UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
DEPARTAMENTO DE COMPUTAÇÃO

IC805 – Linguagem de Programação 3

TCE1 - Trabalho Colaborativo Escrito 1
18/05/2023

Prof. Claver Pari Soto

Data de entrega: 30/05/2023 na hora da aula, impresso.

Muito importante: Para todos os problemas, apresente o **código** solicitado e também o **relatório detalhado** de como conseguiram resolver os problemas, resaltando o que foi aprendido no processo da solução. Esse relatório não pode ser compartilhado entre os grupos. Alem de colocar o código no relatório impresso, esses códigos deverão ser enviados ao e-mail do professor.

Problema 1:

Projete uma classe **`Navio`** que tenha os seguintes membros:

- Um campo para o nome do navio (uma string)
- Um campo para o ano em que o navio foi construído (uma string)
- Um construtor e acessadores (getters) e mutadores (setters) apropriados
- Um método **`toString`** que exiba o nome do navio e o ano em que foi construído.

Projete uma classe **`Cruzeiro`** que estenda a classe **`Navio`**. A classe **`Cruzeiro`** deve ter os seguintes membros:

- Um campo para o número máximo de passageiros (um int)
- Um construtor e acessadores e mutadores apropriados
- Um método **`toString`** que sobrescreva o método **`toString`** da classe mãe. O método **`toString`** da classe **`Cruzeiro`** deve exibir apenas o nome do navio e o número máximo de passageiros.

Projete uma classe **`Cargueiro`** que estenda a classe **`Navio`**. A classe **`Cargueiro`** deve ter os seguintes membros:

- Um campo para a capacidade de carga em toneladas (um int)
- Um construtor e acessadores e mutadores apropriados
- Um método **`toString`** que sobrescreva o método **`toString`** da classe mãe. O método **`toString`** da classe **`Cargueiro`** deve exibir apenas o nome do navio e a capacidade de carga do navio.

Use essas classes em uma classe cliente/teste que tenha um array de elementos de tipo **`Navio`**. Atribua alguns objetos **`Navio`**, **`Cruzeiro`** e **`Cargueiro`** aos elementos do array. O programa deve então percorrer o array, chamando o método **`toString`** de cada objeto.

Problema 2:

Analise e estude todos os detalhes das seguintes classes.

1. Faça a representação das classes usando diagramas UML
2. Descreva com detalhes tudo o que consiga em relação aos objetos de tipo Candidato. Como são criados? como são organizados/estocados?
3. Selecione o segmento de código onde é reconhecido o possível empate dos candidatos mais votados. Explique com detalhes.
4. Acrescente código onde seja conveniente, para encontrar o(s) candidatos com menor número de votos.

```
package eleicoes;
class Candidato {
    String nome;
    int numVotos;

    Candidato(String nome, int numVotos) {
       this.nome = nome;
       this.numVotos = numVotos;
    }
}
package eleicoes;
class ContagemVotos {
    int validos;
    int invalidos;

    ContagemVotos(int validos, int invalidos) {
       this.validos = validos;
       this.invalidos = invalidos;
    }
}
```

```java
package eleicoes;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class VotacaoTeste {
    final static int MaxCandidatos = 7;

    public static void main(String[] args) throws IOException {
       Scanner entrada = new Scanner(new FileReader("votos.txt"));
       PrintWriter saida = new PrintWriter(new FileWriter("resultados.txt"));
       Candidato[] candidato = new Candidato[MaxCandidatos+1];

       for (int c = 1; c <= MaxCandidatos; c++)
          candidato[c] = new Candidato(entrada.nextLine(), 0);

       ContagemVotos contagem = processarVotos(candidato, MaxCandidatos, entrada, saida);
       imprimirResultados(saida, candidato, MaxCandidatos, contagem);
       entrada.close();
       saida.close();
    }

    public static ContagemVotos processarVotos(Candidato[] lista, int max, Scanner entrada, PrintWriter saida) {
       ContagemVotos votos = new ContagemVotos(0, 0);
       int voto = entrada.nextInt();
       while (voto != 0) {
          if (voto < 1 || voto > max) {
             saida.printf("Voto invalido: %d\n", voto);
             ++votos.invalidos;
          }
          else {
             ++lista[voto].numVotos;
             ++votos.validos;
          }
          voto = entrada.nextInt();
       }
       return votos;
    }

    public static void imprimirResultados(PrintWriter out, Candidato[] lista, int max, ContagemVotos votos) {
       out.printf("\nNumero de votantes: %d\n", votos.validos + votos.invalidos);
       out.printf("Numero de votos validos: %d\n", votos.validos);
       out.printf("Numero de votos invalidos: %d\n", votos.invalidos);
       out.printf("\nCandidato          Num de votos\n\n");

       for (int h = 1; h <= MaxCandidatos; h++)
          out.printf("%-18s %3d\n", lista[h].nome, lista[h].numVotos);

       out.printf("\nCandidato(s) eleitos(s)\n");
       int ganhador = encontrarMaisVotado(lista, 1, MaxCandidatos);
       int votosNoGanhador = lista[ganhador].numVotos;
       for (int candidato = 1; candidato <= MaxCandidatos; candidato++)
          if (lista[candidato].numVotos == votosNoGanhador)
                out.printf("   %s\n", lista[candidato].nome);
    }

    public static int encontrarMaisVotado(Candidato[] lista, int pri, int ult) {
       int maisVotado = pri;
       for (int candidato = pri + 1; candidato <= ult; candidato++)
          if (lista[candidato].numVotos > lista[maisVotado].numVotos)
                maisVotado = candidato;
       return maisVotado;
    }
}
```

Problema 3:

Projete uma classe **`BalaDeCanhao`** para modelar uma bala de canhão que é disparada para o ar. Sábe-se que uma bala tem

- Uma posição x e uma posição y.
- Uma velocidade em x e uma velocidade em y.

Forneça os seguintes construtores:

- Um construtor padrão
- Um construtor com uma posição x e a posição y inicialmente em 0.

Forneça os seguintes métodos:

- Um método **`mover(double deltaSegundos)`** que move a bola para a próxima posição. Primeiro calcule a distância percorrida em **`deltaSegundos`** segundos, usando as velocidades atuais, logo atualize as posições x e y; em seguida atualize a velocidade y levando em consideração a aceleração gravitacional de -9,81 m/s2; a velocidade x permanece inalterada.
- Um método **`Ponto getLocalizacao()`** que obtém a localização atual da bala de canhão, arredondada para coordenadas inteiras.
- Um método **`ArrayList<Ponto> atirar(double alpha, double v, double deltaSegundos)`** cujos argumentos são o ângulo **`alpha`** e a velocidade inicial **`v`**. (Calcule a velocidade em x como v cos(alpha) e a velocidade em y como v sen(alpha); então itere chamando o método **`mover`** com o intervalo de tempo dado até que a posição y seja 0. Retorne um ArrayList de localizações após cada chamada para mover.)

Use esta classe em um programa que solicita ao usuário o ângulo inicial e a velocidade inicial. Em seguida, chame o método atirar e imprima as localizações.

Problema 4:

Pense no objeto figura geométrica retângulo. Um retângulo é uma figura geométrica que para ser completamente definida precisa de duas únicas medidas: o comprimento e a largura. Depois de definidas essas características, o retângulo pode nos fornecer seu perímetro e sua área.

1. Projete em Java, uma classe que serva para criar objetos de tipo **`Retangulo`**. A classe deve proteger as variáveis de instância, usando o encapsulamento. A classe deve implementar os métodos para o cálculo do perímetro e da área do objeto **`Retangulo`** em questão, além dos getters e setters.
2. Crie outra classe **`TesteRetangulo`** para instanciar dois objetos tipo **`Retangulo`**. Este teste deve permitir "*settar*" (definir, estabelecer) o comprimento e largura de cada objeto **`Retangulo`**, e fornecer os valores do perímetro e da área dos objetos instanciados.