              //
instâncias criadas
     private static final int MAX_GENOMA = 30;  // no. de
nucleotídeos no genoma
     private static final char[] nucleotideos = new char[]
{'A','C','G','T'};

//--------Variáveis de instância--------------------------

     private final int identidade; // identidade
     private char[] genoma; // genoma

     //--------Construtores-----------------------------------

     /*
        Cria um novo protozoário com genoma aleatório.
      */
     public Protozoario() {
          this.genoma = new char[MAX_GENOMA];
          contador++;
          this.identidade = contador;
          atribuirGenomaAleatorio();
     }

     /*
        Cria um protozoário com um dado genoma.
      */
     public Protozoario(char[] genoma) {
          this.genoma = new char[MAX_GENOMA];
          contador++;
          this.identidade = contador;
```

```
            /* Caso o genoma fornecido tenha tamanho menor, ou
             * contenha nucleotídeos inválidos, são fornecidos
             * nucleotídeos aleatórios
             */
            if
((genoma==null)||(this.genoma.length!=genoma.length)) {
                  atribuirGenomaAleatorio();
            } else {
                  for(int i=0; i<this.genoma.length; i++) {
                        if (ehNucleotideo(genoma[i]))
                              this.genoma[i]=genoma[i];
                        else

      this.genoma[i]=atribuirNucleotideoAleatorio();
                  }
            }
      }

      /*
         Alteração aleatória de um genoma
       */
      public void mutacao() {
            int posicao = (int)(Math.random()*MAX_GENOMA);
            char novoNuc;

            // Certifica que um genoma é alterado
            do {
                  novoNuc = atribuirNucleotideoAleatorio();
            } while(this.genoma[posicao]==novoNuc);

            this.genoma[posicao] = novoNuc;
      }

      /*
         Retorna um clone de um protozoário. A nova
         identificação é garantida dentro do construtor.
       */
      public Protozoario clone() {
            return new Protozoario(this.genoma);
      }

      /*
         Retorna novo indivíduo após pareamento
       */
      public Protozoario parear(Protozoario p) {
            char[] genomaFilho = new char[MAX_GENOMA];
```

```
            for(int i=0; i<this.genoma.length; i++) {
                if (Math.random()<0.5)
                    genomaFilho[i]= this.genoma[i];
                else
                    genomaFilho[i]= p.genoma[i];
            }
            return new Protozoario(genomaFilho);
        }

        /*
           Retorna String que identidade e genoma.
         */
        public String toString() {
            String my = "Protozoario
"+this.identidade+"\nGenoma: ";

            for(int i=0; i<this.genoma.length; i++)
                my += this.genoma[i];

            return my;
        }

        /**********************************************
         * Métodos opcionais que foram utilizados neste
         * gabarito
         **********************************************/

        /*
           Atribuir genoma aleatório a um protozoário
         */
        private void atribuirGenomaAleatorio() {
            if (this.genoma==null)
                return;

            for (int i=0;i<genoma.length;i++)
                genoma[i] = atribuirNucleotideoAleatorio();
        }

        /*
           Retorna um nucleotídeo aleatório.
         */
        private char atribuirNucleotideoAleatorio() {
            int g = (int)(Math.random()*nucleotideos.length);
            return nucleotideos[g];
        }
```

```
    /*
       Testa se um nucleotídeo pertence a um genoma.
     */
    private boolean ehNucleotideo(char c) {
         for (int i=0;i<nucleotideos.length;i++)
              if (c==nucleotideos[i])
                   return true;
         return false;
    }

    public static int retornarNumeroNucleotideos() {
         return MAX_GENOMA;
    }

    /*
       Retorna nucleotídeo numa dada posição
     */
    public char retornaNucleotideo(int posicao) {
         if (posicao<0|| posicao>=MAX_GENOMA)
              return ' ';

         return this.genoma[posicao];
    }

    /*
       Retorna uma cópia do genoma.
     */
    public char[] retornaGenoma() {
         if (this.genoma==null)
              return null;

         char[] copia = new char[this.genoma.length];

         for (int i=0;i<this.genoma.length;i++)
              copia[i] = this.genoma[i];

         return copia;
    }
}
```