Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/2

Programação 3

Lista 5

Instruções

Responda às questões abaixo implementando os programas em Java. Sempre que possível, explique a saída do código e o comportamento esperado.

1. Identificando Exceções

Analise o seguinte código e identifique quais exceções podem ser lançadas em tempo de execução. Justifique sua resposta.

```
public class ExcecaoTeste {
    public static void main(String[] args) {
        int[] numeros = {1, 2, 3};
        System.out.println(numeros[5]); // O que acontece aqui?

        String texto = null;
        System.out.println(texto.length()); // E aqui?

        int resultado = 10 / 0; // E aqui?
    }
}
```

Perguntas:

1. Quais exceções podem ser lançadas no código acima?
2. Em quais linhas os erros ocorrem?
3. Como evitar esses erros?

2. Lidando com `try-catch`

Modifique o código do exercício anterior para capturar todas as exceções possíveis usando `try-catch`, evitando que o programa seja interrompido abruptamente.

3. Criando um Manipulador de Exceções

Crie um programa que peça ao usuário para inserir dois números e exiba o resultado da divisão entre eles. O programa deve:

1. Tratar a divisão por zero (`ArithmeticException`).
2. Tratar entradas não esperadas pelo `Scanner` (`InputMismatchException`).
3. Exibir uma mensagem final informando que a execução foi concluída.

4. Criando Exceções Personalizadas

Implemente uma classe `SaldoInsuficienteException` e use-a na classe `ContaBancaria`.

A classe `ContaBancaria` deve conter:

1. Um método `sacar(double valor)`, que lança `SaldoInsuficienteException` caso o saldo seja menor que o valor do saque.
2. Um método `depositar(double valor)`, que adiciona saldo à conta.
3. O método `main()` deve testar saques válidos e inválidos.

Dica: `SaldoInsuficienteException` deve ser uma **Checked Exception**.

5. Hierarquia de Exceções

Crie um programa que:

1. Contenha duas exceções personalizadas: `LoginInvalidoException` e `ContaBloqueadaException`, ambas derivadas de `Exception`.
2. Simule um sistema de login onde:
   - Um usuário insere um login e senha.
   - Se o login for incorreto, dispare `LoginInvalidoException`.
   - Se o usuário errar três vezes seguidas, dispare `ContaBloqueadaException`.
3. O programa deve capturar e exibir mensagens de erro apropriadas.