Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Campus Maracanaú
Coordenadoria de Computação
Curso de Bacharelado em Ciência da Computação
Disciplina: Programação Orientada a Objetos
Professor: Igor Rafael Silva Valente

ATIVIDADE

Assunto:
Exceções – fundamentos, captura e tratamento.

Orientações:
A atividade deve ser executada individualmente e entregue através do ambiente *Google Classroom*.

Regras de criação dos programas:
Crie um novo projeto Java denominado **AtividadeExcecoes1**. As classes devem possuir os nomes informados no texto. Ao final, o projeto deve ser exportado para um arquivo em formato ZIP.

Nome completo:

Mac Myller da Silva Carlos

1. O que é uma exceção? Explique.
   As exceções ocorrem quando algo imprevisto acontece, elas podem ser provenientes de erros de lógica ou acesso a recursos que talvez não estejam disponíveis.

2. Explique detalhadamente a diferença entre as exceções verificadas e não verificadas.

   Exceções não verificadas:
   - Representam defeitos no programa (bugs).
   - São subclasses de RuntimeException e são normalmente implementadas usando IllegalArgumentException, NullPointerException ou IllegalStateException.
   - Um método não é obrigado a estabelecer uma política para as exceções não checadas lançadas por sua execução (e quase sempre nunca o fazem).

   Exceções verificadas:
   - Representam condições inválidas em áreas fora do controle imediato do programa (problemas de entradas inválidas do usuário, banco de dados, falhas de rede, arquivos ausentes).
   - São subclasses de Exception.
   - Um método é obrigado a estabelecer uma política para todas as exceções checadas lançadas por sua implementação (ou passar a exceção checada mais acima na pilha, ou manipulá-la de alguma forma).

3. Qual a hierarquia de classes do mecanismo de tratamento de exceções da linguagem Java?
   - primeiro vem a classe Throwable
   - depois as classes Error e Exception que são filhas de Throwable
   - e RunTimeException e outras exceções que são filhas de Exception

4. Explique conceitualmente as diretivas a seguir, bem como crie um código-fonte simplificado demonstrando o seu uso (pode ser um código-fonte único demonstrando todas as diretivas):
   a. try... catch
   b. throws
   c. throw

a: try… catch

```
public class Main{
    public static void main(String[] args) {
        try{
            int[] vetor = new int[3];
            vetor[3] = 10; //o erro tá aqui, o vetor tem apenas os índices 0, 1, 2
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Erro ao acessar índice inexistente no vetor");
        }
    }
}
```

b: throws

```
import java.io.FileReader;
import java.io.IOException;
public class Main {
    public static void main(String[] args) {
        try {
            metodo();
        }catch (IOException e) {
            System.out.println("Erro");
        }
    }
    public static void metodo() throws IOException {
        FileReader f = new FileReader("notExist.txt");
    }
}
```

c: throw

```java
public class Main {
    public static void main(String[] args) {
        try{
            lancarUmaException();
            System.out.println("Essa mensagem nao sera impressa");
        }catch(Exception e){
            System.out.println("Erro");
        }
    }
    public static void lancarUmaException() throws Exception {
             throw new Exception(); //toda vez que esse metodo for chamado irá lançar(throw) uma exceção do tipo Exception
    }
}
```

5. Crie a classe `Cadastro` que contém um método `main` e solicita do usuário o fornecimento das informações a seguir: `nomeCompleto` e `telefone` (composto por `ddd` e `numero`). Você pode utilizar as classes `Scanner` ou `JOptionPane` para realizar esta tarefa. Após receber as informações, verifique se `nomeCompleto` é diferente de vazio. Adicionalmente, faça a leitura do telefone em duas etapas. A primeira deve solicitar o `ddd` e a segunda o `numero`. O `ddd` deve possuir dois dígitos inteiros e o `numero` oito ou nove dígitos inteiros. Se o usuário fornecer corretamente estas informações, você deve emitir uma mensagem informando que o cadastro foi realizado e imprimir os valores recebidos. Caso alguma das verificações falhe, você deve notificar o usuário através do lançamento e tratamento de uma exceção. Em seguida, deve reiniciar o processo desde o início, repetindo esta operação até que todas as informações sejam preenchidas.

```
import java.util.Scanner;
public class Cadastro {
    public static void main(String[] args) {

        Scanner scan = new Scanner( System.in );
        String nome=null;
        int ddd, numero;

        while( nome==null ){
            System.out.println("Informe seu nome completo: ");
            nome = scan.nextLine();
        }

        System.out.println("Informe seu numero ");
        while(true){
            try{
                System.out.print("DDD: ");
                ddd = scan.nextInt();
                if(100<=ddd){
                     throw new java.util.InputMismatchException("DDD com mais
de dois digitos");
                }
                break;
            }catch(java.util.InputMismatchException e){
                System.out.println("Erro, DDD invalido, tente novamente");
                scan.nextLine(); //esvaziar buffer
            }
        }

        while(true){
            try{
                System.out.print("Numero: ");
                numero = scan.nextInt();
                if( numero < 900000000 || 1000000000 <= numero   ){
```

```
                        throw new java.util.InputMismatchException("Numero
invalido");
                }
                break;
            }catch(java.util.InputMismatchException e){
                System.out.println("Erro, Numero invalido, tente novamente");
                scan.nextLine(); //esvaziar buffer
            }
        }

        scan.close();

        System.out.println("\n\n\nNome: " + nome);
        System.out.printf("Numero: %d %d\n", ddd, numero);
    }
}
```

Boa sorte!

Prof. Igor.