Programação Orientada a Objetos em

Flávio Miguel Varejão

Departamento de Informática

UFES

Notação para Sistemas de Arquivos

- Plataformas Diferentes
  - diferentes notações para trilhas
  - windows: c:/java/aplic
  - unix: \usr\home\java\aplic
- Classe File
  - notação multiplataforma
    - prefixo
    - sequência de strings
    - File.separator

Identificação de Arquivos

```
File meuArquivo1 = new File("meuArq1.txt");
File meuArquivo2 = new File("c:\\meuDir",
                            "meuArq2.txt");
File meuDir = new File("c:\\meuDir");
File meuArquivo3 = new File(meuDir, "meuArq3.txt");
```

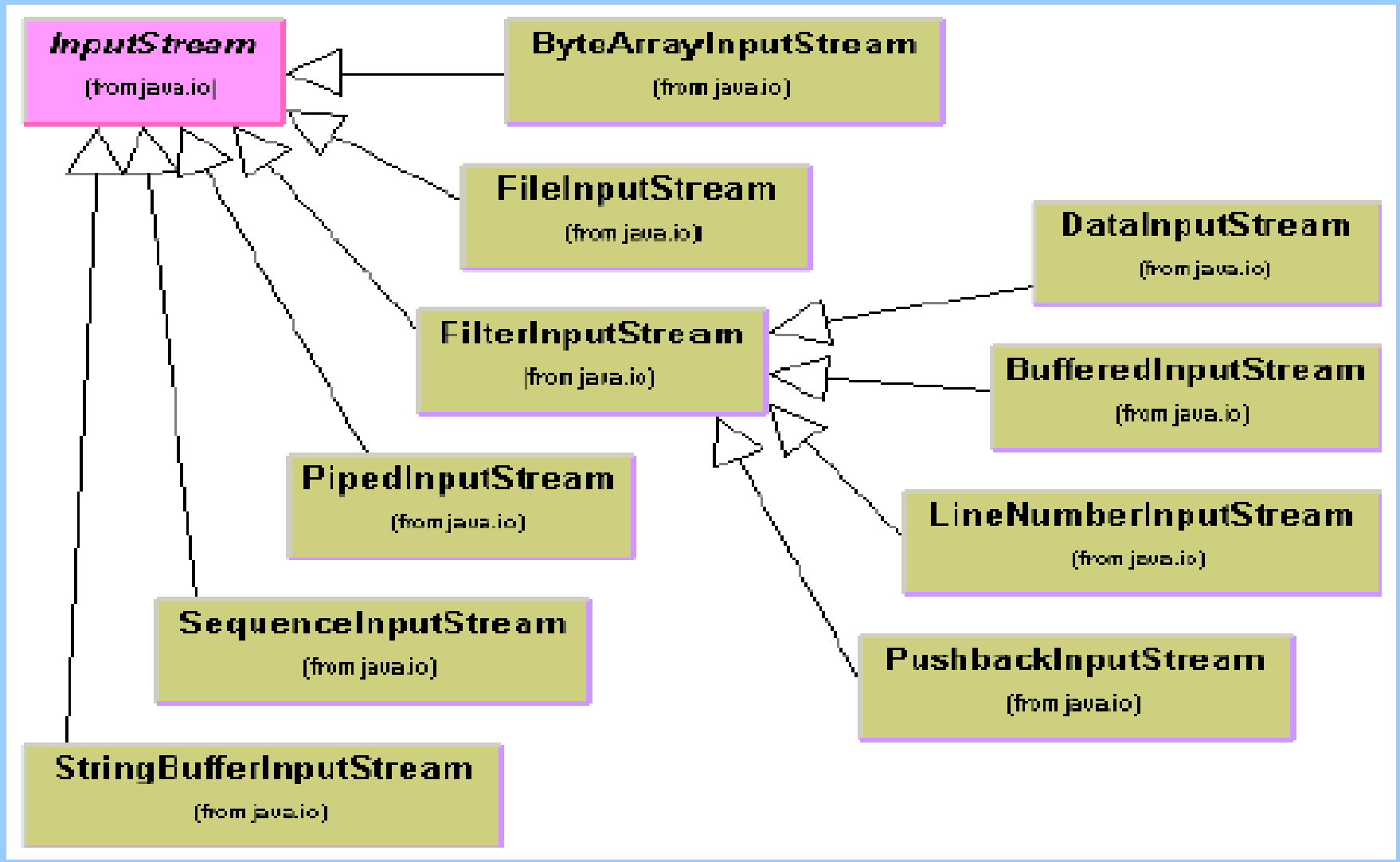
Streams

- Sequências de bytes
- Arquivos, conexões de redes, buffers
- Tipos existentes em Java
  - orientadas a byte (ASCII)
    - InputStream
    - OutputStream
  - orientadas a caractere (Unicode)
    - Reader
    - Writer

Streams

- São necessárias inúmeras combinações para atender demanda
  - Leitura de arquivo bufferizada
  - Leitura de arquivo não bufferizada
  - Escrita em arquivo bufferizada
  - etc
- Padrão de Projeto Decorador
  - adição incremental de funcionalidades
  - uso de filtros

InputStream



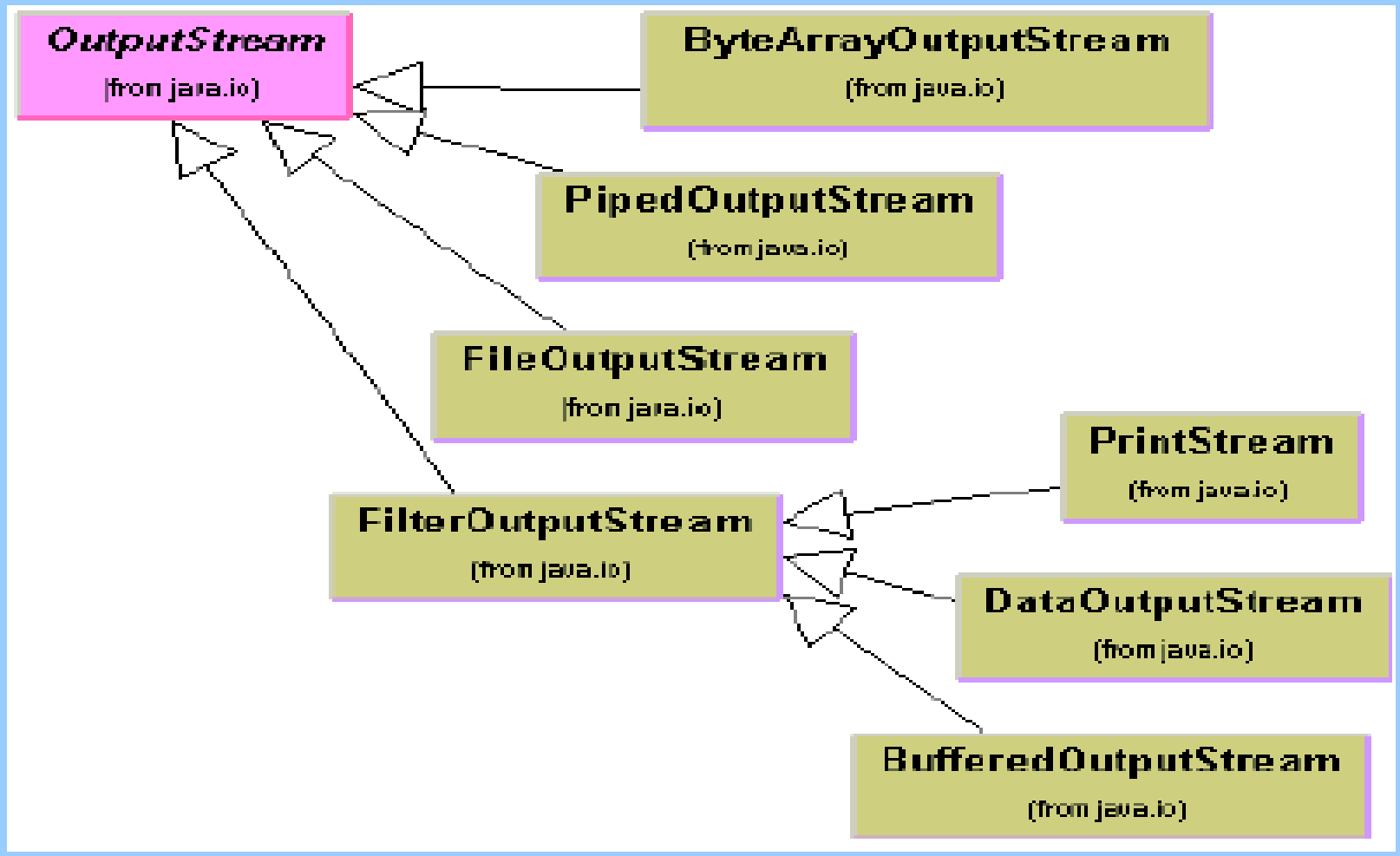
InputStream

```
//UsaInputStream.java
import java.io.*;
public class UsaInputStream {
  public static void main(String [] args) {
    try {
     // constroi objeto stream para leitura de arquivo
     DataInputStream in =
        new DataInputStream (
          new BufferedInputStream (
            new FileInputStream("NomeArquivo.ext")));
     // string auxiliar para linha lida
     String linha;
     // string para armazenar todo texto lido
     String buffer = new String();
     // le primeira linha
     linha = in.readLine();
```

InputStream

```
        // enquanto leitura não nula
        while (linha != null) {
          // adiciona linha ao buffer
          buffer += linha + "\n";
          // le proxima linha
          linha = in.readLine();
        }
        in.close(); // fecha stream de leitura
      } catch (IOException exc) {
        System.out.println("Erro de IO");
        exc.printStackTrace();
      }
}
```

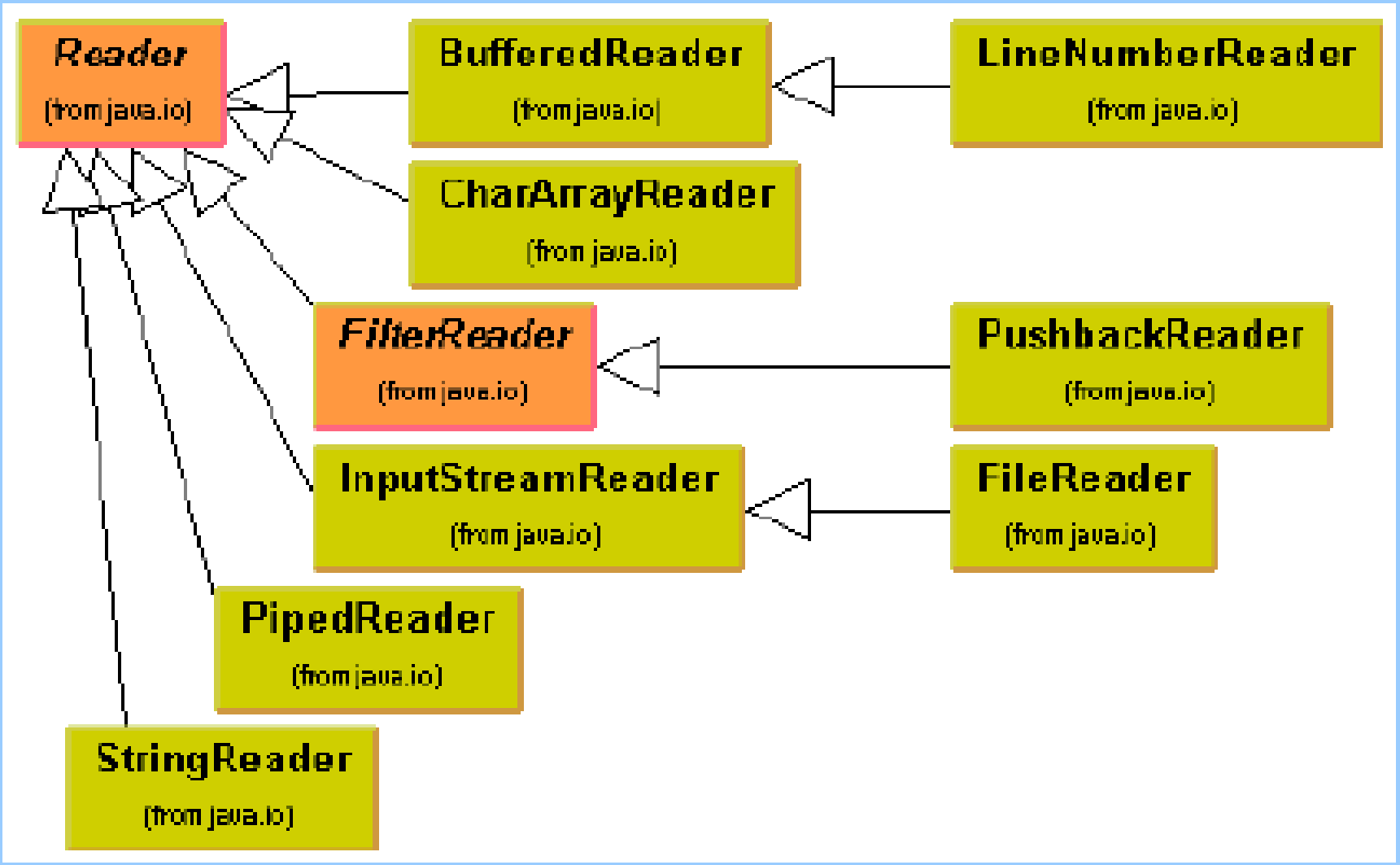
OutputStream



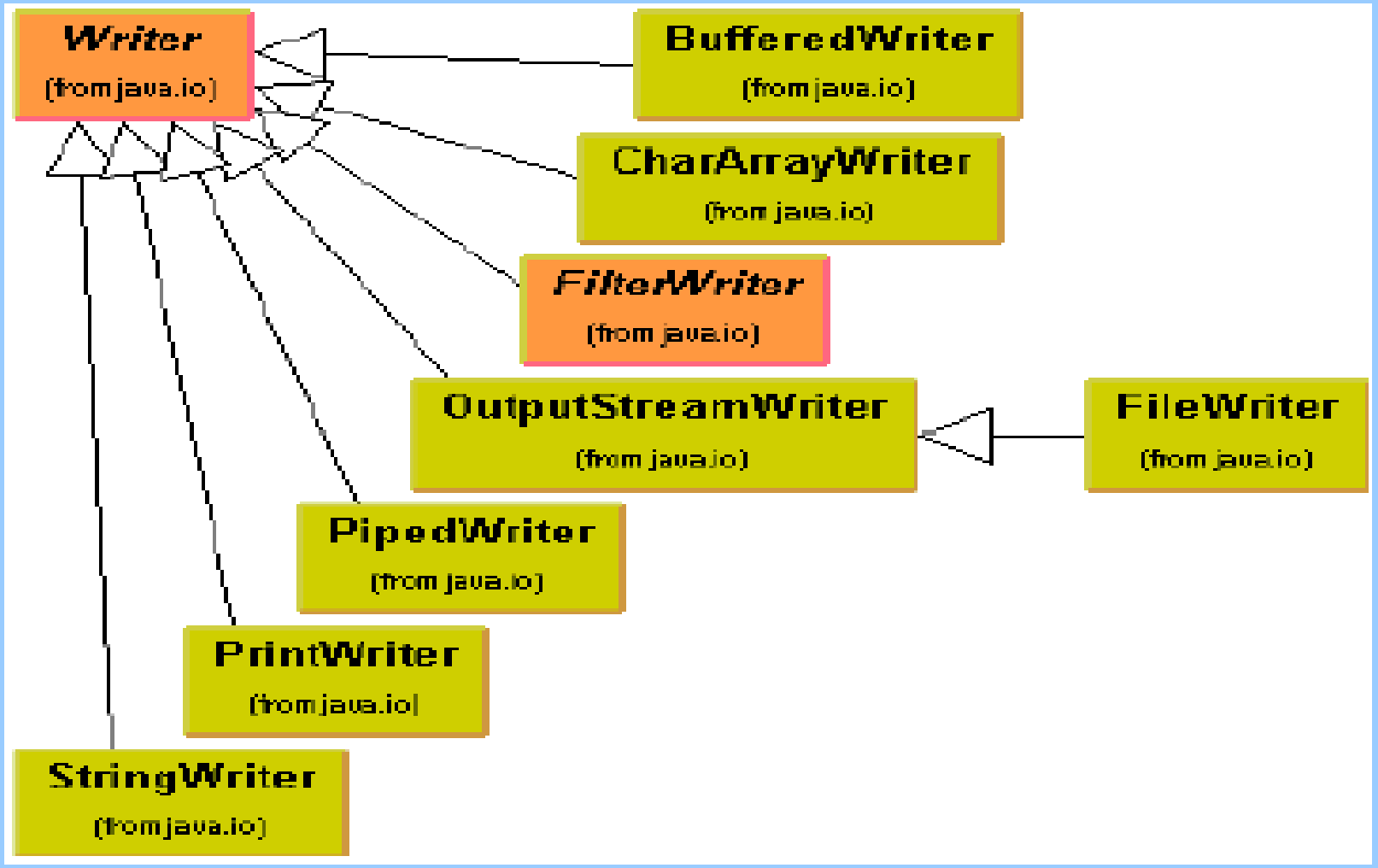
OutputStream

```
//UsaOutputStream.java
import java.io.*;
public class UsaOutputStream {
  public static void main(String [] args) {
    try {
      DataOutputStream out =
        new DataOutputStream(
          new BufferedOutputStream(
            new FileOutputStream("Arquivo.ext")));
      out.writeBytes("Uma frase simples");
      out.writeDouble(265.34554);
      out.close();
    } catch (IOException exc) {
      System.out.println("Erro de IO");
      exc.printStackTrace();
    }
  }
}
```

Reader


Reader
(from java.io)
BufferedReader
(from java.io)
LineNumberReader
(from java.io)
CharArrayReader
(from java.io)
FilterReader
(from java.io)
PushbackReader
(from java.io)
InputStreamReader
(from java.io)
FileReader
(from java.io)
PipedReader
(from java.io)
StringReader
(from java.io)

Writer


Writer
(from java.io)
BufferedWriter
(from java.io)
CharArrayWriter
(from java.io)
FilterWriter
(from java.io)
OutputStreamWriter
(from java.io)
FileWriter
(from java.io)
PipedWriter
(from java.io)
PrintWriter
(from java.io)
StringWriter
(from java.io)

Arquivos Texto

```
import java.io.*;
public class ArquivoTexto {
  File arquivo;
  BufferedReader in;
  PrintWriter out;
    ArquivoTexto(String nomeArq) {
     arquivo = new File (nomeArq);
  }
  ArquivoTexto(String nomeDir, String nomeArq) {
     arquivo = new File (nomeDir, nomeArq);
  }
```

Arquivos Texto

```
void abreLeitura() {
     try {
       in = new BufferedReader(new
                     FileReader(arquivo));
     } catch (FileNotFoundException e) {
       System.err.println ("Arquivo nao encontrado: "
                     + arquivo);
       e.printStackTrace();
     }
  }
  void abreEscrita() {
     try {
       out = new PrintWriter(
                  new FileWriter(arquivo, true));
     } catch (IOException e) {
       e.printStackTrace();
     }
  }
```

Arquivos Texto

```
void fechaLeitura() {
   try {
     in.close();
   } catch (IOException e) {
     e.printStackTrace();
   }
}
void fechaEscrita() {
     out.close();
}
```

Arquivos Texto

```
   String leProxLinha() {
      String s;
      try {
        s = in.readLine();
      } catch (IOException e) {
         e.printStackTrace();
         s=null;
      }
      return s;
   }
   void gravaProxLinha(String linha) {
      out.println(linha);
   }
}
```

Arquivos Texto

```
public class UsaArquivosTexto {
  public static void main (String [] args) {
    System.out.print (
      "Entre com nome do arquivo de entrada: ");
    String nomeEntrada = Console.readString();
    System.out.print (
      "Entre com nome do arquivo de saida: ");
    String nomeSaida = Console.readString();
    ArquivoTexto entrada = new
      ArquivoTexto (nomeEntrada);
```

Arquivos Texto

```
    ArquivoTexto saida = new ArquivoTexto (nomeSaida);
    entrada.abreLeitura();
    saida.abreEscrita();
    String s;
    while (null != (s = entrada.leProxLinha())) {
      saida.gravaProxLinha(" ==== " + s + " ==== ");
    }
    entrada.fechaLeitura();
    saida.fechaEscrita();
  }
}
```

Arquivos de Acesso Direto

```
// UsaRandom.java
import java.io.*;
public class UsaRandom {
  public static void main(String args[]) {
    System.out.print ("Entre com nome do arquivo: ");
    String nome = Console.readString();
    try {
      RandomAccessFile f =
           new RandomAccessFile (nome, "rw");
```

Arquivos de Acesso Direto

```
        try {
          f.seek(f.length());
          f.writeByte('\n');
          for (int i = 0; i < 25; i++) {
            f.writeByte('+');
          }
        } catch (IOException e) {
          System.out.println("Falhou!!!");
        } finally {
          f.close();
        }
      } catch (IOException e) {
        System.out.println("Nao abriu!!!");
      }
    }
}
```

Serialização

```
import java.io.*;
public class Impares implements Serializable {
   private static int j = 1;
   private int i = j;
   private  Impares prox;
   Impares(int n) {
       j = j + 2;
       if(--n > 0)
            prox = new Impares(n);
   }
   Impares() {
       j = j + 2;
       prox = new Impares(9);
   }
```

Serialização

```
public String toString() {
     String s = ""  +  i;
     if(prox != null)
         s += " : " + prox.toString();
     return s;
}
public static void main(String[] args) {
     Impares w = new Impares(6);
     System.out.println("w = " + w);
     try {
         ObjectOutputStream out =
              new ObjectOutputStream(
                  new FileOutputStream("impares.dat"));
         out.writeObject("Armazena Impares");
         out.writeObject(w);
         out.close();
```

Serialização

```
            ObjectInputStream in =
                 new ObjectInputStream(
                     new FileInputStream("impares.dat"));
             String s = (String)in.readObject();
             Impares z = (Impares)in.readObject();
             System.out.println(s + ", z = " + z);
             in.close();
        } catch(Exception e) {
             e.printStackTrace();
        }
    }
}
```

Serialização

```
//Persiste.java
import java.io.*;
class Dado implements Serializable {
   private int i;
   Dado(int x) { i = x; }
   public String toString() {
     return Integer.toString(i);
   }
}
```

Serialização

```
public class Persiste implements Serializable {
   private static int r() {
      return (int)(Math.random() * 10);
   }
   private Dado[] d = {
      new Dado(r()), new Dado(r()), new Dado(r())
   };
   private  Persiste prox;
   private char c;
   Persiste(int i, char x) {
     System.out.println("Construtor Persiste: " + i);
     c = x;
     if(--i > 0)
       prox = new Persiste(i, (char)(x + 1));
   }
```

Serialização

```
Persiste() {
  System.out.println("Construtor Default ");
}
public String toString() {
  String s = ":" + c + "(";
  for(int i = 0; i < d.length; i++)
    s += d[i].toString();
  s += ")";
  if(prox != null)
    s += prox.toString();
  return s;
}
```

Serialização

```
public static void main(String[] args) {
  Persiste w = new Persiste(6, 'a');
  System.out.println("w = " + w);
  try {
    ObjectOutputStream out =
       new ObjectOutputStream(
          new FileOutputStream("persiste.out"));
    out.writeObject("Armazena Persiste");
    out.writeObject(w);
    out.close();
```

Serialização

```
        ObjectInputStream in =
           new ObjectInputStream(
              new FileInputStream("persiste.out"));
        String s = (String)in.readObject();
        Persiste w2 = (Persiste)in.readObject();
        System.out.println(s + ", w2 = " + w2);
      } catch(Exception e) {
        e.printStackTrace();
      }
    }
}
```