Week 1 Java Session

Introduction

Teacher: Troy Vaughn

Email : troy.vaughn@tafeqld.edu.au

Unit Details: Refer to *Java1_Assessment outline.docx*

▼ Logic Activity (No Code Exercise)

For this activity we will be starting by only using 4 commands to show how a lot of programming is just about using common commands in different orders to get different results.

The 4 commands we will use are:

Print Message: <Add Message or Value Here>

Read Input

Perform Math: <Add Calculation>

Store Value: <Give Value Name> = <Add value or source of value here.>

For the 4 commands, any section with the pointy brackets will need to be filled in with the values or messages for that command. This will allow us to add some variation to the 4 simple commands.

The commands will all work as follows:

Print Message - This will print whatever message you have provided to the user's screen.

For example, Print Message: "Hello Class" which is telling the program to print the message "Hello Class" to the screen.

NOTE: Normally you put sentences between double quote marks ""

Read Input - This command will read a users input into the system. This will normally need to be used as part of another command to be effective.

Perform Math - Provide a math operation and the system will work out the result. Normal math rules and order of operations (BOMDAS) will apply here.

For example, Perform Math: 5 + 4. This will calculate the result of the sum which will give you 9.

Store Value - Will take a provided value and store it under the provided name. You can then use this value anywhere by printing its name.

For example, Store Value<Number>: 10. This will store the number 10 under the label number.

We could then access the number somewhere else by writing <Number> as part of another command such as Perform Math: <Number> + 12. By doing this you are telling the program to put whatever the current value of <Number> is at the position where the tag is placed. This would give us the result of 10 plus twelve which would result in an answer of 22.

Chaining Commands

You can also combine some commands together to add multi step commands.

Examples:

Store Value<Number>: Read Input - Tells the system to read and input and store it in a value labelled as Number.

Print Message: “Your answer is: “ +<Number> - Tells the system top print the message and put the value of number in the position where the <Number> value is positioned.

NOTE: When combining values with text or to each other in sentences you need to put a + sign between them to indicate they are being combined.

Our First Non-Code Program

For your first program you need to use these 4 commands to write me a program the does the following.

Say ‘Hello Class’ to the class then print a second message that says ‘Welcome to your first class.’ This should take 2 commands to achieve.

▼ ANSWER:

Print Message: “Hello Class”

Print Message: “Welcome to your first class.”

Program With Some Maths

Write a simple program that calculates the sum of 10 + 6 and stores the result in a value called <Total>. Then print the message “Your total is:” + result of the sum.

NOTE: When combining values to text or to each other you need to put a + sign between them to indicate they are being combined.

▼ ANSWER:

```
Store Value<Total> = Perform Math:<10 + 6>
Print Message: “Your total is:” + <Total>
```

Program With Input

Write a simple program that asks the user for their name. Then read their name and store it in a value called <Name>. Then print a message that says “Hello” + their name.

▼ ANSWER:

```
Print Message: “What is your name?”
Store Value<Name>:  =  Read Input
Print Message: “Hello “ + <Name>
```

Some Maths Plus Inputs

For this program we are going to modify the program from the previous exercise. This time I want you to ask the user for a number then store the result in a value called <Number 1>. Then ask for a second number and store it in a value called <Number 2>. Then modify the sum to use these 2 values now instead of the 10 and 6 and then still print the results.

▼ ANSWER:

```
Print Message: “What is your first number?”
Store Value<Number 1>:  =  Read Input
Print Message: “What is your second number?”
Store Value<Number 2>:  =  Read Input
Store Value<Total> = Perform Math:<<Number 1> + <Number 2>>
Print Message: “Your total is: “<Total>
```

Remember these commands. When we get to our code later you will see how this relates to how you write code.

▼ Introduction To Java

Let’s move on now by looking at what the Java programming language is and how its basic syntax and structure works.

Refer to connect powerpoint : *Getting Started With Java*

▼ Let’s Write Some Java

▼ Our First Java Commands

Now that we have had a quick look at some Java syntax, let's have a go at using the same 4 types of commands as our previous examples. This time we will use the actual Java commands that perform these operations to see how they work.

You will notice as we do some of these that there are sometimes different variations for some of the commands. This will be because Java needs to treat numbers and text differently sometimes so it have a variation of the commands depending on which one it is dealing with.

To test these out we will go to a free online compiler to keep things easier.

Go to : https://www.online-java.com/

We will start initially with all the commands except reading input. This is because the command needed for reading inputs will need some minor setup done before we can use it.

The sections below will outline the formatting and syntax used to execute the same commands as before but in java.

Print Message

System.out.println("***your message***");

This will work the same as our print message command from earlier except you put the message inside the quotation marks of the command.

Store Value

int <name> =

or

String <name> =

These are the 2 variations of our store value command from earlier. The top one stores an integer (int) value using the provided name (no brackets needed.) The bottom one does the same but for a text (string) value.

For examples:

int size = 10;

or

String name = “Troy”;

NOTE: When assigning text to a variable you need to put it inside quote marks “ ” for Java to identify it as text.

Perform Math

This is the same as before. Just provide a math operation and the system will work out the result. Normal math rules and order of operations (BOMDAS) will apply here.

▼ Let’s Use Our New Commands

Now let’s have a go at using the print command to replicate the same message as before.

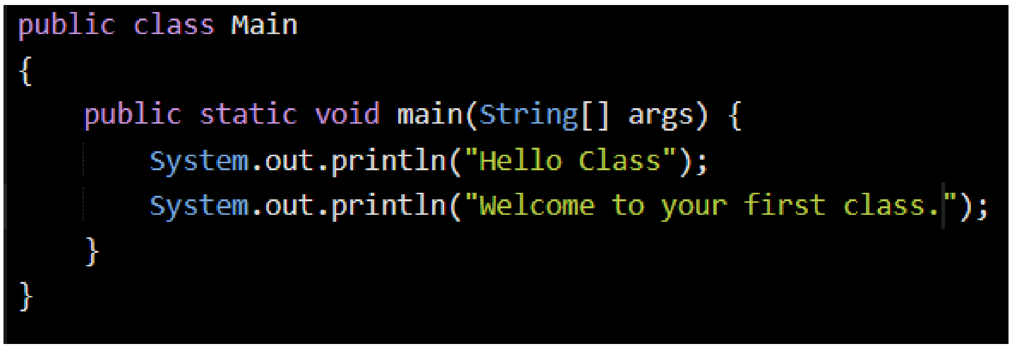
Our First Java Program

Get your program to print ‘Hello Class’ on the screen and then print a second message that says ‘Welcome to your first class.’ This should take 2 commands to achieve.

Remember for this exercise you will need to put your commands inside the main method and will need to follow each command with a semicolon to indicate the end of the statement.

If you think you have it correct press the RUN button at the bottom of the code window and it should print your message in the output window..

▼ ANSWER



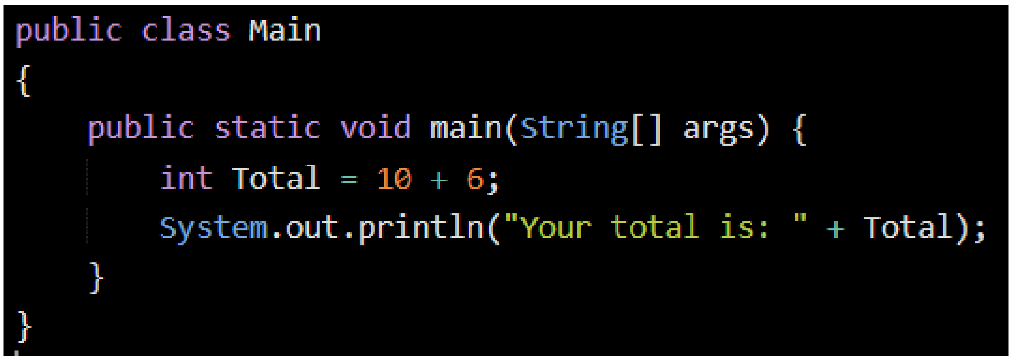
```java
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello Class");
        System.out.println("Welcome to your first class.");
    }
}
```

Program With Some Maths

Now replicate the program that calculates the sum of 10 + 6 and stores the result in a value called <Total>. Then print the message “Your total is:” + result of the sum.

NOTE: Don’t forget that when you are combining values to text or to each other you need to put a + sign between them to indicate they are being combined. You will need to do this to add the result to the message in your print command.

▼ ANSWER:



```java
public class Main
{
    public static void main(String[] args) {
        int Total = 10 + 6;
        System.out.println("Your total is: " + Total);
    }
}
```

▼ Adding Inputs To Our Commands

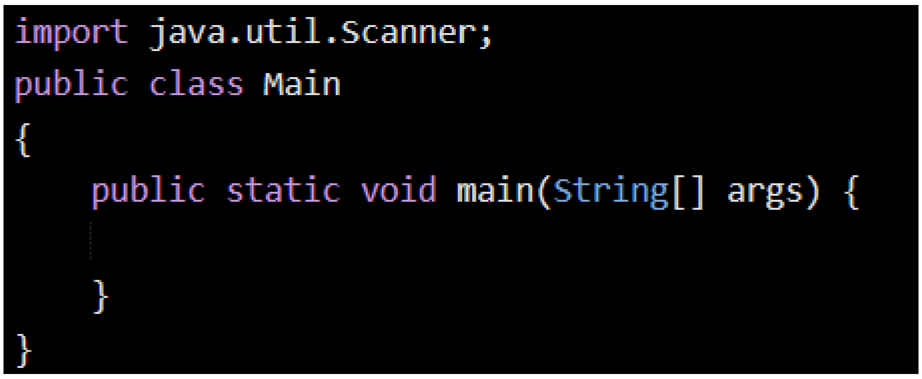
Now let’s look at how we handle reading input in Java.

The reason we had to leave the reading command until last is because the default Java libraries and commands do not have the desired reading functionality.

To allow us to do this we will need to add a component to our code called the Scanner. This component allows us to read inputs from the command window of our code editor and use them in our code.

To add the Scanner we need to just add 2 lines of code to our program.

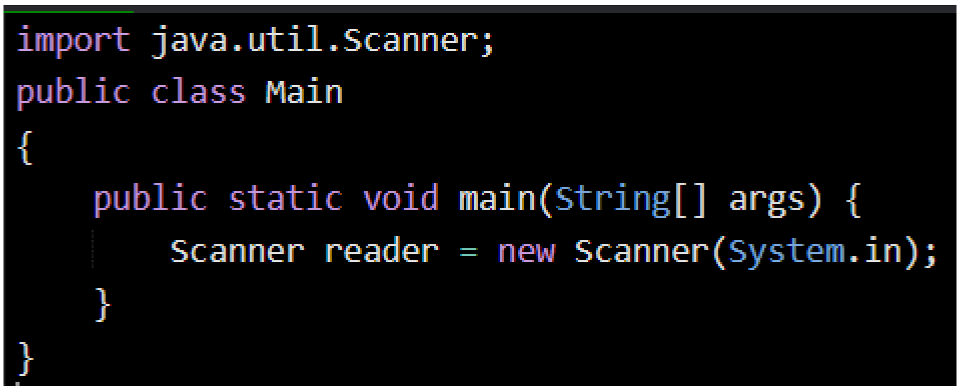
The first is an import statement at the top of the program (above the class) to tell the program it needs access to the library containing the scanner.



```java
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {

    }
}
```

The second line we need is to build a scanner and configure it to be used in our project. This line can be added to the opening of the main method.



```java
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
    }
}
```

This command declares it is creating the scanner and calls it reader (the same way we name an input), then builds a new scanner and tells it to use the System.in as it's input source. System.in is the command window that our outputs have been printing to.

Once these pieces are in place we can use the scanner to read inputs from the user.

Read Inputs

There will be 2 variations of the command needed to use the scanner, one for text and one for numbers.

reader.nextLine();

or

reader.nextInt();

The next line option reads text and the nextInt option reads whole numbers. Remember, when you use these you will normally need to combine it with another command such as Store value to store the input so it can be used later.

Examples:

String name = reader.nextLine();

int size = reader.nextInt();

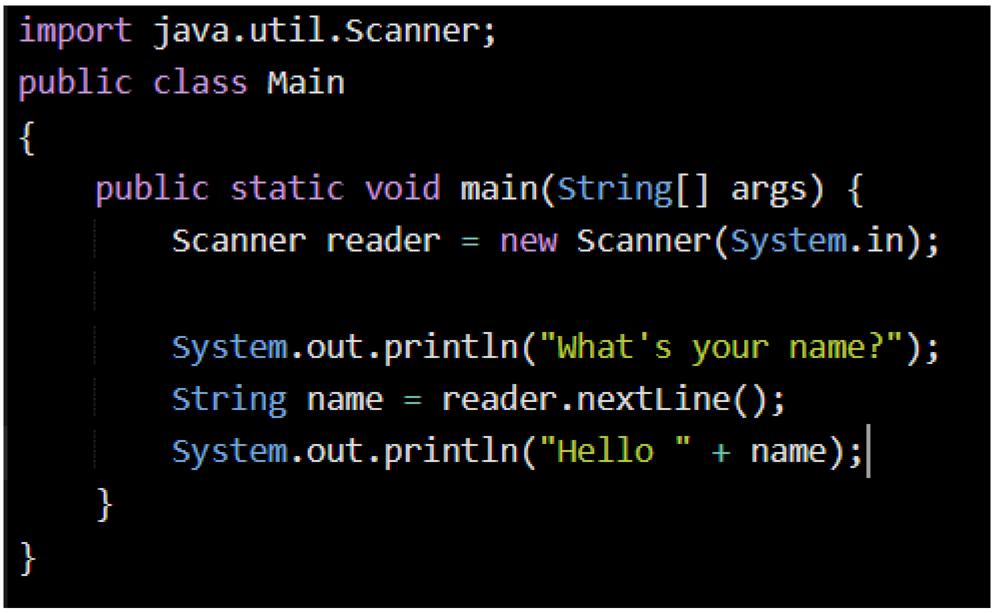
NOTE: The nextLine() version can actually read numbers but it will store them as their text equivalent and will not allow math operations on the values unless you convert them to their numeric equivalents. We will do this another time to show how it is done.

▼ Let's Use Some Inputs

Program With Input

Write a simple program that asks the user for their name. Then read their name and store it in a value called <Name>. Then print a message that says "Hello" + their name.

▼ ANSWER:



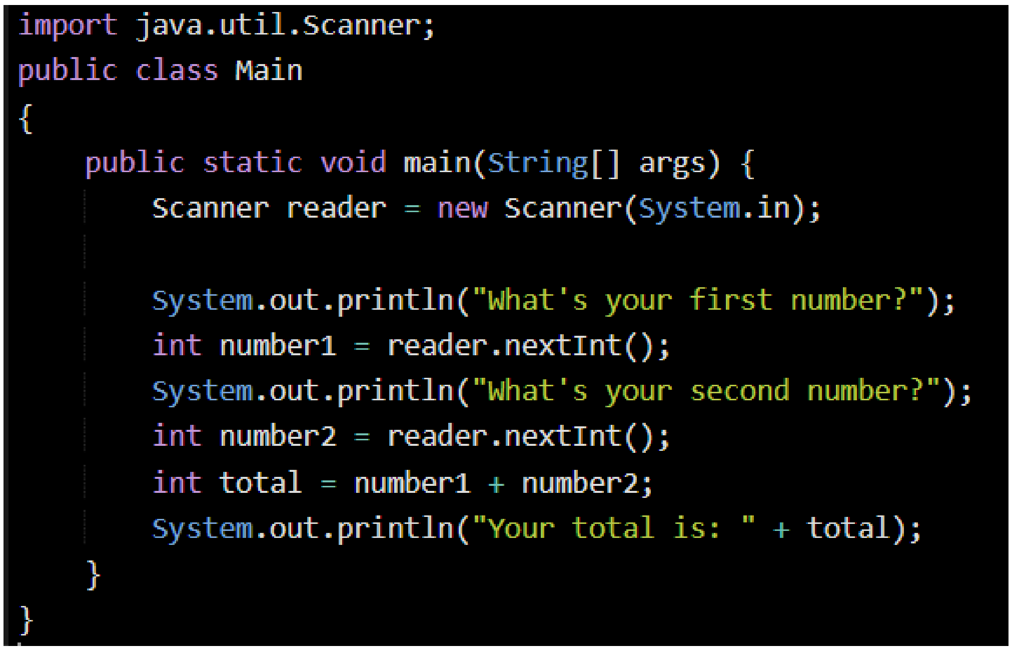
```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);

        System.out.println("What's your name?");
        String name = reader.nextLine();
        System.out.println("Hello " + name);
    }
}
```

Some Maths Plus Inputs

For this program we are going to modify the program from the previous exercise. This time I want you to ask the user for a number then store the result in a value called <Number 1>. Then ask for a second number and store it in a value called <Number 2>. Then modify the sum to use these 2 values now instead of the 10 and 6 and then still print the results.

▼ ANSWER:



```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);

        System.out.println("What's your first number?");
        int number1 = reader.nextInt();
        System.out.println("What's your second number?");
        int number2 = reader.nextInt();
        int total = number1 + number2;
        System.out.println("Your total is: " + total);
    }
}
```

▼ Practice Exercises

Now that we have looked at our starting commands, try to use them to build one of the following programs.

- Area & Perimeter Calculator (Square, Circle or Triangle) - Write a program that takes the provided sizes for one of the shapes and calculates the Area and Perimeter of the shape.
  - Add an extra step to take another value to calculate the volume of the shape.
- Simple Calculators (simple arithmetic) - Write some simple arithmetic programs for the math operations we didn't do in our examples.
- Volume converter - Write a program that converts a provided volume from wither litres(L) to millilitres(ml) or from millilitres(ml) to litres(L)
- Hours to seconds converter - Write a simple converter to take a provide input and convert it from minutes to seconds or from seconds to minutes.

    - Make a more advanced version that takes the input as hours and then tells you what that amount of time would be as minutes or as seconds.
- Custom Exercise - Think up another program you could make with these commands and see if you can build it.

SELF DIRECTED LEARNING

Use some of the commands we have learned in Week 1 to come up with some other small programs to solve simple problems.

Have a look at W3Schools Java tutorial from the Intro through to the Data Types Sections.

https://www.w3schools.com/java/default.asp