DEFINIÇÃO E TRATAMENTO DE EXCEÇÕES

PROF. ME. HÉLIO ESPERIDIÃO

EXCEÇÕES EM JAVA

As exceções em Java se referem aos erros que podem ser gerados durante a execução de um programa.

Como o próprio nome sugere, trata-se de algo que interrompe a execução normal do programa. É um problema que não ocorre frequentemente.

UNCHECKED EXCEPTION

Significa "exceção não verificada". Neste tipo de exceção, o Java não verifica o código-fonte para determinar se a exceção está sendo capturada.

Fazem parte dessas exceções de tratamento opcional, por exemplo, a verificação de acesso a um índice inexistente num vetor, a tentativa de se usar um método de um objeto ainda não instanciado e a conversão de um String em inteiro

CHECKED EXCEPTION

Significa "exceção verificada". Neste tipo de exceção, o compilador Java obriga o programador a tratá-la.

O Java verifica o código-fonte, com a finalidade de determinar se a exceção está sendo capturada.

```
try {
    comandos
} catch (exceção_tipo1 identificador1) {
    comandos
} catch (exceção_tipo2 identificador2) {
    comandos
    ...
} finally {
    comandos
}
```

TRATAMENTO

No Java, a estrutura que trata as exceções é formada pelos comandos try-catch-finally:

TRY {..}

NESTE BLOCO, SÃO ESCRITAS TODAS AS LINHAS DE CÓDIGO QUE PODEM VIR A LANÇAR UMA EXCEÇÃO;

```
catch (tipo_excessao e)
{ … }:
```

NESTE BLOCO É DESCRITA A AÇÃO QUE OCORRERÁ QUANDO A EXCEÇÃO FOR CAPTURADA;

FINALLY

- É opcional e fornece um conjunto de códigos que é sempre executado, independentemente da ocorrência da exceção. O uso do finally pode ser exemplificado por meio de operações de banco de dado

EXEMPLO

```
try {
        System.out.print("Digite a idade: ");
        int idade = sc.nextInt();

        System.out.println(idade);
    } catch (InputMismatchException e) {
        e.printStackTrace();
    }
```

```java
import java.util.Scanner;
public class LeitorTeclado {
    public void lerDados() {
        try {
            Scanner in = new Scanner(System.in)
            int numero = in.nextInt();
            System.out.println(numero);
            in.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        LeitorTeclado l = new LeitorTeclado();
        l.lerDados();
    }
}
```

EXCEPTION

Os recursos para tratamento de exceções são muito importantes para produzir sistemas de qualidade e são mecanismos que a própria linguagem propicia para detectar o local em que um problema ocorreu.

COMUNS

Exceção	Descrição
java.lang.ArrayIndexOutOfBoundsException	tenta acessar uma posição de vetor ou matriz que não existe
java.lang.ArithmeticException	Gerado na divisão de um número int por zero
java.lang. IllegalArgumentException	argumentos errados para um método.
java.io.FileNotFoundException	Quando se tenta fazer a leitura ou escrita em um arquivos que não existe

FINALLY

- O finally não executará somente se a aplicação for terminada.
- Assim, é possível fazer o tratamento final de qualquer evento.
- Garante que sempre se tentará executar o procedimento para finalizar ou tratar algo sem importar se houveram erros a serem tratados.

```java
import java.io.*;
public class ControleArquivos {
    public void escreveArquivo(String caminhoArquivo) {
        FileWriter fw = null;
        File f = new File(caminhoArquivo);
        try {
            fw = new FileWriter(f);
            fw.write(10);
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fw != null) {
                try {
                    fw.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
    public static void main(String[] args) {
        ControleArquivos l = new ControleArquivos();
        l.escreveArquivo("c:\\arquivoTeste.txt");
    }
}
```

EXEMPLO

Veja que, mesmo dentro do finally, é necessário utilizar um try e um catch por imposição do método.

TRATAMENTO PERSONALIZADO DE EXCEÇÕES

- Existem outras formas de fazer o tratamento de exceção relacionado à centralização e ao controle do local onde os tratamentos de erros são implementados.

- A cláusula throws é um recurso que faz com que o método "lance" uma exceção ou mais, que serão especificadas pelo programador, caso ocorra algum problema.

LANÇAMENTO

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
public class EscritorArquivo {

    private String caminhoArquivo;
    private FileWriter fileWriter;
    private File file;
    public EscritorArquivo(String pCaminhoArquivo) {
        this.caminhoArquivo = pCaminhoArquivo;
    }
    public void escreveArquivo() throws IOException {
        this.file = new File(this.caminhoArquivo);
        fileWriter = new FileWriter(this.file);
        fileWriter.write(10);
        if (fileWriter != null) {
            fileWriter.close();
        }
    }
    public static void main(String[] args) {
        EscritorArquivo s = new EscritorArquivo("dados1/arquivo.txt"
        try {
            s.escreveArquivo();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

THROWS

- o uso da cláusula throws da a possibilidade de o programador criar suas próprias exceções.
- Isso é feito por meio do comando throw new Exception().

```java
import java.io.File;
import java.util.Scanner;
public class LeitorArquivo2 {

    private String caminhoArquivo;

    public LeitorArquivo2(String pCaminhoArquivo) {
        this.caminhoArquivo = pCaminhoArquivo;
    }

    public void lerArquivo() throws java.lang.Exception {
        File f = new File(caminhoArquivo);
        if (f.exists() == false) {
            throw new Exception("Arquivo nao encontrado");
        }
        Scanner sc = new Scanner(f);
        String dados = sc.next();
        System.out.println(dados);
    }

    public static void main(String[] args) {

        LeitorArquivo2 le = new LeitorArquivo2("dados1/arquivo.txt");
            le.lerArquivo();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

EXCEÇÕES GERADAS POR THREADS

Exceções	Descrição
IllegalArgumentException	Quando é feito um Thread.sleep (valor), o valor deve estar entre 0 e 999999.
InterruptedException	Quando uma thread é interrompida. Esse processo pode ocorrer quando se tenta parar a thread
SecurityException	É possível criar grupos de threads. Essa exceção é gerada quando a thread não pode ser inserida em um certo grupo