Getting Started With Java – A First Program with Simple Output

The “Hello, world” program is a traditional starting point for the introduction to most programming languages. It also allows us to explore and explain some of the basic structure common to all Java programs.

```
class Greet
{
  // This program prints a simple message
  public static void main (String[] args)
  {
    System.out.println("Hello, world");
  }
}
```

Consider some of the components of this program in detail:

1. **`class`**

   All Java programs are contained in a *class*. The start of a class is indicated by the reserved word **`class`**. There are about fifty (50) reserved words in Java.

2. **`Greet`**

   This is the name to identify this particular class. Class names in Java are traditionally named using PascalCase, where each distinct word in the name (no spaces) starts with a capital letter.

3. **`{ }`**

   Brace brackets are used to indicate the beginning and end of any section of a Java program.

4. **`// This program....`**

   Two slashes indicate a comment in Java. Anything on a line after the // is ignored by the compiler. It is also possible to enclose longer sections of code in a comment using the older style of comments:

   ```
   /*  beginning of comments
   end of comments     */
   ```

5. **`public static void main (String [] args)`**

   Java programs consist of one or more methods (which are sometimes called *functions*, *procedures*, or *subroutines* in other programming languages). If the program is run directly by the Java interpreter, at least one method must be called main. The other terms in the method will be explained later.

6. **`System.out.println("Hello, world");`**

   This statement calls a system method called **`println`** to send a string of characters to the output device (typically an output window on the monitor).

7. Indentation

   Notice that specific parts of the program have been indented. Although not necessary for the program to run properly, it greatly improves the visual appearance and organization of the program.

Getting Started With Java – A First Program with Simple Output

Even with our simple program, we can introduce some variations on output that will prove useful with most of our future Java programs.

1. To print a single blank line, use the line

   ```
   System.out.println();
   ```

2. Some strings are quite long, and will not fit on one line of a program (or they will run off the screen). To split a string across multiple lines of code, write something like the following:

   ```
   System.out.println("This will print a verrrrrry long"
                      + " string on one line by joining"
                      + " these strings into one string");
   ```

3. Each **`println`** statement will advance the printing position to the next line. If we wish to stay on the same line, at the end of the last printed character, use the **`print`** statement instead.

4. Each string is denoted using double quotes ("), which begs the question, “How do you print double quotes?” You can force Java to interpret a double quote as a printable character, rather than a string delimiter, by using the backslash (\) character.

   ```
   System.out.println("He said, \"That's ridiculous.\"");
   ```

The following program illustrates some uses of print and println.

```
class PrintDemo
{
  public static void main (String[] args)
  {
    System.out.println("Never put off till tomorrow");
    System.out.println();
    System.out.print("what you can do");
    System.out.println(" the day");
    System.out.println();
    System.out.println("after" + " " + "tomorrow.");
  }
}
```

This will produce the following output:

```
Never put off till tomorrow

what you can do the day

after tomorrow
```

Using a slash (\) to precede another character and produce a different result is known as an *escape sequence*. Some examples of escape sequences are as follows:

Character (without slash)		Escape Sequence (with slash)	
"	a String delimiter	\"	a printable quotation mark
\	starts an escape sequence	\\	a printable slash
n	the letter 'n', printable	\n	prints a new line

Getting Started With Java – A First Program with Simple Output

Exercises

1. What does this program print? (Determine the output without programming it first!)

```
class Advice
{
      public static void main (String[] args)
      {
            System.out.print("If at first ");
            System.out.println("you don't succeed" + ",");
            System.out.print("failure may be ");
            System.out.println("your style");
      }
}
```

2. Rewrite the following program using the indentation style shown in our previous examples.

```
class Quote{public static void main (String[] args) {
System.out.print("The unexamined life");
System.out.println(" is not worth living.");}}
```

3. Write the Java statements that would print the following exactly as shown.
(Hint: The backslash character, \, needs special treatment just like the quotation character, ")

```
A (forward) slash is "/"
while
a backslash is "\"
```

4. The following program contains a number of errors. Rewrite the program with the errors corrected.

```
class BadForm
      public void main (string[] args);
      {
            System.Out.Println('What's wrong with this?')
      }
```

5. Write a Java program to print your name and address as they would appear on the envelope of a letter addressed to you at your home.