Input / Output in Java

Input/Output in Java can be done using the keyboard and screen, using files, or some combination of these methods. Input typed at the keyboard and output displayed on the screen are often referred to as **console** input/output.

Interactive input/output using the keyboard and screen is the default way of doing input/output in jGRASP. It is also possible to use a dialog box to get input from the user and to display a message. Java refers to these as an input dialog and message dialog respectively.

An object is always needed to perform an input/output operation in Java.

Display Output [standard output]

- Displaying screen output in Java is done using the System.out object. You do not need to create the System.out object. It is always available for you use in a Java program. There are 3 methods that can be used to display output: print( ), printf( ), and println( ).
    - The **print( )** method is often used to display a prompt or a portion of a line. It does not move the cursor to the beginning of a new line automatically.
    - The **println( )** method displays output exactly the same as the print( ) method except that it will always move the cursor to the beginning of a new line as the last action it performs.
    - The **printf( )** method is used to output formatted text and numbers. Like the print( ) method, It does not move the cursor to the beginning of a new line automatically.

Keyboard Input [standard input]

- Accepting keyboard input in Java is done using a Scanner object. There is a System.in object in Java that can be used to get input in Java, but it is not very functional. Instead, Java programmers usually prefer to use a Scanner object that has been "mapped" to the System.in object. The System.in object (like the System.out object) already exists and is available for use. However, you must create a Scanner object.

  Once you have created a Scanner object and "mapped" it to the System.in object you can use the following methods to test for and get input. Input always comes in to a Java program as a string of characters (Java String type). There are methods that can be used to convert the String to a character, an integer number, a floating-point number, etc. Here are commonly used Scanner methods:

```
hasNext()          //is something available to read? Returns true or false.

nextInt()          //get an int
nextDouble()       //get a double

next()             //get a String ( delimited by whitespace )
nextLine()         //get the rest of the line as a String

Note:  There is no method to read a character! Read a string instead.
```

Examples of Interactive input/output in Java

Use a Scanner object "mapped" to the System.in object for input and the standard output object System.out for output.

Input:

1. Use the **Scanner** class to create an input object using the **standard input object** in the System class (**System.in**). The Scanner class is in the package java.util

   ```
   Scanner input = new Scanner( System.in );  //input object is informally called  "input"
   ```

2. Use **Scanner class** methods to get values the user enters using the keyboard.

```
int num1; double num2; String str;

num1 = input.nextInt();       //get an integer and assign it to num1
num2 = input.nextDouble();    //get a real number and assign it to num2
str  = input.nextLine();      //get a String and assign it to str
```

Output:

1. Use the **standard output object** in the System class (**System.out**).

2. Use **PrintStream class** methods (**printf(), print(), println()**) to display the output to the screen using the System.out object. The System.out object is an object of type **PrintStream**.

```
System.out.print("What is your name? ");
String name  = input.nextLine();
System.out.println("Hello there " + name + ", nice to meet you!");

double purchasePrice = 178.34;
System.out.printf("Your total is $%.2f\n", purchasePrice);
```

Input/Output Using a Dialog Box [import javax.swing.JOptionPane;]

- When you use a dialog box in Java, it pops up in the middle of the screen and waits for you to enter input or displays a message and waits for you to click "OK".

- Input/output using a dialog box is done in Java using methods of the JOptionPane class. You do not have to create a JOptionPane object in Java to work with the dialog box.

- Methods used: showInputDialog( ) [used to get input] and showMessageDialog( ) [used to display output].

- If you are using an input dialog box, the input comes in as a String. You must then use a method to convert the input to the data type you need it to be. The method you would use to get the input is showInputDialog( ). Once you have the input stored in a String variable, you can use a Java "wrapper" class method to convert the String.

  Here are the most frequently used "wrapper" class methods: [all classes are in package java.lang]

  ```
  Integer.parseInt( ), Long.ParseLong( ), Double.parseDouble( )
  ```

Input using a dialog box [JOptionPane.showInputDialog( )]

```
String name;  //no conversion necessary
name = JOptionPane.showInputDialog("What is your name?");

String numberString; //accept input as a string, then convert it to a number
int number;          //variable that will store an integer

//input and converstion from String to int
numberString = JOptionPane.showInputDialog( "Enter a number" );
number       = Integer.parseInt( numberString );
```

Output using a dialog box [JOptionPane.showMessageDialog( )]

```
JOptionPane.showMessageDialog(null, "Nice to meet you " + name + "!");
```

File Input/Output

Use a Scanner object for input and a PrintWriter object for output.

Note: <u>Once we have created our objects</u>, the input/output statements look virtually identical to the versions using the standard input/output objects.

Input:

1. Create an input object that is a Scanner class object "wrapped around" a File class object. Note this "wrapping" is similar in concept to wrapping a Scanner class object around the System.in object for input from the keyboard.

   ```
   Scanner inFile = new Scanner( new File("MyData.txt") );
   //replace "MyData.txt" with the actual name of your data file.
   ```

   The data file can be created using a simple editor such as Notepad. If the input file can not be found at runtime, an exception will occur and your program will stop execution.

2. Use Scanner class methods to read values from the input file.

   ```
   int num1, num2, num3;
   …
   num1 = inFile.nextInt();      //get an integer
   num2 = inFile.nextInt();      //get an integer
   num3 = inFile.nextDouble();   //get a real number
   ```

Output:

1. Create a PrintWriter object. The PrintWriter class is a subclass of PrintStream. The methods **`print(), printf(), println()`** will be available for use once the output object is created and will work exactly the same way as for the System.out object.

   ```
   PrintWriter outFile = new PrintWriter("MyReport.txt");
   ```

   The output file will automatically be created by your program. If a file exists with that name, it will be overwritten.

2. Use **PrintWriter class** methods (**`print(), printf(), println()`**) to write the output to the file using the PrintWriter object. Note: the textbook uses the Formatter class rather than the PrintWriter class.

   ```
   int sum = num1 + num2 + num3;
   outFile.printf("The sum of the numbers is %5d\n", sum);
   ```

Note for File Input/Output, you need to close the files when you are finished doing your input and/or output operations. The close( ) method is used to do this.

```
Close the input  file object:    inFile.close();
Close the output file object:    outFile.close();
```

To get access to the file stream and exception classes necessary to do input/output, you must include an import statement that imports classes in java.io.

Note, the names your choose for your objects is entirely up to you. An identifier is used in the declaration of an object. You must follow the same rules as for any identifier. Choose a name that is descriptive.

Here are some example input file variable identifiers: **inFile, inputFile, dataFile**
Here are some example ouput file variable identifiers: **outFile, outputFile, reportFile**

Sample Application to read input from a data file and write output to a report file.

Suggestion: Use a **txt** extension for your input and output files.

I have included a Java program called UpdateBalance that demonstrates reading from an input file and writing to an output file.

The input file will contain three "fields": an integer account number, a real account balance, a string account name.

AccountData.txt file contents: Note all the fields have been separated by whitespace (spaces and newlines).

```
123 1358.72 Fred Flintstone
458 2283.59 Barney Rubble
727 100.00 Pebbles
394 100.00 Bam Bam
```

The UpdateBalance Java program will read a record from the data file, add 10% to the account balance, then print a detail line to the report file (AccountReport.txt).

AccountReport.txt file contents: [generated by UpdateBalance program]

```
Bedrock Bank Statement of Accounts

Account   Owner               Beginning Balance    Ending Balance

123       Fred Flintstone      $ 1,358.72           $ 1,494.59
458       Barney Rubble        $ 2,283.59           $ 2,511.95
727       Pebbles              $   100.00           $   110.00
394       Bam Bam              $   100.00           $   110.00
```