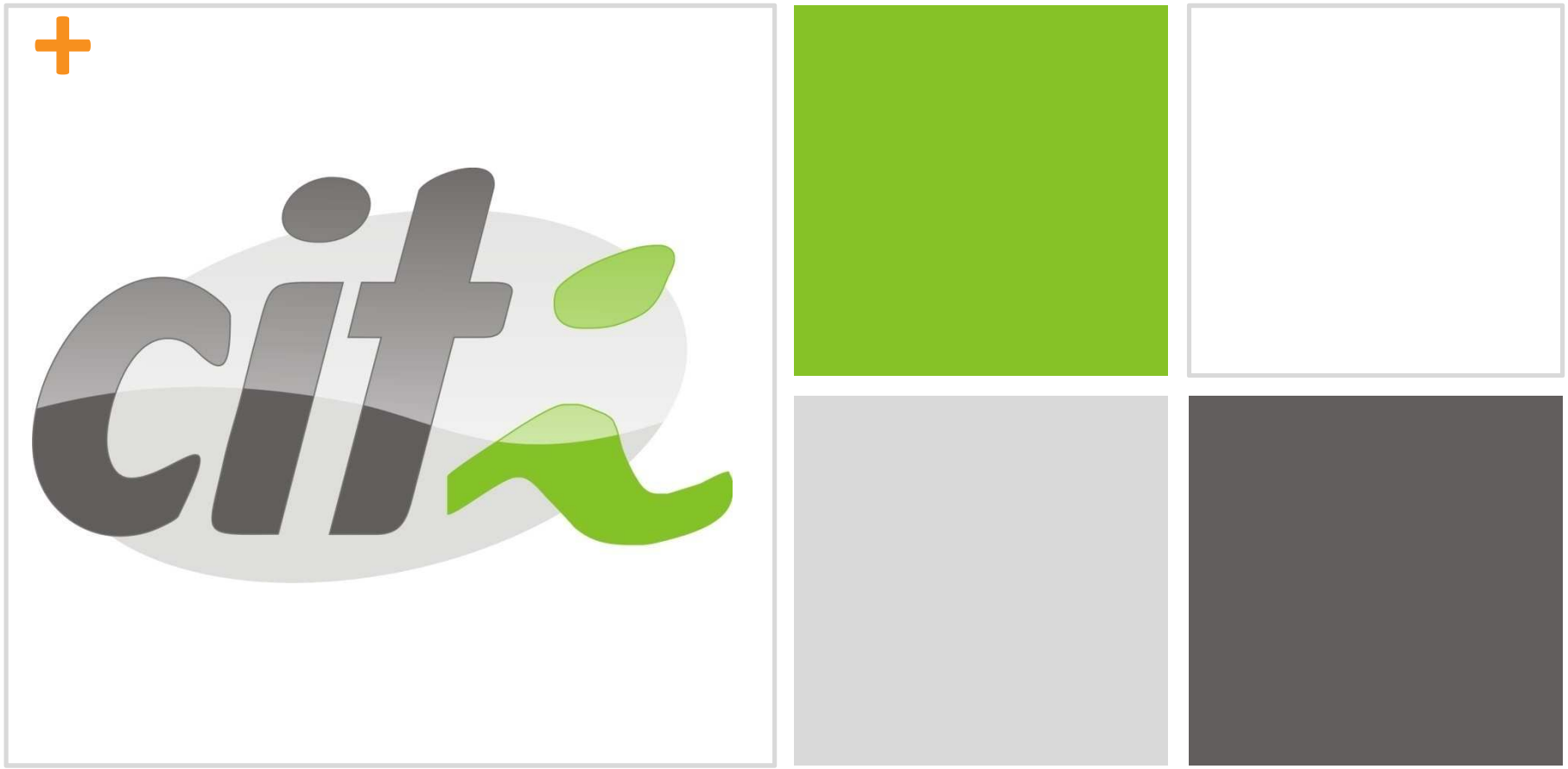


Java Básico

Igor Ebrahim (ies@cin.ufpe.br)

Módulo 8

Exceções

Robustez

- Recuperar-se de falhas
- Informar sobre os erros
- Validar os dados
- Evitar que situações indesejadas ocorram
- Garantir a consistência das operações

Exceções

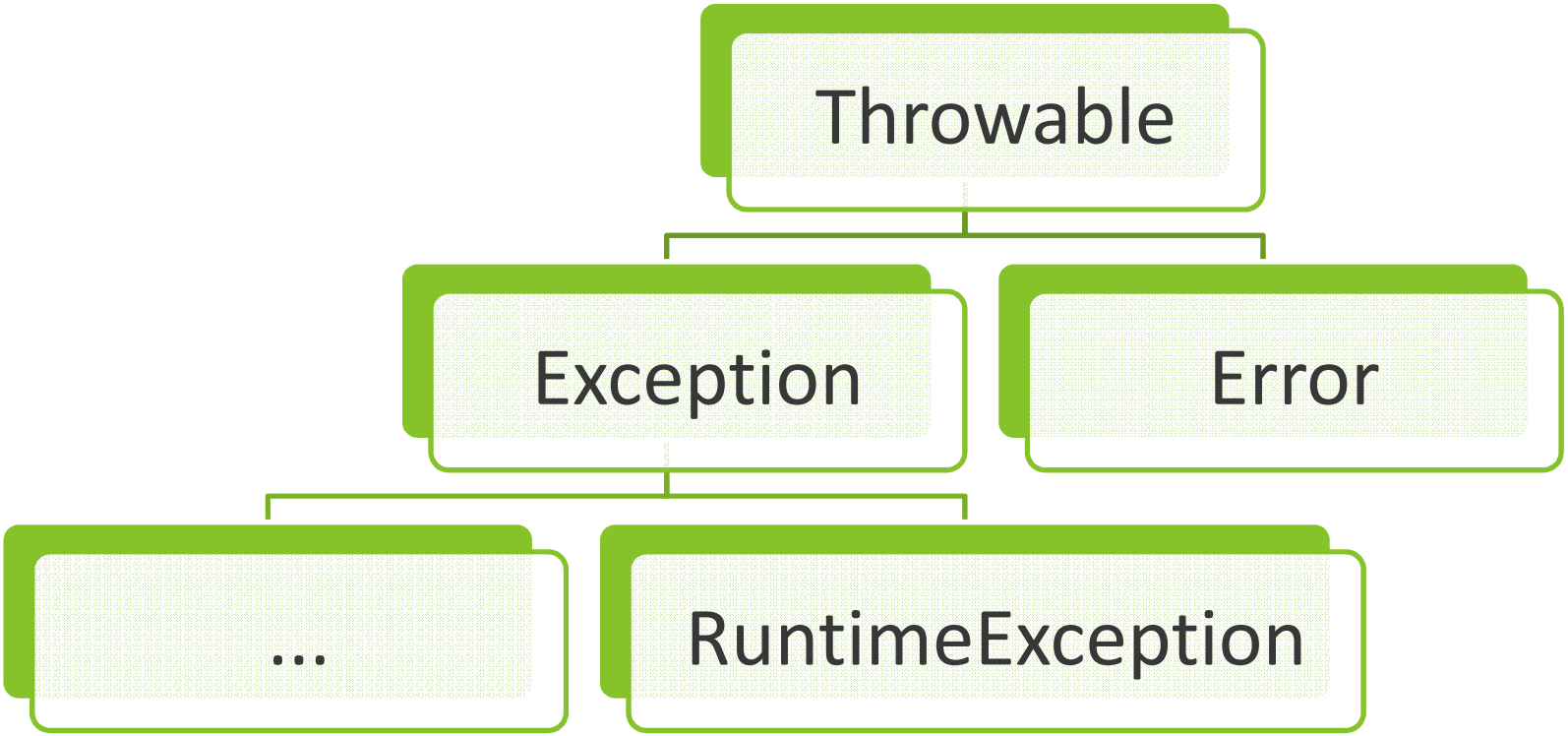
- Mecanismo utilizado por Java para tratamento de erros e situações indesejadas
  - Erros de Programação:
    - Acesso a uma posição inválida de um array
    - Invocação de um método em uma referência nula
  - Situações Indesejadas:
    - Conexão com o servidor de banco de dados falhou
    - Ausência de um arquivo

Exceções

- Também são classes em Java

- As exceções podem ser:
  - Declaradas
  - Lançadas
  - Tratadas

Tipos de Exceções



Exceções

Tipos de Exceções

- Throwable
  - É a super classe de todas as exceções
- Error
  - Erros internos da máquina virtual
- Exception
  - Devem ser declaradas e tratadas
- RuntimeException
  - Erros de programação
  - Não precisam ser declaradas

Exceções

Exceções Não Checadas

- Exceções do tipo **RuntimeException**
- Qualquer método pode gerar essas exceções apesar de não explicitar isto em sua definição
- Tratar estas exceções é tentar corrigir um erro de programação durante a programação não faz muito sentido
- O que se faz é capturar essa exceção e apresentar uma mensagem de erro agradável ao usuário indicando esta ocorrência

Exceções

Exceções Checadas

- Exceções do tipo **Exception**, exceto RuntimeException e suas subclasses
- Devem ser declaradas e tratadas no código

- Exceções podem ser definidas pelo programador e devem ser subclasses de Exception
  - Oferecer informações extra sobre o erro
  - Específicas para uma dada aplicação (exceções de negócio)

Exceções

- Por convenção, é aconselhável que o nome de qualquer exceção definida pelo programador tenha o sufixo Exception:
  - SaldoInsuficienteException
  - ObjetoInvalidoException

```
public class NomeDaExcecaoException extends Exception {
	public NomeDaExcecaoException() {

	}
	// ...
}
```

Exceções

Exemplo de Exceção

```
public class SaldoInsuficienteException extends Exception {

	public SaldoInsuficienteException() {
		super(“Saldo Insuficiente!”);
	}

	// ...

}
```

Exceções

Exemplo de Exceção

```
public class SaldoInsuficienteException extends Exception {

        private double saldo;
        private String numero;

        public SaldoInsuficienteException(double saldo,
                String numero) {
                super("Saldo Insuficiente!");
                this.saldo = saldo;
                this.numero = numero;
        }

        public SaldoInsuficienteException() {
                super("Saldo Insuficiente");
        }

        // ... Getters & Setters

}
```

Exceções

Declaração e Lançamento de Exceções

- Declaradas na assinatura dos métodos, que devem tratar um dado processamento, usando o comando **throws**
- Exceções são lançadas no corpo dos métodos usando o comando **throw**

```
public void debitar(double valor)
                throws SaldoInsuficienteException {
        // ...
        throw new SaldoInsuficienteException();
}
```

Exceções

Quando Lançar Exceções

- Um método que lança exceções é chamado
- Quando é detectada uma situação de erro e uma exceção é levantada com throw
- Erros de programação ocorrem (Java levanta a exceção)
- Erro interno ocorre em Java

```
public void debitar(double valor)
                throws SaldoInsuficienteException {
        // ...
        throw new SaldoInsuficienteException();
}
```

Exceções

Declaração e Lançamento de Exceções

- Se a exceção não for tratada em lugar nenhum, Java assume o controle e pára a aplicação

Exceções

Lançamento de Exceções

```
public class Conta {

   // ...

   public void debitar(double valor)
                throws SaldoInsuficienteException {
        if (valor <= saldo) {
                saldo = saldo – valor;
        } else {
                throw new SaldoInsuficienteException(numero, saldo);
        }
   }

}
```

Exceções

Lançamento de Exceções

```
public class Conta {

   // ...

   public void transferir(Conta c, double v)
               throws SaldoInsuficienteException {
       this.debitar(v);
       c.creditar(v);
   }

}
```

Exceções

Tratamento de Exceções

- Exceções são tratadas usando blocos **try-catch**

```
try {
  // chamada aos métodos que
  // podem lançar exceções
} catch (Exception e) {
  // código para tratar um tipo de exceção
}
```

Exceções

Tratamento de Exceções

```
public class CadastroConta {

   private RepositorioContas contas;

   // ...

   public void debitar(String n, double v)
               throws SaldoInsuficienteException,
                      ContaInexistenteException {
       Conta c = contas.procurar(n);
       c.debitar(v);
   }

}
```

Exceções

Tratamento de Exceções

```
public static void main(String args[]) {
   try {
       CadastroContas contas;
       // ...
       contas.debitar("123-0", 250.0);
       System.out.println("Débite efetuado");
   } catch (SaldoInsuficienteException e) {
       System.out.println(e.getMessage());
   } catch (ContaInexistenteException e) {
       System.out.println(e.getMessage());
   }
}
```

Exceções

Tratamento de Exceções

- A execução do **try** termina ao final do bloco ou assim que uma exceção é levantada
- O primeiro **catch** de um supertipo da exceção é executado e o fluxo de controle passa para o código seguinte ao último catch
- Exceções mais específicas devem ser capturadas primeiro. Caso contrário um erro de compilação é gerado

Exceções

Usando finally

- Trecho de código com finally sempre é executado, independente de ter havido ou não exceção

```
try {
   // ...
} catch (Exception e) {
   // ...
} finally {
   // Sempre executado
}
```

Prática

1. Refatore o seu sistema bancário, aplicando o conhecimento sobre exceções. Devem existir as seguintes exceções:
   - SaldoInsuficienteException;
   - ClienteNaoExisteException;
   - ClienteJaExisteException;
   - ContaNaoExisteException;
   - ContaJaExisteException;
2. Pense em outras exceções que poderiam ser utilizadas.