Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP2 2° 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)

Alguns números possuem uma propriedade interessante: se você recuperar seus dois primeiros dígitos e seus dois últimos dígitos e elevar ao quadrado a soma deles, você obterá a concatenação desses quatro dígitos. Por exemplo, o número 203125 possui essa propriedade, pois $(20 + 25)^2 = 2025$. Por outro lado, o mesmo não é observado para 20326, pois $(20 + 26)^2 = 2116 \neq 2026$. Escreva um programa em Java que receba o nome de um arquivo como argumento, que contém números, um por linha, e retorne na console, para cada número contido no arquivo, **`true`** se o número satisfaz a essa propriedade, e **`false`**, caso contrário.

RESPOSTA:

```
import java.io.*;
import java.util.*;

public class Q1{
  public static void main(String[] args) throws IOException{
    BufferedReader in;
    in = new BufferedReader(new FileReader(args[0]));
    try {
      String s = in.readLine();
```

```
      while(s != null){
        System.out.println(teste(Integer.parseInt(s)));
        s = in.readLine();
      }
      in.close();
    }
    catch (Exception e){
      System.out.println("Excecao\n");
    }
  }

  public static boolean teste (int n){
    int pot = 1;
    while (pot < n) pot *= 10;
    pot /= 100;
    int num1 = n / pot, num2 = n % 100;
    int x = (num1 + num2) * (num1 + num2), y = num1 * 100 + num2;
    if(x == y) return true;
    return false;
  }
}
```

Questão 2) (3.0 pontos)

Dado um arquivo, passado como argumento, contendo um conjunto de datas de nascimento (uma data por linha), **`dd mm aaaa`**, isto é, TRÊS inteiros seguidos, construir um outro arquivo, cujo o nome é **`out-<nome do arquivo de entrada>`** , contendo as idades ordenadas em ordem crescente, isto é, a pessoa mais nova aparecerá primeiro. Por exemplo, se o arquivo de entrada for:

```
4
2 1 1999
1 1 1999
1 2 1999
1 1 2000
```

A saída, para esse exemplo será:

```
1 1 2000
1 2 1999
2 1 1999
1 1 1999
```

Seu programa deve funcionar para qualquer arquivo de entrada. Se não, sua resposta será severamente descontada. Seu programa só pode ler o arquivo uma única vez.

RESPOSTA:

```
import java.io.*;
import java.util.*;

class Data{
  int dd, mm, aaaa;
```

```
  Data(int d, int m, int a){
    dd = d;
    mm = m;
    aaaa = a;
  }

  public String toString(){
    return dd + " " + mm + " " + aaaa + "\n";
  }
}

public class Q2{
  public static void main(String[] args) throws IOException{
    BufferedReader in;
    in = new BufferedReader(new FileReader(args[0]));
    try {
      int n = Integer.parseInt(in.readLine());
      if(n < 0){
        in.close();
        return;
      }
      Data vet[] = new Data[n];
      String s = in.readLine();
      int i = 0;
      while(s != null){
          String vs[] = s.split(" ");
          vet[i++]     =     new     Data(Integer.parseInt(vs[0]),
Integer.parseInt(vs[1]), Integer.parseInt(vs[2]));
          s = in.readLine();
      }
      in.close();
      String out = "out-" + args[0];
      Ordena(vet, out);
    }
    catch (Exception e){
      System.out.println("Excecao\n");
    }
  }

    static  void  Ordena(Data  vet[],  String  nome_arq)  throws
IOException{
    int i;
    for(i = 0; i < vet.length; i++){
      int j, menor = i;

      for(j = i + 1; j < vet.length; j++)
       if((vet[menor].aaaa < vet[j].aaaa) || ((vet[menor].aaaa ==
vet[j].aaaa) && (vet[menor].mm < vet[j].mm)) || ((vet[menor].aaaa
== vet[j].aaaa) && (vet[menor].mm == vet[j].mm) && (vet[menor].dd
< vet[j].dd))) menor = j;
```

```
        if(menor != i){
          int temp = vet[menor].aaaa;
          vet[menor].aaaa = vet[i].aaaa;
          vet[i].aaaa = temp;
          temp = vet[menor].mm;
          vet[menor].mm = vet[i].mm;
          vet[i].mm = temp;
          temp = vet[menor].dd;
          vet[menor].dd = vet[i].dd;
          vet[i].dd = temp;
        }
      }

      BufferedWriter out;
      out = new BufferedWriter(new FileWriter(nome_arq));
      try {
        for(i = 0; i < vet.length; i++)
          out.write(vet[i].toString());
        out.close();
      }
      catch (Exception e){
        System.out.println("Excecao\n");
      }
    }
}
```

Questão 3) (4.0 pontos)

Considere o programa incompleto abaixo que gerencia notas, as quais podem ser textos simples ou cartões de visita (métodos get/set foram omitidos, apesar de serem uma boa prática, para encurtar o código):

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

interface Nota {
        String exibe ();
}
class NotaTexto implements Nota {
        String conteudo;
        public NotaTexto (String c) {
                this.conteudo = c;
        }
        public String exibe() {
                return ">> " + this.conteudo + "\n";
        }
}
class BlocoNotas {
        List<Nota> notas;
        public BlocoNotas () {
                notas = new ArrayList<Nota>();
        }
        public void adiciona(Nota n) {
                notas.add(n);
        }
        public void imprimeNotas() { /* Implementar !! */ }
        public BlocoNotas buscaNotas(String termo) { /* Implementar !! */ }
```

```
}
public class AP2_2019_2_Q3 {
	public static void main(String[] args) {
		NotaTexto nota = new NotaTexto("Estudar Herança, Polimorfismo, ...");
		Cartao cartao = new Cartao("Ouvidoria", "Cederj", "21-2334-1583");
		BlocoNotas anotacoes = new BlocoNotas();
		anotacoes.adiciona(nota);
		anotacoes.adiciona(cartao);
		anotacoes.buscaNotas("Herança").imprimeNotas();
	}
}
```

a) (1.5 pto) Implemente a classe Cartao (cartão de visita), a qual é um tipo de Nota (pode ser deduzido do código acima - 2ª e 5ª linhas do método main()). Um cartão possui um nome, a empresa e um telefone para contato.
b) (1.0 pto) Implemente o método imprimeNotas(), da classe BlocoNotas.
c) (1.5 pto) Implemente o método buscaNotas(). Este retorna um bloco de notas novo, o qual conterá notas que contenham o termo passado por parâmetro (Use o método contains() da classe String aplicado ao retorno do método exibe()).

RESPOSTA:

a)
```
class Cartao implements Nota {
	String nome;
	String empresa;
	String telefone;
	public Cartao (String n, String e, String tel) {
		this.nome = n; this.empresa = e; this.telefone = tel;
	}
	public String exibe() {
		return ">> " + this.empresa + "\n" + this.telefone + "\n";
	}
}
```

b)
```
	public void imprimeNotas() {
		for (Nota n : notas) {
			System.out.println(n.exibe());
		}
	}
```

c)
```
	public BlocoNotas buscaNotas (String termo) {
		BlocoNotas resultado = new BlocoNotas();
		for (Nota n : notas) {
			if (n.exibe().contains(termo))
				resultado.adiciona(n);
		}
		return resultado;
	}
```