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

Trabalho #01 - Jogo General[1]

Data de entrega [Turma 4CP]: **09/10/2023 (até 23h59)**, via moodle.
Data de entrega [Turma 4CPE]: **10/10/2023 (até 23h59)**, via moodle.

O trabalho poderá ser individual ou em dupla.

Serão descontados **2 pontos por dia de atraso**.

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.

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 (humano ou máquina). A aplicação poderá executar n (indeterminadas) rodadas para os jogadores (humanos ou máquinas) participantes do *Campeonato do Jogo General*. Em cada rodada, cada jogador (humano ou máquina), por sua vez, joga os dados e, conforme o resultado obtido, marca a jogada prevista em sua cartela. Uma vez marcada, aquela jogada não poderá ser repetida pelo mesmo jogador até o final da rodada.

Regras básicas:

(1) Sendo 13 o número de jogadas possíveis e 13 o número máximo de linhas para cada coluna na cartela de marcação (Fig.1), uma rodada consiste de 13 jogadas para cada jogador.

(2) Cada jogador (humano ou máquina), 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.

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

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

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 deverá ser mostrado na forma de cartela (Fig.1), 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 rodada.

(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 de *Campeonato Jogo General*, 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 o campeonato uma ou mais vezes, com no máximo dez jogadores (humanos ou máquinas):

   (a) Incluir jogador (solicitar nome e tipo [humano ou máquina?])

   (b) Remover jogador (pelo nome)

   (c) Executar rodada
   >essa opção compreende, para cada jogador, em cada jogada [no total de 13 para cada jogador]:

   - rolar os dados
   - mostrar os valores dos dados obtidos
   - dar a opção de escolher a jogada que deseja marcar [caso o jogador seja humano, caso seja máquina, usar uma estratégia de escolha das opções de jogadas disponíveis]
   - passar a vez para o próximo jogador [caso não exista, repetir o processo para a próxima jogada do jogador]

   (d) Mostrar a cartela de resultados [da última rodada realizada]

   (e) Gravar os dados do campeonato em arquivo

   (f) Ler os dados do campeonato em arquivo

   (g) Sair da aplicação

	*nomeJog*1(*H*/*M*?)	*nomeJog*2(*H*/*M*?)	*nomeJog*3(*H*/*M*?)	*nomeJog*4(*H*/*M*?)	$\cdots$
1					
2					
3					
4					
5					
6					
T					
Q					
F					
S+					
S-					
G					
X					
Total					

Figura 1: Cartela de marcação (gerada ao final de cada rodada)

Exemplo de um esboço de execução da aplicação:

```
>entre com a opção do menu: a
>Nome do jogador: Luciene
>Tipo do jogador [H - humano ou M - máquina]:
>H
>entre com a opção do menu: a
>Nome do jogador: Maria
>Tipo do jogador [H - humano ou M - máquina]:
>M
>entre com a opção do menu: c
rolando dados para Luciene (H)...
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 (M)...
valores obtidos: 1-1-1-1-1
jogada escolhida por Maria(M) [1 - 13]: 12
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - -  -    -    -     -      -     50     -

rolando dados para Luciene (H)...
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)
- -  -  -  -  12  -    -    -     -      -      -     -
>8
seus valores não cumprem o requisito para esta jogada!

rolando dados para Maria (M)...
```

```
valores obtidos: 1-2-3-4-5
jogada escolhida por Maria(M) [1 - 13]: 11
1 2 3 4 5 6 7(T) 8(Q) 9(F) 10(S+) 11(S-) 12(G) 13(X)
- - - - - -  -    -    -     -     40     50    -

[... após finalizar todas as jogadas para todos os jogadores:]

>entre com a opção do menu: d
-- Cartela de Resultados --
      Luciene(H)  Maria(M)
1         x          x
2         x          x
3         x          x
4         x          x
5         x          x
6         12         x
7(T)      x          x
8(Q)      0          x
9(F)      x          x
10(S+)    x          x
11(S-)    x          40
12(G)     x          50
13(X)     x          x
---------------------------
Total  (12+x's)   (90+x's)
```

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, -cada classe executando suas operações específicas, o esboço do diagrama UML proposto foi seguido?).

  Observação: não utilizar a API (*Application Programmig Interface*) de estruturas de dados (Coleções), por exemplo, `ArrayList`, `LinkedList`, `HashSet`, `TreeSet`, etc da linguagem Java (ou qualquer outra escolhida).
- 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 a ser cumprido	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	
Não utilizar classes da API de coleções (estrutura de dados) da linguagem Java (ou outra escolhida)	

Esboço do diagrama UML a ser seguido:

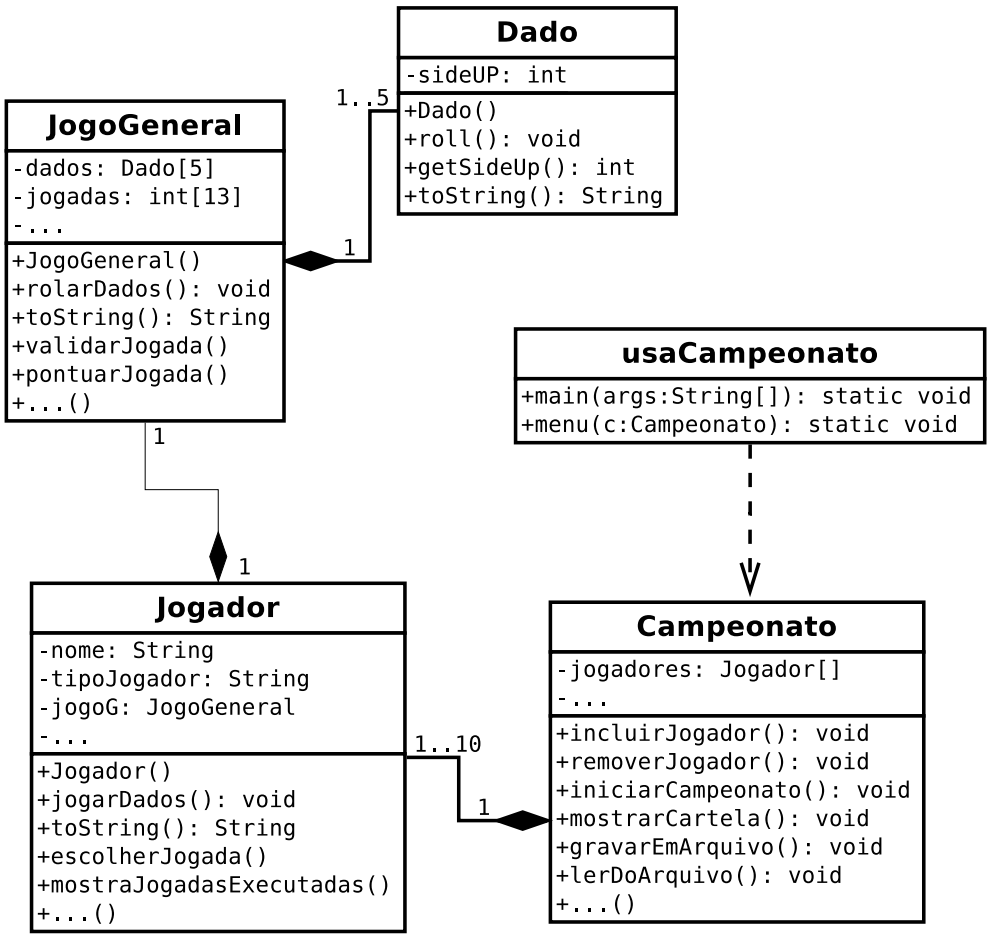

Figura 2: Diagrama UML.

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 }
```