Técnicas Avançados de Programação OO

aula 04

Jobson Luiz Massollar
jobson.luiz@gmail.com

Tratamento de Exceções

- As exceções do Java são classificadas como checked ou unchecked.
- Para as exceções checked, o Java nos obriga a:

1) Tratar a exceções no método onde ela pode ocorrer

OU

2) Avisar que estamos cientes de que aquela exceção pode ocorrer, mas não desejamos tratá-la.

- Para a opção 1 implementamos o bloco try...catch visto anteriormente.

- Para a opção 2 usamos o comando throws

Tratamento de Exceções

- Comando throws

```
public class ImprimeArquivo {
    public static void main(String[] args)
    {
        FileReader fr = new FileReader("arquivo.txt");
        BufferedReader f = new BufferedReader(fr);
        String linha;

        linha = f.readLine();
        while (linha != null) {
          System.out.println(linha);
          linha = f.readLine();
        }

        f.close();
    }
}
```

Esse trecho de programa lê e imprime um arquivo texto.

Esse código pode gerar exceções do tipo **FileNotFoundException** ou **IOException**

O Java não compila esse código !

Tratamento de Exceções

- Comando throws

```
public class ImprimeArquivo {
    public static void main(String[] args) throws Exception
    {
        FileReader fr = new FileReader("arquivo.txt");
        BufferedReader f = new BufferedReader(fr);
        String linha;

        linha = f.readLine();
        while (linha != null) {
          System.out.println(linha);
          linha = f.readLine();
        }

        f.close();
    }
}
```

Para avisar ao compilador que **não desejamos tratar esses erros**, temos que usar o comando **throws** no método onde a exceção é gerada**.**

Assim, a compilação é realizada.

Tratamento de Exceções

- Como tratar situações de erro que são específicas dos nossos programas ?

- Exemplo: para construir os objetos abaixo, que valores seriam inválidos?

```
public Retangulo(int x, int y, int largura, int altura)

public Circulo(int x, int y, int raio)

public Aluno(int matricula, String nome)
```

Tratamento de Exceções

- Como tratar situações de erro que são específicas dos nossos programas ?

- Exemplo: para construir os objetos abaixo, que valores seriam inválidos?

```
public Retangulo(int x, int y, int largura, int altura)

public Circulo(int x, int y, int raio)

public Aluno(int matricula, String nome)
```

Largura e **altura** devem ser maiores que zero.

Raio deve ser maior que zero.

Matricula deve ser um número positivo e **nome** não pode ser vazio.

Tratamento de Exceções

- Como podemos tratar a situação onde Largura ou Altura é menor ou igual a zero?

```
public class Retangulo {
    private int x, y, largura, altura;

    public Retangulo(int x, int y, int largura, int altura)
    {
       this.x = x;
       this.y = y
       this.largura = largura;
       this.altura = altura;
    }
}
```

Tratamento de Exceções

- Resposta:
  - Criando e disparando nossas próprias exceções !

- Como ?

1. Criamos uma instância da classe Exception com o operador new;
2. Disparamos a exceção com o comando throw;
3. Declaramos que o método irá disparar uma exceção com o comando throws.

Tratamento de Exceções

- A classe Exception é a classe mãe de todas as exceções.
- Ela possui alguns métodos úteis e comuns a todas as exceções:
  - Construtor Exception(String msg): permite criar uma exceção e armazenar a mensagem de erro.
  - getMessage(): retorna a mensagem de erro.
  - printStackTrace(): imprime a pilha de chamadas no mesmo formato da JVM.
  - getStacktrace(): retorna a pilha de chamadas. Nesse caso você pode implementar a sua própria impressão ou salvar essa informação em outro local ou formato (por exemplo, para montar um *log* de erros).

Tratamento de Exceções

- Se depararmos com uma situação na qual o objeto não pode ser criado, disparamos uma exceção:

```
public class Retangulo {
    private int x, y, largura, altura;

    public Retangulo(int x, int y, int largura, int altura) throws Exception
    {
       if (largura <= 0 || altura <= 0)
         throw new Exception("Retangulo deve ter largura e altura maior
                              que zero");

       this.x = x;
       this.y = y
       this.largura = largura;
       this.altura = altura;
}
```

Se **largura** ou **altura** forem menores ou iguais a zero a execução do construtor será interrompida e a exceção será disparada

Tratamento de Exceções

- No método onde o Retangulo é criado devemos tratar essa exceção com try...catch:

```
public class EditorGrafico {

    public static void main(String[] args) {
       Retangulo r;

       try {
         r = new Retangulo(0,0, -10, 20);
       } catch(Exception e) {
         System.out.println("Erro: " + e.getMessage());
         r = null;
       }

       if (r != null)
         r.desenhar();
    }
}
```

Em caso de erro na criação do Retângulo, imprime a mensagem de erro e encerra o programa.

Exercícios

- Exercício 10: Crie a classe Circulo com um construtor que recebe a posição (x, y) central e um raio. Caso o raio seja negativo ou zero gere uma exceção.

Tratamento de Exceções

- Na implementação da classe Triangulo, cujo construtor recebe um vetor de vértices com os 3 vértices do triângulo, pergunta-se: o que pode dar errado na construção do triângulo?

```
public class Vertice {
    private float x, y;

    public Vertice(float x, float y) {
      this.x = x;
      this.y = y;
    }

    public float getX() { return x; }
    public float getX() { return y; }
}

public class Triangulo {
    private Vertice[] vertices;

    public Triangulo(Vertice[] vertices)
    {
       this.vertices = new Vertice[3];

       int i = 0;
       for (Vertice v : vertices)
           this.vertices[i++] = new Vertice(v.getX(), v.getY());
    }
}
```

Tratamento de Exceções

- O que pode dar errado ?

```
public class Triangulo {
    private Vertice[] vertices;

    public Triangulo(Vertice[] vertices)
    {
       this.vertices = new Vertice[3];

       int i = 0;
       for (Vertice v : vertices)
           this.vertices[i++] = new Vertice(v.getX(), v.getY());
    }
}
```

- O vetor passado não possui 3 vértices

- O vetor passado não está inicializado corretamente, ou seja, possui elementos nulos

- Os vértices passados não formam um triângulo

Tratamento de Exceções

- Uma outra forma, um pouco mais complexa, de tratar exceções específicas dos nossos programas é criando nossas próprias exceções.

- Como ?
  - Criando uma classe que estende Exception
  - Definimos o(s) construtor(es) adequados
  - Podemos usar alguns métodos herdados da classe Exception

- Como disparamos a nossa exceção ?
  - Criamos uma instância com o operador new
  - Disparamos a exceção com o comando throw

Tratamento de Exceções

- No caso da classe Triangulo:

```
public class Triangulo {
    private Vertice[] vertices;

    public Triangulo(Vertice[] vertices)
    {
       this.vertices = new Vertice[3];

       int i = 0;
       for (Vertice v : vertices)
           this.vertices[i++] = new Vertice(v.getX(), v.getY());
    }
```

Vamos criar nossa própria exceção, chamada **TrianguloInvalidoException**, para informar que houve algum problema na criação do Triangulo

Tratamento de Exceções

- Classe TrianguloInvalidoException: no nosso caso, queremos somente armazenar uma mensagem descrevendo o que houve de errado. Por isso usamos o que já existe na classe Exception.

```
// Normalmente criamos um pacote especifico para implementar as excecoes
package excecoes;

public class TrianguloInvalidoException extends Exception {

    public TrianguloInvalidoException(String msg) {
        // Usamos o construtor da classe Exception para armazenar a
        // mensagem no objeto
        super(msg);
    }
}
```

Tratamento de Exceções

- Implementamos as validações no construtor e disparamos a exceção, caso algum erro seja encontrado.

```
package principal;
import excecoes.TrianguloInvalidoException;

public class Triangulo {
    private Vertice[] vertices;

    public Triangulo(Vertice[] vertices) throws TrianguloInvalidoException {
       // 1o. teste - o vetor de vertices tem que ter tamanho 3
       if (vertices.length != 3) {
         TrianguloInvalidoException e =
              new TrianguloInvalidoException("Quantidade de vertices deve ser igual a 3");

         throw e;
       }

       this.vertices = new Vertice[3];
       int i = 0;
       for (Vertice v : vertices)
           this.vertices[i++] = new Vertice(v.getX(), v.getY());
    }
}
```

Nesse exemplo implementamos somente a primeira validação

Tratamento de Exceções

- Nos métodos onde construímos o Triangulo, devemos tratar a exceção:

```
package principal;
import excecoes.TrianguloInvalidoException;

public class TesteTriangulo {
    public static void main (String[] args) {
      Vertice[] v = new Vertice[3];
      Triangulo t;

      v[0] = new Vertice(0,0);
      v[1] = new Vertice(10,10);
      v[2] = new Vertice(20,0);

      try {
        t = new Triangulo(v);
      } catch(TrianguloInvalidoException e) {
        System.out.println("Erro no triangulo: " + e.getMessage());
      }
    }
```

Exercícios

- Exercício 11: Altere o exercício 10 e crie uma classe CirculoInvalidoException para tratar as exceções de criação do círculo.