Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP1 1° semestre de 2019.

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) (3.0 pontos)

Considere que foi contratado por uma aceleradora de startups (empresas emergentes que desenvolvem negócios escaláveis, em condição de grande incerteza) para criar um banco de dados destas empresas. Deve-se armazenar o nome da empresa, sua área de atuação (tecnologia, social, saúde, finanças, educação, etc), a quantidade de investimentos já feitos nesta, a quantidade de receita até o momento, assim como suas despesas. O objetivo deste banco é a geração de relatório consolidado. Ou seja, para cada empresa basta ter na base 1 único valor para cada das quantidades citadas (investimento, receita e despesa). Para tal:

a) Crie a classe que representa uma startup. Além de tudo que já foi descrito para criação desta classe, crie 2 métodos: i) um método chamado *obtemVolume()*, o qual retorna a quantidade de recurso movimentado pela startup (soma dos investimentos, receitas e despesas); ii) sobrescreva o método *toString()*, herdado da classe Object, retornando todos os atributos da classe.
b) Crie uma classe que representa a aceleradora. Esta se constituirá de um conjunto de startups. Para tal, declare um **vetor** para armazenar este conjunto. Além do próprio vetor, qualquer outro recurso utilizado para manipular este deve ser mantido interno a classe, não podendo ser acessado externamente. Os recursos que podem ser acessados externamente são os seguintes métodos: i) um construtor para a criação de objetos desta classe; ii) um método para adição de empresas; iii) Um método para obtenção do total de recursos da aceleradora (soma dos recursos

movimentados por cada empresa); iv) redefinição do método *toString()* para listagem das empresas com seus respectivos atributos.

c) Crie objetos para as seguintes startups (tabela abaixo) num método *main()* para verificar se os métodos da classe aceleradora estão funcionando corretamente.

Empresa	Área	Investimentos	Receitas	Despesas
99	Transporte	300	500	100
Nubank	Finanças	400	500	200
CargoX	Transporte	200	400	150
Movile	Tecnologia	500	300	150

RESPOSTA:

```
// Item a
class Startup {
	String nome;
	String area; // tecnologia, social, saúde, finanças, educação
	double investimentos;
	double receitas;
	double despesas;

	public Startup(String n, String a, double i, double r, double d) {
		this.nome = n; this.area = a; this.investimentos = i;
		this.receitas = r; this.despesas = d;
	}

	public double obtemInvestimentos () {
		return investimentos;
	}

	public double obtemDespesas () {
		return despesas;
	}

	public double obtemReceitas () {
		return receitas;
	}

	public double situacao () {
		return obtemInvestimentos() + obtemReceitas() - obtemDespesas();
	}

	public double obtemVolume() {
		return obtemInvestimentos() + obtemReceitas() + obtemDespesas();
	}

	public String toString() {
		return "Nome: " + this.nome + "\n" +
		"Área: " + this.area + "\n" +
		"Investimentos: " + obtemInvestimentos() + "\n" +
		"Receitas: " + obtemReceitas() + "\n" +
		"Despesas: " + obtemDespesas() + "\n";
	}
}

// Item b
class Aceleradora {
	private Startup empresas[];
	private int qtdEmpresas;
```

```
	public Aceleradora (int q) {
		empresas = new Startup[q];
		qtdEmpresas = 0;
	}

	public void adicionaEmpresa(Startup s) {
		empresas[qtdEmpresas] = s;
		qtdEmpresas++;
	}

	public double volumeRecursos () {
		double volume = 0;
		for (int i = 0; i < qtdEmpresas; i++) {
			volume += empresas[i].obtemVolume();
		}
		return volume;
	}

	public String toString() {
		String saida = "";
		for (int i = 0; i < qtdEmpresas; i++) {
			saida += empresas[i].toString();
		}
		return saida;
	}
}

public class AP1_2019_1_Q1 {
	public static void main(String[] args) {
		// Item c
		Startup e99 = new Startup("99", "transporte", 300, 500, 100);
		Startup nubank = new Startup("Nubank", "finanças", 400, 500, 200);
		Startup cargox = new Startup("CargoX", "transporte", 200, 400, 150);
		Startup movile = new Startup("Movile", "tecnologia", 500, 300, 150);

		Aceleradora acel = new Aceleradora(10);
		acel.adicionaEmpresa(e99);
		acel.adicionaEmpresa(nubank);
		acel.adicionaEmpresa(cargox);
		acel.adicionaEmpresa(movile);

		System.out.println("Volume de recursos da aceleradora: " + acel.volumeRecursos());

		System.out.println("Relatório da aceleradora:\n" + acel);
	}
}
```

Questão 2) (3.0 pontos)

Implemente uma classe que representa uma única tarefa, a ser utilizada por uma aplicação de lista de tarefas. Uma tarefa tem um título, tem uma informação que indica se está feita ou não, uma data limite para realização (opcional, ou seja, a data pode ser indefinida), e pode ocorrer de forma recorrente, semanal ou mensalmente (acontece toda semana ou todo mês). Além destas informações, devem ser definidas operações para realizar uma tarefa, ou desfazê-la (passar do estado de feita para não feita). Na implementação destas operações, altere os atributos informados de maneira a refletir estas operações. Sugestão: Para modelar datas, utilize a classe *LocalDate*, do pacote *java.time*. Esta classe possui métodos úteis para a manipulação de datas, como os métodos *plusDays(int)*, *plusWeeks(int)*, *plusMonths(int)* e *plusYears(int)*, para adição de dias, semanas, meses e anos, respectivamente.

RESPOSTA:

```
import java.time.LocalDate;

class ToDo {
	String titulo;
	boolean feita;
	LocalDate dataLimite;
	boolean recorrente;
	String razao; // semanal ou mensal

	public ToDo(String titulo, LocalDate dataLimite, boolean recorrente) {
		this.titulo = titulo;
		this.feita = false;
		this.dataLimite = dataLimite;
		this.recorrente = recorrente;
	}

	public ToDo(String titulo) {
		this(titulo, null, false);
	}

	public void feito () {
		if (! recorrente)
			this.feita = true;
		else {
			if (razao == "semanal")
				dataLimite.plusWeeks(1);
			else
				dataLimite.plusMonths(1);
		}
	}

	public void desfeito () {
		this.feita = false;
	}
}
```

Questão 3) (4.0 pontos)

Escreva um programa que receba dois números, maiores que zero, como parâmetros de entrada, e retorne se eles são palíndromos (um número palíndromo é aquele que é igual quando lido nos dois sentidos). Também retorne o mdc (máximo divisor comum) entre eles. O máximo divisor comum entre dois ou mais números naturais é o maior de seus divisores. Dois números naturais sempre tem divisores em comum. Seguem alguns exemplos de entrada e saída deste programa:

Entrada	**Saída**
java Q3 121 11	**121: s** **11: s** **mdc(121,11) = 11**
java Q3 22 20	**22: s** **20: n** **mdc(22,20) = 2**
java Q3 5 131	**5: s** **131: s** **mdc(5,131) = 1**
java Q3 1000 200	**1000: n** **200: n**

	mdc(1000,200) = 200

RESPOSTA:

```
public class Q3{
 public static void main(String[] args){
  int n1 = Integer.parseInt(args[0]), n2 = Integer.parseInt(args[1]);
  if((n1 <= 0)||(n2 <= 0)||(args[0].charAt(0) == '0')||(args[1].charAt(0)
== '0')) return;

  if(ePalindromo(args[0])) System.out.println(args[0] + ": s");
  else System.out.println(args[0] + ": n");

  if(ePalindromo(args[1])) System.out.println(args[1] + ": s");
  eles System.out.println(args[1] + ": n");

  System.out.println("mdc(" + n1 + "," + n2 + ") = " + mdc(n1,n2));
 }

 public static boolean ePalindromo(String str){
   int n = str.length(), metade = str.length()/2, i;
   for(i = 0; i <= metade; i++)
    if(str.charAt(i) != str.charAt(n - 1 - i)) return false;
   return true;
 }

 public static int mdc(int n1, int n2){
   while(n1 != n2)
     if(n1 > n2) n1 -= n2;
     else n2 -= n1;
   return n1;
 }
}
```