UFAPE
UNIVERSIDADE FEDERAL DO AGRESTE DE PERNAMBUCO

Programação Orientada a Objetos

Prof. Daliton da Silva

daliton.silva@ufape.edu.br

Tratamento de Exceções

Motivação e Conceito

- Exceções representam *condições anormais* ou *falhas de fluxo* que interrompem a execução normal
- Fornecem mecanismo estruturado para **capturar**, **propagar** e **recuperar** de erros

Hierarquia Padrão

- Throwable
- ├── Error (não capturar – falhas graves da VM)
- └── Exception
  - ├── Checked (IOException, SQLException…)
  - └── RuntimeException (unchecked – NullPointerException, IllegalArgumentException…)

Checked × Unchecked

- Checked: obrigam a declaração (throws) ou captura
  - Representam falhas recuperáveis previstas
- Unchecked: herdam de RuntimeException
  - Indicam erros de programação ou estados ilegais

Declaração de Exceções (throws)

```
public void read() throws IOException, ParseException
{ ... }
```

- Faz parte da assinatura pública
  - Influencia sobrecarga
  - Subclasses só podem declarar exceções iguais ou mais específicas.

Try - Catch

```
try {
    // código suscetível
} catch (FileNotFoundException e) {
    // tratamento específico
} catch (IOException e) {
    // genérico
} finally {
    // executa sempre (libera recurso)
}
```

Propagação & Empilhamento

- Exceções “sobem” a pilha até encontrar catch compatível
- StackTrace registra cadeia de chamadas → importante para depuração.

Rethrow e Exception Translation

```
catch (SQLException e) {
    throw new DataAccessException("Erro no BD", e); //
encapsula
}
```

Exceções Personalizadas

- Nome e mensagem claros; herdar de Exception ou RuntimeException conforme política

```
public class SaldoInsuficienteException extends
RuntimeException {
    public SaldoInsuficienteException(double saldo) {
        super("Saldo insuficiente: " + saldo);
    }
}
```