Lab 9 – Exceções

## Laboratório Java SE

Neste laboratório faremos uso das estruturas de tratamento de **“erros”**, usando **Exceptions** e **Assertions**. A linguagem de programação Java possui uma estrutura de manipulação de exceção muito bem elaborado, que ajuda os desenvolvedores a separar a lógica de tratamento de exceções de sua lógica de negócios.

Sugerimos que estes exemplos sejam feitos com uso da IDE Eclipse.

### Duração prevista: 60 minutos

***Exercícios***

**Exercício 1**: Criando exceções (30 minutos)

**Exercício 2**: Usando exceções (30 minutos)

# *Exercício 1 – Criando exceções*

1. Quando sacamos e transferimos dinheiro de uma instância da classe **ContaService.java**, **Laboratório 5 exercício 3, item 1,** espera-se que o saldo seja suficiente para cobrir a operação, porém pode ocorrer uma situação de exceção, ou seja, situação inesperada. Neste caso podemos usar os mecanismos de controle de exceção em Java para contornar a situação. Veja o código abaixo, nele criamos uma nova classe de exceção **SaldoInsuficienteException.java**, para indicarmos que não há saldo suficiente para que uma operação saque/transferência ocorra.

**public class** SaldoInsuficienteException **extends** Exception {

**public** SaldoInsuficienteException() {

**super**("Saldo insuficiente.");

}

**public** SaldoInsuficienteException( String mensagem ) {

**super**(mensagem);

}

}

*Listagem 8.1 – SaldoInsuficienteException*

1. Modifique o método **transferir()** da classe **ContaService.java**, como na listagem abaixo, ou seja, quando tentar fazer uma operação em uma conta, que seja necessário verificar se possui saldo e o saldo for insuficiente, então à aplicação lançará uma exceção.

@Override

**public boolean** transferir(Conta destino, **double** valor) **throws** SaldoInsuficienteException{

**if** (getSaldo() - valor >= 0) { **this**.debito(valor); destino.credito(valor);

historicoTransacao(**this**, destino, valor, "transferencia para conta " +

destino.getNumero());

**return true**;

} **else** {

### 1

**Laboratório Java SE**

**throw new** SaldoInsuficienteException();

}

}

Listagem 8.2 – Exceção método transferir() classe ContaCorrente*.java*

1. A compilação da classe **ContaService.java** irá gerar um erro, isto se deve ao fato do método **transferir()** usado na sobrecarga como você já viu é necessário fazer o tratamento da exceção **SaldoInsuficienteException**. Modifique a classe **ContaService.java**, no método **transferir()** para desviar a exceção usando **throws** na assinatura do método. Faça o mesmo tratamento de exceção para o método **sacar().**

# *Exercício 2 – Usando exceções*

1. Crie a classe **MovimentoContaCorrente.java** conforme **Listagem-8.3**, para testar e manipular a exceção que acabou de ser criada e adicionado ao método **transferir().**

**public class** MovimentoContaCorrente {

**public static void** main(String[] args) {

// Cria uma instância de ContaService onde está presente as operações para Objeto Conta ContaService operacoesConta = **new** ContaService();

Conta correntista1 = **new** Conta("Aluno", 1001); Conta correntista2 = **new** Conta("Aluna", 21);

// faz deposito operacoesConta.depositar(correntista1, 100);

// faz Transferência proibida

**try** {

operacoesConta.transferir(correntista1, 600, correntista2);

} **catch** (Exception e) {

System.*out*.println(e.getMessage());

}

// faz Transferência autorizada

**try** {

operacoesConta.transferir(correntista1, 99.00, correntista2);

} **catch** (Exception e) {

System.*out*.println(e.getMessage());

}

ExtratoTXT movimento = **new** ExtratoTXT(correntista1); System.*out*.println(movimento.formatar());

}

}

Listagem 8.3 – MovimentoContaCorrente

### 2