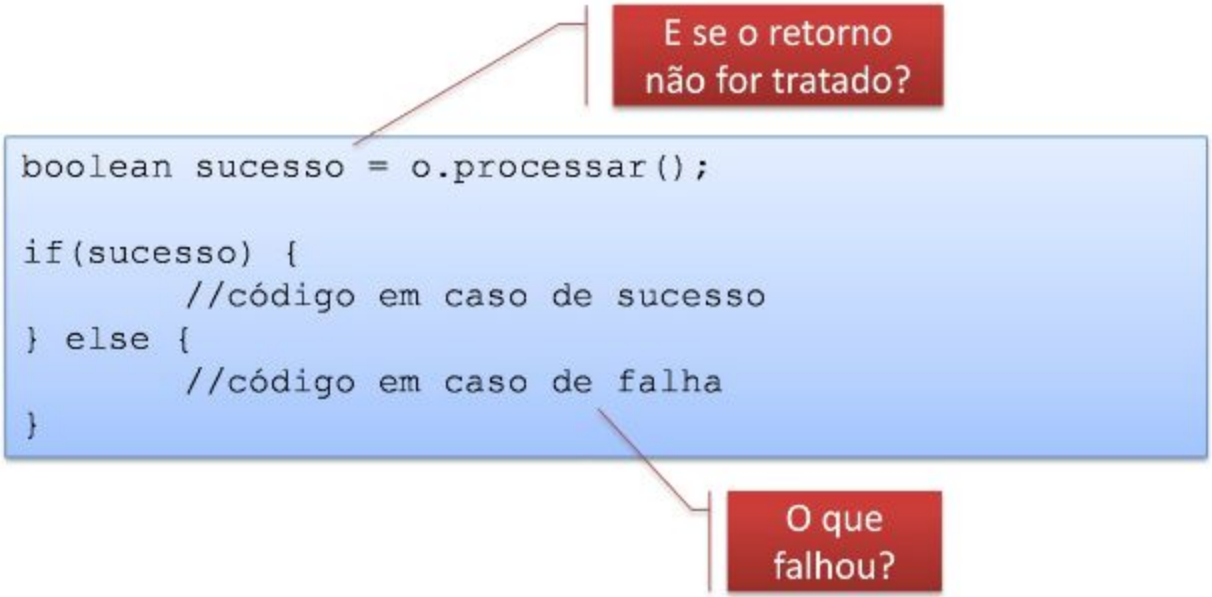
Aula: Tratamento de Exceções

Instituto Metrópole Digital - UFRN
Disciplina: LP2
Docente: Emerson Alencar

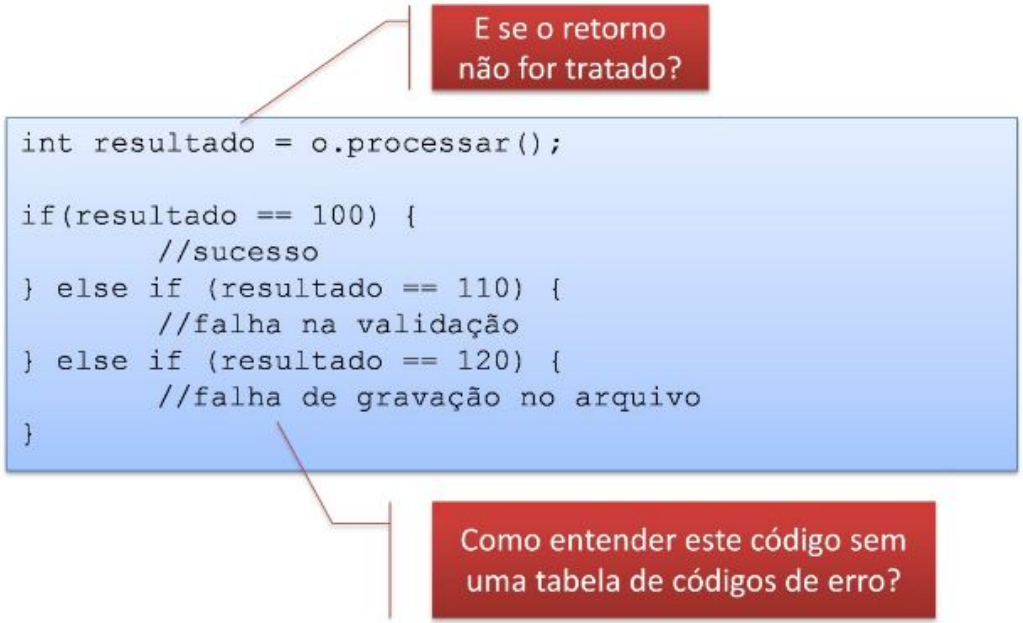
Avisando sobre falhas em métodos

- Situação: Você precisa avisar quem chamou o método e informar que o método não executou como deveria. Como fazer?
- As abordagens mais comuns são:
  - Usar booleanos
  - Usar magic numbers

Problema de usar Booleanos



Problema de usar Magic Numbers



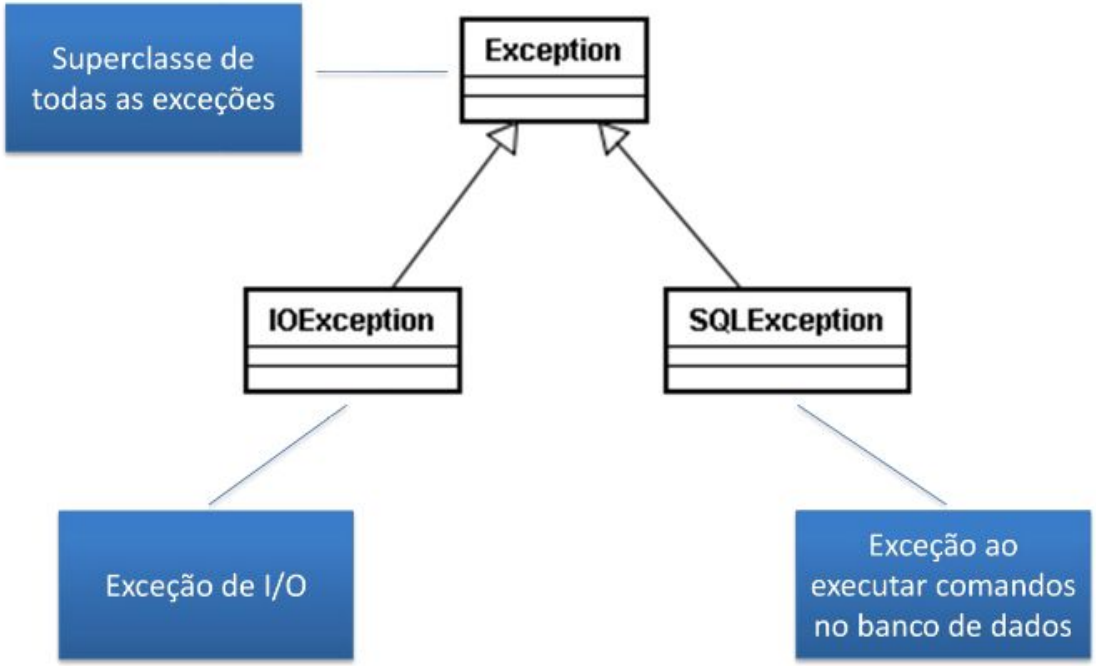
Exceções

- Exceções representam algo estranho ao sistema que normalmente não ocorre
  - Tenho um comportamento anormal do meu código
- Em Java, o tratamento de exceções é feito por um código diferente do código executado quando não ocorre a exceção.

Classes que representam Exceções

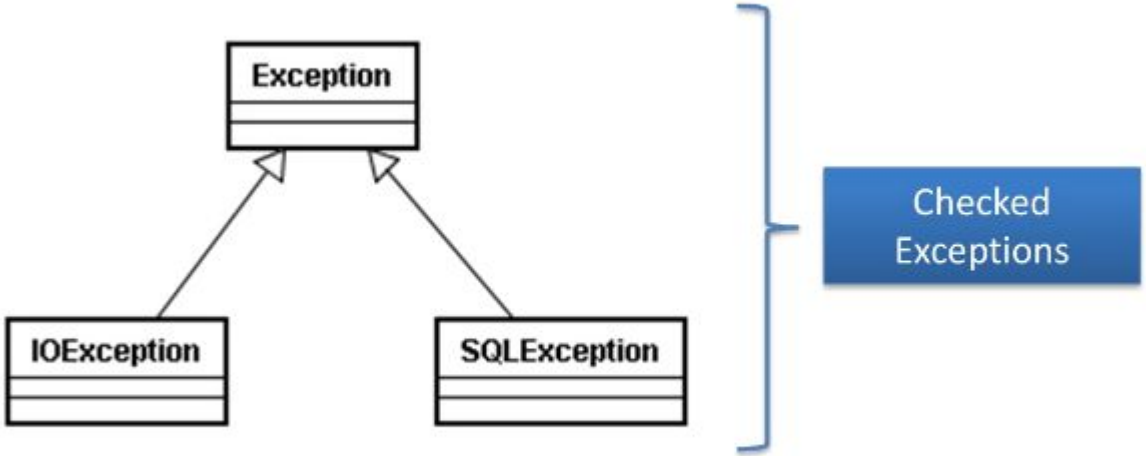
- Exceções são representadas por classes
- As classes devem herdar direta ou indiretamente de *Exception*
- O java tem classes que representam diversos tipos de exceção, mas o programador pode criar exceções específicas de acordo com a necessidade

Exemplos de Exceções



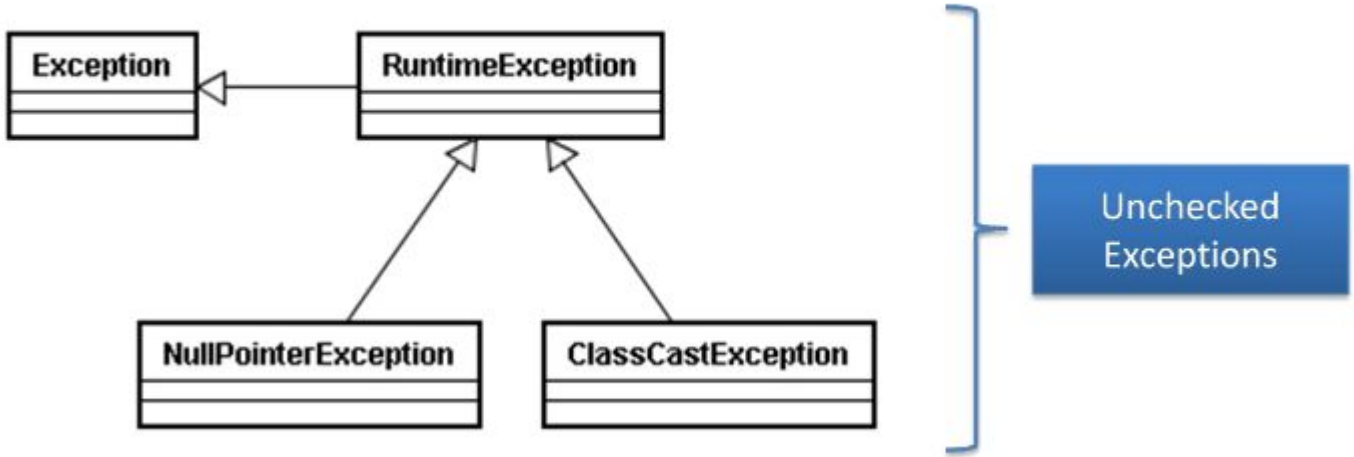
Checked Exceptions

- Herdam direta ou indiretamente de Exception
- Só não podem herdar de RuntimeException

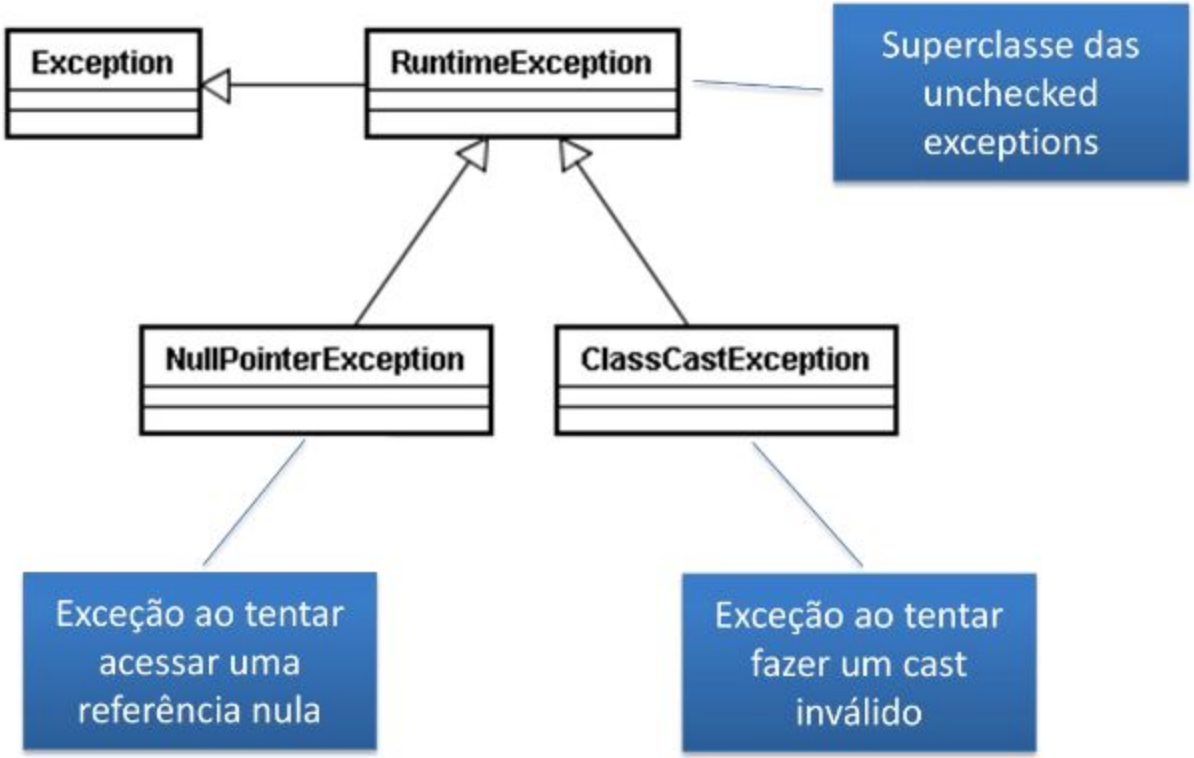


Unchecked Exceptions

- Também são chamadas de *runtime exceptions*
- Herdam direta ou indiretamente de *runtimeException*

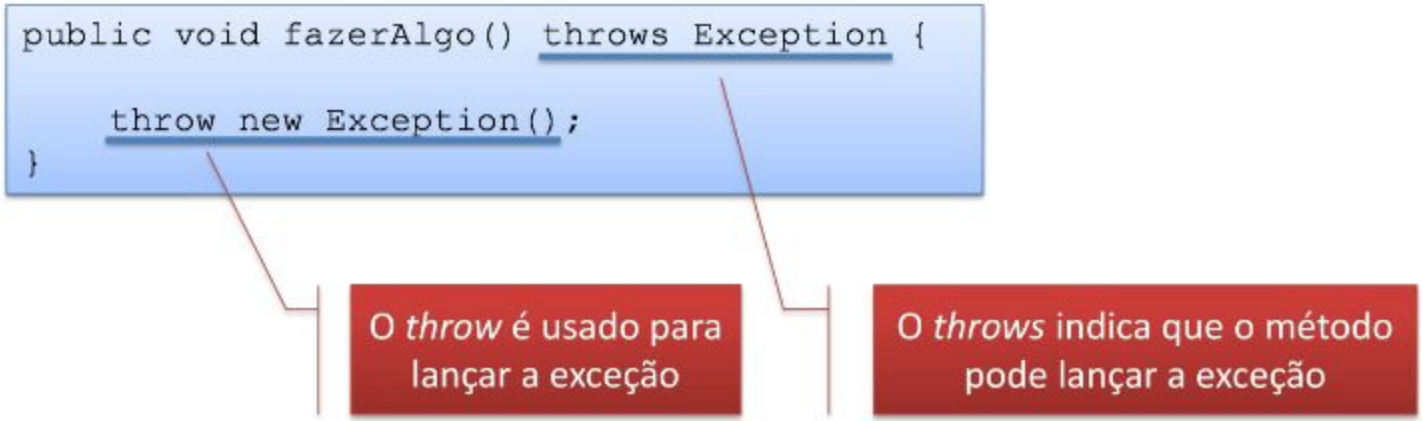


Unchecked Exceptions



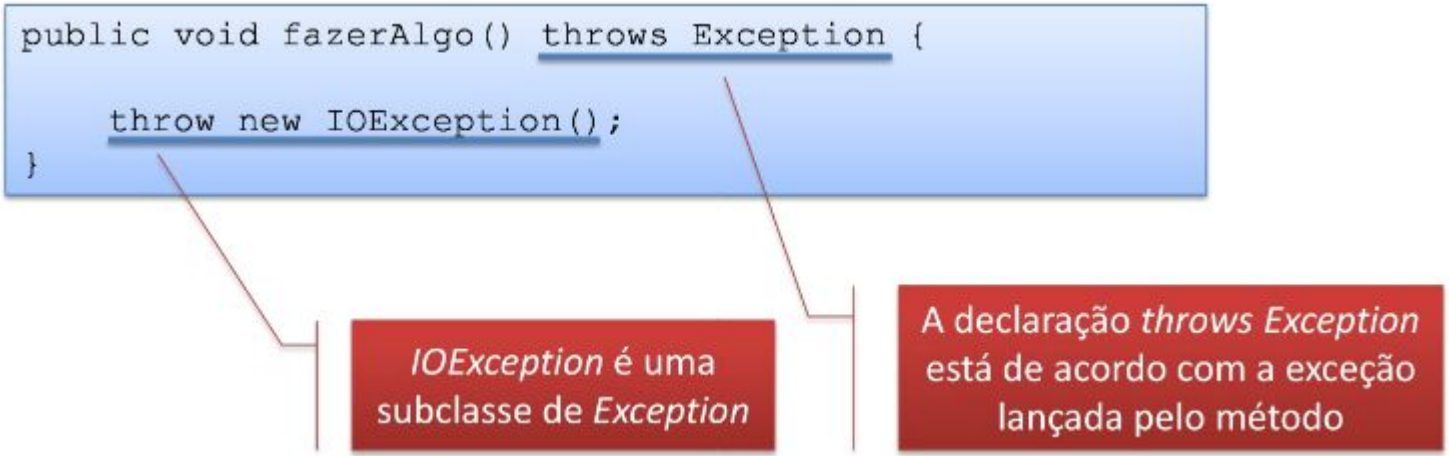
Lançando Exceções

- O lançamento de exceções é feito através do *Throw*



Lançando Exceções

- É possível também lançar subclasses da exceção declarada pelo *throws*



Tratando Exceções

- Exceções podem ser tratadas através do uso do bloco *try/catch*
  - Determinado código tenta (try) executar um método e, caso alguma exceção aconteça, ele pega (catch) a exceção ocorrida e faz o que deseja.
- Após uma exceção ter alcançado o bloco catch, o código volta o seu fluxo normal de execução

Tratando Exceções

```
public void m1() throws Exception {
    throw new Exception();
}
```

```
public void m2() {
    try {
        m1();
    } catch (Exception e) {
        ...
    }
    ...
}
```

Se uma *Exception* acontecer, o fluxo é desviado para o bloco *catch*

Ao fim do bloco *catch*, a execução continua após o bloco

Tratando Múltiplas Exceções

```
public void m1() throws IOException, SQLException {
    ...
}
```

```
public void m2() {
    try {
        m1();
    } catch (IOException e) {
        ...
    } catch (SQLException e) {
        ...
    }
    ...
}
```

Dependendo da exceção, o bloco *catch* correspondente é executado

No máximo um bloco *catch* é executado

Multi-Catch

- É possível fazer o catch de mais de uma exceção ao mesmo tempo;

```
try {
    m();
} catch (MyException1 e) {
    ...
} catch (MyException2 e) {
    ...
} catch (MyException3 e) {
    ...
}
```

```
try {
    m();
} catch (MyException1 | MyException2 | MyException3 e) {
    ...
}
```

Ordem da Exceções no Catch

```
public void m1() throws IOException {
    throw new IOException();
}
```

```
public void m2() {
    try {
        m1();
    } catch (Exception e) {
        ...
    } catch (IOException e) {
        ...
    }
    ...
}
```

Toda exceção será tratada por este bloco *catch*

Este bloco *catch* nunca será executado

Tratando e Lançando Exceções

```
public void m1() throws IOException, SQLException {
    ...
}
```

```
public void m2() throws IOException {
    try {
        m1();
    } catch (SQLException e) {
        ...
    }
    ...
}
```

Apenas a *SQLException* é tratada. A *IOException* é lançada para quem chamou *m2()*

Transformando Exceções

```
public void m1() throws IOException {
    ...
}
```

```
public void m2() throws AppException {
    try {
        m1();
    } catch (IOException e) {
        throw new AppException();
    }
    ...
}
```

A *IOException* é relançada como uma *AppException*

Lançando Unchecked Exceptions

- Estas exceções normalmente são provocadas por problemas de programação, não devendo ser tratadas
- Por este motivo, o Java não obriga o programador a tratar as unchecked exceptions

Lançando Unchecked Exceptions

```
public void m1() {
    throw new NullPointerException();
}
```

O *throws* não é necessário, pois *NullPointerException* é unchecked

```
public void m2() {
    m1();
}
```

Não é necessário tratar a exceção. Caso ela ocorra, será lançada automaticamente para quem chamou *m2()*

Informações

- Se uma exceção for lançada pelo método main(), a JVM termina
- Exceções muito genéricas dificultam no entendimento do problema