Lista de Exercícios 1

Tema 01. Introdução à linguagem Java.

Recomendações:

- A solução dos problemas deve ser realizada utilizando a linguagem Java, e as boas práticas de programação para esta linguagem.
- A solução dos exercícios (apenas código fonte) deve ser colocada no Github no repositório do aluno em uma pasta chamada Lista01.

i. Multímetro quebrado? Uma equipe de manutenção elétrica deve determinar se o multímetro disponível opera corretamente. Para isso devem realizar 10 medições de voltagem em um circuito de corrente direta criado para testes. Em seguida devem calcular a média o desvio padrão das leituras. Se o desvio padrão for superior a 10% do valor médio, o multímetro está com problemas e não pode ser utilizado. Crie um programa que processe os dados das leituras e informe se o multímetro se encontra operacional. Seu programa deve imprimir no final todas as leituras, o valor médio o desvio padrão e uma mensagem informando o estado do dispositivo.

   Instruções:

   - Os dados de entrada podem ser fixos para fins de teste, lidos via teclado, ou lidos de um arquivo texto com formato livre.
   - As expressões para cálculo da média e o desvio padrão devem ser obtidas da literatura acadêmica ou Google.

ii. Jogo do Craps. Um dos jogos de azar mais populares é o jogo conhecido como *Craps*. Escreva um programa em Java que simule o jogo de *Craps*. O jogo pode ser dividido em dois estágios, e as regras do jogo são simples:

   Estágio 1

   - Um jogador joga dois dados. Cada dado tem seis faces. Essas faces contêm 1, 2, 3, 4, 5, ou 6 pontos. Depois dos dados pararem a soma das faces superiores dos dados é calculada. Se a soma for 7 ou 11 no primeiro lançamento o jogador vence.
   - Se a soma for 2, 3 ou 12 no primeiro lançamento (chamado *Craps*), o jogador perde (i.e. a "banca" vence).

- Se a soma for 4, 5, 6, 8, 9 ou 10 no primeiro lançamento esta soma se torna o "ponto" do jogador, e o jogo entra no segundo estágio.

Estágio 2

- Para vencer, o jogador deve continuar lançando dados até "fazer o ponto".
- O jogador perde se tirar um 7 antes de fazer o ponto.

Seu programa deve imprimir o resultado de cada lançamento dos dados (jogada) e mensagens que informem as jogadas realizadas. Considere como exemplo as saídas abaixo.

Saida 1	**Saida 2**
Dado 1: 6, Dado 2 :3	**Dado 1: 3, Dado 2 :5**
Ponto: 9	**Ponto: 8**
Iniciando estágio 2	**Iniciando estágio 2**
Dado 1: 5, Dado 2 :5	**Dado 1: 5, Dado 2 :5**
Jogada 1: 10	**Jogada 1: 10**
Dado 1: 2, Dado 2 :5	**Dado 1: 3, Dado 2 :1**
Jogada 2: 7	**Jogada 2: 4**
Vc perdeu :-(	**Dado 1: 4, Dado 2 :6**
	Jogada 3: 10
	Dado 1: 2, Dado 2 :6
	Jogada 4: 8
	Vc ganhou :-)

iii. Jogo de Batalha Naval. Desenvolva um programa em Java que seja uma recriação do Jogo Batalha Naval, nele dois jogadores escolherão os locais dos navios em um tabuleiro e tentarão afundar os navios do outro jogador escolhendo as coordenadas para atirar. O jogador que primeiro afundar todos os navios do oponente ganha o Jogo. Leia cuidadosamente as instruções abaixo, considere requisitos que seu programa deve atender:

1) Imprima na tela a mensagem **Welcome to Battleship!!!**

2) Peça a cada usuário que insira as coordenadas de cinco navios de comprimento um. Deve haver cinco prompts separados para cada navio (use o exemplo abaixo como guia). Você pode esperar que a entrada do usuário seja dois inteiros separados por um espaço. O primeiro inteiro representa o número da linha e o segundo representa o número da coluna.

   a) Se o usuário introduzir um inteiro inválido imprima

      Invalid coordinates. Choose different coordinates.

   b) Se o usuário introduzir uma coordenada que já foi digitada previamente imprima

You already have a ship there. Choose different coordinates.

c) Após cada jogador inserir sua quinta coordenada, um tabuleiro representando as localizações dos navios do jogador deve ser impresso na tela. Veja no item três sobre como construir o tabuleiro.

3) Crie dois Tabuleiros de Localização de 5x5 na forma de matrizes 2D usando as coordenadas inseridas pelos jogadores. Esses tabuleiros armazenam as localizações dos navios de cada jogador e serão usados para acompanhar os estados de dano dos navios de cada jogador, bem como qualquer disparo falho. O tabuleiro correspondente deve ser impresso no console logo após o jogador inserir as coordenadas de seus navios.

   a) O caractere '-' deve ser usado para representar um espaço vazio.

   b) O caractere '@' deve ser usado para representar um navio que não está afundado. Quando o jogo começa todos os navios do jogador estão em bom estado.

   c) O caractere 'X' representa um espaço com um navio que já está afundado.

   d) O caractere 'O' representa um espaço que foi alvejado, mas como não há um navio naquela coordenada, o disparo falhou.

   e) O tabuleiro de localização de cada jogador deve ter cinco navios de comprimento um. Cinco dos 25 espaços do tabuleiro começarão com navios neles

4) Adicionalmente, você deve gerar mais dois tabuleiros 5x5 na forma de matrizes 2D. Esses Tabuleiros de Histórico de Alvos permitirão que cada jogador rastreie visualmente seus acertos e erros. Após cada acerto ou erro do jogador, seu quadro de histórico de alvos deve ser impresso no console.

   a) Neste tabuleiro, um caractere 'X' deve representar um acerto do jogador, isto é um navio alvejado,

   b) Um caractere 'O' deve representar uma falha do jogador e

   c) Um caractere '-' deve representar um espaço que não foi atacado.

5) Peça ao Jogador 1 para inserir uma coordenada para atirar. Você pode esperar que a entrada do usuário seja dois inteiros separados por um espaço.

   a) Se o usuário introduzir um inteiro inválido imprima

   Invalid coordinates. Choose different coordinates.

   b) Se o usuário introduzir uma coordenada que já foi digitada previamente, imprima

   You already fired on this spot. Choose different coordinates.

   c) Se o usuário introduze uma coordenada que não contêm um navio, imprima a mensagem a seguir e imprima o Tabuleiro de Histórico de Alvos do jogador atualizado, onde [NUM] é substituído com o Id do jogador em ataque

Player [NUM] MISSED!

d) Se o usuário introduze uma coordenada que contêm um navio, imprima a mensagem a seguir e imprima o Tabuleiro de Histórico de Alvos do jogador atualizado, onde [NUM A] é substituído com o Id do jogador em ataque e [NUM B] é substituído pelo Id do jogador atacado.

Player [NUM A] hit Player [NUM B]'s Ship!!!

6) O Jogador 2 terá seu turno após cada turno do Jogador 1, que funcionará da mesma forma que os turnos do Jogador 1 (ver item 5).

7) Quando um navio é atingido por um jogador, o Tabuleiro de Localização (que rastreia os estados de dano) dos navios do jogador correspondente deve ser atualizado. Os disparos falhos também devem ser atualizados no Tabuleiro de Localização.

8) O programa deve terminar normalmente após a vitória de um jogador. Isso ocorrerá quando todos os símbolos '@' no tabuleiro do oponente forem substituídos por símbolos '**X**'.

   a) Imediatamente após o movimento que afunda a localização do último navio, imprima a seguinte mensagem, onde [NUM] é substituído pelo Id do jogador vencedor

   Player [NUM] WINS! You sunk all of your opponent's ships!

   b) Imprima os Tabuleiros de Localização de ambos os jogadores para verificar o resultado do jogo para os jogadores. O Tabuleiro de Localização do Jogador 1 deve ser impresso primeiro

9) Restrições:

   a) Você só pode importar a classe **java.util.Scanner.**

   b) Não pode usar a palavra reservada **var**

   c) Não pode utilizar o método **System.exit()**

Exemplos de Saída. As entradas digitadas pelo usuário aparecem em negrito.

```
Welcome to Battleship!

PLAYER 1, ENTER YOUR SHIPS' COORDINATES.
Enter ship 1 location:
0 1
Enter ship 2 location:
1 3
Enter ship 3 location:
2 1
```

```
Enter ship 4 location:
3 0
Enter ship 5 location:
3 4
   0 1 2 3 4
0  - @ - - -
1  - - - @ -
2  - @ - - -
3  @ - - - @
4  - - - - -

PLAYER 2, ENTER YOUR SHIPS' COORDINATES.
Enter ship 1 location:
0 1
Enter ship 2 location:
0 4
Enter ship 3 location:
2 0
Enter ship 4 location:
5 10
Invalid coordinates. Choose different coordinates.
Enter ship 4 location:
3 1
Enter ship 5 location:
4 4
   0 1 2 3 4
0  - @ - - @
1  - - - - -
2  @ - - - -
3  - @ - - -
4  - - - - @

Player 1, enter hit row/column:
```

```
5 12
Invalid coordinates. Choose different coordinates.
Player 1, enter hit row/column:
3 2
PLAYER 1 MISSED!
   0 1 2 3 4
0  - - - - -
1  - - - - -
2  - - - - -
3  - - O - -
4  - - - - -

Player 2, enter hit row/column:
1 0
PLAYER 2 MISSED!
   0 1 2 3 4
0  - - - - -
1  O - - - -
2  - - - - -
3  - - - - -
4  - - - - -

Player 1, enter hit row/column:
3 2
You already fired on this spot. Choose different
coordinates.
Player 1, enter hit row/column:
0 4
PLAYER 1 HIT PLAYER 2's SHIP!
   0 1 2 3 4
0  - - - - X
1  - - - - -
2  - - - - -
```

```
3  -  -  O  -  -
4  -  -  -  -  -

Player  2,  enter  hit  row/column:
3  3
PLAYER  2  MISSED!
    0  1  2  3  4
0  -  -  -  -  -
1  O  -  -  -  -
2  -  -  -  -  -
3  -  -  -  O  -
4  -  -  -  -  -

Player  1,  enter  hit  row/column:
2  0
PLAYER  1  HIT  PLAYER  2's  SHIP!
    0  1  2  3  4
0  -  -  -  -  X
1  -  -  -  -  -
2  X  -  -  -  -
3  -  -  O  -  -
4  -  -  -  -  -

Player  2,  enter  hit  row/column:
3  4
PLAYER  2  HIT  PLAYER  1's  SHIP!
    0  1  2  3  4
0  -  -  -  -  -
1  O  -  -  -  -
2  -  -  -  -  -
3  -  -  -  O  X
4  -  -  -  -  -
```

```
Player 1, enter hit row/column:
4 4
PLAYER 1 HIT PLAYER 2's SHIP!
   0 1 2 3 4
0  - - - - X
1  - - - - -
2  X - - - -
3  - - O - -
4  - - - - X

Player 2, enter hit row/column:
0 2
PLAYER 2 MISSED!
   0 1 2 3 4
0  - - O - -
1  O - - - -
2  - - - - -
3  - - - O X
4  - - - - -
***Skipping to the last turn***


Player 1, enter hit row/column:
3 1
PLAYER 1 HIT PLAYER 2's SHIP!
   0 1 2 3 4
0  - X - - X
1  - - - - -
2  X - - - -
3  - X O - -
4  - - - - X
PLAYER 1 WINS! YOU SUNK ALL OF YOUR OPPONENT'S SHIPS!
```

```
Final  boards:

    0  1  2  3  4
0   -  @  O  -  -
1   O  -  -  @  -
2   -  @  -  -  -
3   @  -  -  O  X
4   -  O  -  -  -

    0  1  2  3  4
0   -  X  -  -  X
1   -  -  -  -  -
2   X  -  -  -  -
3   -  X  O  -  -
4   -  -  -  -  X
```