P.O.O – Prof. Torrico
Lista de exercícios
Valor: 1,0 pt.
Apresentar ao professor na aula do dia 20/06/2017
Pode ser feita em dupla.

1)
Os dados são pequenos poliedros gravados com determinadas instruções. O dado mais clássico é o cubo (seis faces), gravado com números de um a seis. Para este exercício você deverá criar uma classe Dado para representar um dado e que nos permitirá rolá-lo e tirar valores que variarão de 1 a 6, ou seja, um dado de seis lados.

2)
Escreva uma classe que encapsule uma carta de baralho, com um valor que represente o valor da carta, de um (ás) a treze (rei), e outro valor correspondente ao naipe (1 = ouros, 2 = paus, 3 = copas e 4 = espadas). Escreva nessa classe um método toString que retorne o nome da carta por extenso, usando a instrução switch. **Obs:** Estude a utilização do método toString. Pra que serve esse método? Ele pertence a qual classe?

3)
Escreva uma aplicação em Java que simule uma calculadora bem simples. Esse programa deve ler dois valores de ponto flutuante do teclado e um caractere, correspondente a uma das operações básicas (+, -, * ou /), calcular a operação e imprimir o resultado. A aplicação deve considerar divisões por zero como sendo erros, e imprimir uma mensagem adequada. A aplicação também deve considerar a possibilidade de o usuário entrar um valor diferente de +, -, * ou / como operador, e imprimir uma mensagem de erro. Faça uma interface gráfica usando Java Swing para sua calculadora.

4)
Crie uma classe que represente um jogo da velha, usando uma matriz de duas dimensões para representar as posições do jogo. A matriz deve ser alocada no construtor da classe, ter o tamanho 3 × 3 e ser de um tipo que suporte três estados possíveis: vazio, preenchido com 'O' e preenchido com 'X' (use um enumerador). A classe deve poder ser usada para jogos com dois jogadores. Dica: A classe deve ter os seguintes métodos:
• **jogaO**, que aceita dois valores que são as coordenadas onde um 'O' será jogado, e marca na matriz a posição somente se esta estiver livre.
• **jogaX**, que aceita dois valores que são as coordenadas onde um 'X' será jogado, e marca na matriz a posição somente se esta estiver livre.
• **verifica**, que verifica a matriz para ver se existe algum ganhador (esse método deve verificar se existem três marcas iguais que não sejam vazias em uma horizontal, vertical ou diagonal da matriz).
• **toString**, que retornará uma string com a representação gráfica do jogo com as posições atuais.
Escreva também um programa que use a classe. Este programa deve executar um laço no qual fica perguntando as posições para os jogadores alternadamente, enquanto não houver vitória, desistência ou acabarem as posições vazias da matriz.

5)
Crie uma classe **StringUtils** que contenha vários métodos estáticos para processamento de strings. Crie nesta classe um método estático **desacentua** que recebe como argumento uma string e que substitua todos os caracteres acentuados desta string por caracteres não-acentuados correspondentes. Por exemplo, se a string "Nação" for passada como argumento, esse método deverá retornar "Nacao". O método deve considerar maiúsculas e minúsculas

como sendo diferentes. **Dica**: Várias chamadas ao método replace, em cascata, poderão resolver o problema.

6)
Crie, na classe **StringUtils** o método **alinhaÀDireita**, que recebe como argumentos uma string e um valor numérico, e completa a string com espaços à esquerda até que o comprimento da string fique igual ao valor numérico passado, retornando a string modificada. Escreva também o método **alinhaÀEsquerda**, que faz o mesmo, mas adicionando espaços à direita. Se o comprimento da string passada já for maior que o valor passado como argumento, o método deve retornar a string inalterada.

7)
Crie, na classe **StringUtils**, o método **replica**, que recebe como argumentos uma string e um valor inteiro, e retorna uma string composta de várias repetições da string passada como argumento, onde o número de repetições deve ser o número passado como argumento. Por exemplo, se os argumentos para esse método forem a string “Ha!” e o valor 3, o método deverá retornar “Ha!Ha!Ha!”.

8)
Escreva para a classe **StringUtils** um método estático **conta** que receba como argumentos uma string e um caracter, e retorne um inteiro correspondente ao número de ocorrências do caracter na string passados como argumentos.

9)
Escreva na classe **StringUtils** um método **reverte** que reverta a ordem dos caracteres de uma string passada como argumento e retorne a string revertida. Um exemplo: se a string ”Java” for passada para esse método, ele deve retornar a string ”avaJ”. **Dica**: Use um laço for ou while e o método charAt, e crie uma nova string que receberá os caracteres na ordem invertida. Não use mecanismos da classe StringBuffer ou StringBuilder.

10)
Escreva um método **quantasVezes** para a classe **StringUtils** que receba duas strings como argumentos e retorne o número de vezes que a segunda string aparece na primeira. Por exemplo, se a string “recrearem” e “re” forem passadas como argumentos, o método deverá retornar 3.

11)
Escreva um método **retiraVogais** na classe **StringUtils** que receba uma string como argumento e remova todas as vogais (maiúsculas e minúsculas) dessa string, retornando a string modificada como resultado.