Quarta Lista de Exercícios

1. Escreva o método public ArrayList<Double> readValues(String filename) throws... que lê um arquivo contendo números em ponto flutuante. Lance exceções apropriadas se o arquivo não poder ser aberto ou se alguma das entradas não forem números em ponto flutuante. Para ler o arquivo use as classes FileReader e BufferedReader. **(FEITO)**
2. Escreva o método public double sumOfValues(String filename) throws... que chama o método do exercício 1 e retorna a soma dos valores no arquivo. Propague quaisquer exceções para o chamador. **(FEITO)**
3. Escreva um programa que chame o método anterior e imprima o resultado. Capture as exceções e forneça feedback ao usuário sobre quaisquer condições de erro. **(FEITO)**
4. Implemente um método que contenha o código a seguir, porém sem usar try-with-resource:

try (Scanner in = new Scanner(Paths.get("words.txt"));

PrintWriter out = new PrintWriter("output.txt"))

{

while (in.hasNext())

out.println(in.next().toLowerCase())

}

Em vez disso, use apenas cláusulas catch. Certifique-se de fechar ambos os objetos,

desde que tenham sido devidamente construídos. Você precisa considerar as seguintes

condições:

• O construtor Scanner lança uma exceção.

• O construtor PrintWriter lança uma exceção.

• hasNext(), next() ou println() lançam uma exceção.

• out.close() lança uma exceção.

• in.close() lança uma exceção.

**(FEITO)**

1. O código abaixo ilustra um exemplo de try-catch-finally com problema. Corrija o código com (a) capturando a exceção na cláusula finally, (b) uma instrução try/catch contendo uma instrução try/finally e (c) uma instrução try-with-resources com uma cláusula catch.**(FEITO)**

BufferedReader in = null;

try {

in = Files.newBufferedReader(path, StandardCharsets.UTF\_8);

//lê do arquivo

} catch (IOException ex) {

System.err.println("Caught IOException: " + ex.getMessage());

} finally {

if (in != null) {

in.close(); //pode lançar uma exceção

}

}

1. Implemente uma classe genérica Pilha<E> que gerencia um ArrayList de elementos do tipo E. Forneça os métodos push() para empilhar, pop() para desempilhar e isVazia() para testar se a pilha está vazia. **(FEITO)**
2. Reimplemente a classe Pilha<E>, usando um array para armazenar os elementos. Se necessário, aumente o array no método push. Forneça duas soluções, uma com um array E[] e outra com um array Object[]. Ambas as soluções devem compilar sem warnings. Qual você prefere e por quê? **(FEITO)**

**R: Prefiro criar um array de Object, pois não é necessário lidar com conversões, e também pelo fato de que quando se cria um array genérico será preciso chamar Object, o que torna bem mais prático lidar com um array de Object:**

**E[ ] array = (E[ ]) new Object[tamanho];**

**Object[ ] array = new Object[tamanho];**

1. Implemente uma classe Table<K, V> que gerencia uma ArrayList de objetos Entry<K,V>.

**(FEITO)**

1. Forneça métodos para obter o valor V associado a uma chave K, para armazenar um valor V para uma chave K e para remover uma chave. **(FEITO)**
2. Refaça o exercício 9, transformando Entry em uma classe aninhada. Essa classe deveria

ser genérica? **(FEITO)**

**R:** **Sim, a classe aninhada Entry deve ser genérica, pois a mesma está sendo criada dentro de uma classe genérica**