

# 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...

- Interfaces...
  - Uma interface pode herdar de uma ou mais interfaces...
    - “um contrato pode ser anexado a um ou mais outros contratos...”

# Tratamento de Erros

- Durante a execução de um programa erros podem acontecer...
- Exemplo:
  - divisao ..

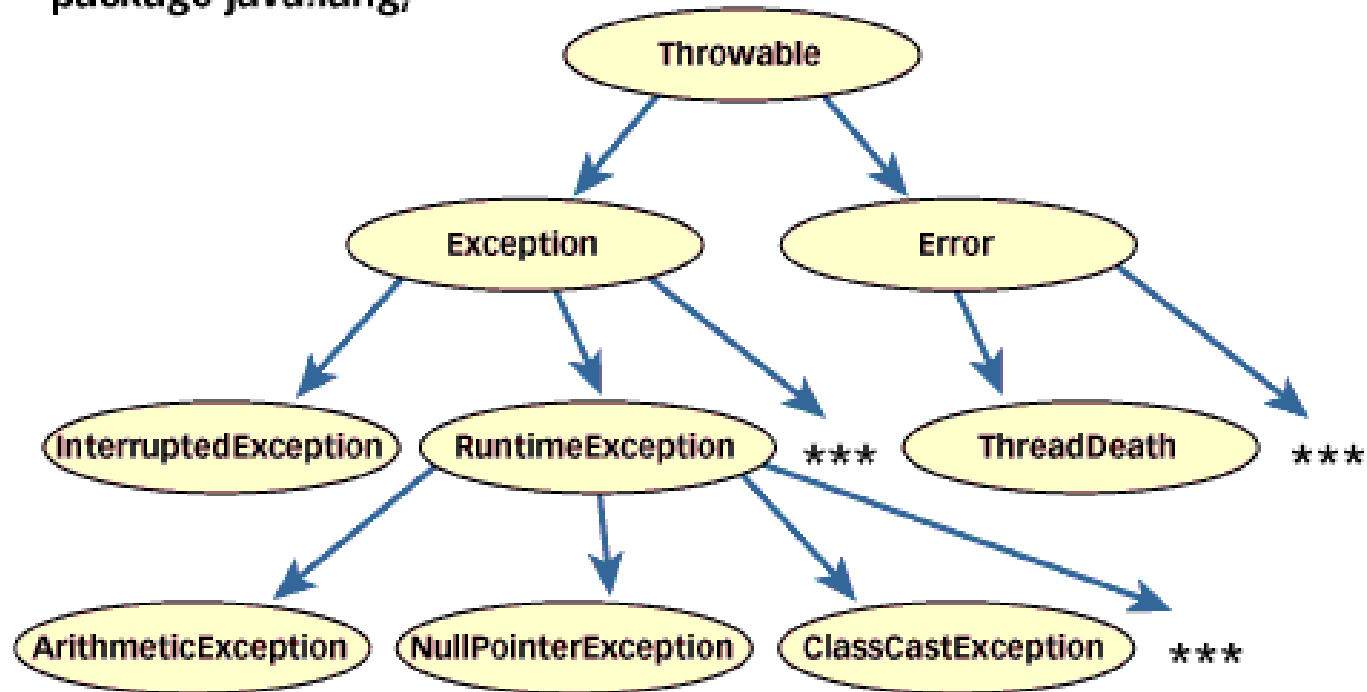
# Tratamento de Erros

- Em java existem 2 grandes categorias de erros..

Erros e Exceções (Errors and Exceptions)

# Tratamento de Erros

package java.lang;



# Tratamento de Erros

- Error
  - Erros que não podem ser recuperados, ou sejam que são fatais para a execução do programa...
- Exemplos
  - Leitura de um arquivo corrompido...
    - `java.io.IOException`

# Tratamento de Erros

- Exceções são possibilidades de execução do código que podem dar errado e que precisam ser tratadas...
  - As exceções em java são também subdividas em 2 outras categorias...
    - checked exceptions
    - unchecked exceptions

# Tratamento de Erros

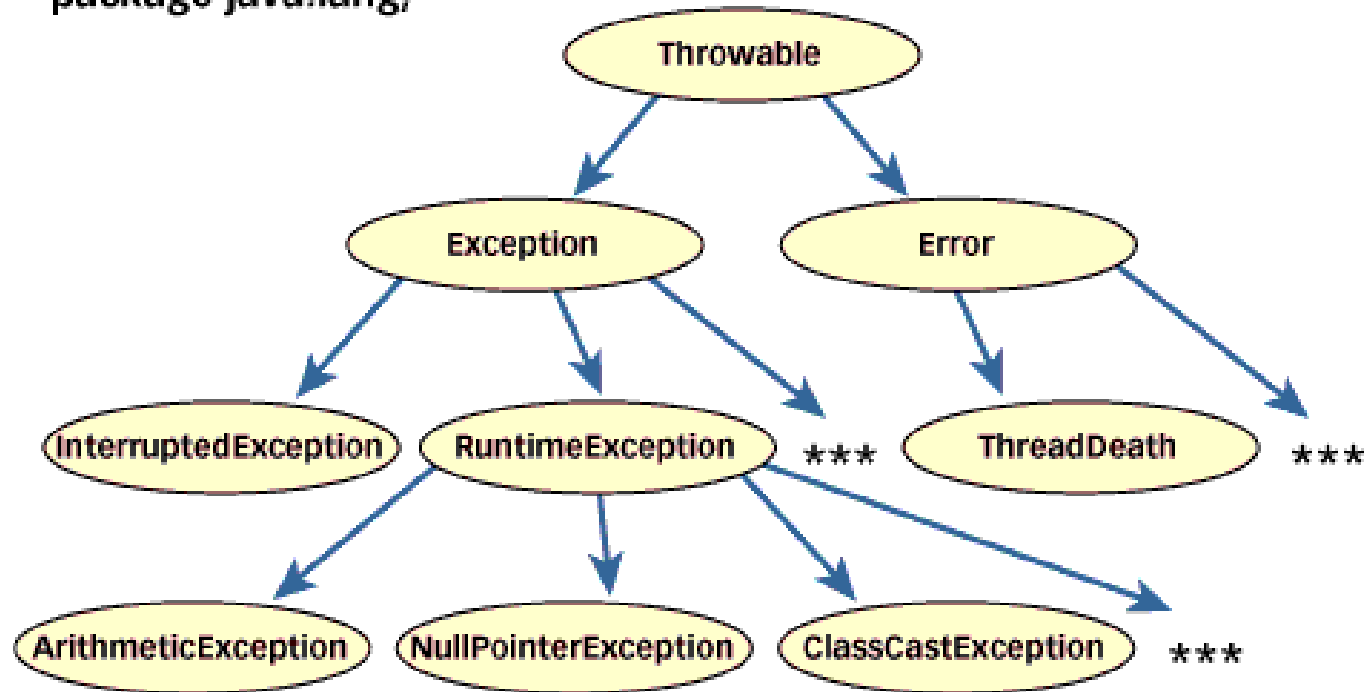
- Exceptions
  - Exceptions são objetos do tipo Exception (Throwable)
  - Dizemos que uma exceção foi lançada(throw) e que ela terá de ser **pega|pegada|** (catch)

# Tratamento de Erros

- 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**)

# Tratamento de Erros

package java.lang;



# Tratamento de Erros

- Obs: somente exceções que podem ser lançadas.. objetos comuns não podem ser lançados...

# Tratamento de Erros

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

# Tratamento de Erros

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

# Tratamento de Erros

- 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;`

# Tratamento de Erros

- `public void andar(int passos) throws Exception;`

```
public class Principal {  
    public static void main(String args[]){  
        Humano pessoa = new Humano();  
        pessoa.andar(3); <<< pode dar erro  
    }  
}
```

# Tratamento de Erros

```
public class Principal {  
    public static void main(String args[]){  
        try{  
            Humano pessoa = new Humano();  
            pessoa.andar(3) ; <<< pode dar erro  
        }catch(Exception e) { ..... }  
    }  
}
```

# Tratamento de Erros

- Para um código específico lançar uma exceção utilizamos

```
throw new Exception();
```

exemplo:

```
if(x<) throw new Exception();
```

- Como exceções são objetos elas precisam ser instanciadas...

# Tratamento de Erros

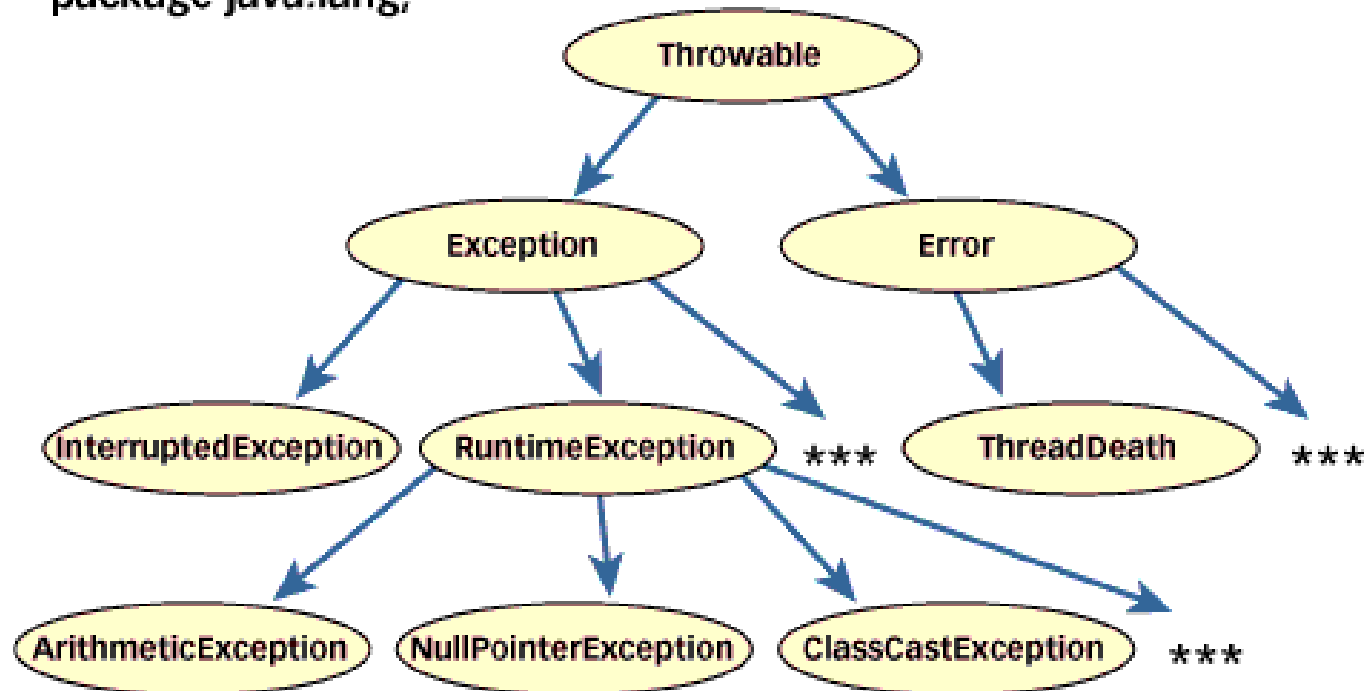
- **try** – tenta executar um código
- **catch** – pega alguma exceção que possa ser lançada
- **throws** (indica que um comportamento pode ter uma exceção)
- **throw** (lança a exceção)

# Tratamento de Erros

- Um comportamento pode lançar mais de um erro diferente.. então deve ser possível pegar erros diferentes que possam ocorrer..

# Tratamento de Erros

package java.lang;



# Tratamento de Erros

- `catch(Exception exceptionObj)`
  - Pega todas as exceções do tipo **Exception**, mas se todas as exceções herdarem de `Exception` ele automaticamente pega todas as exceções...

# Tratamento de Erros

- Mas é interessante deixar o seu código o mais claro e específico possível por isso o uso de `catch(Exception e)` não é muito recomendado...
- `catch(ArithmeticException e) { ... }`

# Tratamento de Erros

- Um corpo de código dentro de `try { ... }` pode dar vários erros então dessa forma como queremos ter um código mais claro e específico `catch(Exception e)` não é legal logo deve haver uma forma de pegar outras exceções específicas e diferentes...

# Tratamento de Erros

```
try {  
    ...  
} catch(Exception1 obj){  
    ... corpo da exceção 1  
} catch(Exception2 obj){  
    ... corpo da exceção 2  
}
```

# Tratamento de Erros

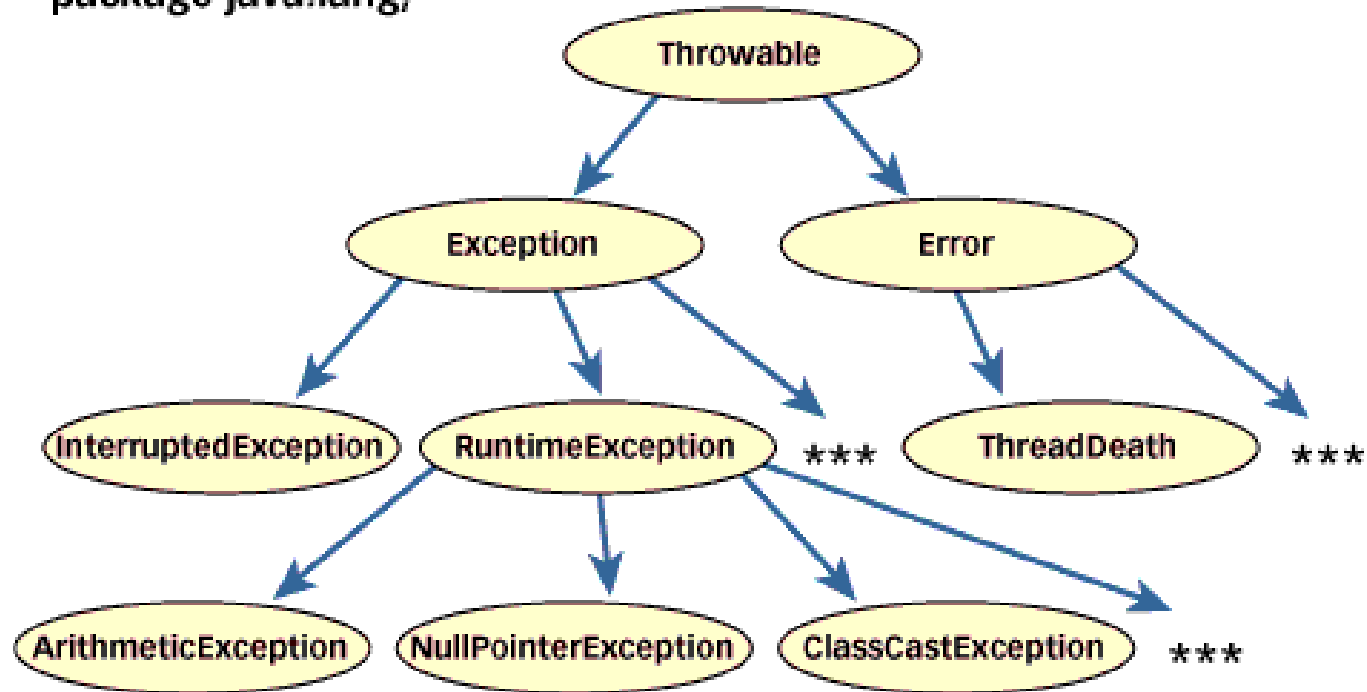
- Caso ocorra uma exceção que não foi pegada o programa vai interromper sua execução...

# Tratamento de Erros

- Os diferentes “**catch**” de um corpo **try** precisam estar numa ordem determinada...
- da exceção específica para a mais geral... pq?

# Tratamento de Erros

package java.lang;



# Tratamento de Erros

- Quando um erro acontece, vamos supor que vc estava no meio de alguma operação e precisa então depois de ter pegado esse erro limpar o que foi feito ou então fazer alguma operação pra tratar isso... como fazer?

# Tratamento de Erros

```
try {  
    ...  
}catch(Exception e) {  
    ...  
}
```

- como recuperar o que eu comecei a fazer antes de dar o erro?

# Tratamento de Erros

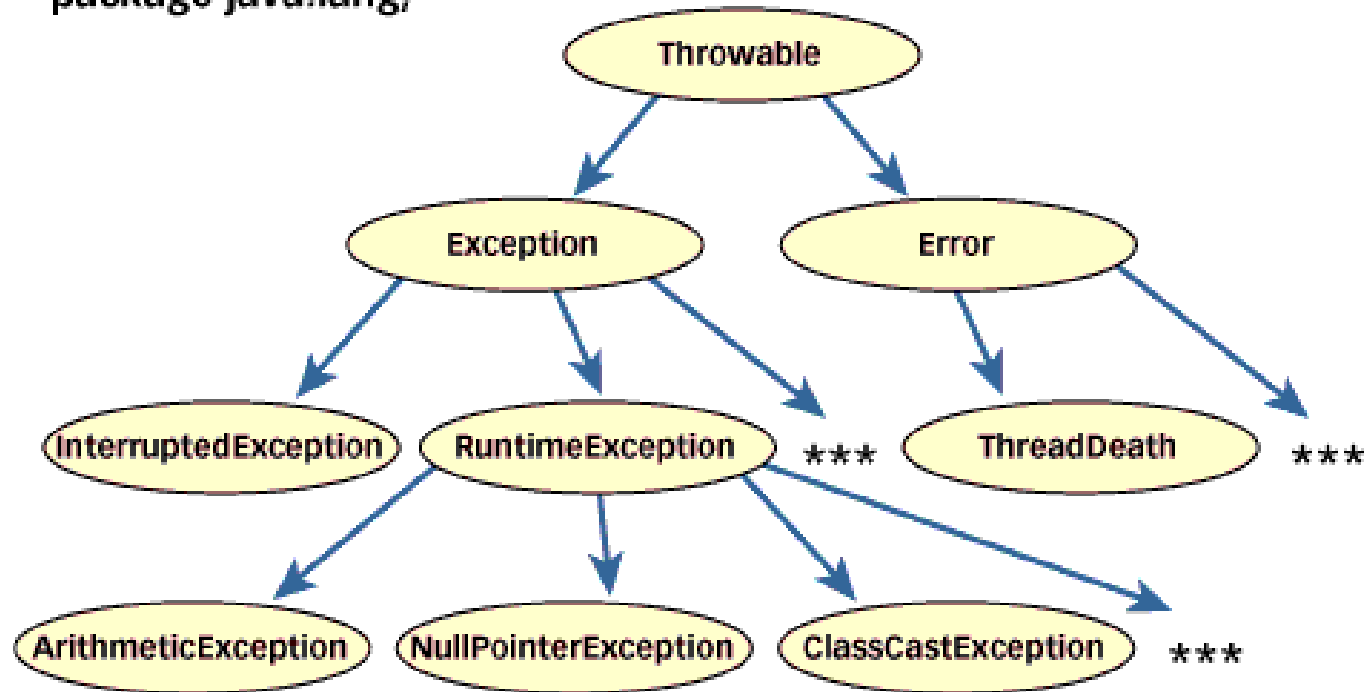
- **finally** { ... }
  - O corpo do **finally** é executado depois da saída do try, seja ela por meio de um erro ou que ele tenha executado sem erro...

# Tratamento de Erros

- **Exemplo:**
  - abertura e fechamento de arquivos...
  - abertura -> **try**
  - fechamento -> **finally**

# Tratamento de Erros

package java.lang;



# Tratamento de Erros

- Criando as próprias Exceptions, ou erros específicos..
- ex. Máquina de Café
  - Erros: muito quente ou muito frio

# Tratamento de Erros

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

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

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

# Tratamento de Erros

- A experiência do Café...
  - **Humano**
    - beberCafe(CopoCafe copo) throws ...
  - **CopoCafe**
    - Temperatura
  - **Cafeteria**
    - servirCafe(Humano pessoa, CopoCafe cafe);