

Revisão...

- Classes abstratas...
 - Não podem ser instanciadas...
 - Obriga todas as classes que herdam a implementar os comportamentos abstratos...

Revisão...

- Interfaces...
 - Classes abstratas puras
 - Funcionam como um contrato entre duas classes...
 - Uma classe implementa o que está definido numa ou mais interfaces...

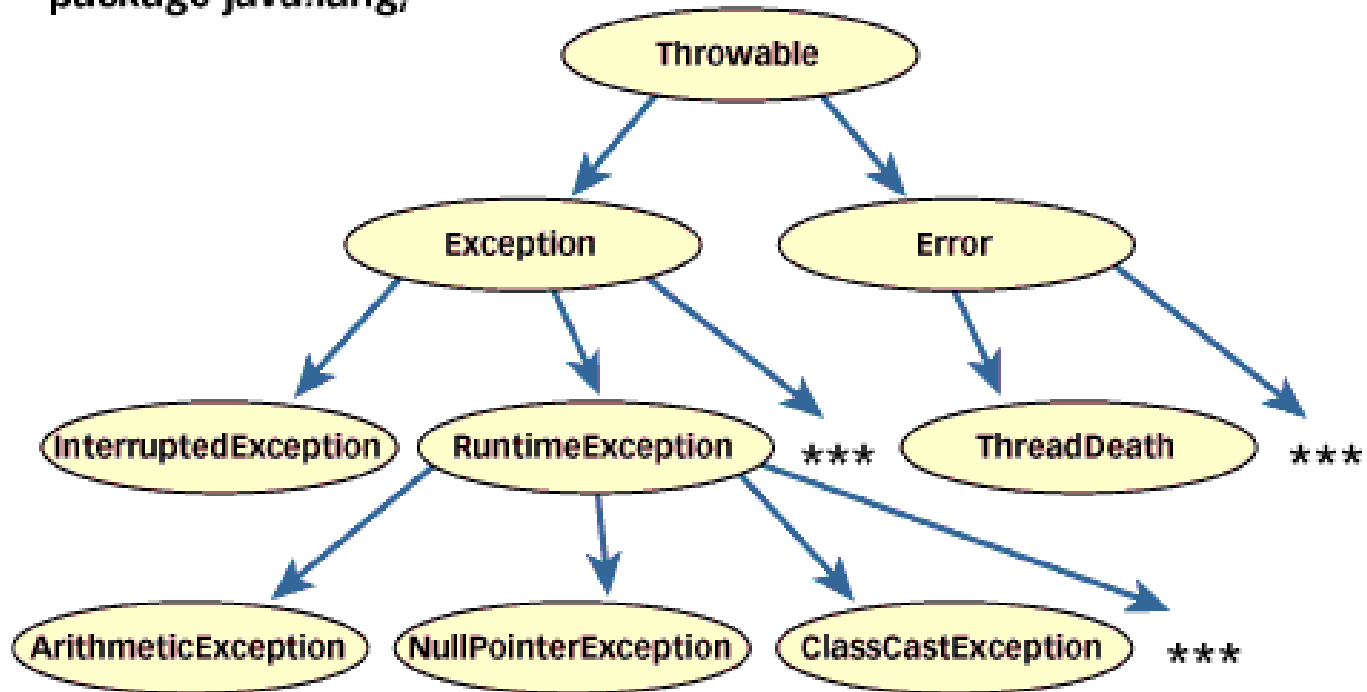
Revisão

- Em java existem 2 grandes categorias de erros..

Erros e Exceções (Errors and Exceptions)

Revisão

package java.lang;



Revisão

- Um código que pode dar erro ou ter uma exceção ele precisa primeiro tentar executar(**try**), caso ele não consiga uma exceção será lançada(**throw**) e ela precisará então ser pegada (**catch**)

Revisão

```
public float dividir(int a, int b) {  
    float c = (float)(a/b);  
    return c;  
}
```

Revisão

```
public float dividir(int a, int b) {  
    try{  
        float c = (float)(a/b);  
        return c;  
    }catch(Exception e) {  
        .. o que fazer com a excessao...  
    }  
}
```

Revisão

- Um método pode lançar uma exceção ou um erro de tal forma que quem for usar esse método deverá saber que ele pode lançar essa exceção...
- `public void andar(int passos) throws Exception;`

Revisão

- **try** – tenta executar um código
- **catch** – pega alguma exceção que possa ser lançada
- **throws** (indica que um comportamento pode ter uma exceção)
- **throw** (lança a exceção)

Revisão

- Uma exceção nova é uma `Exception`, ou seja ela herda de `Exception`

```
public class MuitoFrioException extends Exception {  
    ...  
}
```

```
public class criarCafe() throws MuitoFrioException { ... }
```

Pacotes Java

- Existem diversos pacotes que possuem funcionalidades prontas e são disponibilizados para o desenvolvedor...
- Não vamos reinventar a roda!

Pacotes Java

- `java.lang`
 - Funcionalidades essenciais à linguagem Java
- `java.io`
 - Leitura e escrita em arquivos
- `java.util`
 - Trabalhar com coleção de objetos

Java.lang

- Pacote que contém as classes essenciais para a linguagem java
- Documentação *

java.lang

- Object
- System
 - out
 - in
- Wrapper classes
 - Integer, Double, Float, Boolean, Byte, Short, Long ...

java.lang

- Faça um programa que receba 2 números na entrada e escreva a soma deles na tela
- Faça um programa que receba o primeiro nome e a idade, crie uma pessoa com esses dados e depois escreva na tela usando o método toString() de **Pessoa**

java.lang

- Alguns exemplos...

java.io

- Pacote com classes que ajudam na manipulação de fluxos de dados, como arquivos por exemplo...

java.io

- Documentação*

java.io

- **Escrevendo para arquivos em Java!**
 - Primeiro eu preciso de um arquivo a ser criado
 - Depois eu preciso de um fluxo de dados que escreverá nesse arquivo...

java.io

- **FileOutputStream** saída;
 - associado ao arquivo
- **PrintStream** fluxo;
 - associado ao fluxo de dados

java.io

- **Lendo de arquivos em Java!**
 - Primeiro eu preciso de um arquivo que será lido
 - Depois eu preciso de um fluxo de dados que vai ler desse arquivo...

java.io

- **FileInputStream** entrada;
- **DataInputStream** fluxo;

java.io

- Outra formas de ler arquivos seria usando outras classes do pacote `java.io` ou criando suas próprias classes de acesso a `arquivos...`

java.io

- Exemplo
 - Vamos escrever os dados de uma Pessoa num arquivo e depois ler esse arquivo e armazenar em Pessoa novamente!
 - Pessoa
 - nome: string
 - idade: int
 - sexo: char
 - telefone: String

java.util

- Pacote responsável por funcionalidades como collections, vetores, etc...
- Documentação*

java.util

- Arrays & Vectors
 - Arrays e Vectors apesar de possuírem conceitos semelhantes eles funcionam de formas diferentes...

java.util

- Exemplo

- Arrays possuem o seu tamanho fixados na hora da sua criação...

- `int numeros[] = new int[5];`

- Collections podem ter seus tamanhos alterados

- `Vector nomes = new Vector();`

- `numeros.addElement("casa");`

java.util

- Obs:
 - Vectors trabalham sobre Object!
 - Tipos primitivos não são permitidos
 - Wrapper Classes

Como armazenar e recuperar um objeto Pessoa em um Vector ?

java.util

- **Vectors**

- void add(Object o);
- int size();
- Object elementAt(int index);
- Object firstElement();
- Object lastElement();
- void removeElementAt(int index);
- Enumeration elements();

java.util

- Crie um Vector e insira 4 números diferentes..depois percorra esse vector e escreva esses números na tela

