accenture

Strategy | Digital | Technology | Operations

High performance. Delivered.

Application Delivery Fundamentals: Java

Module 8: Exceptions and Assertions

Module Objectives

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- No final deste módulo, você será capaz de:
  - Explique o conceito de exceções e asserções.
  - Explique o uso de exceções e asserções.
  - Gerencie exceções usando try-catch-finally.
  - Crie exceções personalizadas e condições de exceção.
  - Use declarações de asserção para melhorar a qualidade do código.

Exceptions

“Exceptions”

Exceptions

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

Exceções

- Mecanismo para tratar os erros durante a execução do programa.
- Exceções são classes que representam erros durante a execução de um programa java.
- Os objetos que representam as exceções podem ser “lançados”, capturados e manipulados.

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- A exceção é:
  - Um evento que acontece durante a execução do programa e impede que o programa continue normalmente.
  - Uma condição de erro que altera o fluxo normal de controle em um programa.
  - Um sinal que indica alguma condição inesperada ocorreu no programa.

Exceptions

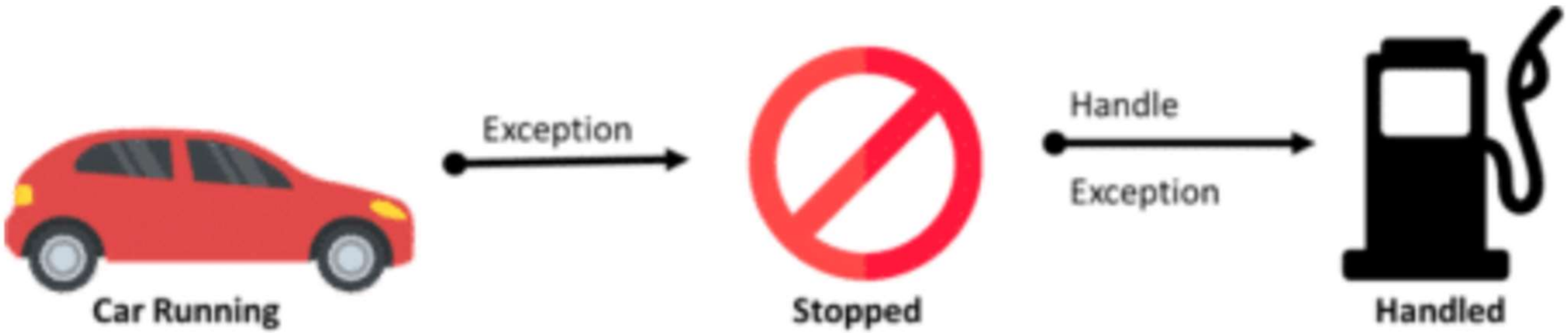
2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

Condições de Erro

- Tentar abrir um arquivo que não existe
- Tentar usar uma conexão de rede interrompida
- Tentar acessar posição de arrays fora da faixa
- Tentar carregar uma classe que não está no classpath

Exceptions

- A exceção:



Exceptions

- A exceção:

Exemplo

```
public class Semaforo {
  public static void main (String args[ ]) {
    int i = 0;
    String semaforo [ ] = {"Verde", "Amarelo","Vermelho"};
    while (i < 4) {
      System.out.println (sinais[i]);
      i++;
    }
  }
}
Ao executar o resultado é
java Semaforo
Verde
Amarelo
Vermelho
java.lang.ArrayIndexOutOfBoundsException: 3
at Semaforo.main(Semaforo.java:12)
```

Handling Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- O mecanismo de tratamento de exceções é construído em torno do **throw-and-catch paradigm:**
  - **‘to throw’ significa que uma exceção ocorreu.**
  - **‘to catch’ significa lidar com uma exceção.**
- Se uma exceção não for capturada, ela será propagada para a pilha de chamadas até que um manipulador seja encontrado.

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Tratamento de exceção:

Exceptions

- Tratamento de exceção:

```
Conta c = new Conta();
try {
      c.saque(100);
} catch (EstouroSaqueException e) {
  System.out.println("Erro: " + e.getMessage() );
}
c.mostraSaldo();
```

Using try-catch-finally Blocks

```
try {
    /*
     * some codes to test here
     */
} catch (SQLException sx) {
    /*
     * handle Exception1 here
     */
} catch (IOException ix) {
    /*
     * handle Exception2 here
     */
} catch (Exception ex) {
    /*
     * handle Exception3 here
     */
} finally {
    /*
     * always execute codes here
     */
}
```

Try block inclui o contexto em que uma possível exceção pode ser lançada

Cada **Catch() block** é um manipulador de exceções e pode aparecer várias vezes

O opcional **Finally block** é sempre executado antes de sair da instrução **Try.**

Refer to the TryCatchFinallySample.java sample code.

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Instrução *finally*

A instrução finally define um bloco de código que sempre será executado, mesmo que uma exceção seja lançada ou não:

```
boolean transacaoOk = false;
try {
  abreTransacao( );
  debito.saque ( valor );
  credito.deposito( valor );
  transacaoOk = true;
} finally {
  if ( transacaoOk )
    fechaTransacao( );
  else
    desfazTransacao( );
}
```

Using try-catch-finally Blocks (cont.)

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Isole o código que pode gerar uma exceção no bloco try.
- Para cada bloco **catch** () individual, você escreve o código que deve ser executado se uma exceção desse tipo específico ocorrer no bloco try.
- No bloco **finally**, você escreve um código que será executado independentemente de um erro ou não. Isso é opcional.

Refer to the TryCatchFinallySample.java sample code.

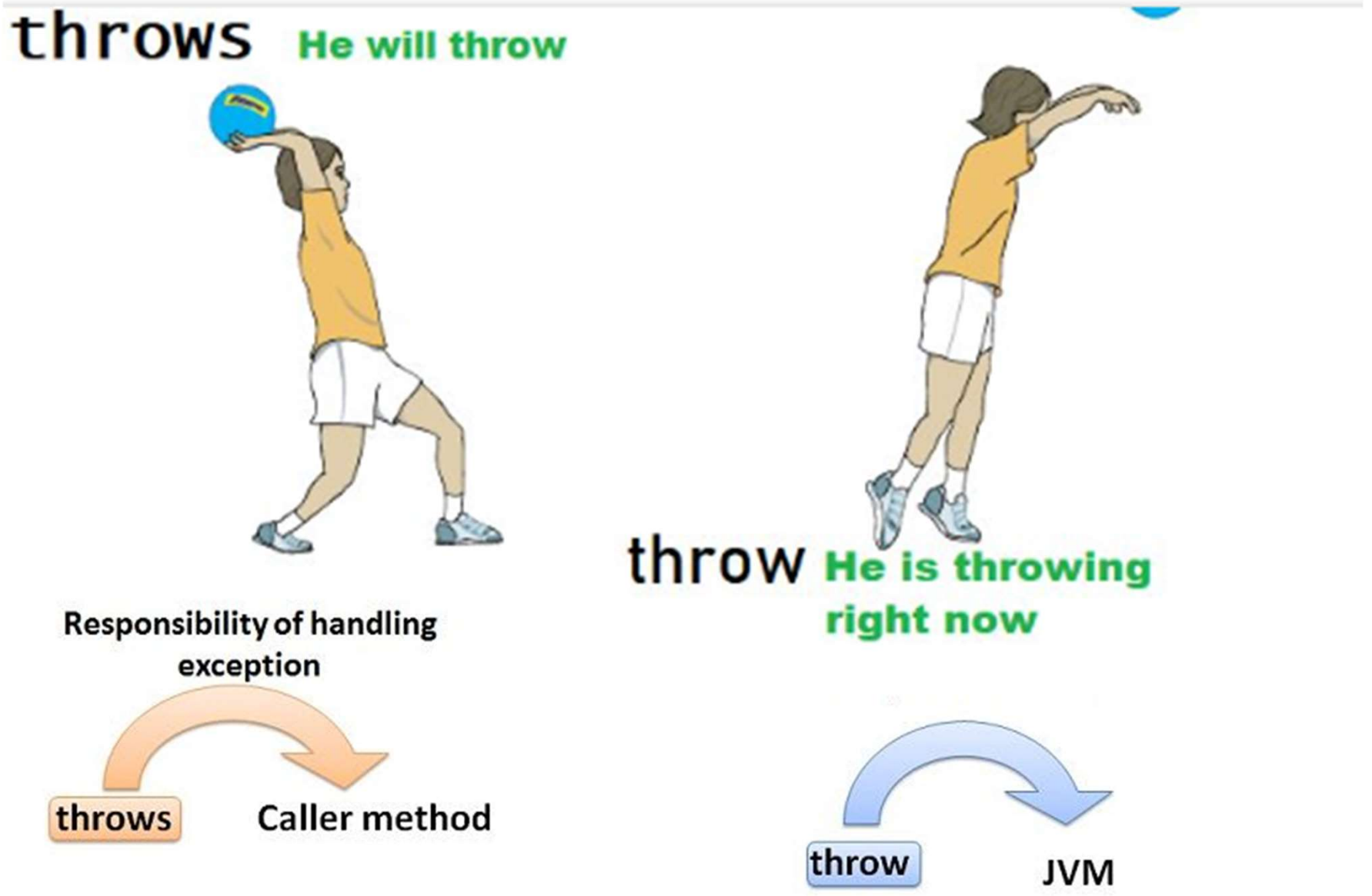
Exceptions

Instrução *throw e clausula throws*

```
public void deposito(double valor) throws
DepositoInvalidoException {
      if (valor > 0) {
            saldo = saldo + valor;
      } else {
            throw new DepositoInvalidoException("Este
valor é invalido para depósito: " + valor);
      }
}
```

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207



Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Tratar ou lançar?

- O tratamento de exceções pode ser de 2 modos:
- Tratar dentro do próprio método usando try,catch e finally
- Lançar para quem chamou o método onde a exceção foi gerada usando a clausula throws

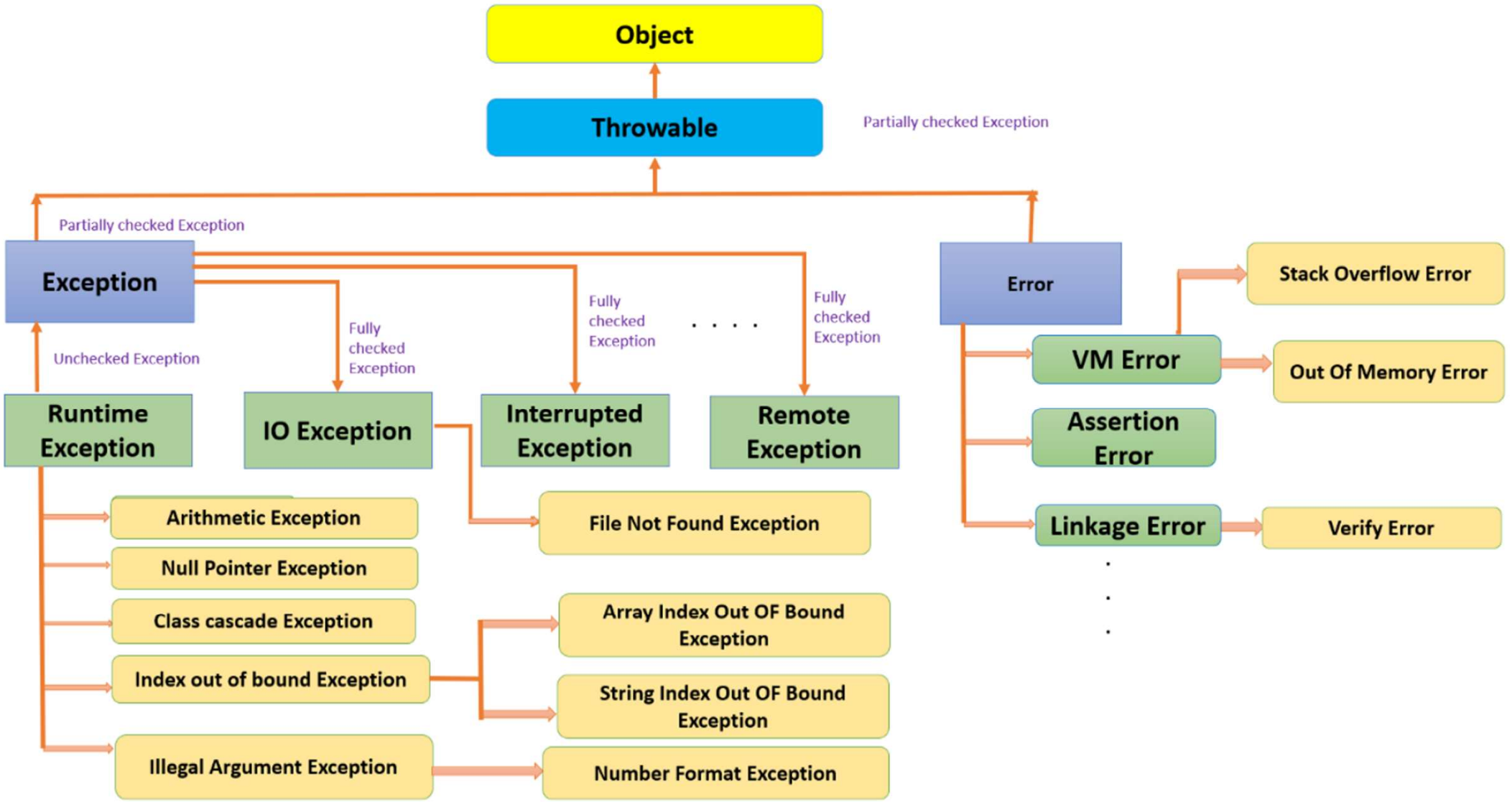
Activity – TryCatchFinally

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Nesta atividade, você :
  - Abra o arquivo ‘TryCatchFinallyActivity.java’ no pacote sef.module8.activity.
  - Leia as instruções e crie o código para concluir este programa.

Exception Class Hierarchy

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



Un-Checked Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- As exceções **Un-checked** representam erros geralmente causados por código ou lógica incorreta do programa, como parâmetros inválidos passados para um método.
- Eles são uma subclasse da classe RuntimeException.
- O aplicativo não é necessário para lidar com essas exceções, pois elas devem ser recuperadas corrigindo o código do programa.
- **Examples**: IllegalArgumentException, NumberFormatException.

Refer to the ArrayExceptionSample.java sample code.

Checked Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- As exceções **Checked** representam erros causados por fatores fora do código do aplicativo.
- Eles são subclasses da classe Exception, excluindo a classe RuntimeException.
- O aplicativo é necessário para lidar com esses cenários excepcionais através de construções try-catch.
- **Examples**: IOException, SQLException

Refer to CheckedExceptionSample.java sample code.

Activity – SQL Exception

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Nesta atividade, você:
  - Abra o arquivo ‘SQLExceptionActivity.java’ no pacote sef.module8.activity.
  - Leia as instruções e crie o código para concluir este programa.

Errors

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Os erros representam erros críticos que não devem ocorrer e dos quais não se espera que o aplicativo se recupere;
- Os erros geralmente são gerados a partir de erros na lógica ou no design do programa e devem ser tratados através da correção do design ou código.
- **Examples**: OutOfMemoryError, StackOverFlowError

Refer to ErrorSample.java sample code.

Specifying Exceptions

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- As exceções também podem ser tratadas propagando-as na pilha de chamadas em vez de manipulá-las no método atual.
- Um método pode declarar que uma de suas instruções pode gerar uma exceção e que é deixada para quem está chamando o método para lidar com isso.

```
<method signature> throws <Exception1>,<Exception2>

public void connectToDB (String query)throws SQLException,IOException{
      //code here
}
```

- Qualquer declaração que possa gerar uma exceção verificada declarada pelo método é considerada 'manipulada' e não precisa de um bloco try-catch.

Refer to SpecifyExceptionSample.java sample code.

Handling Exception Through Declaration

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Pode ser dito ao código para lançar explicitamente uma exceção (Checked or Unchecked).
- As exceções são representadas como objetos Java e podem ser criadas como qualquer outro objeto e depois 'lançadas' usando a palavra-chave throw.

```
Example:
public void setAge(int age){
   if(age < 0 ){
            //create an instance and throw at the same time
throw new IllegalArgumentException("parameter age cannot be less than 0");
   }
}
```

Refer to the ExceptionDeclarationSample.java sample code.

Customizing Exceptions

- Exceções na API padrão podem não ser suficientes para cobrir os cenários necessários ao aplicativo.
- Uma exceção personalizada pode ser declarada subclassificando a classe Exception.
- A exceção personalizada deve conter dados e comportamento apropriados para ajudar na identificação e correção do problema.

i Refer to the CustomException.java and CustomExceptionSample.java sample code.

Activity 2 – Custom Exception

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Nesta atividade, você:
  - Abra os arquivos ‘CustomExceptionActivity.java’ e ‘CustomExceptionActivityTest.java’ no pacote sef.module8.activity.
  - Leia as instruções e crie o código para concluir este programa.

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Criando Exceções

```
public class DepositoInvalidoException extends Exception {

        public DepositoInvalidoException (String motivo) {
        // chama construtor da classe pai
                super(motivo);
        }
}
```

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Existem muitos métodos que são úteis ao implementar o tratamento de exceções em Java:

- 1. **getMessage()-** Esta função retorna uma mensagem resumida sobre a exceção que ocorreu. Ele também é inicializado no construtor de uma classe Throwable. A sintaxe deste método é public String getMessage();

- 2. **getCause()-** Esta função retorna a causa da exceção que surge no programa na forma de um Throwable Object. A sintaxe é public Throwable getCause();

- 3. **toString()-** Este método retorna a string que contém o nome da string anexada à mensagem de exceção. A sintaxe é public String toString();

Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- 4. **printStackTrace()-**Esta função imprime o rastreamento de pilha do sistema para o fluxo de saída de erro. A sintaxe desta função é public void printStackTrace();

- 5. **getStackTrace()-** Esta função é responsável por retornar um array que contém os elementos de rastreamento de pilha. O último elemento é a parte inferior da pilha e o primeiro elemento é o topo do rastreamento de pilha.

- 6. **fillinStackTrace()-** Retorna um objeto do tipo throwable. Este objeto é o novo rastreamento de pilha da exceção. No entanto, o rastreamento de pilha mais antigo não é excluído. A sintaxe é public throwable fillinStackTrace();

Atividade 3 - Exceptions

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Exercício

1. Crie as seguintes exceções no pacote **erros**:
   a. **EstouroSaqueException** - Acontece quando é feita uma tentativa de tirar mais dinheiro do que a conta possui.
   b. **DepositoInvalidoException** – Acontece quando uma quantia inválida de dinheiro (quantia < zero) é depositada.
2. Reescreva as operações de saque e depósito para lançar estas exceções.

Assertion Statements

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Uma **assertion** é uma construção de linguagem de programação que verifica se uma expressão especificada é verdadeira.
  - As asserções são usadas para ajudar o programador a melhorar a qualidade do código. A verificação feita usando asserções não faz parte da lógica de código real.
  - As asserções podem ser usadas para:
  - Valide as pré-condições antes de inserir uma seção do código.
  - Valide as pós-condições após executar uma seção do código.
  - Validando invariantes de classe sempre que o estado do objeto é modificado.

Using Assert Statements

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Assertions pode ser inserido em qualquer lugar do código usando a seguinte sintaxe:
  - assert <boolean expression>
  - assert<boolean expression> : <String expression>

- Se a expressão booleana for falsa, à instrução lançará um AssertionError e exibirá a expressão String (se especificada);

Refer to the AssertSample.java sample code.

Assertion Sample Code


```
import java.util.Scanner

   public class AssertTest
    {
         public static void main( String args[] )
         {
              Scanner input = new Scanner( System.in );
              System.out.print( "Enter a number between 0 and 10: " );
              int number = input.nextInt();

              // assert that the absolute value is between 1-10
              assert ( number > 0 && number <= 10 ) : "bad number: " + number;

              System.out.printf( "You entered %d\n", number );
         } // end main

    } // end class AssertTest
```

OUTPUT 1	OUTPUT 1
Enter a number between 0 and 10:5.	Enter a number between 0 and 10:50
You entered 5	Exception in thread “main” java.lang.AssertionError: bad number: 50 at AssertTest.main(AssertTest.java:15)

Enabling/Disabling Assertions

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Para habilitar assertions em tempo de execução, use os seguintes comandos:
  - java enableassertion <java class file> OR
  - java –ea <java class file>
  - E.g., java –ea AssertionMain
- Para desativar assertions em tempo de execução, use os seguintes comandos:
  - java disableassertion <java class file> OR
  - java –da <java class file>
  - E.g., java –da AssertionMain
- Para ativar assertions no tempo de execução (no Eclipse), use os seguintes comandos:
  - Right-click on the file and select Run As >Run Configurations
  - Click on the Arguments tab
  - In the VM arguments text box, enter -ea

Activity – Exception Sequence

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- Nesta atividade, você :
  - Abra o arquivo ‘ExceptionSequenceActivity.java’ no pacote sef.module8.activity.
  - Leia as instruções e crie o código para concluir este programa.

Questions and Comments

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- What questions or comments do you have?