Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática

Declaração e lançamento de exceções

NOTAS DE AULA

Linguagem de Programação II
2º semestre de 2015
Prof. Tomaz Mikio Sasaki
Versão 2.0

Referência

As referências para esta aula são:

HORSTMANN, C.S.; CORNELL, G. **Core Java, Volume I - Fundamentals.** Prentice Hall, 2012.

Caelum - Excessões e controles de erros. Disponível em: http://www.caelum.com.br/apostila-java-orientacao objetos/excecoes-e-controle-de-erros

Tutorials Point - Java - Exceptions. Disponível em: http://www.tutorialspoint.com/java/java_exceptions.htm

Observação: As notas de aula são material de apoio para estudo e não têm o objetivo de apresentar o assunto de maneira exaustiva. Não deixe de ler o material de referência da disciplina.

Exceções personalizadas

É possível criar sua própria classe de exceção.

Exemplo de uma exceção personalizada *unchecked*:

```
public class SaldoInsuficienteException extends RuntimeException {

    public double excedente;

    public SaldoInsuficienteException(String message, double excedente) {
        super(message);
        this.excedente = excedente;
    }

    public double getExcedente() {
        return excedente;
    }

}
```

Lançamento de uma exceção

O lançamento de uma exceção é realizado utilizando a sentença **throw**.

Exemplo:

```
public class ContaCorrente {

  private double saldo;

  public ContaCorrente(double saldoInicial) { this.saldo = saldoInicial; }

  public void depositar(double deposito) { this.saldo += deposito; }

  public void sacar(double valor) {
    if (this.saldo < valor) {
      throw new SaldoInsuficienteException("Saldo Insuficiente", valor-saldo);
    } else {
      this.saldo -= valor;
    }
  }

}
```

Exemplo de uma exceção personalizada *checked*

```
public class SaqueNecessitaAutorizacaoException extends Exception {

}
```

Declaração de lançamento de exceção *checked*

Quando uma exceção *checked* pode ser lançada por um método, é necessário declarar isto na assinatura do método utilizando a sentença **throws**.

Exemplo:

```
public class ContaCorrente {

    private double saldo;
    public static double VALOR_MAXIMO_SAQUE = 100000;

    public ContaCorrente(double saldoInicial) { this.saldo = saldoInicial; }

    public void depositar(double deposito) { this.saldo += deposito; }

    public void sacar(double saque) throws SaqueNecessitaAutorizacaoException {
        if (this.saldo < saque) {
            throw new SaldoInsuficienteException("Saldo Insuficiente",
                                                 saque - saldo);
        } else {
            if (saque > VALOR_MAXIMO_SAQUE) {
                throw new SaqueNecessitaAutorizacaoException();
            }
            this.saldo -= saque;
        }
    }

}
```

Tratamento das exceções

Ao chamar o método **sacar**, o compilar avisa que é necessário tratar a exceção **SaqueNecessitaAutorizacaoException**.

```
public class SaqueApp {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Entre o saldo inicial da conta: ");
        double saldoInicial = entrada.nextDouble();
        ContaCorrente conta = new ContaCorrente(saldoInicial);
        System.out.print("Entre o valor do saque: ");
        double saque = entrada.nextDouble();
        conta.sacar(saque);
    }
}
```

O compilador indica um erro na linha destacada.

Tratamento das exceções (cont.)

Ao tratar a exceção **SaqueNecessitaAutorizacaoException**, o compilador deixa de apresentar a mensagem de erro.

```
public class SaqueApp {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Entre o saldo inicial da conta: ");
        double saldoInicial = entrada.nextDouble();
        ContaCorrente conta = new ContaCorrente(saldoInicial);
        System.out.print("Entre o valor do saque: ");
        double saque = entrada.nextDouble();
        try {
            conta.sacar(saque);
        } catch (SaqueNecessitaAutorizacaoException ex) {
            System.out.println("Valor do saque excede o máximo permitido." +
                               "Solicite autorização para seu gerente.");
        }
    }
}
```

No entanto, a exceção **SaldoInsuficienteException** pode ocorrer em tempo de execução.

Tratamento das exceções (cont.)

Apesar do compilador não obrigar o programador a tratar **SaldoInsuficienteException**, isto pode ser feito.

```
public class SaqueApp {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Entre o saldo inicial da conta: ");
        double saldoInicial = entrada.nextDouble();
        ContaCorrente conta = new ContaCorrente(saldoInicial);
        System.out.print("Entre o valor do saque: ");
        double saque = entrada.nextDouble();
        try {
            conta.sacar(saque);
        } catch (SaqueNecessitaAutorizacaoException ex) {
            System.out.println("Valor do saque excede o máximo permitido." +
                               "Solicite autorização para seu gerente.");
        } catch (SaldoInsuficienteException ex) {
            System.out.println("Seu saque excede seu saldo em " +
                               ex.getExcedente());
        }
    }
}
```

"Relançamento" de uma exceção

Se um método que você está implementando chama um método que lança uma exceção *checked* e você não quer tratar esta exceção no seu método, você pode "relançá-lo" utilizando a sentença **throws**.

Suponha que você adicionou o método **transferir** na sua classe **ContaCorrente**.

```
    public void transferir(ContaCorrente destino, double valor)
                                    throws SaqueNecessitaAutorizacaoException {
        this.sacar(valor);
        destino.depositar(valor);
    }
```

Note que você está usando o método **sacar** que lança a exceção *checked* **SaqueNecessitaAutorizacaoException**. Para não ter que tratar esta exceção com um *try-catch*, você pode fazer o "relançamento" da exceção utilizando **throws** na assinatura do método **transferir**.

Bom estudo!