WIT COMP1000

Java Basics

Java Origins

- Java was developed by James Gosling at Sun Microsystems in the early 1990s
- It was derived largely from the C++ programming language with several enhancements
- Java is a *high level* programming language
  - » Provides many useful features that make it easier to write complex code
  - » As opposed to *low level* languages that provide direct access to computer subsystems like memory but require much more care from programmers

Java

- Like most programming languages, Java programs are written using a fixed *syntax*
  - » Syntax: a grammar that distinguished well formed statements from those that are not
- Special *keywords* and characters are used to tell the computer how to do what you want it to do
- Java forces the programmer to think as logically as the CPU
  - » Step-by-step, following a strict order

Hello World

```
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }
}
```

Tells Java that this file contains executable code

"main" is where your program actually starts executing; for now simply use it to mean "start here"

Print the line "Hello World" to the screen

Eclipse: New Project and Source File

- **File** menu -> **New** -> **Java Project**
  - » Enter a **Project name**, for example: Hello World
  - » Click **Finish**
- The project will show up in the left pane (**Package Explorer**)
- **Right click** on the project -> **New** -> **Class**
  - » Enter a **Name**, for example: HelloWorld
  - » Click the button labelled **public static void main(String[] args)**
  - » Click **Finish**

Adding Code

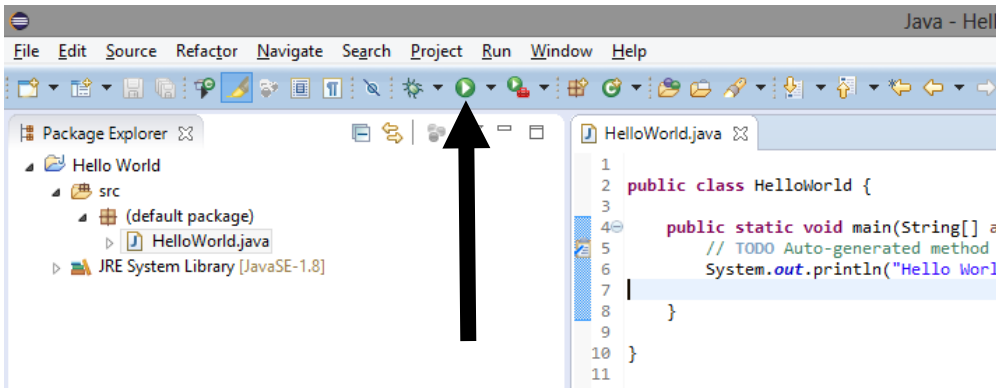
- You can add your program code in between the curly braces after the main line
- For now, add the following line:

```
System.out.println("Hello World!");
```

- Note that you can ignore the line that starts with //
  - Lines that start with // are *comments* that are ignored by the compiler

Eclipse: Running Your Program

- When you are ready to test your program, you can run it directly in Eclipse
- Click the Run button
  - Or **Run** menu -> **Run**
  - Or (Windows) **Ctrl-F11**
  - Or (Mac) **Shift-Cmd-F11**



- You'll be prompted to save your changes
- Then you'll see your program output in the **Console** pane at the bottom

Errors

- Save your work often! (Win **Ctrl-s** or Mac **Cmd-s**)
- If there are any errors in your syntax you will get a message indicating it
  - » Normally click **Cancel** as there is no need to run the program if it was incorrect
- Error icons show up to the left of the line numbers where the errors are detected
  - » Hover over the icon to get the error message
  - » Also, go to the bottom pane then select the **Problems** tab to see a list of all errors detected

`System.out.println()`

- When you want to output text to the screen for the user to see, you use: `System.out.println()`
- This is a built-in Java *method* that makes it easy to print a line of text
- Syntax: `System.out.println("Text")`
- It automatically includes a special character at the end of your text so that the next time you use the method any text will begin on the following line

Notes about `println()`

- You can insert special characters into the output by putting a backslash in front of some normal characters
  - » New line: `\n`
  - » Horizontal tab: `\t`
  - » Backslash: `\\`
  - » Double quote: `\"`
- You can combine multiple pieces of text with the plus sign
  - » We'll see why that's important soon

Examples

- `System.out.println("Hello World!");`
- `System.out.println("Howdy");`
- `System.out.println("This is a tab:\t.");`
- `System.out.println("Hello" + " World!");`
- `System.out.println("Avengers" + " assemble" + "!");`

Variables

- The first fundamental concept in programming is that of a *variable*
  - » Variables are related to mathematical variables in algebra, but work quite differently
- Variables are stored in memory while the program is running
  - » Every variable has a *value* that is stored in a particular memory location
- Each variable has a *name* that the programmer uses to access and modify that variable's value

Variables

- Each variable holds exactly one value
- Over time, as a program executes, **the value of a variable can change**
  - This is fundamentally different from algebraic variables which are used to represent one unknown value
  - Instead, programming variables store values so that we can use those values throughout a program

Variable Names

- In Java, variable names:
  - Must start with either a letter (uppercase or lowercase) or an underscore
  - Must contain only letters, digits, and underscores
  - Are case sensitive
- Examples: `count`, `x`, `user_input2`, `hit_points`
- Invalid names: `42`, `5x`, `#yolo`, `file.cpp`, `a-b`

Variable Declarations

- Every variable must be *declared* before you can use it in your program
- To declare a variable, you must give it a specific *type*:
  - `int`: integer (whole number), positive or negative
  - `double`: numbers with a fractional component
  - `boolean`: boolean value (true or false)
  - `char`: a single character

Variable Declarations

- Syntax: `TYPE NAME;`
  - The type comes first, then then variable name, followed by a semicolon
  - The semicolon tells Java that this declaration is done, and we'll see that many Java statements need one
- Examples:
  - `int count;`
  - `int num_vals;`
  - `double average;`
  - `char first_initial;`

Variable Initialization

- Before you can use a variable, you MUST give it a value
- Otherwise, your program won't know how to use the variable
  - » In fact, it will have a random value based on what memory looked like before, which leads to unpredictable behavior
- Java will catch the error and notify you of it, but be careful of this in other languages

Variable Initialization

- You can *initialize* a variable when you declare it or you can do so afterwards
- Syntax after declaration: `NAME = VALUE;`
- Syntax during declaration: `TYPE NAME = VALUE;`
- Examples

```
» count = 0;
» ultimate_answer = 42;
» int num_vals = 10;
» double pi = 3.14159;
```

Example

```
public class HelloWorld {

    public static void main(String[] args) {
        int magic_points = 100;
        System.out.println("Number of magic points: " + magic_points);
    }

}
```

Printing Variables

- `System.out.println()` is used to output the current value of a variable
- Don't put the name of the variable in quotes
  - This is one of the most common mistakes made by new programmers
- Any values in quotes are printed out literally
- Any values not in quotes are assumed to be variable names

More Printing Notes

- You can mix literal text in quotes with variable names with the plus sign as we saw earlier
- If you don't want a new line automatically added to the end of your output message you can use `System.out.print()` instead
  - Otherwise it works the same as `System.out.println()`

Another Example

```
public class HelloWorld {

    public static void main(String[] args) {
        int magic_points = 100;
        int hit_points = 200;
        System.out.print("Number of magic points: ");
        System.out.println(magic_points);
        System.out.println("Number of hit points: " + hit_points);

        System.out.println("hit_points");
    }

}
```

Sequential Execution

- Java programs start executing with the first line after the `{` after the `main()` line
- Each line is executed in order, one after the other until you get the `}` after the `main()` line
- We'll soon see how to affect this linear execution order, but even then programs execute one line at a time
- You have to train yourself to think one statement at a time
  - » This is one of the single most important skills you can have as a novice programmer

Program Input

- We also need a way to get user input into our programs while they are running
- Java doesn't (easily) allow reading directly from `System.in`
- Instead, you use a `Scanner` object that handles reading the input and ensures that the type of data you read matches what you want
  - The syntax is weird at first, but you'll get use to it

Input with a `Scanner`

- First, you have to declare and initialize the `Scanner` object
  - `Scanner input = new Scanner(System.in);`
- Then you call different methods on the `Scanner` object to read different types of values from the keyboard
  - Read an `int`: `variable = input.nextInt();`
  - Read a `double`: `variable = input.nextDouble();`

Example

```
import java.util.Scanner;

public class InputOutput {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int magic_points = 100;
        int hit_points;
        System.out.print("Enter the number of hit points: ");
        hit_points = input.nextInt();

        System.out.print("Number of magic points: ");
        System.out.println(magic_points);
        System.out.println("Number of hit points: " + hit_points);
    }
}
```

This line tells Java that you are going to use the Scanner

Notes about `Scanner`

- All input is automatically separated by whitespaces (spaces, new lines, tabs)
  - In other words, when you ask for a value from the `Scanner` it won't continue executing your program until a non-whitespace value is entered
  - Multiple input values can be separated on the same line by whitespaces
- In order to use the Scanner, you must include an extra line of code at the top of your source file: `import java.util.Scanner;`

Comments

- In Java source code, you can (and will!) include *comments* that explain in plain English what is happening in the code
- There are two types of comments in Java
- Single line comments start with `//`
  - » Everything after the `//` until the end of the line is ignored by the compiler
- Multi-line comments start with `/*` and end with `*/`
  - » Everything between the `/*` and the `*/` is ignored by the compiler

Commented Example

```
import java.util.Scanner;

public class HelloWorld {
     public static void main(String[] args) {
          // set up the Scanner to work with user input
          Scanner input = new Scanner(System.in);

          // initialize the number of magic points to 100
          int magic_points = 100;

          /* initialize the number of hits points based on
           * whatever the user types in */
          int hit_points;
          System.out.print("Enter the number of hit points: ");
          hit_points = input.nextInt();

          // print out results
          System.out.print("Number of magic points: ");
          System.out.println(magic_points);
          System.out.println("Number of hit points: " + hit_points);
     }
}
```

Wrap Up

- Java programs are executed one statement at a time, starting after `main()` and going down from there
- `System.out.println()` and `System.out.print()` are used to print values to the screen
  - » Use double quotes to print literal words
  - » Don't use quotes to print variables
- A `Scanner` object is used to read values from the user
- You should always include comments to explain your code for others who might read it