PO24CP - Programação Orientada a Objetos
Profa. Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Trabalho #01 - Jogo General[1]

Data de entrega: **31/10/2018 (até 23h55)**, via moodle.

O trabalho poderá ser individual ou em dupla.

Serão descontados **2 pontos por dia de atraso**, com data limite até 04/11 às 23h55 (via moodle).

Em caso de **cópia** de código, **os alunos envolvidos** terão nota igual a **zero** no trabalho 1 e em todas as outras avaliações da disciplina, ou seja, P1, T2, P2 e APS's.

General é um jogo de dados para dois ou mais jogadores. Para jogar General são necessários cinco dados comuns (hexaédricos) e uma cartela de marcação. O objetivo do jogo é marcar o maior número de pontos, através de algumas combinações de resultados nos dados.

Para este trabalho, as regras do jogo General serão simplificadas e o campeonato poderá ser realizado desde que exista ao menos um jogador, por exemplo. O jogo consistirá de um certo número de rodadas: em cada uma delas, cada jogador, por sua vez, joga os dados e, conforme o resultado obtido, marca a jogada prevista em sua cartela. Uma vez marcada, aquela jogada não pode ser repetida pelo mesmo jogador até o final da partida.

Regras básicas:

(1) Sendo 13 o número de jogadas possíveis e 13 o número de linhas de cada coluna na cartela de marcação (Fig.1), um jogo consiste de 13 rodadas, ou 13 jogadas para cada jogador.

(2) Cada jogador, em sua vez, tem apenas uma chance de arremessar os dados.

(3) O resultado obtido ao final do arremesso deve ser classificado, pelo próprio jogador, como uma das seguintes 13 possibilidades:

Jogada de 1: um certo número de dados (de 0 a 5) marcando o número 1; sendo que a jogada vale mais pontos conforme a quantidade de dados que marcarem o número 1. Por exemplo: 1-1-1-4-5 vale 3 pontos.

Jogadas de 2, 3, 4, 5 e 6: correspondentes à jogada de 1 para os demais números. Por exemplo: 3-3-4-4-5 vale 6 pontos se for considerada uma jogada de 3; ou 8 pontos se for considerada uma jogada de 4; ou ainda 5 pontos se for uma jogada de 5.

Trinca (T): três dados marcando o mesmo número. Vale a soma dos 5 dados. Exemplo: 4-4-4-5-6 vale 23 pontos.

Quadra (Q): quatro dados marcando o mesmo número. Vale a soma dos 5 dados. Exemplo: 1-5-5-5-5 vale 21 pontos.

Full-hand (F) ou Full-house: uma trinca e um par (exemplo: 2-2-2-6-6). Vale 25 pontos para qualquer combinação.

[1] `https://pt.wikipedia.org/wiki/General_(jogo)`

	Jogador 1	Jogador 2	Jogador 3
1			
2			
3			
4			
5			
6			
T			
Q			
F			
S+			
S-			
G			
X			
Total			

Figura 1: Cartela de marcação

Seqüência alta (S+): 2-3-4-5-6. Vale 30 pontos.

Seqüência baixa (S-): 1-2-3-4-5. Vale 40 pontos.

General (G): cinco dados marcando o mesmo número (por exemplo: 4-4-4-4-4). Vale 50 pontos.

Jogada aleatória (X) : qualquer combinação. Vale a soma dos 5 dados. Por exemplo: 1-4-4-5-6 vale 20 pontos.

(4) O resultado é mostrado na forma de cartela, na coluna do jogador e na linha correspondente à jogada. Aquela linha (e portanto aquela jogada) não poderá mais ser utilizada pelo jogador na mesma partida.

(5) Se um determinado resultado não cumprir os requisitos para a jogada escolhida, o jogador zera a respectiva jogada. E ainda, se um determinado resultado não puder ser classificado como nenhuma das jogadas ainda restantes para aquele jogador, ele deverá escolher qual das jogadas restantes será descartada, marcando 0 (zero) para a jogada correspondente.

(6) Ao final de 13 rodadas, com a cartela toda preenchida, somam-se os valores de cada coluna, e o jogador que obtiver mais pontos será considerado o vencedor.

Com base no detalhamento anterior, faça:

1. Descreva o diagrama UML das classes do simulador tomando como modelo o esboço apresentado na Figura 2 (gerar o arquivo pdf do diagrama).

2. Com base no diagrama UML (Fig. 2), desenvolva um aplicativo Java com um menu iterativo que permita ao usuário simular um Campeonato de jogo General com no máximo cinco jogadores:

   (a) Incluir jogador

   (b) Remover jogador

   (c) Iniciar/reiniciar o campeonato
   >essa opção compreende, para cada jogador:

- rolar os dados
- mostrar os valores dos dados obtidos
- dar a opção de escolher a jogada que deseja marcar

(d) Mostrar a cartela de resultados

(e) Gravar os dados do campeonato em arquivo

(f) Ler os dados do campeonato em arquivo

(g) Sair da aplicação

Exemplo de um esboço de execução da aplicação:

```
entre com a opção do menu: a
Nome d@ jogador@: Luciene
entre com a opção do menu: a
Nome d@ jogador@: Maria
entre com a opção do menu: c
rolando dados para Luciene...
valores obtidos: 4-4-5-6-6
para qual jogada deseja marcar: [1 - 13] Luciene?
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - -  -    -    -     -      -      -     -
6
rolando dados para Maria...
valores obtidos: 1-1-1-1-1
para qual jogada deseja marcar: [1 - 13] Maria?
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - -  -    -    -     -      -      -     -
12
rolando dados para Luciene...
valores obtidos: 1-5-5-5-3
para qual jogada deseja marcar: [1 - 13] Luciene?
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - x  -    -    -     -      -      -     -
8
seus valores não cumprem o requisito para esta jogada!
rolando dados para Maria...
valores obtidos: 1-2-3-4-5
para qual jogada deseja marcar: [1 - 13] Maria?
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - -  -    -    -     -      -      x     -
11
... após finalizar todas as rodadas para tod@s @s jogador@s:
d
-- Cartela de Resultados --
        Luciene     Maria
1
2
3
4
5
```

```
6          12
7(T)
8(Q)       0
9(F)
10(S+)
11(S-)                  40
12(G)                   50
13(X)
---------------------------
Total      12           90
```

Esboço do diagrama UML a ser seguido:

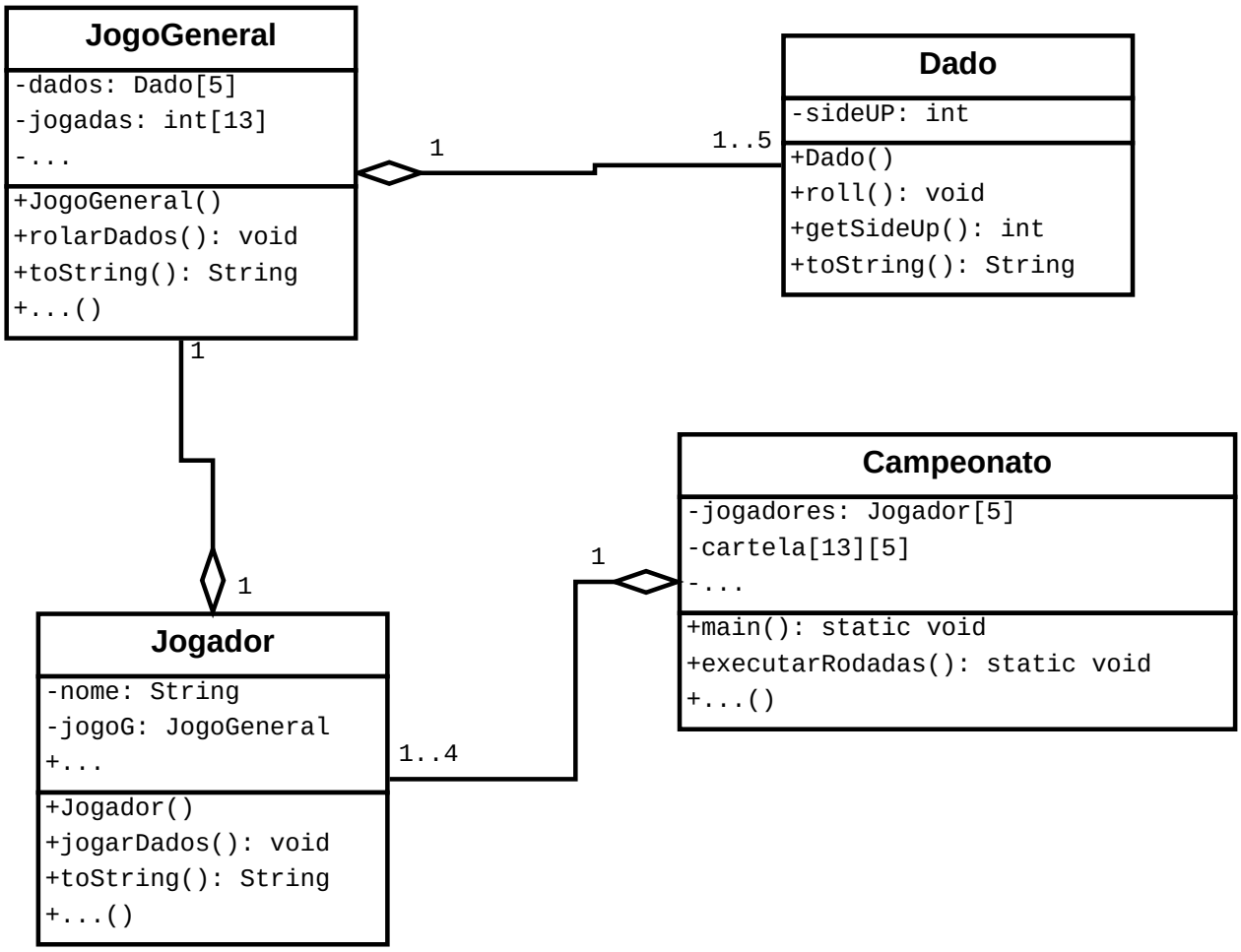

Figura 2: Diagrama UML.

Avaliação:

O trabalho será avaliado em função da:

- Correção (o aplicativo cumpre com as exigências);
- Documentação (o aplicativo está devidamente comentado);

- Paradigma orientado a objetos (o aplicativo está seguindo os princípios da programação OO: -encapsulamento, -associação de classes?)
- Modularidade (o aplicativo está bem estruturado onde necessário, com métodos (funções) parametrizados);
- Robustez (o aplicativo não trava em tempo de execução).

Detalhamento de itens a serem avaliados:

Item	Atendeu?
Respeitar o princípio do encapsulamento de dados	
Usar modificadores de acesso adequados (`private` e `public`)	
Criar getters e setters que forem necessários	
Criar métodos construtores parametrizados	
Fazer sobrecarga de pelo menos um método (qualquer um)	
Criar associação entre classes (Agregação ou Composição)	
O aplicativo não deve travar em tempo de execução	
Seguir o diagrama UML apresentado	

Exemplo de como fazer um menu iterativo em linguagem java:

```
1  import java.util.Scanner;
2  /**
3   * Exemplo de como criar um menu interativo com o laco
4   * do..while
5  */
6  public class ExemploMenu{
7
8   public static void main(String[] args){
9
10   Scanner teclado = new Scanner(System.in);
11   int opcao = 0;
12
13   do{
14         System.out.println("..:: Menu interativo ::..");
15         System.out.println("1 - Ola mundo");
16         System.out.println("2 - Ola POO");
17         System.out.println("3 - Sair");
18         System.out.print("Entre com uma opcao: ");
19         opcao = teclado.nextInt();
20
21         switch(opcao){
22          case 1:
23                 System.out.println("Ola mundo");
24                 break;
25          case 2:
26                 System.out.println("Ola POO");
27                 break;
28          case 3:
29                 System.out.println("Saindo");
30                 break;
31          default:
32                 System.out.println("Opcao invalida. Tente novamente");
```

```
33              }
34      }while(opcao != 3);
35
36              }
37  }
```