PARADIGMAS DE PROGRAMAÇÃO

2023/2024

P.PORTO
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Aula 12

1. Palavras Reservadas
2. Input/Output em Java
3. Serialização
4. Wrapper types
5. Palavras Reservadas Usadas
6. Links Úteis

Ricardo Santos | rjs@estg.ipp.pt

Palavras Reservadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Input/Output em *Java*

- Em Java existe um conceito - *stream*
- Uma *stream* é uma abstracção para uma qualquer forma de entrada ou saída de dados
- As *streams* podem ser catalogadas em dois grandes grupos:
  - ***Streams* de *Bytes***
  - ***Streams* de *Caracteres***

Usar *Streams* de *Bytes*

- No topo da hierarquia das *streams* de *bytes* estão as classes `InputStream` e `OutputStream`
- Consulte a **API** para ver métodos de `InputStream` e `OutputStream`

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

```
// Read an array of bytes from the keyboard.

import java.io.*;

class ReadBytes {
   public static void main(String args[])
                throws IOException {

      byte data[] = new byte[10];
      System.out.println("Enter some characters.");

      System.in.read(data);

      System.out.print("You entered: ");
      for(int i=0; i < data.length; i++)
         System.out.print((char) data[i]);
      }
}
```

```
// Demonstrate System.out.write().

class WriteDemo {
   public static void main(String args[]) {
      int b;

      b = 'X';

      System.out.write(b);

      System.out.write('\n');
   }
}
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Ler e Escrever Ficheiros a partir de *Streams* de *Bytes*

- Existem várias formas para ler e escrever sequências de *bytes* e armazenar os valores em formato binário
- As subclasses de `FileOutputStream` e `FileInputStream` disponibilizam vários métodos para esse efeito
- Consulte a API para ver métodos de `FileInputStream` e `FileOutputStream`

```
class ShowFile {
   public static void main(String args[]) throws IOException {
      int i;
      FileInputStream fin;
      try {
         fin = new FileInputStream(args[0]);
      } catch(FileNotFoundException exc) {
         System.out.println("File Not Found");
         return;
      } catch(ArrayIndexOutOfBoundsException exc) {
         System.out.println("Usage: ShowFile File");
         return;
      }
      // read bytes until EOF is encountered
      do {
         i = fin.read();
         if(i != -1) System.out.print((char) i);
      } while(i != -1);
      fin.close();
   }
}
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

```
class CopyFile {
   public static void main(String args[]) throws IOException {
      int i;
      FileInputStream fin;
      FileOutputStream fout;
      try {
      // open input file
         try {
            fin = new FileInputStream(args[0]);
         } catch(FileNotFoundException exc) {
            System.out.println("Input File Not Found");
            return;
         }
      // open output file
      try {
         fout = new FileOutputStream(args[1]);
         } catch(FileNotFoundException exc) {
            System.out.println("Error Opening Output File");
            return;
         }
      } catch(ArrayIndexOutOfBoundsException exc) {
         System.out.println("Usage: CopyFile From To");
         return;
      }
```

```
        // Continuação da classe CopyFile



        // Copy File
        try {
          do {
             i = fin.read();
             if(i != -1) fout.write(i);
          } while(i != -1);

        } catch(IOException exc) {
           System.out.println("File Error");
        }
        fin.close();
        fout.close();
    }
}
```

Ler e Escrever Dados Binários

- Até agora temos lido e escrito *bytes* que contêm caracteres **ASCII**
- Também é possível ler e escrever outros tipos de dados
- Para ler e escrever valores binários de tipos primitivos *JAVA* é necessário usar a `DataInputStream` e a `DataOutputStream`

```
class RWData {
   public static void main(String args[]) throws IOException {
      DataOutputStream dataOut;
      DataInputStream dataIn;
      int i = 10;
      double d = 1023.56;
      boolean b = true;

      try {
         dataOut = new DataOutputStream(new
                        FileOutputStream("testdata"));
      }
      catch(IOException exc) {
         System.out.println("Cannot open file.");
         return;
      }
```

```
// Continuação da classe RWData


try {
   System.out.println("Writing " + i);
   dataOut.writeInt(i);
   System.out.println("Writing " + d);
   dataOut.writeDouble(d);
   System.out.println("Writing " + b);
   dataOut.writeBoolean(b);
   System.out.println("Writing " + 12.2 * 7.4);
   dataOut.writeDouble(12.2 * 7.4);
}

catch(IOException exc) {
   System.out.println("Write error.");
}
```

```
// Continuação da classe RWData


dataOut.close();

System.out.println();

// Now, read them back.
try {
   dataIn = new DataInputStream(new
                    FileInputStream("testdata"));
}
catch(IOException exc) {
   System.out.println("Cannot open file.");
   return;
}
```

```
      // Continuação da classe RWData

      try {
         i = dataIn.readInt();
         System.out.println("Reading " + i);
         d = dataIn.readDouble();
         System.out.println("Reading " + d);
         b = dataIn.readBoolean();
         System.out.println("Reading " + b);
         d = dataIn.readDouble();
         System.out.println("Reading " + d);

      }
      catch(IOException exc) {
         System.out.println("Read error.");
      }
      dataIn.close();
   }
}
```

Output:

```
Writing 10
Writing 1023.56
Writing true
Writing 90.28

Reading 10
Reading 1023.56
Reading true
Reading 90.28
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Streams de Caracteres

- No topo da hierarquia das *streams* de caracteres estão as classes abstractas `Reader` e `Writer`
- Consulte a **API** para ver métodos de `Reader` e `Writer`

- Como o `System.in` é uma *stream* de *bytes* temos de o "embrulhar" em algum tipo de `Reader`

- A melhor classe para ler o input da consola é a `BufferedReader` mas não pode ser criada directamente a partir do `System.in`
- Para isso é necessário usar o `InputStreamReader` que converte o *bytes* em caracteres

```
class ReadChars {
   public static void main(String args[]) throws IOException {

      char c;

      BufferedReader br = new BufferedReader(new
                          InputStreamReader(System.in));

      System.out.println("Enter characters, period to quit.");

      // read characters

      do {
         c = (char) br.read();
         System.out.println(c);
      } while(c != '.');

   }
}
```

```
class ReadLines {
   public static void main(String args[]) throws IOException {

      // create a BufferedReader using System.in
      BufferedReader br = new BufferedReader(new
                          InputStreamReader(System.in));
      String str;
      System.out.println("Enter lines of text.");
      System.out.println("Enter 'stop' to quit.");
      do {
         str = br.readLine();
         System.out.println(str);
      } while(!str.equals("stop"));
   }
}
```

```
public class PrintWriterDemo {
   public static void main(String args[]) {
      PrintWriter pw = new PrintWriter(System.out, true);
      int i = 10;
      double d = 123.65;
      pw.println("Using a PrintWriter.");
      pw.println(i);
      pw.println(d);
      pw.println(i + " + " + d + " is " + (i+d));
   }
}
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Streams de caracteres com ficheiros I/O

- Existem várias formas para ler e escrever *streams* de caracteres em *Java*
- No entanto para simplificar a escolha sugerem-se as `FileReader` e `FileWriter`
- Consulte a **API** para ver métodos destas classes

```
class KtoD {
   public static void main(String args[]) throws IOException {

      String str;
      FileWriter fw;

      BufferedReader br = new BufferedReader(
                          new InputStreamReader(System.in));

      try {

         fw = new FileWriter("test.txt");

      }
      catch(IOException exc) {
         System.out.println("Cannot open file.");
         return ;
      }
```

```
    // Continuação da classe KtoD



    System.out.println("Enter text ('stop' to quit).");

    do {
       System.out.print(": ");
       str = br.readLine();

       if(str.compareTo("stop") == 0) break;

       str = str + "\r\n"; // add newline

       fw.write(str);

    } while(str.compareTo("stop") != 0);

    fw.close();
  }
}
```

```
class DtoS {
   public static void main(String args[]) throws Exception {
      FileReader fr = new FileReader("test.txt");
      BufferedReader br = new BufferedReader(fr);

      String s;

      while((s = br.readLine()) != null) {
         System.out.println(s);
      }

      fr.close();
   }
}
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

Serialização

- O Java permite a gravação directa de objectos
- Para isso, o objecto deve declarar implementar a interface `java.io.Serializable`

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

```
// Write Object


ObjectOutputStream out = new ObjectOutputStream(
        new FileOutputStream(test.txt));
Student james = new Student();
Employee john = new Employee();

out.writeObject(james);
out.writeObject(john);
```

```
// Read Object


ObjectInputStream in = new ObjectInputStream(
        new FileInputStream(test.txt));
Student james = (Student)in.readObject();
Employee john =
       (Employee)in.readObject();
```

Wrapper types para converter *strings* numéricas

Wrapper	Conversion Method
Double	static double parseDouble(String str) throws NumberFormatException
Float	static float parseFloat(String str) throws NumberFormatException
Long	static long parseLong(String str) throws NumberFormatException
Integer	static int parseInt(String str) throws NumberFormatException
Short	static short parseShort(String str) throws NumberFormatException
Byte	static byte parseByte(String str) throws NumberFormatException

```
class AvgNums {
   public static void main(String args[]) throws IOException {

      // create a BufferedReader using System.in
      BufferedReader br = new BufferedReader(new
                          InputStreamReader(System.in));

      String str;
      int n;
      double sum = 0.0;
      double avg, t;

      System.out.print("How many numbers will you enter: ");
      str = br.readLine();
      try {
         n = Integer.parseInt(str);
      }
      catch(NumberFormatException exc) {
         System.out.println("Invalid format");
         n = 0;
      }
```

```
    // Continuação da classe AvgNums


    System.out.println("Enter " + n + " values.");
    for(int i=0; i < n ; i++)  {
       System.out.print(": ");
       str = br.readLine();
       try {
          t = Double.parseDouble(str);
       } catch(NumberFormatException exc) {
          System.out.println("Invalid format");
          t = 0.0;
       }
       sum += t;
    }
    avg = sum / n;
    System.out.println("Average is " + avg);
  }
}
```

Output:

```
How many numbers will you enter: 5
Enter 5 values.
: 1.1
: 2.2
: 3.3
: 4.4
: 5.5
Average is 3.3
```

Palavras Reservadas Usadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Links Úteis

- http://docs.oracle.com/javase/tutorial/essential/io/charstreams.html