TINKER ACADEMY

Programming Using Java

Handout 3: Data Types Operators and Control Statements

Note your Student ID. You will need to use it throughout the Course.

Setup Instructions In Classroom

Connect to the Local Class Network

1. Select WiFi “TINKER ACADEMY”
2. This network has only LOCAL access and does NOT connect to the internet

Update the Course

1. Ensure you are connected to “TINKER ACADEMY”
2. Restart the VM. Login into the VM.
3. Open Firefox in the VM
4. Your Instructor would tell you what to type in the browser. (Typically it is 192.168.1.5)
5. You should see a page with a list of entries.
6. Click on CourseUpdate<Date>.zip. This will download CourseUpdate<Date>.zip onto your VM
7. Open Nautilus. Click on Downloads. You should see the file CourseUpdate<Date>.zip
8. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
9. Open the extracted folder
10. Double click Course Update. Select “Run” in the window.

Update the Course (Alternate Approach In Class Using USB)

1. Borrow a USB drive from the Instructor
2. If you are on VirtualBox
    a. Click on Devices in the Top level Menu
    b. Select Drag ‘n’ Drop
    c. Select Bidirectional
3. If you are on VirtualBox (Another Way)
    a. Shutdown Virtual Machine
    b. Click on VM in the VirtualBox Manager
    c. Click on the Settings
    d. Click General
    e. Click Advanced Tab

   f. Select “Bidirectional” under Drag ‘n’ Drop
   g. Click OK
   h. Start Virtual Machine

4. If you are on VMWare
   a. Open the virtual machine settings editor (VM > Settings),
   b. Click the Options tab
   c. Select Guest isolation.
   d. Deselect Disable drag and drop to and from this virtual machine
5. Open Nautilus, Click on Desktop
6. Drag the file CourseUpdate<Date>.zip from Windows or Mac onto Desktop in your Virtual Machine
7. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
8. Open the extracted folder
9. Double click Course Update. Select “Run” in the window.
10. Eject the USB Drive and hand it back to the Tinker Academy instructor

Setup Instructions At Home

Connect to your Home WiFi Network

Updating the Course (Using Wifi)

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Update”. Choose “Run”.
   If you see a window popup with the message “update course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.
4. Follow the instructions in this handout (last 2 pages) on the quiz and homework steps.

Submitting Quiz and Homework

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Submit”. Choose “Run”.
   If you see a window popup with the message “submit course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.

Virtual Machine Installation

Installing the Virtual Machine (VM)

1. Borrow the USB drive from your Tinker Academy instructor

2. Create the folder “tinkeracademy” (without the quotes) under Documents using Finder or Windows Explorer. Type it in *exactly* as indicated.
3. Copy the folder “installers” from the USB drive to under “tinkeracademy” using Finder or Windows Explorer
4. Eject the USB Drive and hand it back to the Tinker Academy instructor
5. Locate the VirtualBox installer under “tinkeracademy” using Finder or Windows Explorer

If your Laptop is	Double click on
Windows 7	VirtualBox-4.3.12-93733-Win.exe
Windows 8	VirtualBox-4.3.14-95030-Win.exe
Mac	VirtualBox-4.2.26-95022-OSX.dmg

6. Install the VirtualBox application
7. Congratulations, You completed a major milestone. Give yourself a pat on the back :)

Importing the Virtual Machine (VM)

1. Locate the Virtual Machine “tinkeracademy.ova” under “tinkeracademy”
2. Double click on “tinkeracademy.ova”. You should get the import screen in VirtualBox with an “Import” Button. Click on the “Import” button to Import the Virtual Machine.

Starting the Virtual Machine (VM)

1. Once the Import is complete and successful, you should see the VM “TinkerAcademy” in the side panel in VirtualBox.
2. If it says “Powered Off” click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
3. If it says “Running” click on the Show Button (Green Arrow) in the VirtualBox Toolbar. This should display the VM window.
4. Once the VM starts up you will be presented with a login screen. Type in “password” without the quotes. Type it in exactly as indicated and hit “Enter”.
5. Once the login is completed you should see a Desktop with a few icons. The Screen might go fuzzy for a few seconds before displaying the Desktop. *That is ok.*
6. Congratulations. You are now running Linux within your laptop.
7. Double click on the “Firefox” icon in the Sidebar. This should launch Firefox. Verify you have network access. Close “Firefox”

Launching the Virtual Machine in Full Screen

1. Use the VirtualBox menu View->Switch to Fullscreen to switch the VM to fullscreen mode
2. Use the same VirtualBox menu View->Switch to Fullscreen to switch the VM back out of fullscreen mode

Shutting Down the Virtual Machine

1. Click on the red close window button (to the top left on a Mac, top right in Windows).
2. You will prompted with a confirmation message asking if you want to “Power Off” the machine. Click the button to confirm power off.
3. In a few minutes the VM will shut down and you should see the VirtualBox side panel with the “Tinker academy” VM indicating “Powered Off”.

Restarting the Virtual Machine

1. Start VirtualBox
2. Click on the VM “TinkerAcademy” in the VirtualBox side panel.
3. Click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
4. Once the VM startup you will be presented with a login screen.

Right Click in VM on Mac

1. Open System Preferences, Trackpad
2. Enable “Secondary Click”, Toggle the small arrow to the right and select “Click with two fingers”.

Getting Ready to Program

Import StarterPack3.zip
We will be using StarterPack3.zip for this class.

Click on StarterPack3.zip under “Courses”. Right Click. Select “Extract Here”
Start Eclipse from Desktop. Right Click over Package explorer. Select “Import”. Browse and select the extracted folder “StarterPack”. Click “Finish” to complete the Import.

Structure of StaterPack3.zip
StarterPack3 is an Eclipse Java Project. The Project has

1. A Java Source File called “StarterPack3.java”
2. A Java Source File called “MyJavaClass1.java”
3. Project references to the Java Runtime Environment 1.6 System Library.

Run the Program
Right Click over project name “StarterPack3”. Select “Run As…”. Select “Java Application”.
The text “Starter Pack 3” should be displayed in the Console window (bottom)

If the “Console” window is not visible click on the “Window” Toolbar (top), select “Show View”, select “Console”. If all else fails, click on the “Window” Toolbar (top), select “Reset Perspective…” click “OK”. This will put Eclipse back into the default Factory state. You can now use “Show View” to view the “Console”.

Variables

What is a **variable**?
A variable is a very important concept in programming.

It is a location in the Virtual Machine’s memory used to store values.
Every variable has a name, a datatype (covered in next section) and an initial value.

A variable has a name and an initial value. The name follows the same rules and convention as that for naming classes and methods.

- Starts with a lowercase letter (a-z) or an underscore (_) *
- Cannot start with a number
- Uses camel case notation such as “myFirstVariableInJava”. Note that each word except the first has its first letter capitalized.

- Capitalization matters! “myFirstVariableInJava” is not the same as “MyFirstVariableInJava”

* unless its an public static constant (which we will cover later)

A variable occupies space in the virtual machine’s memory.

Declaring Variables

Variables are created in Java using a **variable declaration.**

What is a **variable declaration**?
A variable declaration is a special term used in programming languages (Java, C, C++).

A variable declaration tells the Java Compiler that it needs to allocate space in the Virtual Machine’s memory for that variable.

The declaration specifies

- name of the variable
- datatype of the variable
- initial value of the variable

Here is an example of a variable **declaration**

```
int i = 0;
```

The name of the variable is **i**.
The initial value of the variable is **1**.
The **datatype** of the variable is **int**. We will cover **datatypes** and **int** in the next section.

The code below declares 2 variables

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open MyJavaClass1.java under “src”.
4. Add the code under //TODO (1)

```
int i = 0;
```

5. Add the code under //TODO (2)

```
int rem = 0;
```

6. Save the File
7. Open Problems View. You should see no Errors.

Assigning Values To Variables

The value of a variable can be changed using an **assignment statement**.

What is an **assignment statement**?
An assignment statement is an important concept in programming.

An assignment statement sets or resets the value of the variable.

The code below uses an assignment statement to **assign** a new value to the variable “result”.

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open MyJavaClass1.java under “src”.
4. Add the code under //TODO (3)

```
rem = num % 2 ;
```

In the code above num % 2 is an example of an **expression.**
The value is the **remainder** after dividing the value of variable num by 2.
If num has the value 2, then the variable **rem** will have the value 0.
The symbol % is the mod operator.
We will cover the % (mod) operator and Expressions later in this Handout

5. Save the File
6. Open Problems View. You should see no Errors.

What is an **expression**?
In computer science, an expression is something that will return a value.

The expression

num % 2

returns the **remainder** of dividing the value of variable **num** by 2

The symbol % is called the mod operator

What is an **operator**?
In computer science, an operator is a special symbol that will perform operations on operands.

The mod (%) operator calculates the integer remainder after dividing the first number by the second num

num % 2
% is the operator
num is the first operand
2 is the second operand

We will cover expressions and operators in more detail later.

Data Types

What is a **datatype**?
A datatype is a very important concept in programming.

Every variable has an associated datatype which identifies

The possible values allowed for that variable
The possible operations that are allowed using the variable
The maximum amount of space that the variable will require in virtual memory

Why does Java need datatypes?

The Java Compiler takes your Java Source File(s) and converts into bytecode for the JVM.

HelloWorld.java → 11001010100101010

The Java Compiler tries very hard to help you create the most efficient program for you.

By program, we mean the Java byte code.

By efficient, we mean it will run the fastest possible while consuming the least amount of the CPU time or the memory usage.

The Java Compiler therefore requires that you provide enough information in your program so that the Java Compiler

1. Creates the smallest and fastest set of byte code instructions for the JVM
2. Catches as many logic errors in your program as possible (before it is actually run)
3. Use the space available in the virtual machine's memory as efficiently as possible

You provide that information by specifying the datatype.
The datatype identifies

The possible values that the variable can contain.
The possible operations that are allowed using that variable.
The maximum amount of space that the variable will need to use in the virtual memory.

For example, the variable **i** below is specified to be of **datatype int.**

```
int i = 1;
```

The Java Compiler now knows that

1. The possible values that **result** can hold are integers (both positive and negative). The maximum value is 2,147,483,647 and the minimum value is -2,147,483,648.
2. The variable will always occupy 32 bits (or 4 bytes since 1 byte is 8 bits) of the computers memory

C and C++ work similarly.

Such languages are called statically typed languages, since the type of the variable must be declared before its first used.

In general, variables are **declared** using the form

```
type var-name;
```

where **type** is the datatype of the variable and **var-name** is its name.

Defining Data Types

Java lets you define new datatypes.

You may not realize it, but you have already defined a new datatype.

Every class you create is a new datatype.

In Handout 2, the class MyJavaClass1 is a datatype.

You can now use it to declare a variable myJavaClass1 using the code below.

The code declares a variable myJavaClass1 of type MyJavaClass1.

```
MyJavaClass1 myJavaClass1;
```

The code creates a Java object of datatype MyJavaClass1.

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open StarterPack3.java under “src”.
4. Add the code under //TODO (4)

```
MyJavaClass1 myJavaClass1;
```

5. Save the File
6. Open Problems View. You should see no Errors.

Built-In Data Types

Java provides a set of datatypes. These datatypes are always available for your program.

Java contains three categories of built-in datatypes

- Primitive DataTypes
- Runtime DataTypes
- Special DataTypes

The built-in datatypes are available to a program as part of the Java Runtime Environment (JRE).

Java Runtime Environment (JRE)

What is a **runtime environment**?
Every programming language has a kind of runtime environment.

Every program needs to ultimately interact with the Operating System in order to perform its instructions successfully.

The runtime environment is the gateway through which the program interacts with the Operating System.

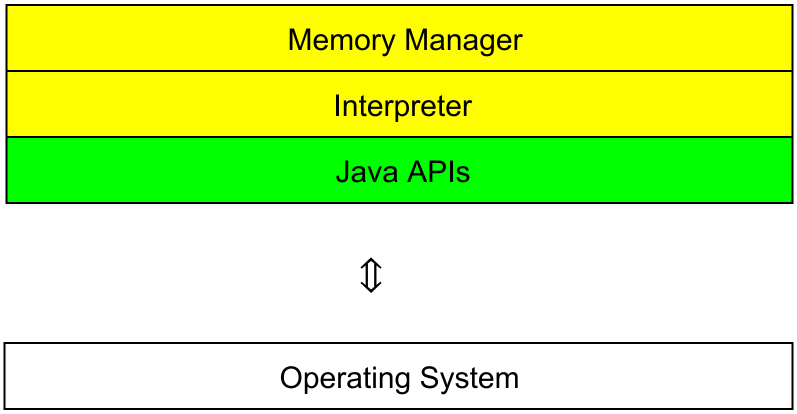
As we learnt from the previous class,

When you ran the program using “Run As...Java Application”, Eclipse

1. Compiled the Java source file and created a new Java Class File called “HelloWorld.class” using a special program called the “Java Compiler”. This process is called compilation. The Java Class File is called the compiled file. It contains the byte code that the Java Virtual Machine needs to run the Program.
2. Started the Java Virtual Machine (JVM) and provided the JVM with the byte code in the Java Class File. The JVM ran the program printing the text “Starter Pack 3” to the console.

The Java Virtual Machine (JVM) runs in a special environment called the Java Runtime Environment (JRE).

Visual Model of the Java Runtime Environment



Memory Manager

Interpreter

Java APIs

⇳

Operating System

Memory Manager	Manages JVM Memory for classes, methods, variables, literals
Interpreter	Interprets Byte Code (Runs the program)

Java API	Provides Built-in data types, predefined Classes for program

🟨 = Java Virtual Machine

🟨 + 🟩 = Java Runtime Environment (JRE)

Primitive Data Types

Primitive Datatypes are datatypes that are **used to construct other datatypes**.

Java has 8 primitive datatypes

DataType	Description	Possible Values
boolean	Represents true/false values	2 values, **true** or **false**
char	A single Character such as ‘c’	All possible characters across all languages
byte	8 bit integer	-128 to 127
short	16 bit integer	-32,768 to 32,767
int	32 bit integer	-2^{31} to $2^{31}-1$
long	64 bit integer	-2^{63} to $2^{63}-1$
float	32 bit decimal	Approximately -3.4028e+38 to +3.4028e+38
double	64 bit decimal	Approximately -1.7977e+308 to +1.7977e+308

Why are there so many ways to represent integer?

Byte, short, int, long all represent integers. However the range of values vary between the da types.

Smaller Datatypes require less space in memory. Use smaller datatypes such as **byte or short** to declare variables when you are sure the possible value for that variable will not exceed the range of values.

Use larger datatypes such as int or long when the variable can have values in the range of values for the data type

```
byte myByteVariable = 1;
```

You can print the value of all the integer variables using System.out.println

```
System.out.println(“The value of myByteVariable is “ + myByteVariable);
```

Why is there a special datatype for decimal numbers?

Decimal numbers are represented using an approximation called a floating point representation. The floating point representation allows Java to represent huge numbers suc as the distance between galaxies and small numbers such as the distance between subatomic particles.

In general floating point operations such as multiplying 2 floating point numbers lead to precision errors.

Whenever possible, use integer datatypes instead of decimal datatypes to represent numbers

Always use double instead of float to declare variables to hold decimal numbers.

```
double myDoubleVariable = 1/3;
```

You can print the value of all the double variables using System.out.println

```
System.out.println(“The value of myDoubleVariable is “ + myDoubleVariable);
```

What is a boolean?

The boolean type represents true/false values. boolean datatypes can have only 2 possible values

true
false

Both true and false are reserved words in Java and you cannot used them for anything else.

A boolean variable can be assigned a value using the reserved words **true** or **false**

```
boolean myBooleanVariable = true;
```

You can print the value of the boolean variable using the System.out.println

```
System.out.println(“The value of myBooleanVariable is “ + myBooleanVariable);
```

What is a char?

A char represents a character in ANY language.

Java characters are based on Unicode, which is an international standard that standardizes a the characters in use around the world.

A char variable can be assigned a value by enclosing the character in single quotes.

```
char myCharVariable = ‘y’;
```

You can print the value of the char variable using the System.out.println

```
System.out.println(“The value of myCharVariable is “ + myCharVariable);
```

Runtime DataTypes

Runtime datatypes are classes that are part of the Java Runtime Environment (JRE) and are always available for your program.

These classes are fundamental to the Java language.

The important Runtime DataTypes are listed below.

DataType	Description	Possible Values

String	Represents a Java String	any valid string of characters
Object	Represents a Java Object	any Java object
ArrayList	Represents a list of Java Objects	can contain any Java object
HashMap	Represents an association between Java Objects	can contain any Java object
HashSet	Represents a **unique** collection of Java Objects	can contain any Java object

The following code declares the variable “myStringVariable1” of datatype String with the initial value “this is a String”.

```
String myStringVariable1 = “this is a String”;
```

The following code declares the variable “myObjectVariable1” of datatype Object with the initial value as a new Java Object object. The variable’s datatype is Object.

```
Object myObjectVariable1 = new Object();
```

The following code declares the variable “myArrayListVariable1” of datatype ArrayList with the initial value as a new Java ArrayList object. The variable’s datatype is ArrayList.

```
ArrayList myArrayListVariable1 = new ArrayList();
```

The following code declares the variable “myHashMapVariable1” of datatype HashMap with the initial value as a new Java HashMap object. The variable’s datatype is HashMap.

```
HashMap myArrayListVariable1 = new HashMap();
```

The following code declares the variable “myHashSetVariable1” of datatype HashSet with the initial value as a new Java HashSet object. The variable’s datatype is HashSet.

```
HashSet myHashSetVariable1 = new HashSet();
```

We will be covering String, Object, ArrayList, HashMap and HashSet in the next class.

Special DataTypes

Java uses arrays to represent a list of values of the same type.

What is an array?
An array is a list of values of the same type.

An array is an example of a data structure. A data structure is a special arrangement of data that is useful in developing efficient computer programs.

In the code below the variable “stringArray1” is an “array of **String** values”

```
String[] stringArray1;
```

In the code below the variable “intArray1” is an “array of **int** values”

```
int[] intArray1;
```

DataType	Represents an array of ...
boolean[]	boolean values
byte[]	byte values
char[]	char values
short[]	short values
int[]	int values
long[]	long values
float[]	float values
double[]	double values

Java uses the keyword **void** to indicate that a method does not return a value.

In the method definition below, the method “main” does not return a value. This is indicated using the keyword void.

```
public static void main(String[] args) {
        System.out.println("Hello World");
}
```

Literals

Literals are human readable values.

You specify the value. The Java compiler is smart enough to figure out the datatype.

Literal	Represents the value...	DataType
true	true	**boolean**
false	false	**boolean**
100	100	**int**
12.8	**double** 12.8	**double**
‘a’ (uses single quotes)	a	**char**

Special Literals

Java defines special literals that are very useful

Double quotes “ and ” represents a String literal.
Square Brackets “[“ and “]” represent arrays

Special Literal	Represents the value...	DataType
“abc” (uses double quotes)	**string** of characters	**String**
{1, 2, 3, 4} (uses parenthesis)	an array of int values	Array of int values
{‘h’,’e’,’l’,’l’,’o’} (uses single quotes)	an array of char values	Array of char values
{“a”, “String”, “array”}	an array of String values	Array of String values

(uses double quotes)		

The special literal null is used to indicate that the variable has no value.

If you declare a Java object without specifying a initial value, then it has the special value null.

In the code below, there is no initial value for myObjectVariable2, so it is given the special value null.

```
Object myObjectVariable2;
```

In the code below, myObjectVariable2 has been set to null.

```
Object myObjectVariable2 = null;
```

This is very useful as you will see in later classes.

Expressions

What is an **expression**?
In computer science, an expression is something that will return a value.

Every expression has a datatype.
The datatype is the datatype of the value returned by the expression.

Expressions are identified based on the datatype

The expressions below are intuitive based on your math knowledge.

i is a variable of datatype **int**

num is a variable of datatype **int**

result is a variable of datatype **int**

myJavaClass1 is a variable of datatype **MyJavaClass1**

Expression	This is a ...	Which returns ..

<table>
<tr><td>i == 0</td><td><b>boolean</b> expression</td><td>true if the value of i is 0 , else false</td></tr>
<tr><td>i != 0</td><td><b>boolean</b> expression</td><td>true if the value of i is not equal to 0, else false</td></tr>
<tr><td>i &lt; num</td><td><b>boolean</b> expression</td><td>true if the value of i is less than the value of num, else false</td></tr>
<tr><td>i &lt;= num</td><td><b>boolean</b> expression</td><td>true if the value of i is less than <b>or equal</b> the value of num, else</td></tr>
<tr><td>i &gt; num</td><td><b>boolean</b> expression</td><td>true if the value of i is greater than the value of num, else false</td></tr>
<tr><td>i &gt;= num</td><td><b>boolean</b> expression</td><td>true if the value of i is greater than <b>or equal</b> to the value of num, else false</td></tr>
<tr><td>++i</td><td><b>int</b> expression</td><td>the integer value of i + 1.
The value if i now changes to i + 1.
In a sense ++i is a expression and a statement
<table><tr><td>i</td><td>++i</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>2</td></tr></table></td></tr>
<tr><td>--i</td><td><b>int</b> expression</td><td>the integer value of i - 1
The value if i now changes to i - 1.
In a sense --i is a expression and a statement
<table><tr><td>i</td><td>--i</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>-1</td></tr></table></td></tr>
<tr><td>num % 2</td><td><b>int</b> expression</td><td>the integer remainder after integer dividing num by 2
<table><tr><td>num</td><td>num % 2</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>0</td></tr><tr><td>3</td><td>1</td></tr></table></td></tr>
<tr><td>result * i</td><td><b>int</b> expression</td><td>the integer after multiplying result by i
<table><tr><td>result</td><td>i</td><td>result * i</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>2</td><td>2</td></tr><tr><td>2</td><td>3</td><td>6</td></tr><tr><td>6</td><td>4</td><td>24</td></tr></table></td></tr>
<tr><td>result + i</td><td><b>int</b> expression</td><td>the integer after adding result with i
<table><tr><td>result</td><td>i</td><td>result + i</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>2</td><td>3</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>2</td><td>3</td><td>5</td></tr></table>You will use this in Homework3</td></tr>
<tr><td>result / 2</td><td><b>int</b> expression</td><td>the integer after dividing result by 2<table><tr><td>result</td><td>result / 2</td></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>1</td></tr><tr><td>4</td><td>2</td></tr></table></td></tr>
</table>

Special Expressions

The expressions below are not so intuitive but are extremely important

myJavaClass1 is a variable of datatype **MyJavaClass1**

args is a variable of datatype String[] (array of values of datatype String) with the values as shown below

Values in args

0	zeoth
1	first
2	two
3	third

i is a variable of datatype **int**
j is a variable of datatype **int**

<table>
<tr><th>Expression</th><th>This is a ...</th><th>Which returns ..</th></tr>
<tr><td>**(i + 1)**</td><td>**int** expression</td><td>the integer after adding 1 to i.
This is different from the expression ++i.
++i both increases i and has the value of (i+1)
In this expression i will not increase.</td></tr>
<tr><td>**new** MyJavaClass1()</td><td>**object** expression</td><td>an object of datatype MyJavaClass1</td></tr>
<tr><td>**args[i]**</td><td>**String expression**</td><td>the String corresponding to the value at location i. The location starts at 0.
Yes it's counter intuitive, **but location starts at 0 in Java, C, C++, JavaScript, Python** and a number of other languages
An exception to this is Lua.
<table><tr><td>i</td><td>args[i]</td></tr><tr><td>2</td><td>second</td></tr><tr><td>3</td><td>third</td></tr><tr><td>0</td><td>zeroth</td></tr><tr><td>4</td><td>ERROR!</td></tr></table>Since args has only 4 values and no fifth value, the JRE will display an error
We will cover ERRORS in the a future class.</td></tr>
</table>

i == 2 && j == 2	**boolean** expression	true if i is equal to 2 AND j is equal to 2 else false
i == 2 || j == 2	**boolean** expression	true if i is equal to 2 AND j is equal to 2 else false

Operators

+, -, %, /, *, <, <= etc used in the expressions above are called operators. Operators are special symbols that perform operations on operands.

In the table below
intVar1 is a variable of datatype int with value 2
intVar2 is a variable of datatype int with value 2

Operators using 2 Operands

Operator	Operand 1	Operator	Operand 2	Result
plus	4	+	2	6
minus	4	-	2	2
multiplication	4	*	2	8
division	4	/	2	2
mod	4	%	2	0
less than	4	<	2	false
less than or equal	4	<=	2	false
greater than	4	>	2	true
greater than or equal	4	>=	2	true
equals	4	==	2	false
not equals	4	!=	2	true
assignment	a	=	2	a has the value 2
plus assignment	a	+=	2	a has 2 added

minus assignment	a	-=	2	a has 2 subtracted
multiply assignment	a	*=	2	a is multiplied by 2
divide assignment	a	/=	2	a is divided by 2
mod assignment	a	%=	2	a has mod 2 applied
logical AND	intVar1==2	&&	inVar2==2	true
logical AND	intVar1==2	&&	intVar2==1	false
logical OR	intVar1==2	||	intVar2==1	true
logical OR	intVar1==1	||	intVar2==1	false

Remember,
intVar1 is a variable of datatype int with value 2
intVar2 is a variable of datatype int with value 2

Operators using 1 Operand

Operator	Operator	Operand	Result
prefix increment	++	intVar1	3 (also, intVar1 is now 3)
prefix decrement	--	intVar1	1 (also, intVar1 is now 1)

Statements

What is a statement?
A statement is a request to so something.

The following are statements in Java

1. Variable declarations
2. Assigning Values to variables
3. Increment/Decrement Operators
4. Creating new objects using the new operator
5. Invoking methods
6. Return Statement

7. Control Statements

Statements using 1, 2 and 3 were covered earlier in this Handout.

Creating new objects using the new operator

Its very easy to create to create a statement using the new operator after knowing the new expression

The code creates a Java object of datatype MyJavaClass1 using the new operator.

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open StarterPack3.java under “src”.
4. Add the code under //TODO (5)

```
myJavaClass1 = new MyJavaClass1();
```

5. Save the File
6. Open Problems View. You should see no Errors.

Invoking methods

The method isEven is defined in the class MyJavaClass1.

The code creates invokes the method isEven and uses the value as the initial value of the **boolean** variable **isEven**. The value is then printed out to the console.

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open StarterPack3.java under “src”.
4. Add the code under //TODO (6)

```
boolean isEven = myJavaClass1.isEven(2);
System.out.println(“Is 2 an even number?  “ + isEven);
int factorial = myJavaClass1.factorial(4);
System.out.println(“The factorial of 4 is  “ + factorial);
```

7. Save the File
8. Open Problems View. You should see no Errors.
9. Run the program
10. The Console window should display the following

```
Starter Pack 3
Is 2 an even number? false
The factorial of 4 is 1
```

Hmm, 2 is not an even number? and the factorial of 4 is 1?

That does not look right. We need to fix that.

What is a **factorial**?

A factorial is a number multiplied by itself and all numbers less than it upto and including 1.

Return statement

What is a return statement?
The return statement attempts to return control to the invoker of the method.

After the return is successful, the next statement after the method invocation can be invoked.

In Java (C and C++), the return statement can also return a value;

The order of execution of the statements is indicated below

Order	StarterPack3.java
1	boolean isEven = myJavaClass1.isEven(2);
3	System.out.println(“is 2 an even number? “ + isEven);

Line	MyClassJava1.java
	public boolean isEven(int num) {
2	return false;

	}

Methods that do not return a value (use the keyword void in the method signature), do not require a return statement *.

However a method that returns a value requires a return statement.

The Java Compiler will cross check to make sure a return is guaranteed from all control statements. If the check fails, the compiler will show an error.

* they may use a return keyword for early exit (as we will see in later classes)

Control Statements

What is a control statement?
A control statement is a request to the runtime environment to change the order in which statements are executed

In general, control statements are used to

- choose which statements to execute next
- repeat a list of statements

Control statements use execution blocks to collect statements that should be executed together

An execution block in Java is any code between an opening parenthesis and closing parenthesis

The code below is in a block

```
{
        int result = 1;
        result = result * 2;
}
```

if-statement

The if-else statement is used to control if a block of statements should be executed next

```
if (num % 2 == 0)
{
    // if control block
}
```

// if control block will be executed if num is an even number (remainder would be 0)

if-else statement

The if-else statement is used to control which of a block of statements should be executed next

```
if (num % 2 == 0)
{
    // first control block
}
else
{
    // second control block
}
```

// first control block will be executed if num is an even number (remainder would be 0)
// second control block will be executed if num is an odd number (remainder would be 1)

for statement

The for statement is used to repeat a block of statements a certain the number of times.

```
for (i = 0; i < num; ++i)
{
    // block to repeat
}
```

In the table below the value of num is 4

i	i < num ?	repeat ?	total # of repeats	++i
0	true	true	1	1
1	true	true	2	2
2	true	true	3	3

3	true	true	4	4
4	false	false		

The statements in // block to repeat will be executed 4 times.

num	total # of repeats
4	4
5	5
6	6
some number n	?

Adding Control Statements

We are going to use control statements to the isEven and factorial functions

1. Open Eclipse
2. Click project name “StarterPack3” in Package Explorer
3. Open StarterPack3.java under “src”.
4. Add the code under //TODO (7)

```
if (num % 2 == 0)
{
        return true;
}
```

The code above adds an if statement to check if num is even

11. Save the File
12. Open Problems View. You should see no Errors.
13. Run the program
14. The Console window should display the following

```
Starter Pack 3
Is 2 an even number? true
The factorial of 4 is 1
```

15. Add the code under //TODO (7)

```
for (i = 0; i < num; ++i)  {
            int value = (i + 1);
            result = result * value;
}
```

The code above adds a for loop to loop **num** times

16. Run the program
17. The Console window should display the following

```
Starter Pack 3
Is 2 an even number? true
The factorial of 4 is 24
```

That looks right!

That was a LOT we covered!

You made it this far! Awesome!

We will cover **Classes, Objects and Methods** in the next class class. For now you need to make sure you have a good conceptual understand of Data Types Operators and Control Statements.

Quiz 3: Data Types Operators and Control Statements

Make sure you read this Handout!

Open the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Quiz3.odt under “Courses” “TA-JAV-1” “quiz” “quiz3”

Complete the Quiz

1. Attempt each question. Type in the answers in the “Answer:” box.
2. Save the file using File->Save or Ctrl-S

Submit the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Homework” in this Handout.

Homework 3: Data Types Operators and Control Statements

Make sure you read this Handout!

Overview

In this Homework, you will create an instance method called “addNumbers” that will take 1 number of type int and return the sum of all the numbers from 0 to the number.

The method signature is as below

```
public int addNumbers(int num)
```

If num is...	the method should return...	because...
0	0	0
1	1	0 + 1 = 1
2	3	0 + 1 + 2 = 3
3	6	0 + 1 + 2 + 3 = 6
4	10	0 + 1 + 2 + 3 + 4 = 10

Your method will need to contain the following

- a **for loop** to loop through num times
- a variable **result** to hold the result
- a variable **i** to use in the for loop

Open the Homework

Make sure you are on the Home WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Extract Homework3.zip under “Courses” “TA-JAV-1” “homework” “homework3”

- Select Homework zip file

- Right Click, Select Extract Here

Import the Homework3 project in Eclipse

- Right Click over Project Navigator
- Select Import Project
- Browse to the Homework folder
- Click OK

Complete the Homework

You will need to refer to this handout (Handout3). Make sure you read it thoroughly.

Homework Part 1
Open the Java Source File “MyJavaClass1.java”.

Add a method called **addNumbers** after //TODO (1)

The method should use the following signature

```
public int addNumbers(int num)
```

Remember to use { and } to begin and end the method definition.

The method should return the sum of all the numbers from 0 to the number.

Homework Part 2
Invoke the method “addNumbers” in the class from the static main method in the class “HelloWorld” after //TODO (2)

You would need to do the following

- Use the new operator to create a new instance of MyJavaClass1 and assign the value to the variable **myJavaClass1**
- invoke the method addNumbers using the input 4 and assign the result to a variable **result**

Homework Part 3
Print the result using the following code after the code in Part 2.

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

If your program is correct, it should print the following to the console

Homework 3
10

Make sure you save your program.

Test your program. If your program does not run successfully you will not get any credit.

Submit the Homework

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Homework” in this Handout.