Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP1 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) (4.0 pontos)

Implemente uma classe que represente uma lista simplesmente encadeada usando **VETORES** de elementos inteiros, chamada **`ListaEncadeadaInteirosComVetores`**. Sua lista deve implementar os métodos:

```
void insereNoInicio(int valor),
int removeDoInicio(),
void insereNoFim(int valor) ,
int removeDoFim() e
boolean estaVazia().
```

RESPOSTA:

```
class No{
  //todo no de lista tem um campo de informação
  //e um de próximo elemento.
  int info, prox;

  No(int info){
    this.info = info;
    prox = -1;
  }

  public String toString(){
    return info + "\t";
  }
}
```

```
class ListaEncadeadaInteirosComVetores{
  //O "tamanho" da lista é 10 neste caso.
  private final int TAM_MAX_LISTA = 10;
  //A lista propriamente dita.
  No[] lista;
  //O primeiro indice da lista.
  int prim;

  //Inicialmente, a lista não possui nenhum elemento.
  ListaEncadeadaInteirosComVetores(){
    lista = new No[TAM_MAX_LISTA];
    prim = -1;
  }

  void insereNoInicio(int info){
    int i;
    //Acha a primeira posição livre do vetor.
    for(i = 0; i < lista.length; i++) if(lista[i] == null) break;
    //Se encontrou uma posição livre, faça prim apontar para esta
    //posição e o campo prox deve ser atualizado.
    if(i < lista.length){
      lista[i] = new No(info);
      lista[i].prox = prim;
      prim = i;
    }
  }

  int removeDoInicio(){
    //Se a lista está vazia, retorne uma constante que significa erro.
    if(prim == -1) return Integer.valueOf(Integer.MIN_VALUE);
    //Senão, retire o primeiro elemento da lista, atualizando prim.
    int resp = lista[prim].info, ind = prim;
    prim = lista[prim].prox;
    lista[ind] = null;
    return resp;
  }

  void insereNoFim(int info){
    int i;
    //Acha a primeira posição livre do vetor.
    for(i = 0; i < lista.length; i++) if(lista[i] == null) break;
    //Senão, insira na última posição da lista.
    if(i < lista.length){
      lista[i] = new No(info);
      int q = prim, ant = prim;
      //Procure a última posição da lista marcada com ant.
      while(q != -1){
        ant = q;
        q = lista[q].prox;
      }
      //Se a lista está vazia, a inserção será na posição zero.
      if(ant == -1) prim = i;
      //Senão, a inserção deve ser na posição correta.
      else lista[ant].prox = i;
    }
  }
```

```
  int removeDoFim(){
    //Se a lista está vazia, retorne uma constante que significa erro.
    if(prim == -1) return Integer.valueOf(Integer.MIN_VALUE);
    int q = prim, ant = prim;
    //Senão, retire o último elemento da lista.
    //Ache a posição do último elemento.
    while(lista[q].prox != -1){
      ant = q;
      q = lista[q].prox;
    }
    //Se só tem um elemento, retire-o da lista e atualiza prim.
    int resp = lista[ant].info;
    if(ant == prim){
      prim = -1;
      lista[ant] = null;
    }
    //Senão, faça o penúltimo ser o último elemento da lista.
    else{
      int ind = lista[ant].prox;
      lista[ant].prox = -1;
      lista[ind] = null;
    }
    return resp;
  }

  //A lista está vazia se prim está com o valor igual a -1.
  boolean estaVazia(){
    return prim == -1;
  }

  //A impressão usa toString de nó. A ideia é passar por cada
  //elemento imprimindo a informação.
  public String toString(){
    String resp = "";
    int ind = prim;
    while(ind != -1){
      resp += lista[ind].toString();
      ind = lista[ind].prox;
    }
    return resp;
  }
}
```

Questão 2) (3.0 pontos)

Defina uma classe *IntervaloDeTempo,* cujos objetos representam um intervalo de tempo em número de horas, minutos e segundos. Ou seja, existem campos na classe para cada um desses valores. O construtor de objetos dessa classe deve receber como argumento um número inteiro positivo, representando o número de segundos decorridos desde o instante inicial 00:00:00 horas, e retornar um objeto da classe *IntervaloDeTempo* correspondente. Por exemplo, a expressão *new IntervaloDeTempo(3500)* deve retornar um objeto que represente 0 horas, 58 minutos e 20 segundos. Além do construtor citado, defina para a mesma classe:

a) 2 métodos de soma, sendo que um recebe como parâmetro o tempo a ser somado em segundos, enquanto que o outro recebe um objeto IntervaloDeTempo;
b) 1 método chamado *toString()*, que à partir do conteúdo do objeto, retorne uma string com o formato dado acima: "x horas, y minutos e z segundos".

Evite, sempre que possível, a replicação de código.

RESPOSTA:

```
class IntervaloDeTempo {
	// Não coloquei os modificadores de acesso
	//(private ou protected) apenas para tornar o
	//código menor. Mas estes são recomendáveis
	//na criação de aplicações reais.
	int hora, min, seg;

	// Construtor que converte segundos em h,m,s
	public IntervaloDeTempo(int tempo) {
		hora = tempo / 3600;
		min = (tempo - hora*3600) / 60;
		seg = tempo - hora*3600 - min*60;
	}

	// Método que realiza some de objetos
	// Chamada do método simplifica para o caso em que a
	//a soma de segundos ou minutos exceder 60 unidades
	public IntervaloDeTempo soma (IntervaloDeTempo t) {
		this.hora = this.hora + t.hora;
		this.min = this.min + t.min;
		this.seg = this.seg + t.seg;
		this.simplifica();
		return this;
	}

	// Método que adiciona unidades de segundo ao objeto
	public IntervaloDeTempo soma (int t) {
		IntervaloDeTempo aux = new IntervaloDeTempo(t);
		aux.soma(this);
		return aux;
	}

	// Método que simplifica o objeto. Após uma soma, caso
	// o campo segundos seja maior que 60, cada 60 unidades destas
	// s# o transformadas em minutos.
	// Este método não era pedido na questão.
	public void simplifica () {
		if (this.seg >= 60) {
			this.seg = this.seg % 60;
			this.min = this.min + this.seg/60;
		}

		if (this.min >= 60) {
			this.min = this.min % 60;
			this.hora = this.hora + this.min/60;
		}
	}
```

```
	// Método que retorna uma string do objeto
	public String toString () {
		String saida = hora + " horas, " + min + " minutos e " + seg + "
segundos";
		return saida;
	}
}

// Classe de teste, apenas para verificar a corretude
//do código acima. Não exigida na questão!
public class AP1_2014_2_Q2 {
	public static void main(String[] args) {
		IntervaloDeTempo t = new IntervaloDeTempo(3500);
		System.out.println(t.toString());
		System.out.println(t.soma(100).toString());
		System.out.println(t.toString());
	}
}
```

Questão 3) (3.0 pontos)

Suponha a classe Livro definida abaixo, a qual será utilizada num sistema que manipula informações de publicações:

```
public class Livro {
	int isbn;
	String titulo;
	String autor;
	String editora;
	java.util.GregorianCalendar datapublicacao;
	float preco_compra;
}
```

(Obs.: A classe `java.util.GregorianCalendar` é utilizada para a manipulação de datas em Java)

Além de livros, o sistema deve armazenar também informações de revistas (título, isbn, mês, ano e preço de compra), gibis (isbn, título, edição e preço de compra) e cds e dvds (issn, titulo, tamanho e preço de compra).

a) Crie novas classes de forma que estas informações possam ser manipuladas.
b) Implemente construtores que inicializem todos os atributos das classes.
c) Crie um objeto do tipo revista no método main e imprima na tela o preço deste objeto revista criado.

Reutilize construções, utilizando os mecanismos de O.O. em Java, sempre que possível. A classe fornecida pode ser modificada.

RESPOSTA:

```java
import java.util.GregorianCalendar;

// Como há vários atributos comuns entre as publicações
//existentes, foram declaradas superclasses para conter o
//que é comum entre as subclasses.
// Supondo que preco é um campo chave, foi definido como
//privado. Entretanto, isso não era exigido na questão.
class Publicacao {
	String titulo;
	private float preco;

	public Publicacao(String titulo, float preco) {
		this.titulo = titulo;
		this.preco = preco;
	}

	public float retornaPreco() {
		return preco;
	}
}

// Superclasse comum a Livro, Revista e Gibis
class PubLivro extends Publicacao {
	String isbn;

	public PubLivro(String titulo, float preco, String isbn) {
		super(titulo, preco);
		this.isbn = isbn;
	}
}

class Livro2 extends PubLivro {
	String autor;
	String editora;
	java.util.GregorianCalendar datapublicacao;

	public Livro2(String titulo, float preco, String isbn, String autor,
			String editora, GregorianCalendar datapublicacao) {
		super(titulo, preco, isbn);
		this.autor = autor;
		this.editora = editora;
		this.datapublicacao = datapublicacao;
	}
}

class Revista extends PubLivro {
	int mes, ano;

	public Revista(String titulo, float preco, String isbn, int mes, int
ano) {
		super(titulo, preco, isbn);
		this.mes = mes;
		this.ano = ano;
	}
}

class Gibi extends PubLivro {
```

```
	String edicao;

	public Gibi(String titulo, float preco, String isbn, String edicao) {
		super(titulo, preco, isbn);
		this.edicao = edicao;
	}
}

class CdDvd extends Publicacao {
	String issn;
	int tamanho;

	public CdDvd(String titulo, float preco, String issn, int tamanho) {
		super(titulo, preco);
		this.issn = issn;
		this.tamanho = tamanho;
	}
}

// Classe principal que exemplifica o uso das classes declaradas
public class AP1_2014_2_Q3 {
	public static void main(String[] args) {
		Revista revista = new Revista("Auto Esporte", 10.0f, "0001", 9,
2014);
		System.out.println("Preço da revista: " +
revista.retornaPreco());
	}
}
```