Aula 23 – Arquivos e Exceções

Norton Trevisan Roman

18 de junho de 2018

Arquivos

- Nossos programas têm um grande inconveniente

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?
  - Guardando em arquivos

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?
  - Guardando em arquivos
  - Em java há vários modos

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?
  - Guardando em arquivos
  - Em java há vários modos
  - Um modo é criar um objeto `File`

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?
  - Guardando em arquivos
  - Em java há vários modos
  - Um modo é criar um objeto `File`

```
import java.io.File;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    ...
  }
}
```

Arquivos

- Nossos programas têm um grande inconveniente
  - Dados de configuração, que deveriam ser preservados, precisam ser fornecidos a cada vez que o programa roda
- Como preservá-los então?
  - Guardando em arquivos
  - Em java há vários modos
  - Um modo é criar um objeto `File`

```
import java.io.File;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    ...
  }
}
```

- Instâncias da classe `java.io.File` representam caminhos (paths) para possíveis locais no sistema de arquivos

Arquivos

- Apenas representam um arquivo ou diretório

```
import java.io.File;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    ...
  }
}
```

Arquivos

- Apenas representam um arquivo ou diretório
- Não necessariamente o caminho existe

```
import java.io.File;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    ...
  }
}
```

Arquivos

- Apenas representam um arquivo ou diretório
- Não necessariamente o caminho existe
- Não criam o arquivo ou diretório

```
import java.io.File;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    ...
  }
}
```

Arquivos

- Mas isso só não basta

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                new FileWriter(arq,false);

    return(true);
  }
}
```

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo
- Mas onde?

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,false);

    return(true);
  }
}
```

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo
- Mas onde?
  - Ou no início do arquivo

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,false);

    return(true);
  }
}
```

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo
- Mas onde?
  - Ou no início do arquivo
    - Se já existir, irá sobrescrever

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,false);

    return(true);
  }
}
```

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo
- Mas onde?
  - Ou no início do arquivo
    - Se já existir, irá sobrescrever
  - Ou em seu final (append)

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,true);

    return(true);
  }
}
```

Arquivos

- Mas isso só não basta
- Temos que escrever o string no arquivo
- Mas onde?
  - Ou no início do arquivo
    - Se já existir, irá sobrescrever
  - Ou em seu final (append)
    - Se não existir, irá criar

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,true);

    return(true);
  }
}
```

Arquivos

- E...

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                 new FileWriter(arq,true);

    return(true);
  }
}
```

Arquivos

- E...

Saída

```
$javac Projeto.java
Projeto.java:52: unreported exception
java.io.IOException; must be caught
or declared to be thrown
        FileWriter writer =
            new FileWriter(arq,true);
              ^
1 error
```

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,true);

    return(true);
  }
}
```

Arquivos

- E...

Saída

```
$javac Projeto.java
Projeto.java:52: unreported exception
java.io.IOException; must be caught
or declared to be thrown
        FileWriter writer =
            new FileWriter(arq,true);
              ^
1 error
```

```
import java.io.File;
import java.io.FileWriter;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    FileWriter writer =
                  new FileWriter(arq,true);

    return(true);
  }
}
```

- Capturada ou declarada como lançada?

Exceções

- Exceções são como o Java trata de seus erros.

Exceções

- Exceções são como o Java trata de seus erros.
- São objetos "lançados" de um trecho de código para fora do método em que ele está

Exceções

- Exceções são como o Java trata de seus erros.
- São objetos "lançados" de um trecho de código para fora do método em que ele está
- Métodos que fizeram chamada ao método problemático podem

Exceções

- Exceções são como o Java trata de seus erros.
- São objetos "lançados" de um trecho de código para fora do método em que ele está
- Métodos que fizeram chamada ao método problemático podem
  - Capturar e tratar a exceção

Exceções

- Exceções são como o Java trata de seus erros.
- São objetos "lançados" de um trecho de código para fora do método em que ele está
- Métodos que fizeram chamada ao método problemático podem
  - Capturar e tratar a exceção
  - Passá-la adiante, para o método que os invocar

Exceções

- Passando adiante a batata quente:

Exceções

- Passando adiante a batata quente:
  - Basta adicionar `throws` (e o tipo da exceção repassada) na assinatura do método em que ela pode ocorrer

Exceções

- Passando adiante a batata quente:
  - Basta adicionar `throws` (e o tipo da exceção repassada) na assinatura do método em que ela pode ocorrer

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome)
                          throws IOException {
    File arq = new File(nome);
    FileWriter writer =
                  new FileWriter(arq,true);
    return(true);
  }
}
```

Exceções

- Passando adiante a batata quente:
  - Basta adicionar `throws` (e o tipo da exceção repassada) na assinatura do método em que ela pode ocorrer
  - O método que chamar `gravaArq` que se vire para tratá-la, ou passe adiante

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome)
                          throws IOException {
    File arq = new File(nome);
    FileWriter writer =
                 new FileWriter(arq,true);
    return(true);
  }
}
```

Exceções

- Capturando e Tratando Exceções

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);

    return(true);
  }
}
```

Exceções

- Capturando e Tratando Exceções
  - Bloco `Try – Catch – Finally`

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {


    }
    catch(IOException ioe) {

    }
    finally {

    }
    return(true);
  }
}
```

Exceções

- Capturando e Tratando Exceções
  - Bloco `Try – Catch – Finally`
  - `Try`: Testa o comando que pode gerar a exceção

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {

    }
    finally {

    }
    return(true);
  }
}
```

Exceções

- Capturando e Tratando Exceções
  - Bloco `Try – Catch – Finally`
  - `Try`: Testa o comando que pode gerar a exceção
  - `Catch`: Captura a exceção, executando um código que o programador define para tratá-la

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    finally {

    }
    return(true);
  }
}
```

Exceções

- Finally (Opcional)

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    finally {// totalmente opcional
      // Fazendo alguma limpeza
    }
    return(true);
  }
}
```

Exceções

- `Finally` (Opcional)
  - O código dentro dele sempre será executado, mesmo se houver uma exceção

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    finally {// totalmente opcional
      // Fazendo alguma limpeza
    }
    return(true);
  }
}
```

Exceções

- `Finally` (Opcional)
  - O código dentro dele sempre será executado, mesmo se houver uma exceção
  - Se o código dentro do `try` ou `catch` contiver um `return`, o código dentro do `finally` será executado antes do retorno do método

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    finally {// totalmente opcional
      // Fazendo alguma limpeza
    }
    return(true);
  }
}
```

Exceções

- E se nome for null?

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {

    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E se nome for null?
  - Ou testamos e retornamos

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      return(false);


    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E se nome for `null`?
  - Ou testamos e retornamos
  - Ou lançamos uma exceção

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      return(false);


    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E se nome for `null`?
  - Ou testamos e retornamos
  - Ou lançamos uma exceção
- Como lançar?

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      return(false);


    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E se `nome` for `null`?
  - Ou testamos e retornamos
  - Ou lançamos uma exceção
- Como lançar?
  - Criando um objeto da exceção desejada

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      IOException ex = new
          IOException("Parâmetro nulo");

    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E se nome for null?
  - Ou testamos e retornamos
  - Ou lançamos uma exceção
- Como lançar?
  - Criando um objeto da exceção desejada
  - E usando throw

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      IOException ex = new
          IOException("Parâmetro nulo");
      throw ex;
    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

Saída

```
$ javac Projeto.java
Projeto.java:60: unreported exception
java.io.IOException; must be caught or
declared to be thrown
         throw ex;
         ^
1 error
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome) {
    if (nome == null) {
      IOException ex = new
          IOException("Parâmetro nulo");
      throw ex;
    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

Saída

```
$ javac Projeto.java
Projeto.java:60: unreported exception
java.io.IOException; must be caught or
declared to be thrown
         throw ex;
         ^
1 error
```

- Temos também que declarar que o método lança a exceção

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class Projeto {
  ...
  public boolean gravaArq(String nome)
                        throws IOException {
    if (nome == null) {
      IOException ex = new
          IOException("Parâmetro nulo");
      throw ex;
    }
    File arq = new File(nome);
    try {
      FileWriter writer =
                new FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções

- E o que acontece ao fazermos:

```
class Projeto {
  ...
  public static void main(String[] args)
  {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    Projeto p = new Projeto();
    p.adicionaRes(new Residencia(cr,
                                 null));
    p.adicionaRes(new Residencia(cq,
                                 null));


      p.gravaArq(null);




  }
}
```

Exceções

- E o que acontece ao fazermos:

Saída

```
$ javac Projeto.java
Projeto.java:86: unreported
exception java.io.IOException;
must be caught or declared to
be thrown
        p.gravaArq(null);
                   ^
1 error
```

```
class Projeto {
  ...
  public static void main(String[] args)
  {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    Projeto p = new Projeto();
    p.adicionaRes(new Residencia(cr,
                                 null));
    p.adicionaRes(new Residencia(cq,
                                 null));


      p.gravaArq(null);



  }
}
```

Exceções

- E o que acontece ao fazermos:

Saída

```
$ javac Projeto.java
Projeto.java:86: unreported
exception java.io.IOException;
must be caught or declared to
be thrown
        p.gravaArq(null);
                   ^
1 error
```

- Temos que adicionar o bloco `try ... catch`

```
class Projeto {
  ...
  public static void main(String[] args)
  {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    Projeto p = new Projeto();
    p.adicionaRes(new Residencia(cr,
                                 null));
    p.adicionaRes(new Residencia(cq,
                                 null));

    try {
      p.gravaArq(null);
    }
    catch (IOException ioe) {
      System.out.println(
                        ioe.getMessage());
    }
  }
}
```

Exceções

- E agora?

```
class Projeto {
  ...
  public static void main(String[] args)
  {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    Projeto p = new Projeto();
    p.adicionaRes(new Residencia(cr,
                                 null));
    p.adicionaRes(new Residencia(cq,
                                 null));

    try {
      p.gravaArq(null);
    }
    catch (IOException ioe) {
      System.out.println(
                       ioe.getMessage());
    }
  }
}
```

Exceções

- E agora?

Saída

```
$ java Projeto
Parâmetro nulo
```

```
class Projeto {
  ...
  public static void main(String[] args)
  {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    Projeto p = new Projeto();
    p.adicionaRes(new Residencia(cr,
                                 null));
    p.adicionaRes(new Residencia(cq,
                                 null));

    try {
      p.gravaArq(null);
    }
    catch (IOException ioe) {
      System.out.println(
                       ioe.getMessage());
    }
  }
}
```

Exceções

- E se `IOException` não for a ideal?

Exceções

- E se `IOException` não for a ideal?
- E se não houver ideal?

Exceções

- E se `IOException` não for a ideal?
- E se não houver ideal?
  - Cria-se uma exceção, subclasse de `Exception` (ou de qualquer outra exceção)

Exceções

- E se `IOException` não for a ideal?
- E se não houver ideal?
  - Cria-se uma exceção, subclasse de `Exception` (ou de qualquer outra exceção)

```
class MinhaExcecao extends Exception
{
  public MinhaExcecao() {}

  public MinhaExcecao(String msg){
    super(msg);
  }
}
```

Exceções

- Usamos essa exceção como qualquer outra

Exceções

- Usamos essa exceção como qualquer outra

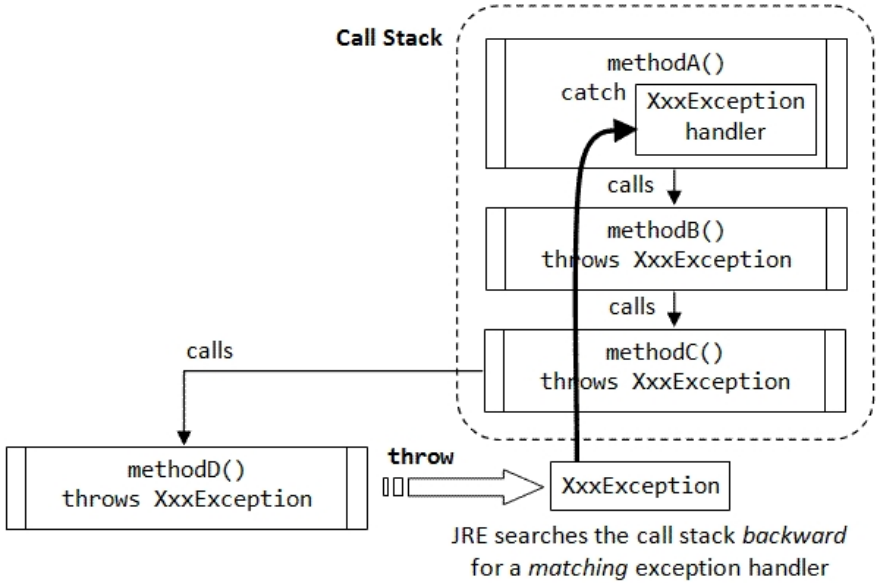
```
class Projeto {
  ...
  public boolean gravaArq(String nome)
                          throws MinhaExcecao {
    if (nome == null) {
      MinhaExcecao ex = new
             MinhaExcecao("Parâmetro nulo");
      throw ex;
    }

    File arq = new File(nome);
    try {
      FileWriter writer = new
                          FileWriter(arq,true);
    }
    catch(IOException ioe) {
      return(false);
    }
    return(true);
  }
}
```

Exceções – Pilha de Chamadas

- O que acontece quando alguem lança uma exceção?

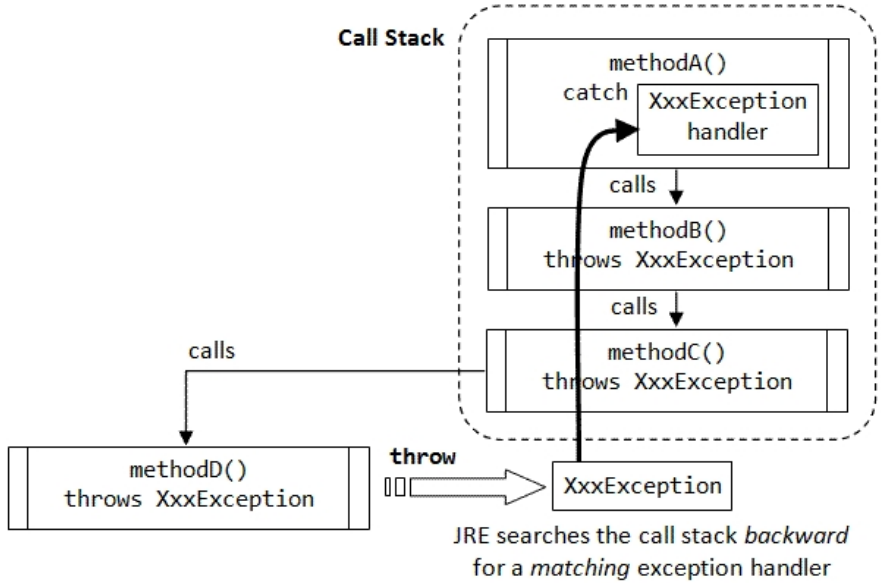
Exceções – Pilha de Chamadas

- O que acontece quando alguem lança uma exceção?
  - À medida em que métodos são chamados dentro de outros métodos, o JRE armazena todos eles em uma estrutura chamada Pilha de Chamadas



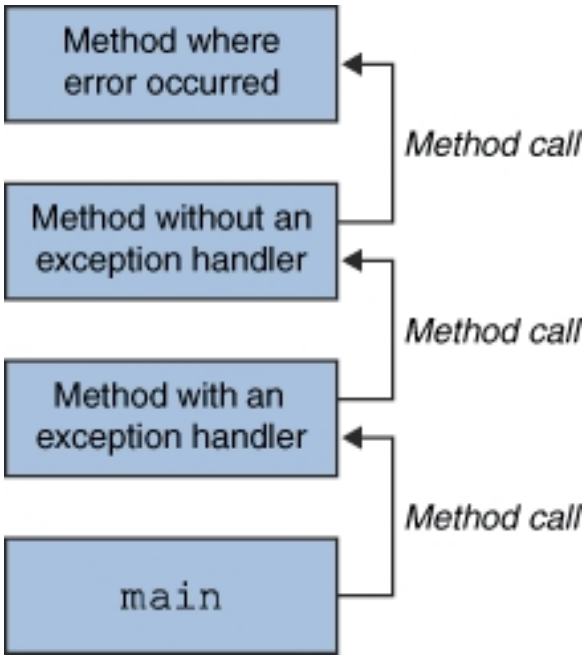
Exceções – Pilha de Chamadas

- O que acontece quando alguem lança uma exceção?
  - À medida em que métodos são chamados dentro de outros métodos, o JRE armazena todos eles em uma estrutura chamada Pilha de Chamadas
  - Se um determinado método lança uma exceção, o JRE busca seu tratamento na pilha, indo desde o método que chamou o método que lançou a exceção, até atingir main, se nenhum método no caminho tratar da exceção.

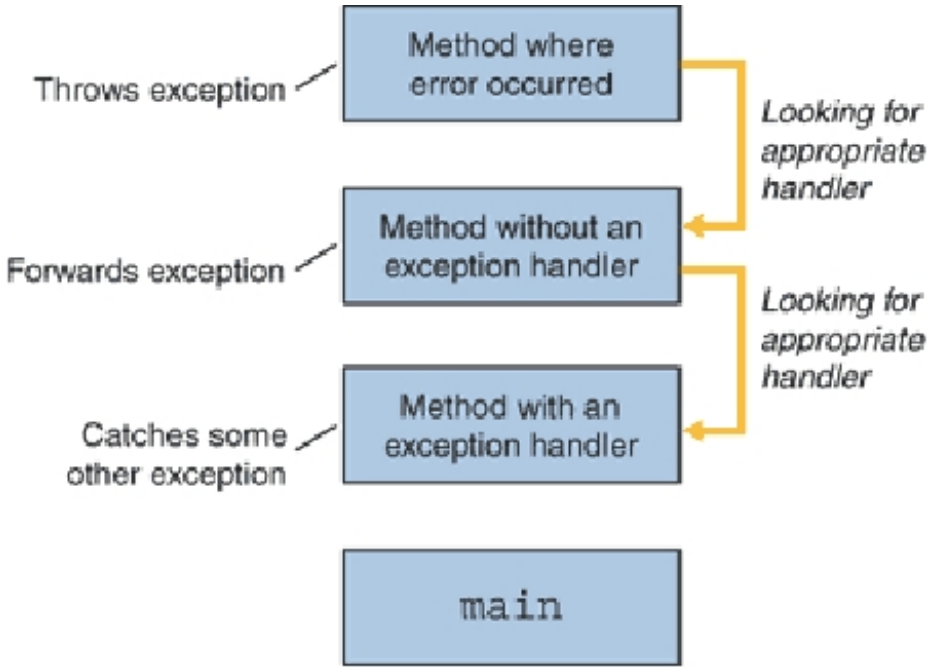


Pilha de Chamadas

Chamada



Busca por tratamento da Exceção



Arquivos

- E como podemos ver a pilha de execução?

Arquivos

- E como podemos ver a pilha de execução?
  - `printStackTrace`

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  try {
    p.gravaArq(null);
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }



}
```

Arquivos

- E como podemos ver a pilha de execução?
  - `printStackTrace`
- A existência de múltiplas exceções abre para a possibilidade de múltiplos blocos catch.

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  try {
    p.gravaArq(null);
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }



}
```

Arquivos

- Criamos um para cada tipo de exceção tratada

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  try {
    p.gravaArq(null);
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }
  catch (MinhaExcecao me) {
    System.out.println(me.getMessage());
  }
}
```

Arquivos

- Criamos um para cada tipo de exceção tratada
- Com código específico para cada exceção

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  try {
    p.gravaArq(null);
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }
  catch (MinhaExcecao me) {
    System.out.println(me.getMessage());
  }
}
```

Arquivos

- Criamos um para cada tipo de exceção tratada
- Com código específico para cada exceção
- Nos permite tomar decisões diferentes conforme o tipo de erro encontrado.

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  try {
    p.gravaArq(null);
  }
  catch (IOException ioe) {
    ioe.printStackTrace();
  }
  catch (MinhaExcecao me) {
    System.out.println(me.getMessage());
  }
}
```

Algumas Exceções

```
int x = Integer.parseInt("five");
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
```

Algumas Exceções

```
int x = Integer.parseInt("five");
```
java.lang.NumberFormatException

```
int[] y = {1, 3};
System.out.println(y[3]);
```
java.lang.ArrayIndexOutOfBoundsException

```
int[] z = {1, 3};
z[-1] = 2;
```
java.lang.ArrayIndexOutOfBoundsException

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
java.lang.ArrayIndexOutOfBoundsException
```

```
int w = 10 / 0;
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
java.lang.ArrayIndexOutOfBoundsException
```

```
int w = 10 / 0;
java.lang.ArithmeticException
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
java.lang.ArrayIndexOutOfBoundsException
```

```
int w = 10 / 0;
java.lang.ArithmeticException
```

```
class Aula23 {
  public static void main(String[] args)
  {
    try {
      int w = 10 / 0;
    }
    catch(Exception e) {

    }
  }
}
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
java.lang.ArrayIndexOutOfBoundsException
```

```
int w = 10 / 0;
java.lang.ArithmeticException
```

```
class Aula23 {
  public static void main(String[] args)
  {
    try {
      int w = 10 / 0;
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

Algumas Exceções

```
int x = Integer.parseInt("five");
java.lang.NumberFormatException
```

```
int[] y = {1, 3};
System.out.println(y[3]);
java.lang.ArrayIndexOutOfBoundsException
```

```
int[] z = {1, 3};
z[-1] = 2;
java.lang.ArrayIndexOutOfBoundsException
```

```
int w = 10 / 0;
java.lang.ArithmeticException
```

```
class Aula23 {
  public static void main(String[] args)
  {
    try {
      int w = 10 / 0;
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

Saída

```
$ java Aula23
java.lang.ArithmeticException: / by zero
        at Aula23.main(Aula23.java:5)
```

Por fim...

- Assim como podemos capturar múltiplas exceções, também podemos repassá-las

Por fim...

- Assim como podemos capturar múltiplas exceções, também podemos repassá-las
- Ex:
  - `public boolean gravaArq(String nome) throws MinhaExcecao, IOException {...}`

Checked × Unchecked

Há dois tipos básicos de exceção

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):
  - Não precisam ser capturadas ou propagadas

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):
  - Não precisam ser capturadas ou propagadas
  - Passam "despercebidas" pelo compilador

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):
  - Não precisam ser capturadas ou propagadas
  - Passam "despercebidas" pelo compilador
  - O programador só descobre que elas existem quando há um erro que pára o programa

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):
  - Não precisam ser capturadas ou propagadas
  - Passam "despercebidas" pelo compilador
  - O programador só descobre que elas existem quando há um erro que pára o programa
  - Estendem `java.lang.RuntimeException`

Checked × Unchecked

Há dois tipos básicos de exceção

- Checked:
  - Devem ser explicitamente capturadas ou propagadas
  - Estendem `java.lang.Exception`
- Unchecked (ou Run-Time):
  - Não precisam ser capturadas ou propagadas
  - Passam "despercebidas" pelo compilador
  - O programador só descobre que elas existem quando há um erro que pára o programa
  - Estendem `java.lang.RuntimeException`
  - MUITO CUIDADO COM ESSAS!

Arquivos

- Voltemos ao código.

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Voltemos ao código.
- Por simplicidade, retornamos `false` no caso de `nome == null`

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Voltemos ao código.
- Por simplicidade, retornamos `false` no caso de `nome == null`
- O que isso faz?

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Voltemos ao código.
- Por simplicidade, retornamos `false` no caso de `nome == null`
- O que isso faz?
  - Cria um arquivo de nome `nome`

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
```

```
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Voltemos ao código.
- Por simplicidade, retornamos `false` no caso de `nome == null`
- O que isso faz?
  - Cria um arquivo de nome `nome`
  - Se já existir, sobrescreve

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Falta gravar as informações relevantes lá

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Falta gravar as informações relevantes lá
  - No caso, a área da casa e da piscina

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
    No p = this.condominio.getCabeca();
    while (p != null) {
      Residencia r = p.getRes();
      if (r.casa != null) {
        writer.write("Casa:Área:" +
                          r.casa.area()+"\n");
      }
      if (r.piscina != null) {
        writer.write("Piscina:Área:" +
                       r.piscina.area()+"\n");
      }
      p = p.getProx();
    }

  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Falta gravar as informações relevantes lá
  - No caso, a área da casa e da piscina
  - Repare nos \n

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
    No p = this.condominio.getCabeca();
    while (p != null) {
      Residencia r = p.getRes();
      if (r.casa != null) {
        writer.write("Casa:Área:" +
                          r.casa.area()+"\n");
      }
      if (r.piscina != null) {
        writer.write("Piscina:Área:" +
                       r.piscina.area()+"\n");
      }
      p = p.getProx();
    }

  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- Falta gravar as informações relevantes lá
  - No caso, a área da casa e da piscina
  - Repare nos \n
- E fechar o arquivo (impedindo novas gravações)

```
public boolean gravaArq(String nome) {
  if (nome == null) return(false);
  File arq = new File(nome);
  try {
    FileWriter writer = new
                        FileWriter(arq,false);
    No p = this.condominio.getCabeca();
    while (p != null) {
      Residencia r = p.getRes();
      if (r.casa != null) {
        writer.write("Casa:Área:" +
                          r.casa.area()+"\n");
      }
      if (r.piscina != null) {
        writer.write("Piscina:Área:" +
                       r.piscina.area()+"\n");
      }
      p = p.getProx();
    }
    writer.close();
  }
  catch(IOException ioe) { return(false); }
  return(true);
}
```

Arquivos

- E como usamos isso?

Arquivos

- E como usamos isso?

```
public static void main(String[] args) {
  CasaRet cr = new CasaRet(10,5,1320);
  CasaQuad cq = new CasaQuad(10,1523);

  Projeto p = new Projeto();
  p.adicionaRes(new Residencia(cr, null));
  p.adicionaRes(new Residencia(cq, null));

  if (!p.gravaArq("oba"))
    System.out.println("Problema");
}
```

Arquivos

- Vimos como escrevemos em um arquivo.

Arquivos

- Vimos como escrevemos em um arquivo.
- Mas, e como lemos de um arquivo?

Arquivos

- Vimos como escrevemos em um arquivo.
- Mas, e como lemos de um arquivo?
  - Com `Scanner`

```
import java.util.Scanner;
import java.io.FileNotFoundException;
...
public static void main(String[] args) {
  try {
    File arq = new File("oba");
    Scanner sc = new Scanner(arq);
    while (sc.hasNext()) {
      System.out.println(sc.next());
    }
    sc.close();
  }
  catch (FileNotFoundException fnfe) {
    System.out.println(fnfe.getMessage());
  }
}
```

Arquivos

- Vimos como escrevemos em um arquivo.
- Mas, e como lemos de um arquivo?
  - Com `Scanner`
- Teremos que capturar exceção

```
import java.util.Scanner;
import java.io.FileNotFoundException;
...
public static void main(String[] args) {
  try {
    File arq = new File("oba");
    Scanner sc = new Scanner(arq);
    while (sc.hasNext()) {
      System.out.println(sc.next());
    }
    sc.close();
  }
  catch (FileNotFoundException fnfe) {
    System.out.println(fnfe.getMessage());
  }
}
```

Arquivos

- Scanner quebra os símbolos usando como delimitador o espaço

```
import java.util.Scanner;
import java.io.FileNotFoundException;
...
public static void main(String[] args) {
  try {
    File arq = new File("oba");
    Scanner sc = new Scanner(arq);
    while (sc.hasNext()) {
      System.out.println(sc.next());
    }
    sc.close();
  }
  catch (FileNotFoundException fnfe) {
    System.out.println(fnfe.getMessage());
  }
}
```

Arquivos

- Scanner quebra os símbolos usando como delimitador o espaço
- E como ler a linha inteira?

```
import java.util.Scanner;
import java.io.FileNotFoundException;
...
public static void main(String[] args) {
  try {
    File arq = new File("oba");
    Scanner sc = new Scanner(arq);
    while (sc.hasNext()) {
      System.out.println(sc.next());
    }
    sc.close();
  }
  catch (FileNotFoundException fnfe) {
    System.out.println(fnfe.getMessage());
  }
}
```

Arquivos

- Scanner quebra os símbolos usando como delimitador o espaço
- E como ler a linha inteira?
  - Com `nextLine()`

```
import java.util.Scanner;
import java.io.FileNotFoundException;
...
public static void main(String[] args) {
  try {
    File arq = new File("oba");
    Scanner sc = new Scanner(arq);
    while (sc.hasNext()) {
      System.out.println(sc.nextLine());
    }
    sc.close();
  }
  catch (FileNotFoundException fnfe) {
    System.out.println(fnfe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Encriptar o conteúdo de um arquivo oba, contendo:

```
teste1
teste2 teste3
```

Arquivos – Um exemplo mais complexo

- Encriptar o conteúdo de um arquivo oba, contendo:

```
teste1
teste2 teste3
```

- Como fazer?

Arquivos – Um exemplo mais complexo

- Encriptar o conteúdo de um arquivo oba, contendo:

  `teste1`
  `teste2 teste3`

- Como fazer?

```
import java.io.File;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileInputStream;

public class Encriptador {
  public static void main(String[] args){
    ...
  }
}
```

Arquivos – Um exemplo mais complexo

- Encriptar o conteúdo de um arquivo oba, contendo:

  `teste1`
  `teste2 teste3`

- Como fazer?
  - Usamos as classes específicas para lidar com bytes de dados

```
import java.io.File;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileInputStream;

public class Encriptador {
  public static void main(String[] args){
    ...
  }
}
```

Arquivos – Um exemplo mais complexo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
                new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo
  - read retorna -1 ao final do arquivo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo
  - read retorna -1 ao final do arquivo
- Encriptamos

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo
  - read retorna -1 ao final do arquivo
- Encriptamos
- Gravamos no novo arquivo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
                new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo
  - read retorna -1 ao final do arquivo
- Encriptamos
- Gravamos no novo arquivo
- Apagamos o antigo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- Criamos um novo arquivo, oba2
- Lemos cada linha do antigo
  - read retorna -1 ao final do arquivo
- Encriptamos
- Gravamos no novo arquivo
- Apagamos o antigo
- Substituimos pelo novo

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- E como encriptamos?

Arquivos – Um exemplo mais complexo

- E como encriptamos?
- Da maneira mais simples:

Arquivos – Um exemplo mais complexo

- E como encriptamos?
- Da maneira mais simples:
  - Invertendo os bits do byte lido

Arquivos – Um exemplo mais complexo

- E como encriptamos?
- Da maneira mais simples:
  - Invertendo os bits do byte lido
- ~ Corresponde a um não bit a bit

Arquivos – Um exemplo mais complexo

- E como encriptamos?
- Da maneira mais simples:
  - Invertendo os bits do byte lido
- ~ Corresponde a um não bit a bit
- Os bits da variável são analisados individualmente

Arquivos – Um exemplo mais complexo

- E como encriptamos?
- Da maneira mais simples:
  - Invertendo os bits do byte lido
- ~ Corresponde a um não bit a bit
- Os bits da variável são analisados individualmente
  - Bit 1 vira 0 e bit 0 vira 1

Arquivos – Um exemplo mais complexo

Ex:

Arquivos – Um exemplo mais complexo

Ex:

- ~11001010 $\rightarrow$ 00110101

Arquivos – Um exemplo mais complexo

Ex:

- ~11001010 $\rightarrow$ 00110101
- ~00110101 $\rightarrow$ 11001010

Arquivos – Um exemplo mais complexo

Ex:

- ~11001010 $\rightarrow$ 00110101
- ~00110101 $\rightarrow$ 11001010

- Essa característica (reversibilidade) é o que o torna útil ao nosso propósito

Arquivos – Um exemplo mais complexo

- E qual a saída de nosso código?

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
              new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- E qual a saída de nosso código?

Saída

```
$ cat oba
teste1
teste2 teste3
$ java Encriptador
$ cat oba
▮▮▮▮▮▮▮
$ java Encriptador
$ cat oba
teste1
teste2 teste3
```

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
                new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Arquivos – Um exemplo mais complexo

- E qual a saída de nosso código?

Saída

```
$ cat oba
teste1
teste2 teste3
$ java Encriptador
$ cat oba
▮▮▮▮▮▮▮
$ java Encriptador
$ cat oba
teste1
teste2 teste3
```

- Note que o mesmo código que encripta, também decripta

```
public static void main(String[] args) {
  File arq = new File("oba");
  File arq2 = new File("oba2");
  try {
    FileOutputStream writer = new
           FileOutputStream(arq2,false);
    FileInputStream reader =
                new FileInputStream(arq);
    int b;
    while ((b = reader.read()) != -1) {
      b = ~b;
      writer.write(b);
    }
    writer.close();
    reader.close();
    arq.delete();
    arq2.renameTo(arq);
  }
  catch(IOException ioe) {
    System.out.println(ioe.getMessage());
  }
}
```

Operadores Bit a Bit

Aplicam-se a inteiros (`int`, `long`, `char`, `byte`)

Operador	Ação	Exemplo
~	*not* inverte bits	~11001010 00110101
&	*and* filtra bits	11001010 & 01100110 01000010
|	*or* "liga" bits	11001010 | 01100010 11101010
^	*xor* ou exclusivo "desliga" bits	11001010 ^ 01100010 10101000

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas

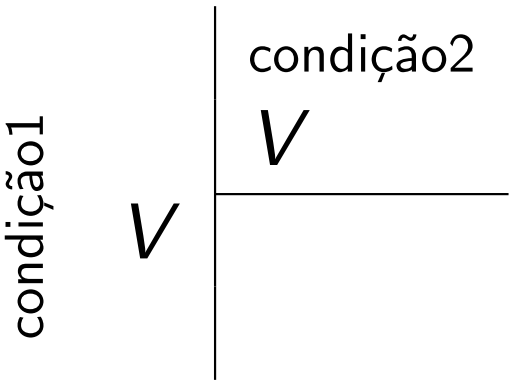
Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?




```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1 \ condição2	V	F
V		

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1 \ condição2	V	F
V	F	

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1	condição2 V	condição2 F
V	F	V

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1	condição2 V	condição2 F
V	F	V
F		

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1	condição2 V	condição2 F
V	F	V
F	V	

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Ou Exclusivo

- Verdadeiro se ou uma das condições é verdadeira ou a outra é, nunca as duas ou nenhuma delas
- **XOr**: Comando1 será executado?

condição1	condição2 V	condição2 F
V	F	V
F	V	F

```
if ((condição1 || condição2)
    &&
    !(condição1 && condição2))
                        comando1;
```

Arquivos

Atenção!

- Nessa aula não discutimos sobre arquivos que contêm todo o leque de caracteres Unicode ou mesmo todas as extensões do ASCII. Focamos apenas em arquivos ASCII "puros".
- Aqueles que, contudo, procuram esse tipo de informação, podem dar uma olhada na classe `java.io.OutputStreamWriter` sobre como criar um Writer levando em conta a codificação dos caracteres do arquivo.

Videoaula

https://www.youtube.com/watch?v=dHou1G8iYo4

e

https://www.youtube.com/watch?v=4WYyd7MNDqQ

(cobrem parcialmente o conteúdo)

Referências Adicionais

- http://www.guj.com.br/articles/13
- http://www.devmedia.com.br/articles/viewcomp.asp?comp=1636
- http://download.oracle.com/javase/6/docs/api/java/io/File.html
- http://download.oracle.com/javase/6/docs/api/java/io/FileWriter.html
- http://tutorials.jenkov.com/java-exception-handling/basic-try-catch-finally.html

Referências Adicionais

- http://download.oracle.com/javase/tutorial/essential/exceptions/index.html
- Horstmann, C.S.; Cornell, G.: Core Java 2: Volume I - Fundamentals. Prentice Hall. 2002.
- http://www3.ntu.edu.sg/home/ehchua/programming/java/J5a_Exception.html
- Bloch, J.: Effective Java: Programming Language Guide. Addison-Wesley. 2001.
- http://download.oracle.com/javase/tutorial/java/nutsandbolts/op3.html