Java Basics
Part 4 - Streams

Manuel Oriol, April 27th, 2006
• Streams are useful to apply to different inputs and outputs a single treatment with different results
• Basically, receive and send bytes
• Streams are responsible for handling the outer part of the communication
Input Streams

- InputStream regroup all objects that can receive information
- Can build readers on top of them, to handle the inner part of the communication
- API
Output Stream

• regroup all objects that can send information
• can build filters around them
• API
Example: PrintStream

- System.out
- overloaded print/println/printf methods
- API

note: printf is a variable arguments method...
Variable Argument Methods

• Arguments are automatically boxed into an array (http://java.sun.com/developer/JDCTechTips/2005/tt1018.html):

```java
import java.util.*;
public class MyArgs {
    public static void main(String args[]) {
        method1("Hello", "World");
        method1(args);
        method2(12, 'a', "Hello", Math.PI, 1/3.0);
        method2(18, 94.0);
    }
    private static void method1(String... args) {
        System.out.println(Arrays.asList(args) + " // " + args.length);
    }
    private static void method2(int arg, Object... args) {
        System.out.println(Arrays.asList(args) + " / " + arg);
    }
}
```
Example: FileInputStream

- Reads from an file
- See API
Reader

- For reading character streams...
- BufferedReader
- API

String readline()
System.out

• is a PrintStream

• can be changed (e.g. output in a file, socket...)

• by default is set to
System.in

- By default reads on the terminal
- Can be changed
- easier to build a BufferedReader on it
Using Streams for Keyboard Interactions

PrintStream out = new PrintStream(new FileOutputStream("myfile.txt"));

out.println("My text");
out.close();

BufferedReader reader = new BufferedReader(new FileInputStream("myfile2.txt"));

// this time we append
out = new PrintStream(new FileOutputStream("myfile.txt"), true);
out.print("\t" + in.readLine());
public static void main(String[] args) {
    String s;
    standard = new BufferedReader(new InputStreamReader(System.in));
    // checks arguments number
    if (args.length!=1) System.exit(0);
    // open the file name
    try {out = new FileOutputStream(args[0]);}
    catch (FileNotFoundException e){System.exit(0);}
    // users have to leave by using Control-C
    while(true){
        try {
            // read and write
            s=standard.readLine();
            out.write(s.getBytes());
            out.write("
".getBytes());
        } catch (IOException e){
            System.out.println("I/O error");
            System.exit(0);
        }
    }
}
Basic Serialization

• ObjectOutputStream
• ObjectInputStream
Example OOS/OIS

```java
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("myfile.txt"));

out.writeObject("test");

out.close();

...

ObjectInputStream ois= new ObjectInputStream(new FileInputStream("myfile.txt"));

String s;

// this time we read the object

s=(String)ois.readObject();
```
Socket streams

- SocketInputStream
- SocketOutputStream
Example Socket Streams

Socket s;

...

PrintStream out = new PrintStream(s.getOutputStream());

out.print("EOF");

...

BufferedReader reader=new BufferedReader(s.getInputStream());

// this time we read a line

String s=reader.readLine();