Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

**Curso de Tecnologia em Sistemas de Computação**
**AD2 de Programação III**
**2° semestre de 2008.**

**Nome:**
**Matrícula:**
**Pólo:**

**Obs:** *A solução para o exercício proposto deve ser entregue por escrito e em formato digital.*

**Questão Única)**

Na Matemática, um intervalo é um conjunto de números reais com a propriedade que qualquer número pertencente entre 2 números num conjunto também pertence a este conjunto. Por exemplo, o conjunto de todos os números que satisfaçam $0 \leq x \leq 1$ é um intervalo que contém os valores 0, 1 e todos que estão entre estes 2 números. Intervalos são elementos fundamentais na matemática intervalar, uma técnica de computação numérica que garante resultados, mesmo na presença de incertezas e/ou aproximações.

Crie um tipo de dados (classe) para modelar um intervalo de números na reta dos reais. Por exemplo, [-3, 7) representa o intervalo que compreende os valores de -3 a 7, incluindo o -3 e excluindo o 7. Defina as seguintes operações (métodos) sobre os intervalos:

a) *a.contém(v)* retorna verdadeiro se o valor *v* pertence ao intervalo *a*; caso contrário, retorna falso
b) *a.intercepta(b)* retorna verdadeiro se há interseção entre os intervalos *a* e *b*; caso contrário, a operação retorna falso
c) *a.media()* retorna a média dos valores pertencentes ao intervalo
d) na aritmética de intervalos temos a combinação dos seus limites; implemente o método *a.produto(b)* que retorna um novo intervalo *c* com os seguintes limites: [min (infa*infb, infa*supb, supa*infb, supa*supb), max (infa*infb, infa*supb, supa*infb, supa*supb)], onde os prefixos *inf* indica o limite inferior do intervalo e *sup* o superior.
e) *a.uniao(b)* deve representar todos os valores que fazem parte dos intervalos *a* e *b*; se podemos ter intervalos sem interseção, como podemos representar esta união? Adeque a solução apresentada para trabalhar com estes novos tipos de intervalos.

Resposta:

A resposta foi elaborada utilizando 4 classes: a classe Limite, que modela um limite de um intervalo; a classe IntervaloSimples, que modela um intervalo comum, composto de 2 limites simples; a classe Intervalo, que contém uma coleção (*ArrayList*), a qual é importante para modelarmos o conjunto de intervalos disjuntos proposto no item e), e; a classe AD2_2008_2, que contém o método principal (*main*) para iniciação do exemplo.

Obs: Quanto à dúvida da manipulação dos limites, teria sido mais simples se tivéssemos proposto apenas no domínio dos inteiros. Entretanto, valeu por estimular as perguntas, as quais foram bem encaminhadas pela tutoria.

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

/**
 * Classe que implementa um limite, a ser utilizado na definição da
classe
 * intervalo (Exercício da AD2 de Prog III / 2008-2)
 * @author Carlos Bazilio
 */
class Limite {
	private float valor;
	private boolean aberto;

	public Limite(float li, boolean a) {
		valor = li;
		aberto = a;
	}

	public float getValor() {
		return valor;
	}

	public void setValor(float valor) {
		this.valor = valor;
	}

	public boolean isAberto() {
		return aberto;
	}

	public void setAberto(boolean aberto) {
		this.aberto = aberto;
	}

	public boolean equals(Limite l) {
		return (this.valor == l.valor &&
				this.aberto == l.aberto);
	}
}

/**
 * Classe que implementa um intervalo simples
```

```
 * (Exercício da AD2 de Prog III / 2008-2)
 * @author Carlos Bazilio
 */
class IntervaloSimples {
      private Limite limiteInf;
      private Limite limiteSup;

      public IntervaloSimples(float linf, boolean abertolinf, float
lsup, boolean abertolsup) {
            limiteInf = new Limite(linf, abertolinf);
            limiteSup = new Limite(lsup, abertolsup);
      }

      public boolean contem (float v) {
            return ((v > limiteInf.getValor() && v <
limiteSup.getValor()) ||
                        (!limiteInf.isAberto() && v ==
limiteInf.getValor()) ||
                        (!limiteSup.isAberto() && v ==
limiteSup.getValor()));
      }

      public boolean intercepta (IntervaloSimples i) {
            return (this.contem(i.limiteInf.getValor()) ||
this.contem(i.limiteSup.getValor()) ||
                        i.contem(this.limiteInf.getValor()) ||
i.contem(this.limiteSup.getValor()) ||
                        this.limiteInf.equals(i.limiteInf) ||
                        this.limiteSup.equals(i.limiteSup));
      }

      public float media () {
            return ((limiteInf.getValor() + limiteSup.getValor()) / 2);
      }

      public IntervaloSimples produto (IntervaloSimples i) {
            float infProd, supProd;
            infProd =
Math.min(this.limiteInf.getValor()*i.limiteInf.getValor(),

this.limiteInf.getValor()*i.limiteSup.getValor());
            infProd = Math.min(infProd,
this.limiteSup.getValor()*i.limiteInf.getValor());
            infProd = Math.min(infProd,
this.limiteSup.getValor()*i.limiteSup.getValor());

            supProd =
Math.min(this.limiteInf.getValor()*i.limiteInf.getValor(),

this.limiteInf.getValor()*i.limiteSup.getValor());
            supProd = Math.min(supProd,
this.limiteSup.getValor()*i.limiteInf.getValor());
            supProd = Math.min(supProd,
this.limiteSup.getValor()*i.limiteSup.getValor());

            return new IntervaloSimples(infProd, true, supProd, true);
      }
```

```
	public void exibe () {
		char abre = '[', fecha = ']';
		if (limiteInf.isAberto())
			abre = '(';
		if (limiteSup.isAberto())
			fecha = ')';
		System.out.println("Intervalo: " + abre +
limiteInf.getValor() + "," +
                                                limiteSup.getValor() + fecha);
	}
}

/**
 * Classe que implementa um intervalo com possíveis sub-intervalos sem
 * interseção
 * (Exercício da AD2 de Prog III / 2008-2)
 * @author Carlos Bazilio
 */
class Intervalo {
	private List<IntervaloSimples> intervalos;

	public Intervalo() {
		intervalos = new ArrayList<IntervaloSimples>();
	}

	public Intervalo(IntervaloSimples i) {
		intervalos = new ArrayList<IntervaloSimples>();
		intervalos.add(i);
	}

	public void uniao (Intervalo i) {
		intervalos.addAll(i.intervalos);
	}

	public void exibe () {
		Iterator<IntervaloSimples> it = intervalos.iterator();
		while (it.hasNext()) {
			IntervaloSimples interv = it.next();
			interv.exibe();
		}
	}
}

/**
 * Classe de teste da classe intervalo
 * (Exercício da AD2 de Prog III / 2008-2)
 * @author Carlos Bazilio
 */
public class AD2_2008_2 {
	public static void main(String[] args) {
		IntervaloSimples interv_simples = new IntervaloSimples(2,
false, 5, true);
		Intervalo interv = new Intervalo(interv_simples);
		interv.exibe();
	}
}
```