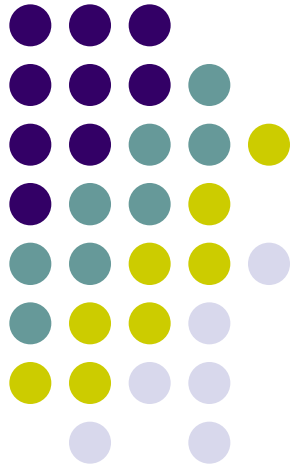
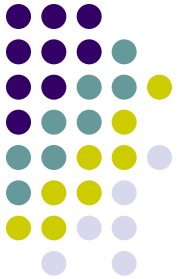
COM220
Aula 14: Tratamento de Exceções

Prof. Laércio Baldochi

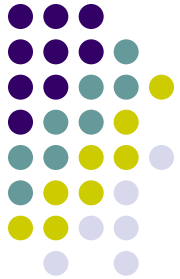




Conteúdo

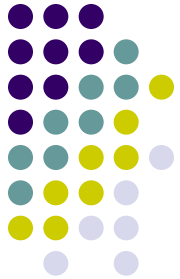
- O que é uma exceção
- Tratamento de exceções em Java
  - Blocos try... catch
  - Blocos try... catch... Finally
- Propagação de exceções
- Exemplos

O que é um exceção?



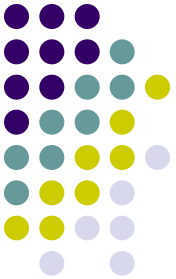
- É uma indicação de que ocorreu um problema durante a execução de um programa
- É uma condição anormal que surge em uma sequência de código ou em uma operação realizada
- O mecanismo de tratamento de exceções da linguagem Java ajuda a escrever programas mais claros, mais robustos e tolerantes a falhas

Exceção
Quando ocorre?



- Pode ocorrer quando
  - É feita manipulação de estruturas fora de um intervalo
    - Ex: acesso de um índice inválido de um vetor
  - For realizada a tentativa de abertura de um arquivo que não existe
  - Uma conexão de rede for interrompida
  - Um banco de dados reportar problemas de acesso
    - Tentativa de manipular uma tabela que não existe, ou um campo que não existe em uma tabela
  - Etc...

try ... catch

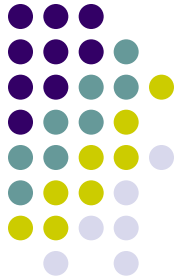


- O tratamento de exceções em Java é feito através de blocos try ... catch
- Podem existir várias declarações de catch
  - Uma para cada exceção que se deseja capturar

```
try {
  //bloco de código
} catch (TipoDeExceção referênciaParaExceção) {
    //código a ser executado caso a exceção geral seja disparada
}
```

- Se um bloco *try* é executado e nenhuma exceção é disparada, todos os tratadores de exceções são desconsiderados e o controle é retomado na primeira instrução após a finalização do bloco

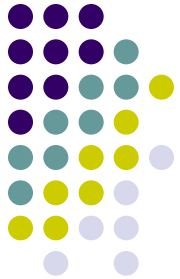
finally



- Quando programas utilizam recursos externos, deve-se ter o cuidado de retornar tais recursos ao final da execução
- A função do bloco finally é garantir a execução de um trecho de código, independentemente da ocorrência de exceções

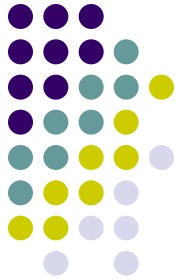
```
try {
  //bloco de código
} catch (TipoDeExceção referênciaParaExceção) {
    //código a ser executado caso a exceção geral seja disparada
} finally {
   //código a ser executado independente da exceção
}
```

Importante
Diferença entre erro e exceção



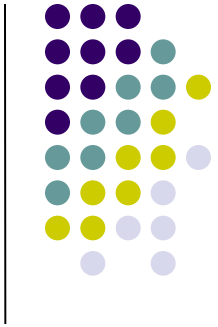
- Erro
  - Em geral um problema grave e difícil ou impossível de ser tratado
    - Perda de ponteiro
- Exceção
  - Pode ser causada por erro de projeto ou de implementação
    - ArrayIndexOutOfBoundsException
  - Pode ser causada por falha ou indisponibilidade temporária de recursos
    - SQLException

Capturando exceções



- Os tratadores de exceção estão contidos em blocos catch
- Cada bloco catch inicia com a palavra-chave catch seguida por parênteses que contém um nome de classe (especifica o tipo de exceção a ser capturado) e um nome de parâmetro
- O tratador de exceção pode fazer referência ao objeto disparado através desse parâmetro
- Após a chave pode-se descrever o código que irá tratar a exceção

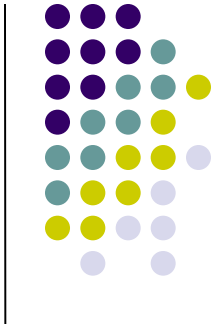
Capturando exceções
Exemplo



```
import java.util.*;

public class ExemploException1 {

  private Vector v = new Vector();

  public ExemploException1() {
    v.add("Disciplina 1");
    v.add("Disciplina 2");
    imprimeVetor();
  }

  public void imprimeVetor()  {
    System.out.println("O número de elementos do vetor é "+v.size());
    for (int i = 0; i <= v.size(); i++) {
      System.out.println(v.elementAt(i).toString());
    }
  }

  public static void main (String par[]) {
    ExemploException1 exemploExc = new ExemploException1();
  }
}
```

Irá produzir a exceção ArrayIndexOutOfBoundsException

Capturando exceções Exemplo



```
import java.util.*;

public class ExemploException1 {
  private Vector v = new Vector();
  public ExemploException1() {
    v.add("Disciplina 1");
    v.add("Disciplina 2");
    imprimeVetor();
  }

 public void imprimeVetor()  {
    System.out.println("O número de elementos do vetor é "+v.size());
    try {
      for (int i = 0; i <= v.size(); i++) {
        System.out.println(v.elementAt(i).toString());
      }
    } catch (ArrayIndexOutOfBoundsException exc) {
      System.out.println("Índice fora do limite");
    }
  }

  public static void main (String par[]) {
    ExemploException1 exemploExc = new ExemploException1();
  }
}
```

A exceção ArrayIndexOutOfBoundsException será capturada no bloco try/catch

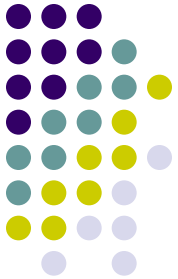
Capturando exceções Exemplo

```
import java.util.*;
public class ExemploException1 {
  private Vector v = new Vector();
  public ExemploException1() {
    v.add("Disciplina 1");
    v.add("Disciplina 2");
    imprimeVetor();
  }
  public void imprimeVetor()  {
    System.out.println("O número de elementos do vetor é "+v.size());
    try {
      for (int i = 0; i <= v.size(); i++) {
        System.out.println(v.elementAt(i).toString());
      }
    } catch (ArrayIndexOutOfBoundsException exc) {
      System.out.println("Índice fora do limite -> "+exc.getMessage());
    } finally  {
      System.out.println("Processo finalizado");
    }
  }
  public static void main (String par[]) {
    ExemploException1 exemploExc = new ExemploException1();
  }
}
```

A exceção ArrayIndexOutOfBoundsException será capturada no bloco try/catch e o bloco *finally* sempre será executado.

Através do objeto exc (instância de ArrayIndexOutOfBoundsException) é possível recuperar maiores detalhes do erro (método getMessage()).

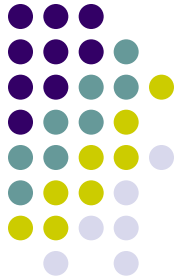
Propagação de exceções throws



- Permite que uma exceção gerada em um determinado método seja propagada para o método chamador

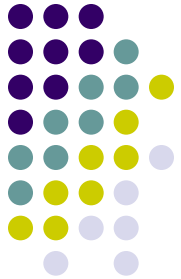
```
public void metodo() throws ExcecaoGeral {
   //código – aqui pode ocorrer uma exceção
}
public void metodoChamador() {
  try {
     //código com chamada ao método() com “throws”
  } catch (TipoDeExceção referênciaParaExceção) {
     // aqui a exceção é tratada  }
  } finally { //código }
}
```

Disparando exceções



- Uma exceção também pode ser disparada através da instrução *throw*
  - A instrução *throw* é utilizada explicitamente para lançar uma exceção provocando a interrupção do fluxo de execução
- O TipoDeExceção pode ser de qualquer classe Throwable (pacote java.lang)
  - As duas subclasses imediatas são Exception e Error

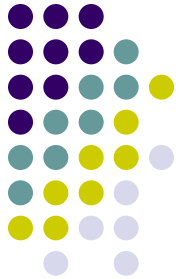
Disparando exceções
Exemplo 1



```
import java.util.*;
public class ExemploThrow {
  private Vector v = new Vector();
  public ExemploThrow() throws Exception {
    v.add("Disciplina 1");
    v.add("Disciplina 2");
    imprimeVetor();
  }
  public void imprimeVetor() throws Exception {
    System.out.println("O número de elementos do vetor é "+v.size());
    try {
      for (int i = 0; i <= v.size(); i++) {
        System.out.println(v.elementAt(i).toString());
      }
    } catch (ArrayIndexOutOfBoundsException exc) {
      throw new Exception("Índice fora do limite -> "+exc.getMessage());
    } finally  {
      System.out.println("Processo finalizado");
    }
  }
  public static void main (String par[]) {
    try {
      ExemploThrow explthrow = new ExemploThrow();
    } catch (Exception exc) {
      System.out.println(exc.getMessage());
    }
  }
}
```

A exceção ArrayIndexOutOfBoundsException é capturada e propagada através do comando ***throw.***

Disparando exceções
Exemplo 2



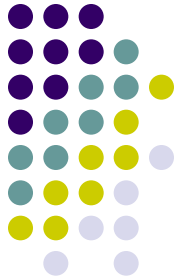
```
public class Disciplina {
  private int codigo;
  private String nome;
  private int cargaHoraria;

  public Disciplina(int pCodigo, String pNome, int pCargaHoraria) throws Exception{
    setCodigo(pCodigo);
    setNome(pNome);
    setCargaHoraria(pCargaHoraria);
  }
  //Métodos de atribuição
  public void setCodigo(int pCodigo)  { codigo = pCodigo;  }

  public void setNome(String pNome) { nome = pNome;   }

  public void setCargaHoraria(int pCargaHoraria) throws Exception {
    if ((pCargaHoraria < 30) || (pCargaHoraria > 110)) {
      throw new Exception("A carga horária deve estar entre 30 e 110 horas.");
    }
    cargaHoraria = pCargaHoraria;
  }

  //Métodos de recuperação
  public int getCodigo() { return codigo;   }
  public String getNome() { return nome;  }
  public int getCargaHoraria() {  return cargaHoraria;  }
}
```

1

2

Faz a validação da carga horária e se estiver fora do intervalo dispara uma exceção que será propagada para o método chamador. Nesse caso quem realizou a chamada foi o próprio construtor da classe.

Disparando exceções
Exemplo 2

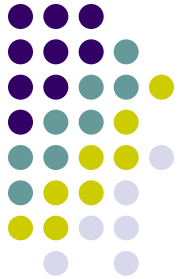


```
import java.util.*;

public class ControleDisciplina {
  private Vector listaDisciplina = new Vector();

  //Método de inserção
  public void insereDisciplina(int pCodigo,  String pNome, int pCargaHoraria) throws Exception {
    Disciplina objDisc = new Disciplina(pCodigo, pNome, pCargaHoraria);
    listaDisciplina.add(objDisc);
  }

  //Método de composição dos dados da disciplina utilizado para visualização
  private String getDisciplina(Disciplina objPDisciplina) {
    return "Código: " + objPDisciplina.getCodigo()+
          "  Nome: " + objPDisciplina.getNome()+
          "  Carga Horária: "+ objPDisciplina.getCargaHoraria()+"\n";
  }

  //Método para obtenção das lista de disciplinas
  public String getListaDisciplinas() {
    String result = "";
    Disciplina objDisciplina = null;
    for (int intIdx = 0; intIdx < listaDisciplina.size(); intIdx++) {
      objDisciplina = (Disciplina)listaDisciplina.elementAt(intIdx);
      result += getDisciplina(objDisciplina);
    }
    if (result.equalsIgnoreCase(""))
      return "Não existem disciplinas cadastradas.";
    else
      return result;
  }
```

Caso ocorra alguma exceção na classe entidade (Disciplina) durante a criação do objeto objDisc uma exceção será capturada no método insereDisciplina e enviado para a classe limite (LimiteDisciplina).

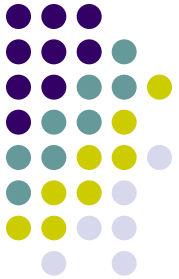
Isso acontece porque na declaração do método existe a palavra reservada *throws*.

Disparando exceções
Exemplo 2



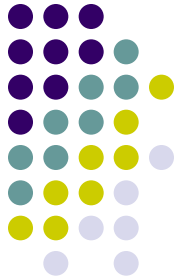
```
  //Método para obtenção de uma disciplina
  public String getDisciplina(int pCodigo) {
    Disciplina objDisciplina = null;
    for (int intIdx = 0; intIdx < listaDisciplina.size(); intIdx++) {
      objDisciplina = (Disciplina)listaDisciplina.elementAt(intIdx);
       if (objDisciplina.getCodigo() == pCodigo)
        return getDisciplina(objDisciplina);
    }
    return "Não foi encontrada nenhuma disciplina com o código "+pCodigo+".";
  }
}
```

Disparando exceções
Exemplo 2



```
import javax.swing.*;
public class LimiteDisciplina  {
  //Instancia o Controlador
  private ControleDisciplina objCtrDisc = new ControleDisciplina();

  public void capturaDados() {
    int escolha = 0;
    String escolhaInformada = "";
    //Variáveis utilizadas para recuperar as informações da interface do
        usuário
    int codigo = 0;
    String nome = "";
    int cargaHoraria = 0;

    do {
      do {
        try {
          escolhaInformada =
            JOptionPane.showInputDialog(
                "Escolha uma opção do menu:\n"+
                "[1] Adiciona disciplina\n"+
                "[2] Lista disciplinas\n"+
                "[3] Finaliza");
          escolha = Integer.parseInt(escolhaInformada);
        } catch (Exception exc) {}
      } while ((escolha < 1) || (escolha > 3));
```

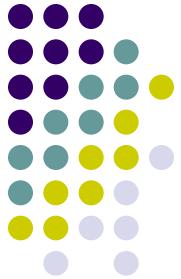
Disparando exceções - Exemplo 2



```
switch (escolha) {
      case 1:
        try {
          //Requisita o Código
          codigo = Integer.parseInt(JOptionPane.showInputDialog ("Informe o código"));
          //Requisita o Nome
          nome = JOptionPane.showInputDialog ("Informe o nome");
          //Requisita a Carga Horária
          cargaHoraria = Integer.parseInt(
                        JOptionPane.showInputDialog ("Informe o carga horária"));
           //Adiciona o objeto para a lista de pacientes
           objCtrDisc.insereDisciplina(codigo, nome, cargaHoraria);
        } catch (Exception exc) {
          JOptionPane.showMessageDialog(null, exc.getMessage(), "Erro",
             JOptionPane.ERROR_MESSAGE);
        }
        break;
      case 2:
        JOptionPane.showMessageDialog(null,
          objCtrDisc.getListaDisciplinas(), "Relação de Disciplinas",
          JOptionPane.INFORMATION_MESSAGE);
        break;
      case 3:
        System.exit(0);
    }
  } while (true);
 }
 public static void main (String par[]) {
  LimiteDisciplina objLimDisc = new LimiteDisciplina();
  objLimDisc.capturaDados();
 }
}
```

Através do bloco try/catch são capturadas as possíveis exceções que podem ocorrer tanto na digitação dos dados quanto na inserção das disciplinas.

Exercício
Entrega: hoje



- Com base no exemplo da Disciplina, crie um programa com classe entidade Aluno, classe controler CtrAluno e classe limite LimiteAluno. Para cada aluno, registrar nome e idade (sendo idade entre 15 e 90 anos). Prepare um sistema com as seguintes opções no menu: (i) inclusão de um aluno; (ii) remoção de um aluno, (iii) apresentação da lista de alunos e (iv) saída do sistema. Note que a idade do aluno deve ser validada com o uso de exceções.