FACEC
Faculdade

Programação II

Tratamento de Exceções

Prof. Emmanuel Neri

2019

FACEC
Faculdade

Cronograma

- Tipos de exceções
- Tratamentos

FACEC
Faculdade

Exceções

“uma indicação de um problema que ocorre durante a execução de um programa”

DEITEL

2019

FACEC
Faculdade

Tratamento

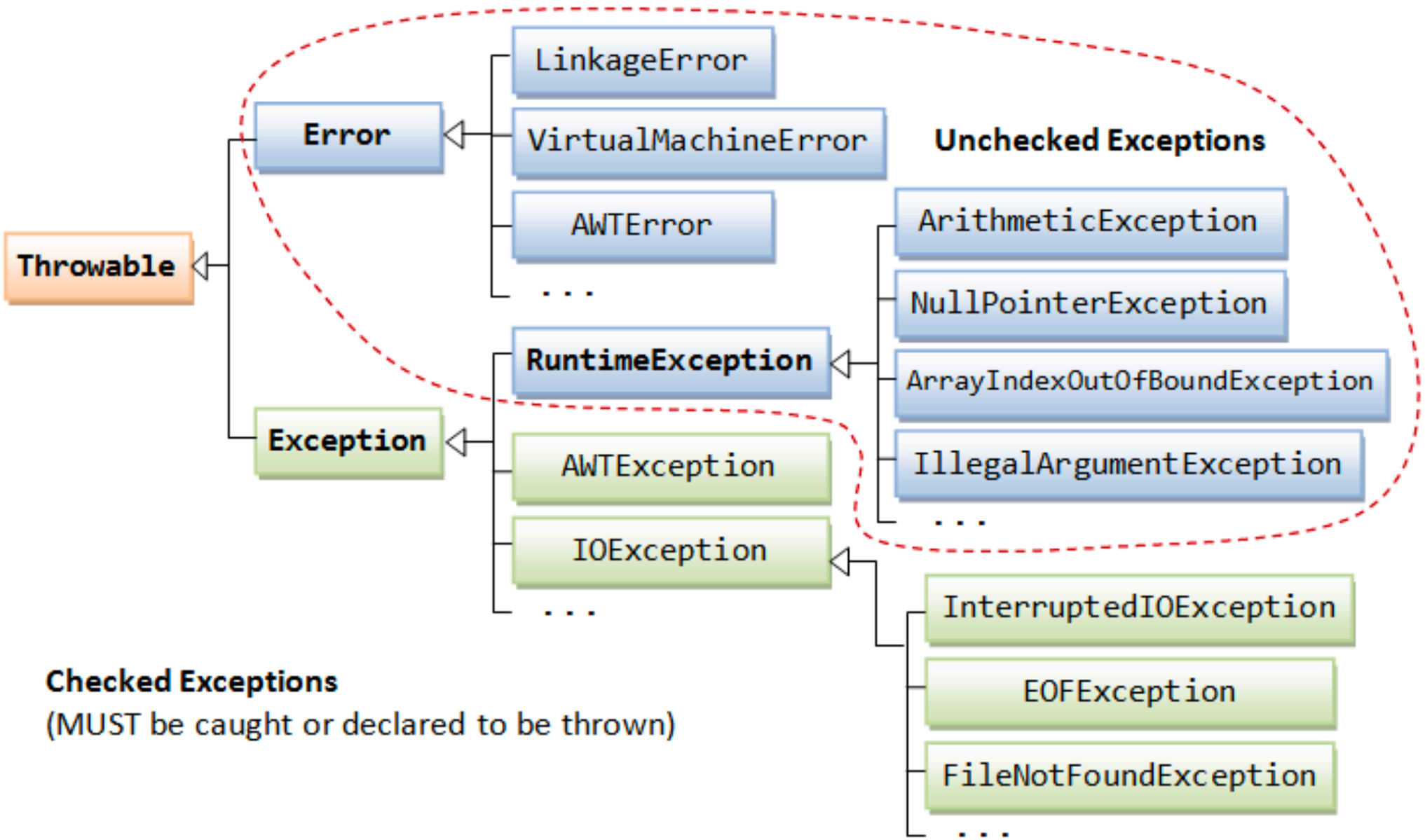
“resolver exceções que poderiam ocorrer para que o programa continue ou termine elegantemente"

Paul J. Deitel

2019

FACEC
Faculdade

Exceções em Java



Checked Exceptions
(MUST be caught or declared to be thrown)

FACEC
Faculdade

Throwable

- getMessage();

- PrintStackTrace();

FACEC
Faculdade

Tipos

- verificadas (checked)

- não verificadas (unchecked)

FACEC
Faculdade

Exceções verificadas

- O compilador impõe a restrição capturar ou sinalizar para exceções verificadas

- Toda exceção que estende Exception e não estende RuntimeException

- Portanto, exceções verificadas devem ser capturadas (catch) ou sinalizadas (throws)

FACEC
Faculdade

Exceções não verificadas

- Exceções que não precisam ser sinalizadas, capturadas ou tratadas

- O compilador Java não verifica se tais exceções podem ser lançadas Exceções não verificadas devem herdar direta ou indiretamente de RuntimeException

- Error são consideradas exceções não verificadas

FACEC
Faculdade

try/catch

```
try {

} catch (Exception ex) {

}
```

FACEC
Faculdade

try/catch

```
try {

} catch (NullPointerException npex) {

} catch (Exception ex) {

}
```

FACEC
Faculdade

try/catch

```
try {
    Integer.valueOf("10.2");
} catch (Exception ex) {
    ex.printStackTrace();
}
```

```
java.lang.NumberFormatException: For input string: "10.2"
    at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.valueOf(Integer.java:766)
    at br.com.emmanuelneri.excecoes.Teste.main(Teste.java:7)
```

FACEC
Faculdade

Finally

```
try {

} catch (Exception ex) {

} finally {
    System.out.println(“sempre executa”);
}
```

FACEC
Faculdade

throw

```
if(valor < 0) {
    throw new RuntimeException("Valor precisa
ser maior que zero");
}
```

FACEC
Faculdade

throws

```
public static void validaNumero(int valor) throws Exception {
    if(valor < 0) {
        throw new Exception("Valor precisa ser maior que zero");
    }
}
```

FACEC
Faculdade

Exceções personalizadas

```
public class ValorInvalidoException extends Exception {
}
```

FACEC
Faculdade

Exceções personalizadas

```
public class ValorInvalidoException extends Exception {

    public ValorInvalido(String message) {
        super(message);
    }
}
```

```
public class ValorInvalidoException extends Exception {

    public ValorInvalido() {
        super("Valor inválido");
    }
}
```

FACEC
Faculdade

Exceções personalizadas

```
try {
    validarValor(valor);
} catch (Exception ex) {
        }
```

```
try {
    validarValor(valor);
} catch (ValorInvalidoException viex) {

}
```

FACEC
Faculdade

Conclusão

- Sempre devemos tratar nossos códigos
- Sempre que possível criei uma exceção específica para o problema;

FACEC
Faculdade

Exercícios

- Criar um tratamento para evitar divisão por zero em uma classe que faz o calculo de distribuição de ingressos por pessoa;

- Criar um tratamento, com uma exceção específica, em uma classe conta para que o usuário não ultrapasse o limite da conta no banco;

- Criar um método que tenta converter uma String para Inteiro, caso der errado tente converter converter para double.

FACEC
Faculdade

Exercícios

- Criar hierarquia de exceção de pelo menos dois níveis abaixo de RuntimeException, após criada, criei um tratamento em que ela possa ser aplicada.