**What are the Steps to install java?**

* Navigate to http://www.oracle.com/technetwork/java/javase/downloads/index.html. You can download a simple Java Software Development Kit (JDK) installer for Windows, macOS, or Linux directly from Oracle.
* Click the “Download” button beneath “JDK.” This will open a new page containing several download options.
* Scroll to the latest version of Java SE Development Kit. You should always use the latest stable version of the toolkit. There may be more than one version listed, so look closely at the release number.
* Click “Accept License Agreement.” Before you can click the download link, you must accept the license agreement. The option is just beneath the JDK version number.
* Click the download link next to your operating system. Once you click the link, follow the prompts to select a save location on your computer and start the download.

Install the JDK on your computer > Double-click the JDK installer file> Allow the app to make changes to your computer> Click “Next” to continue> Click “Next” to accept the default installation settings> Click “Close” when the installation is complete> Press Windows+X to launch the Power User menu> Click “System.” A screen with your system specs will appear> Click the “Advanced System Settings” link> Navigate to the “Advanced” tab.> Click the “Environment Variables” button> Double-click the "Path" variable under “System Variables> Click “New” if you’re using Windows 10> Set the variable in earlier versions of Windows> Click “OK” until you’ve closed all open windows> Press Win+R to launch the Run box> Type cmd and press Enter> Type path and press Enter> Type java –version and press Enter.

**What are the steps to install eclipse?**

**Eclipse Downloading:**

* Click Eclipse 3.2
* A page, with the label eclipse downloads, will be displayed in your browser.
* In the first section, find the text Download Now: Eclipse SDK 3.2, Windows (120 MB)
* Click the icon to the left of this text, or the part of the text labeled Eclipse SDK 3.2
* If you are using a Mac, it should sense it and display Mac instead of Windows. If you are not using the operating system it recognizes, click the link labeled Other downloads for 3.2 instead.
* Click the icon to the left of the text Download from: for using the default mirror site (or click a site from the list displayed in the box).
* Click the Save button to download, somewhere on your disk, the file named eclipse-SDK-3.2-win32.zip (for PCs). The exact form of this interaction will depend on the version of your operating system. Store this zip file somewhere permanent on your disk drive, so that you can unzip it (recreate Eclipse) later, if necessary
* Terminate (X) any windows remaining from this process.

**Eclipse Installation:**

* Unzip this file that you just downloaded.

On most Windows machines, you can Right-click the file > Move to the WinZip command > Click Extract to here > It creates a folder named eclipse. You can leave this folder here or move it elsewhere on your disk drive.

* Create a shortcut on your desktop to the eclipse.exe file.

On most Windows machines, you can

Right-press the file eclipse.exe > Drag it to the desktop > Release the right button > Click Create shortcut here > Now you are ready to perform a one-time only setup of Eclipse.

* Double-click the shortcut to Eclipse that you just created. In the Workspace Launcher window, in the box following Workspace:, should appear something like C:\Documents and Settings\username\workspace (where username is your login on the machine). If you want, you can type in (or browse) another location for the workspace file to be created, but I advise accepting the default.
* Check the box labeled Use this as the default and do not ask again.

Aside: you will be using one workspace during the semester, checking projects in and out of this workspace. If you ever want to re-enable the display of this window, once Eclipse starts, you can

Select Window | Preferences > Click the + in front of General or double-click General (after the +)>Click Startup and Shutdown > Check the box labeled Prompt for workspace on startup.

* Click OK
* If you want easy access to the workspace, create a shortcut to it (see the instructions above), putting the shortcut on your desktop.
* Download (right-click and Save As) Course Library Jar into the same folder that you are using as your workspace (or move this file in the workspace fodler after downloading it elsewhere).
* In Eclipse, select Windows | Preferences
* Click the + in front of Java or double-click Java (after the +)
* Click Installed JREs
* Under the Name column, double-click jre1.5.0\_08.
* Click the Add External Jars... button.
* In the Jar Selection window, navigate to your workspace folder, and double-click the file cs15-1xx.library.
* In the Edit JRE window, click OK
* In the Installed JREs window, click OK
* Terminate (X) the Welcome window
* You can always get it back by selecting Help | Welcome
* Terminate (X) the Eclipse window.

**Steps to create workspace**

You can create multiple workspaces in Eclipse. You have to just specify the path of the workspace during Eclipse startup. You can even switch workspaces via File→Switch workspace.

You can then import project to your workspace, copy paste project to your new workspace folder, then File→Import→Existing project in to workspace→select project.

**Create new project?**

Click "File" → "New" → "Java Project"> Give the project a name> Select the location for the project files.> Select Java Runtime Environment (JRE) you want to use> Select your project layout.> Click "Next" to open the "Java Settings" window> Use the Source tab to define your build path> Use the Libraries tab to add libraries to the project> Click "Finish" to start working on your new project.

**Create .java file/class**

After creating the project right click on it -> go to new and click on class ->select the desired package -> Enter name and check the public static void main(String[] args) if you want main method -> click on Finish.

**How to create packages and what is best way to give name?**

To create a package, you choose a name for the package and put a package statement with that name at the top of every source file that contains the types (classes, interfaces, enumerations, and annotation types) that you want to include in the package.

The package statement (for example, package graphics;) must be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

Package names are written in all lower case to avoid conflict with the names of classes or interfaces.

**What main method will do?**

A Java application (like DateApp in the code listed above) must contain a main() method whose signature looks like this **public static void main(String args[])**

The method signature for the main() method contains three modifiers:

* **public** indicates that the main() method can be called by any object. Missing Page covers the ins and outs of access modifiers supported by the Java language: public, private, protected, and the implicit, friendly.
* **static** indicates that the main() method is a class method. Class Members vs. Instance Members later in this lesson talks in more detail about class methods and variables.
* **void** indicates that the main() method has no return value.

The main method in the Java language is similar to the main function in C and C++. When the Java interpreter executes an application (by being invoked upon the application's controlling class), it starts by calling the class's main method. The main method then calls all the other methods required to run your application.If you try to invoke the Java interpreter on a class that does not have a main method, the interpreter refuses to compile your program and displays an error message.

**What is data type and what are different data types?**

A data type is a set of values and the allowable operations on those values.

There are two fundamental data types in Java: primitive types and reference types. Primitive types are:

* boolean
* char
* byte
* short
* int
* long
* float
* double

There is a specific keyword for each of these types in Java. Primitive types are not objects in Java. Primitive data types cannot be stored in Java collections which work only with objects. They can be placed into arrays instead.

**Byte** - made of 8bits and used to save space in Arrays, which is 4 times smaller than int.

Default value = 0, min = -128(-2^7), max = +127(+2^7-1)

**Short** - made of 16bit signed 2’s complement integer, which is 2 times smaller than int

Default value = 0, min = -32,768(-2^15), max = +32,767(+2^15-1)

**Int** - made of 32bit signed 2’s complement integer

Default value = 0, min =- 2,147,483,648(-2^31), max = 2,147,483,647(+2^31-1)

**Long** - made of 64bit signed 2’s complement integer

Default value = 0, min=-2^63,max=+2^63 -1

**Float** - It is a single-precision 32-bit IEEE 754 floating point

Default value = 0.0f

**Double** - It is a double-precision 64-bit IEEE 754 floating point

Default value = 0.0d

**Boolean** - False and true, Default value=false

**Char**- It a single 16-bit unicode character, min value = 0, max = 65,535

The reference types are:

* class types
* interface types
* array types

There is also a special null type which represents a non-existing value.

**What is variable?**

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type. Variables are typically used to store information which your Java program needs to do its job. This can be any kind of information ranging from texts, codes (e.g. country codes, currency codes etc.) to numbers, temporary results of multi-step calculations etc.

**Creating method with void.**

It is a set of commands that can be used over again. It doesn’t return any value.

This would be written

public void add(int a, int b)

{

// do stuff here

}

Since the return type is void, you will have to write what you want the method to do inside of the method such as by printing it out from the method

public void add(int a, int b)

{

System.out.println(a+b);

}

To use the above method, you would simply call that method and inserting the two integers that will go into the integer parameters:

add(7, 4);

**Creating variable, we can create variables inside method.**

If you declare your object inside a method, it will be visible only in this method. Basically, if you put brackets around it, it's only visible/accessible from within these brackets.

Static Methods can access class variables without using object of the class. It can access non-static methods and non-static variables by using objects. Static methods can be accessed directly in static and non-static methods.

class Example1{

static int i;

static String s;

public static void main(String args[]) //Its a Static Method

{

Example1 obj=new Example1();

//Non Static variables accessed using object obj

System.out.println("i:"+obj.i);

System.out.println("s:"+obj.s);

}

}

**Creating method with return data type, int/string/double/float/date**

Step 1- create a class with all methods

public class MethodsReturn {

public int add(int a, int b) {

return a + b;

}

public String names() {

return "harshini";

}

public double substract(int a, int b) {

return a - b;

}

public float multiple(int a, int b) {

return a \* b;

}

public boolean result() {

return false;

}

public String date(){

return null;

}

Step2: now create main class, then create object for the above class and invoke all the methods

import java.util.Date;

public class Methods {

public static void main(String[] args) {

// TODO Auto-generated method stub

int m = 1;

int n = 2;

int a = 10;

int b = 15;

Person(m, n);

MethodsReturn mr = new MethodsReturn();

Date date = new Date();

System.out.println(mr.add(a, b));

System.out.println(mr.substract(a, b));

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

System.out.println(mr.multiple(a, b));

System.out.println(date);

}

public static void Person(int m, int n) {

int sum = m + n;

System.out.println(sum);

}

}

**Method that will return hard coded value**

It returns hard coded value harshini

public String names() {

return "harshini";

}

Output: harshini

method that will return property value

Here the method add will return value based on the data members.

public int add(int a, int b) {

return a + b;

}

Output: 25

**Write code to add items to integer, string array**

// (1) create a java int array

int[] intArray = new int[3];

// (2) some time later ... assign elements to the array

intArray[0] = 1;

intArray[1] = 2;

intArray[2] = 3;

// (3) print our java int array

for (int i=0; i<intArray.length; i++)

{

System.out.println(intArray[i]);

}

// Create three-element String array.

String[] elements = { "cat", "dog", "mouse" };

**Creating object?**

An object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class −

* **Declaration** − A variable declaration with a variable name with an object type.
* **Instantiation** − The 'new' keyword is used to create the object.
* **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

MyObject object = new MyObject();

Ex: Employee emp1 = new Employee();

**Calling method with no return and parameter**

void skip(int n)

// Function skips n lines on output

{

int i; // a local variable to this function

// now loop n times

for (i = 0; i < n; i++)

cout << endl;

}

**Calling static method**

Static methods in Java can be called without creating an object of class. we write static keyword when defining main, because program execution begins from main and no object has been created yet.

class Languages {

public static void main(String[] args) {

display();

}

static void display() {

System.out.println("Java is a programming language.");

}

}

**Create classes under multiple packages**

**Write code to handle exceptions with try/catch/finally**

class TestExceptions {

static void myMethod(int testnum) throws Exception {

System.out.println ("start Method");

if (testnum == 12)

throw new Exception();

System.out.println("end Method");

return;

}

public static void main(String args[]) {

int testnum = 12;

try {

System.out.println("try - first statement");

myMethod(testnum);

System.out.println("try - last statement");

}

catch ( Exception ex) {

System.out.println("An Exception");

}

finally {

System. out. println( "finally") ;

}

System.out.println("Out of try/catch/finally - statement");

}

}

**What is final keyword, create final class, final method, final property**

The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

* variable
* method
* class

Final variable:

public static final String LOAN = "loan";

LOAN = new String("loan") //invalid compilation error

Final method:

class PersonalLoan{

public final String getName(){

return "personal loan";

}

}

class CheapPersonalLoan extends PersonalLoan{

@Override

public final String getName(){

return "cheap personal loan"; //compilation error: overridden method is final

}

}

Final class:

final class PersonalLoan{

}

class CheapPersonalLoan extends PersonalLoan{ //compilation error: cannot inherit from final class

}

**Write code for interface and create class to implement that interface**

interface MyInterface

{

public void method1();

public void method2();

}

class XYZ implements MyInterface

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[])

{

MyInterface obj = new XYZ();

obj. method1();

}

}

**Write code for creating abstract class**

An abstract class is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

abstract class A

{

abstract void callme();

}

class B extends A

{

void callme()

{

System.out.println("this is callme.");

}

public static void main(String[] args)

{

B b = new B();

b.callme();

}

}