Aula 6

Programação Orientada a Objetos

Prof. Leonardo Gomes

1

Conversa Inicial

2

- **Tratamento de exceções**
- **Criando as próprias exceções**
- **Igualdade**
- **Métodos especiais: toString()**
- **Singleton**

3

Tratamento de exceções

4

Erro

- **É um problema que ocorre em tempo de execução**
- **Geralmente associado ao sistema operacional**
- **java.lang.OutOfMemoryError**

5

Exceção

- **É um problema que ocorre em tempo de execução**
- **Geralmente pode ser tratado e manejado de alguma forma**
- **FileNotFoundException**

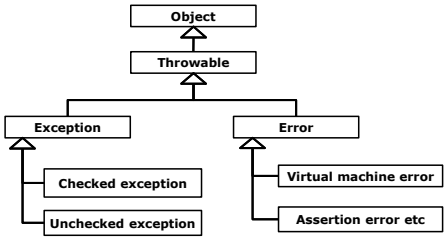
6

Erros e exceções

- **Ambos são identificados pela JVM**
- **Geram subclasses de Throwable**
- **Posteriormente são lançadas para que o *software* gere o tratamento adequado**

7

Hierarquia



8

Passos

- **A JVM busca na *call stack* um método que contenha um bloco de código capaz de tratar a exceção**
- **A busca inicia no método que gerou a exceção e segue ordem reversa de chamada**

9

- **Ao encontrar um bloco tratador apropriado, este recebe o controle do fluxo de código junto ao objeto exceção**
- **Se a JVM não encontrar um bloco tratador de exceções na *call stack*, a JVM utiliza um bloco tratador de exceções padrão que irá encerrar o programa**

10

Try/Catch

```
public class MinhaClasse {
public static void main(String[] args) {
 try {
    int[] meusNumeros = {1, 2, 3};
    System.out.println(meusNumeros[10]);
    } catch (Exception e) {
      System.out.println("Problema = " + e);
    } finally {
      System.out.println("Terminado o try catch.");
    }
  }
}
```

11

Criando as próprias exceções

12

Erros e exceções

- Ambos podem ser muito úteis
- Evitam problemas maiores e comportamentos imprevisíveis
- Mais fáceis de detectar

13

Exemplo

```
class Principal{
    static void funcao(){
        try{
          throw new NullPointerException("Problema!");
        }
        catch(NullPointerException e){
            System.out.println("funcao() :" + e);
            throw e; // jogando a exceção novamente
        }
    }
    public static void main(String args[]){
        try{
            funcao();
        }
        catch(NullPointerException e){
            System.out.println("main() :" + e);
        }

    }
}
```

14

Tipos de exceção

- ***Checked*: exceções verificadas em tempo de compilação**
  - **IOException**
- ***Unchecked*: não são verificadas**
  - **RuntimeException**

15

Exemplo

```
public class UsuarioInexistenteException extends Exception {
  public UsuarioInexistenteException(String mensagem){
    super(mensagem);
  }
}
```

16

Exemplo

```
public class Gerenciador {
    public Usuario buscar(String usuarioID) throws UsuarioInexistenteException {
        if (usuarioID.equals("123456")) {
            return new Usuario();
        } else {
            throw new UsuarioInexistenteException("Nao existe usuario " + studentID);
        }
    }
}
```

17

Exemplo

```
public class Teste {
  public static void main(String[] args) {
    Gerenciador gerenciador= new Gerenciador();
     try {
        Usuario usr = gerenciador.buscar("0000001");
     } catch (UsuarioInexistenteException ex) {
        System.err.print(ex);
     }
  }
}
```

18

Igualdade

19

- **Comando ==**
- **Funciona com primitivas**
- **Não funciona da mesma forma com objetos**

20

Exemplo

```
public class Teste {
   public static void main(String[] args)
   {
       String s1 = new String("Ola");
       String s2 = new String("Ola");
       String s3 = s1;
       System.out.println(s1 == s2); //falso
       System.out.println(s1.equals(s2)); //verdadeiro

       System.out.println(s1 == s3); //verdadeiro
   }
}
```

21

Exemplo equals

```
public class Usuario {
 int id; String nome, cpf;

 public boolean equals(Object outro) {
       // Compara consigo mesmo
       if (outro == this) {
          return true;
       }
       //Objeto outro é uma instância de Usuario?
       if (!(outro instanceof Usuario)) {
          return false;
       }
       // type cast para Usuario
       Usuario o = (Usuario) outro;
       //Compara os atributos são iguais
   if( this.id == o.id &&
     this.nome.equals(o.nome) &&
     this.cpf.equals(o.cpf)) {
       return true;
   }
   return false;
 }
}
```

22

Exemplo equals

```
public class Principal {

  public static void main(String[] args) {

    Usuario m1 = new
Usuario(1,"Mario","111.222.333-44");
    Usuario m2 = new Usuario(1,"Mario","111.222.333-44");
   Usuario m3 = new Usuario(2,"Luigi","555.666.777-88");

    //verdadeiro
    System.out.println(  m1.equals(m2)  );
    //falso
    System.out.println(  m1 == m2  );
    //falso
    System.out.println(  m1.equals(m3)  );
  }
}
```

23

Opção no Eclipse

24

Métodos especiais: toString()

25

toString()

- Muitas vezes é conveniente tratar um objeto como uma string
- Por exemplo, para imprimir dados sobre o objeto
- O método toString() faz essa conversão de forma conveniente

26

toString()

- Feita a impressão sem a sobreescrita do método toString, teremos apenas informações sem sentido do endereço de memória da instância

```
Usuario usr = new Usuario();
System.out.println(usr);
```

27

Criação do método toString()

```
public class Usuario {
  int id;
  String nome;
  String cpf;

  @Override
  public String toString() {
    return "Usuario, id=" + id + ", nome=" + nome + ", cpf=" + cpf;
  }
}
```

28

- Também pode ser criado de forma automatizada pela IDE Eclipse

29

Singleton

30

Design pattern

- Padrões de desenvolvimento muito comuns
- Problemas clássicos e recorrentes
- Soluções padronizadas na literatura

31

Singleton

- Semelhante à variável global
- Única instância no projeto inteiro
- Diversos usos, por exemplo: uma classe de log

32

Passos

- Ter um constructor privado
- Criar um método estático que retorna um objeto da classe instanciada
  - Conceito chamado de "*lazy initialization*"

33

Singleton

```
class Singleton
{
    // variável estática que armazenara a nossa única instancia
    private static Singleton instancia = null;
    // variavel para teste
    public int numero;
    //construtor privado
    private Singleton(){
        numero =20;
    }

    // método estático para criar a instancia
    public static Singleton getInstance(){
        if (instancia == null)
            instancia = new Singleton();

        return instancia;
    }
}
```

34

Singleton

```
public static void main(String args[]) {
    // instantiating Singleton class with variable x
    Singleton x = Singleton.getInstance();
    // instantiating Singleton class with variable y
    Singleton y = Singleton.getInstance();
    // instantiating Singleton class with variable z
    Singleton z = Singleton.getInstance();
    // changing variable of instance x
    x.numero+=10;

    System.out.println("x: " + x.numero);
    System.out.println("y: " + y.numero);
    System.out.println("z: " + z.numero);
    System.out.println("\n");
    //Todos imprimem 30
    z.numero-=5;
    System.out.println("x: " + x.numero);
    System.out.println("y: " + y.numero);
    System.out.println("z: " + z.numero);
    System.out.println("\n");
    //Todos imprimem 25
}
```

35