PatrickVideos.com

Java Programming

Step by Step – sample programs

Supplement to the video tutorial.

Patrick Washington

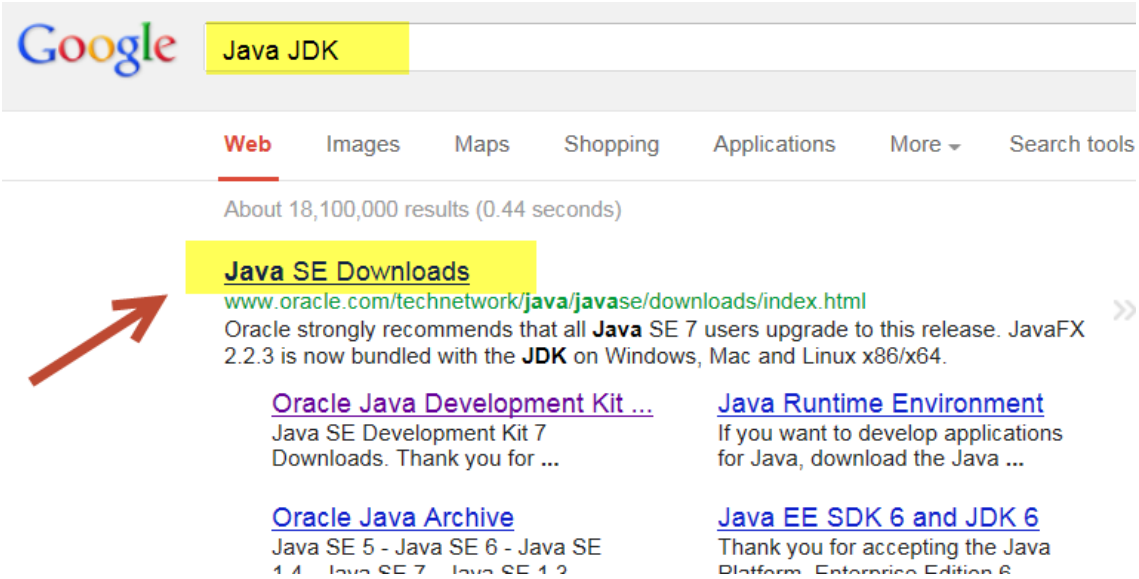
Contents

Install Java and Eclipse

Download and Install Java [00h:05m:10s]

1) Download Java JDK exe from Oracle website. You can search in google.com for Java JDK . Click on the first link.

http://www.oracle.com/technetwork/java/javase/downloads/index.html



2) Click on JDK download.

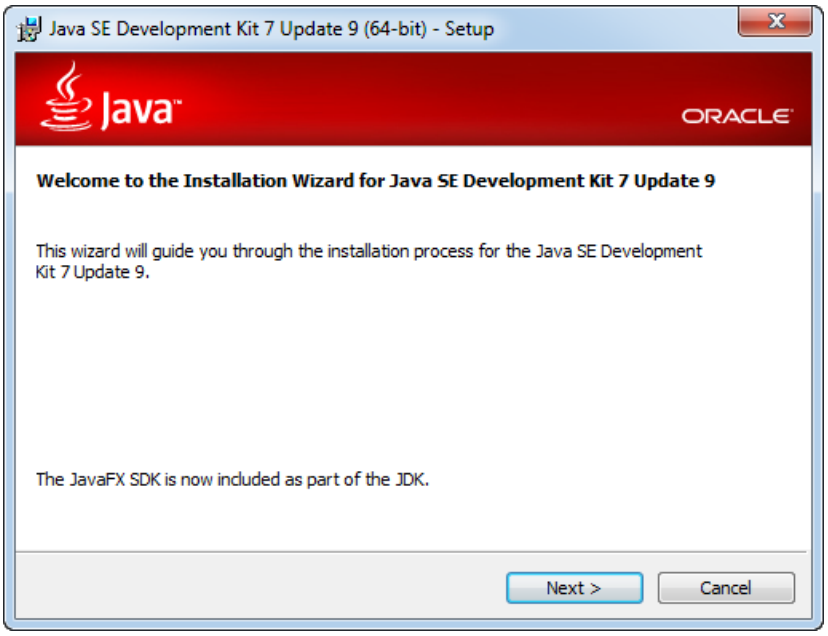
3) Click Accept License Agreement and depending on your Operating System, select the appropriate download.

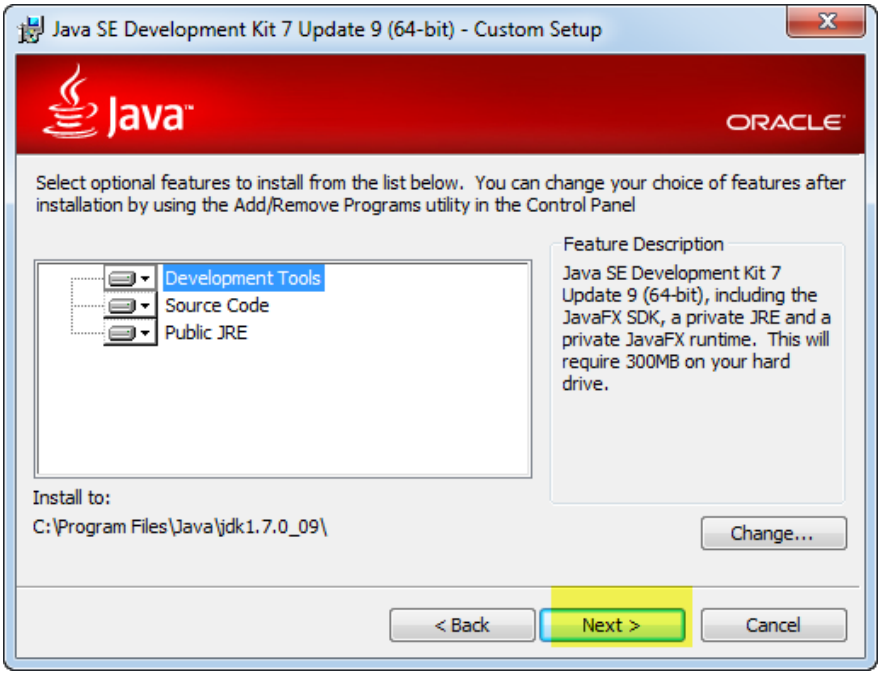
Java SE Development Kit 7u9

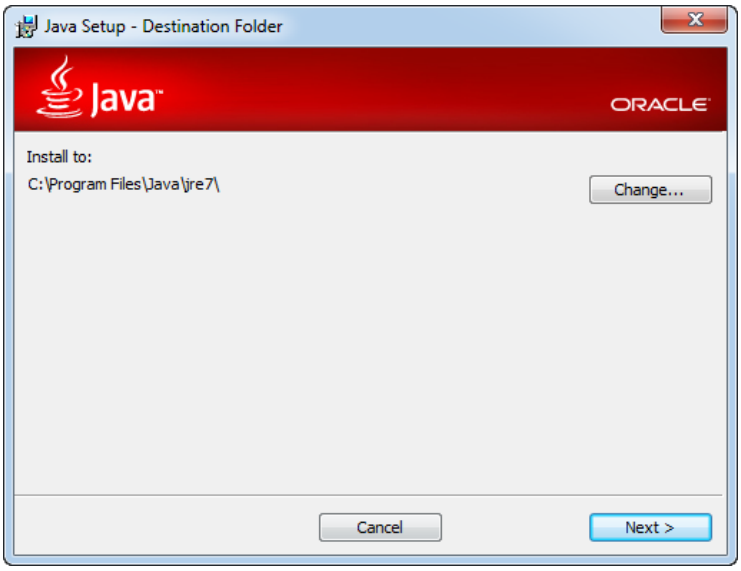
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement **Decline License Agreement**

Product / File Description	File Size	Download
Linux x86	120.63 MB	jdk-7u9-linux-i586.rpm
Linux x86	92.85 MB	jdk-7u9-linux-i586.tar.gz
Linux x64	118.82 MB	jdk-7u9-linux-x64.rpm
Linux x64	91.59 MB	jdk-7u9-linux-x64.tar.gz
Mac OS X	143.47 MB	jdk-7u9-macosx-x64.dmg
Solaris x86	135.14 MB	jdk-7u9-solaris-i586.tar.Z
Solaris x86	91.51 MB	jdk-7u9-solaris-i586.tar.gz
Solaris SPARC	135.7 MB	jdk-7u9-solaris-sparc.tar.Z
Solaris SPARC	95.15 MB	jdk-7u9-solaris-sparc.tar.gz
Solaris SPARC 64-bit	22.8 MB	jdk-7u9-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.51 MB	jdk-7u9-solaris-sparcv9.tar.gz
Solaris x64	22.48 MB	jdk-7u9-solaris-x64.tar.Z
Solaris x64	14.94 MB	jdk-7u9-solaris-x64.tar.gz
Windows x86	88.35 MB	jdk-7u9-windows-i586.exe
Windows x64	90.03 MB	jdk-7u9-windows-x64.exe



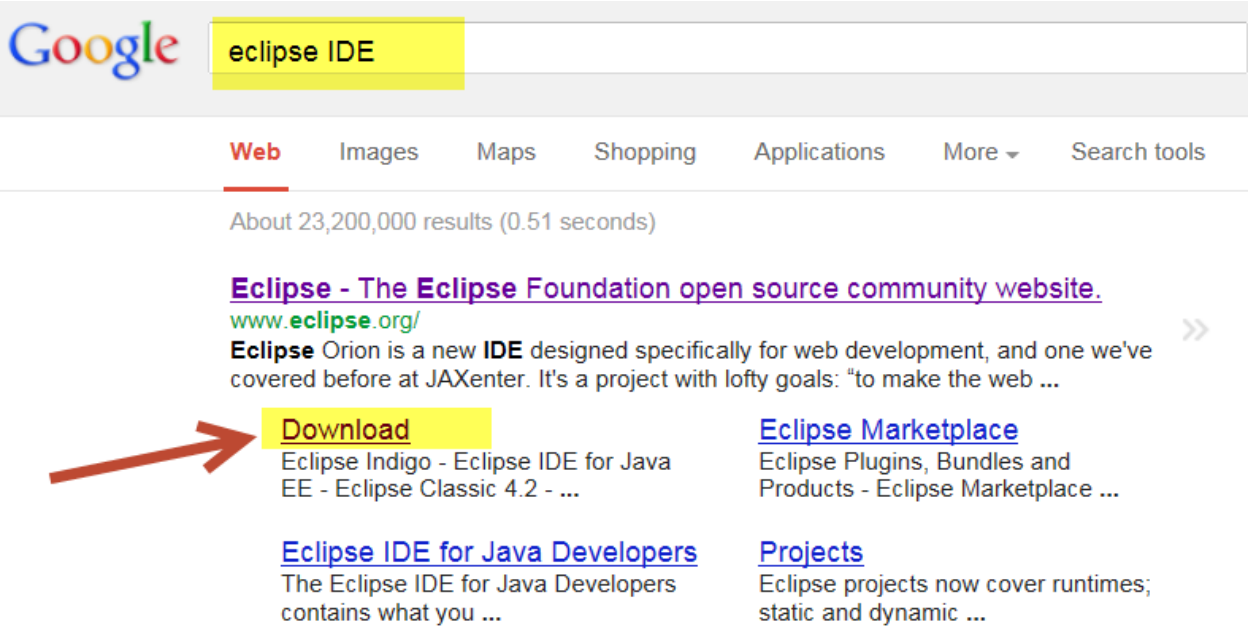

Java SE Development Kit 7 Update 9 (64-bit) - Custom Setup
Java
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Source Code
Public JRE
Feature Description
Java SE Development Kit 7 Update 9 (64-bit), including the JavaFX SDK, a private JRE and a private JavaFX runtime. This will require 300MB on your hard drive.
Install to:
C:\Program Files\Java\jdk1.7.0_09\
Change...
< Back
Next >
Cancel


Java Setup - Destination Folder
Java
ORACLE
Install to:
C:\Program Files\Java\jre7\
Change...
Cancel
Next >


Java SE Development Kit 7 Update 9 (64-bit) - Complete
Java
ORACLE
Java SE Development Kit 7 Update 9 (64-bit) Successfully Installed
Register Java for FREE and get:
- Notification of new versions, patches, and updates
- Special offers on Oracle products, services and training
- Access to early releases and documentation
When installation is complete, product and system data will be collected and the JDK product registration form will be presented. If you do not register, this information will not be saved.
More Information
Close

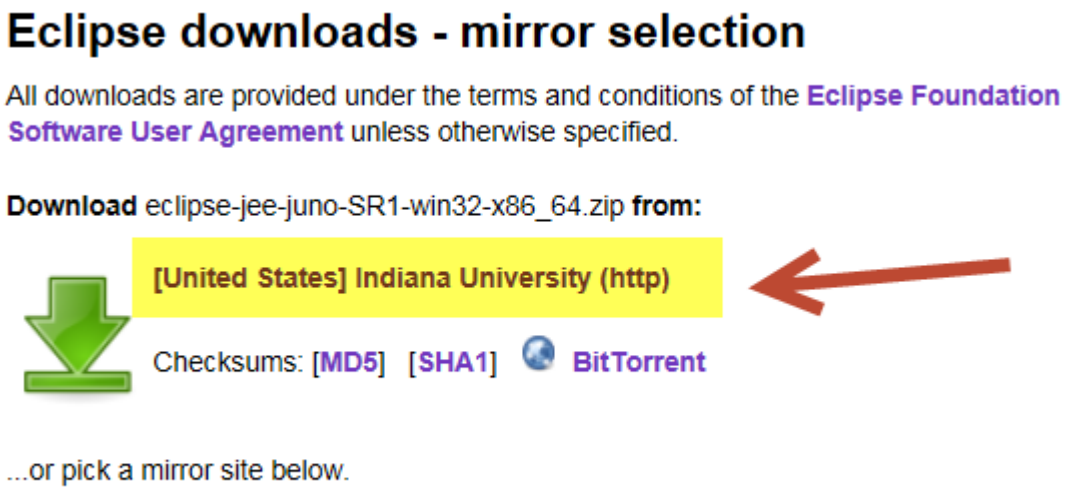
Download and Install Eclipse IDE (Integrated Development Environment) [00h:07m:14s]

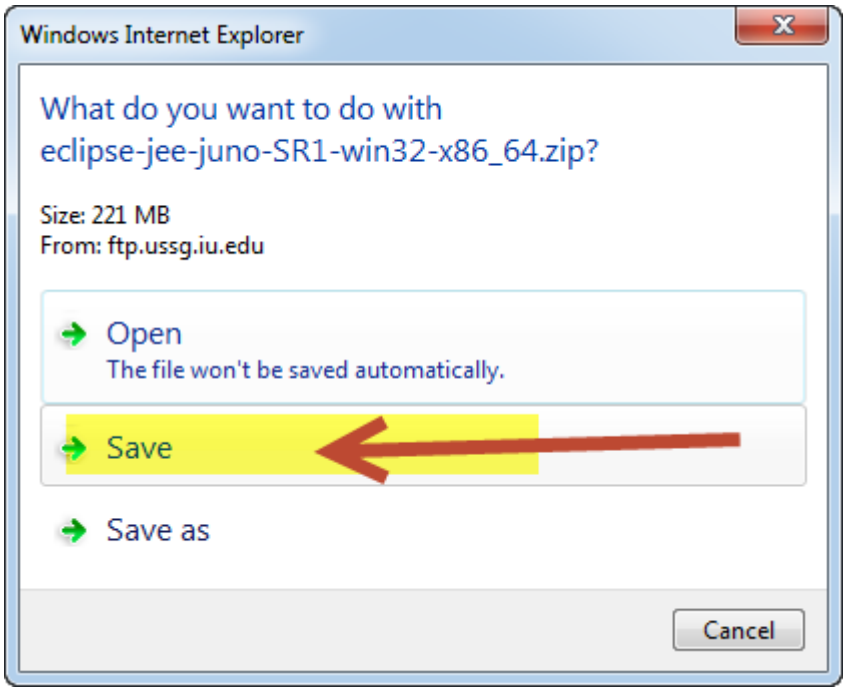
1) Search for Eclipse IDE in google.com. Eclipse IDE makes writing Java programs easier.

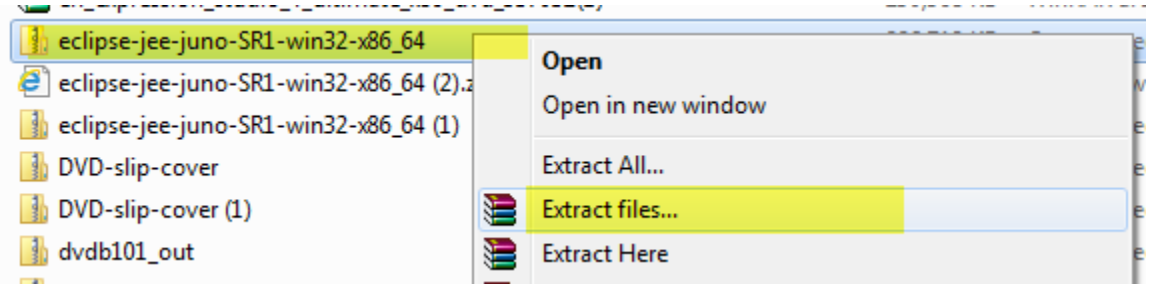


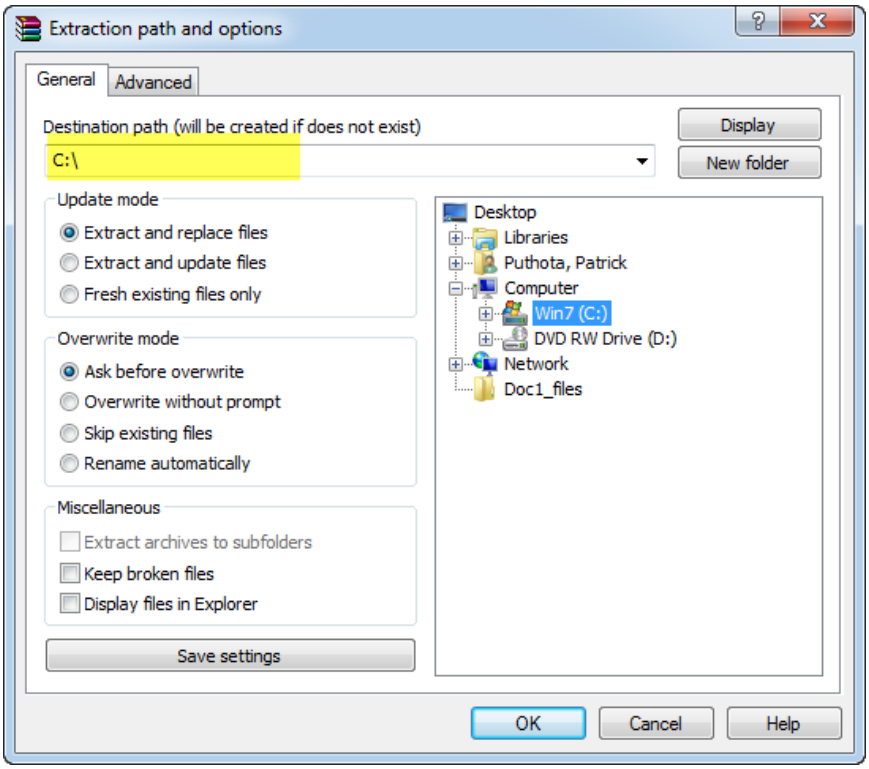
2) Select the appropriate download depending on the OS.

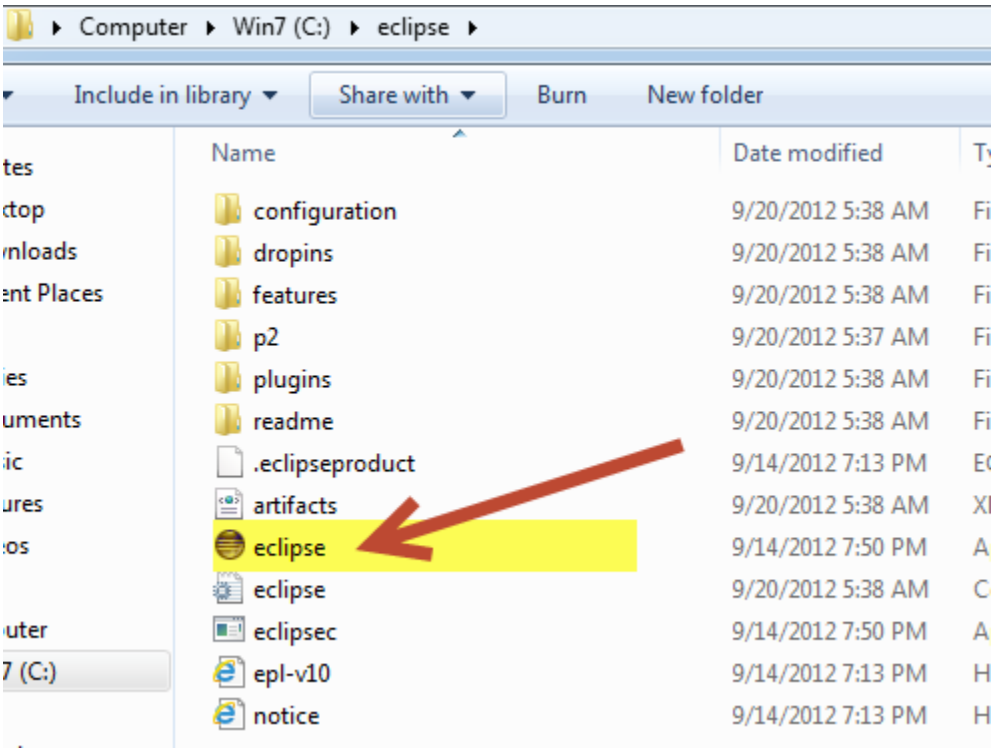
3) Click on one of the mirror sites

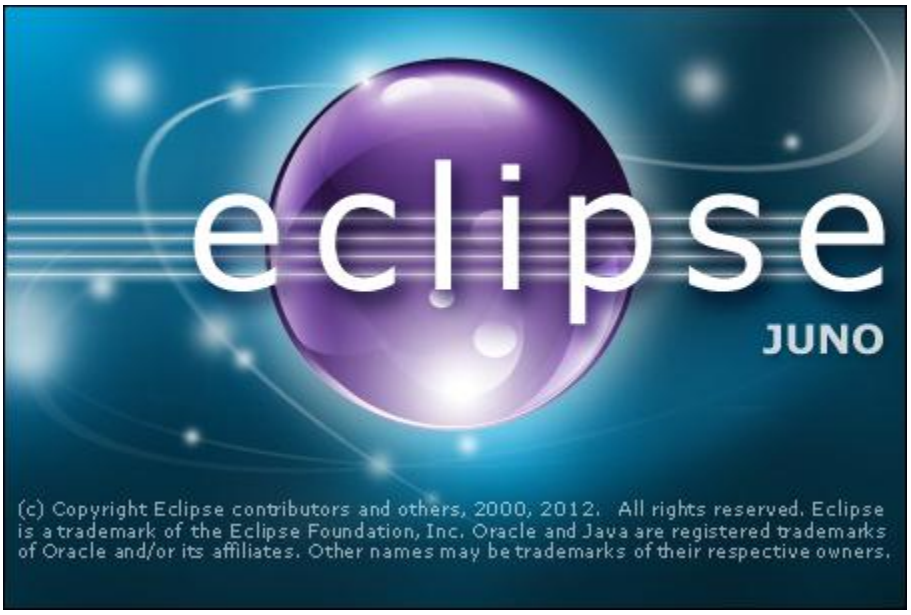




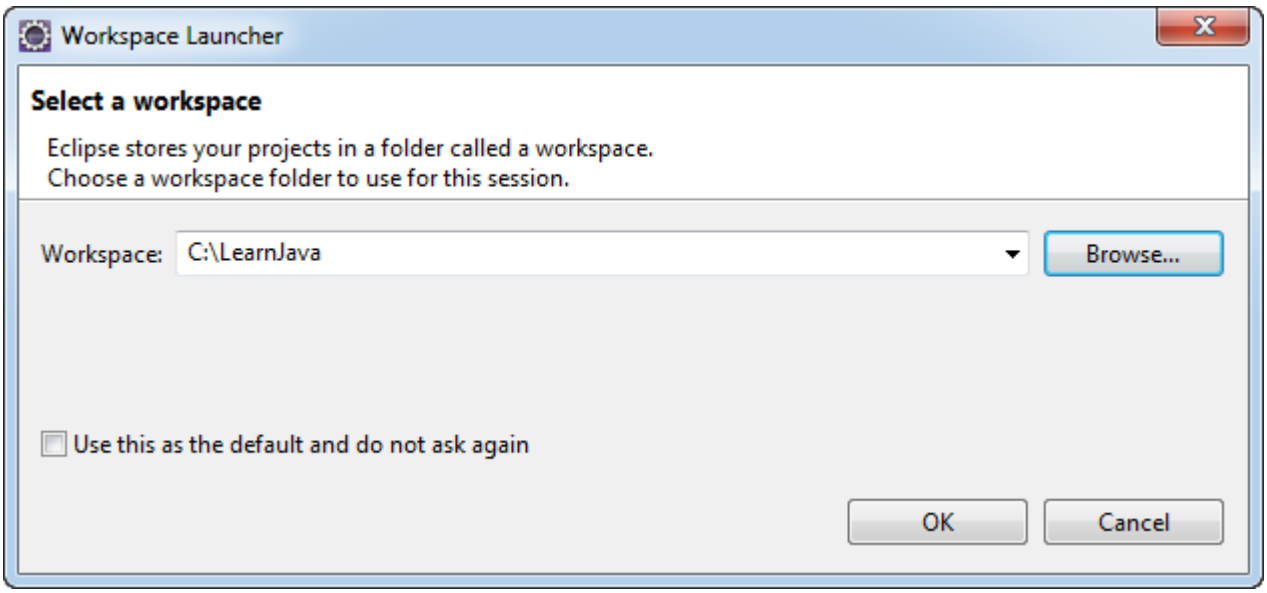

eclipse-jee-juno-SR1-win32-x86_64
eclipse-jee-juno-SR1-win32-x86_64 (2).z
eclipse-jee-juno-SR1-win32-x86_64 (1)
DVD-slip-cover
DVD-slip-cover (1)
dvdb101_out
Open
Open in new window
Extract All...
Extract files...
Extract Here


Extraction path and options
General
Advanced
Destination path (will be created if does not exist)
C:\
Display
New folder
Update mode
Extract and replace files
Extract and update files
Fresh existing files only
Overwrite mode
Ask before overwrite
Overwrite without prompt
Skip existing files
Rename automatically
Miscellaneous
Extract archives to subfolders
Keep broken files
Display files in Explorer
Save settings
Desktop
Libraries
Puthota, Patrick
Computer
Win7 (C:)
DVD RW Drive (D:)
Network
Doc1_files
OK
Cancel
Help

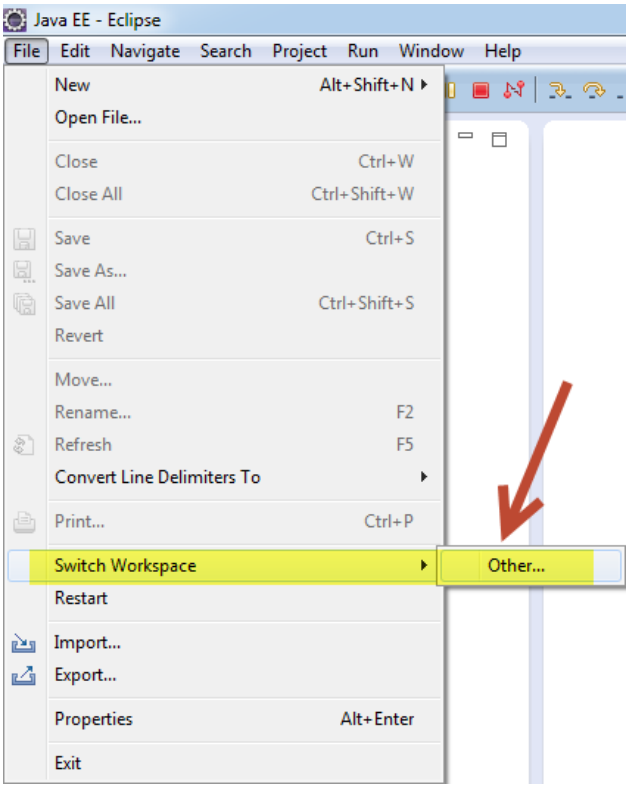

Computer ▸ Win7 (C:) ▸ eclipse ▸
Include in library
Share with
Burn
New folder
Name
Date modified
configuration 9/20/2012 5:38 AM
dropins 9/20/2012 5:38 AM
features 9/20/2012 5:38 AM
p2 9/20/2012 5:37 AM
plugins 9/20/2012 5:38 AM
readme 9/20/2012 5:38 AM
.eclipseproduct 9/14/2012 7:13 PM
artifacts 9/20/2012 5:38 AM
eclipse 9/14/2012 7:50 PM
eclipse 9/20/2012 5:38 AM
eclipsec 9/14/2012 7:50 PM
epl-v10 9/14/2012 7:13 PM
notice 9/14/2012 7:13 PM


eclipse
JUNO
(c) Copyright Eclipse contributors and others, 2000, 2012. All rights reserved. Eclipse is a trademark of the Eclipse Foundation, Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

4) Select a folder where you want to store your programs.

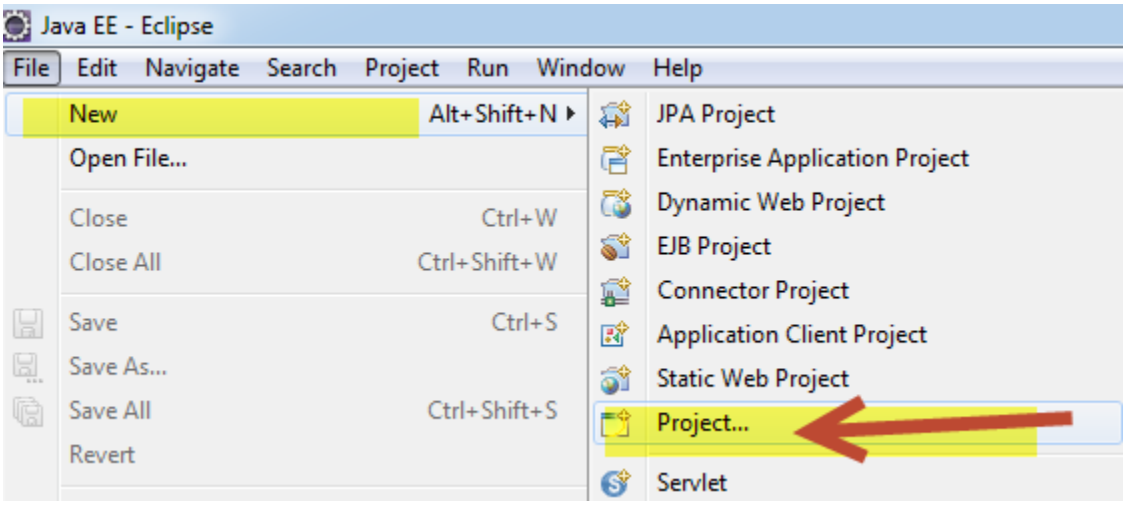


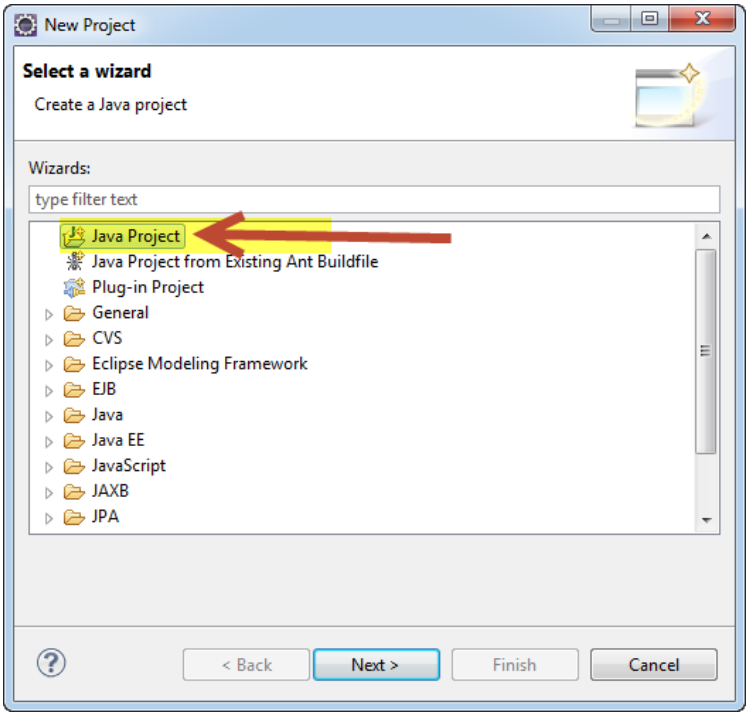
5) You can always change your workspace(the place where you store your programs by clicking on File – Switch Workspace



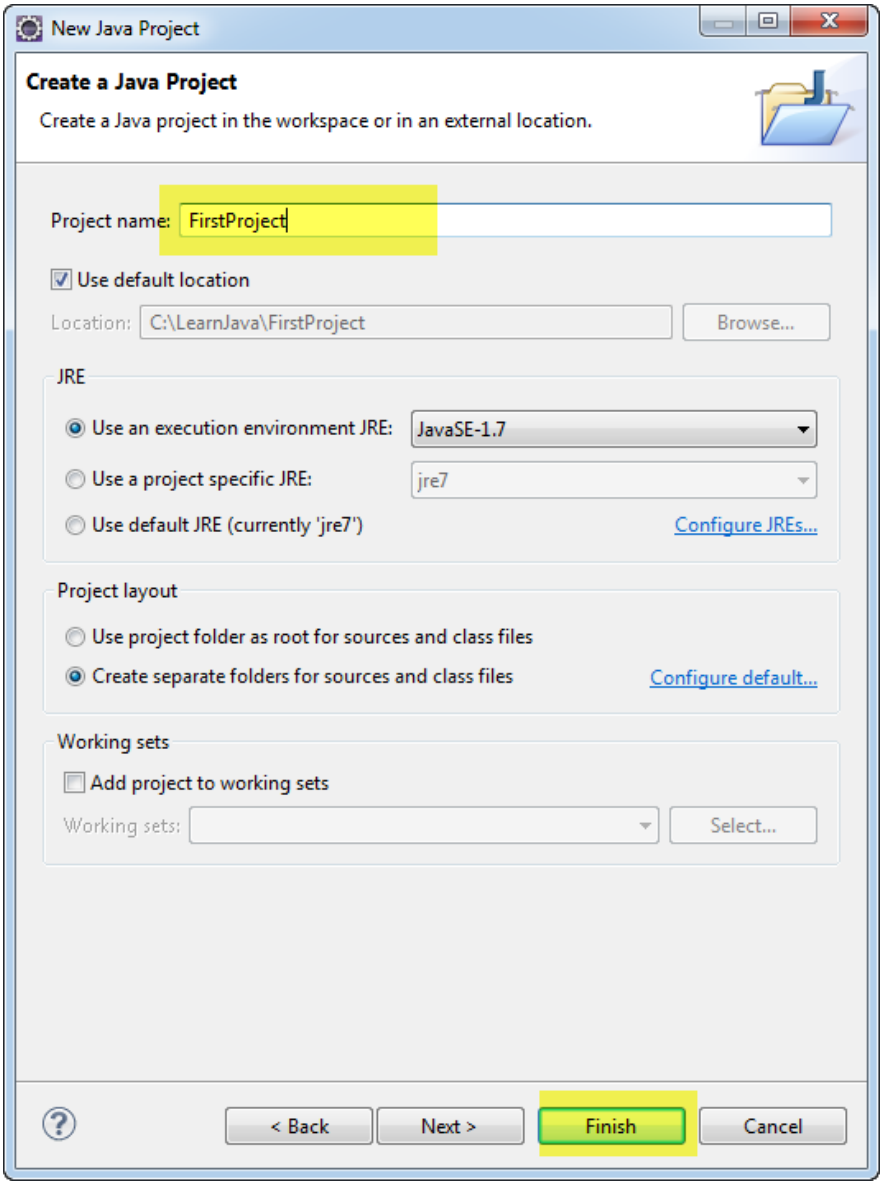
Java Syntax & Statements

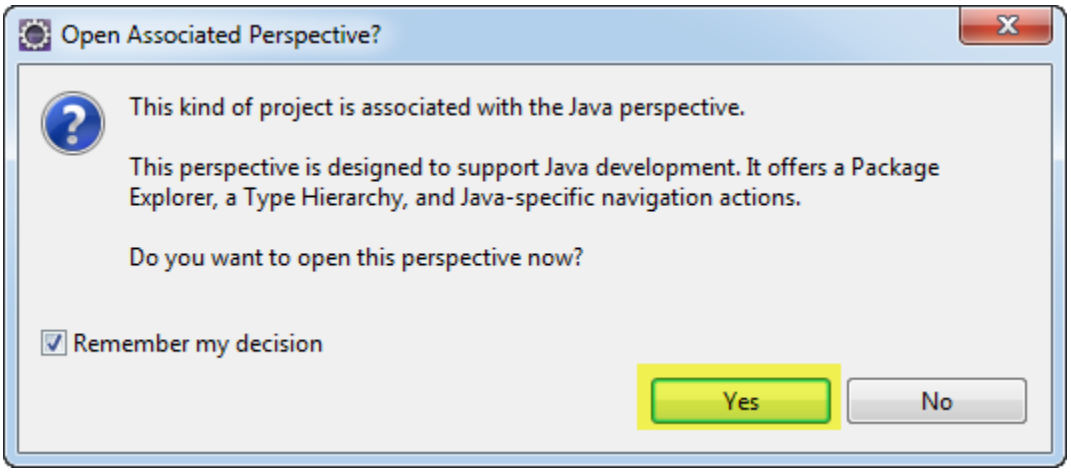
Create Java Project [00h:08m:54s]



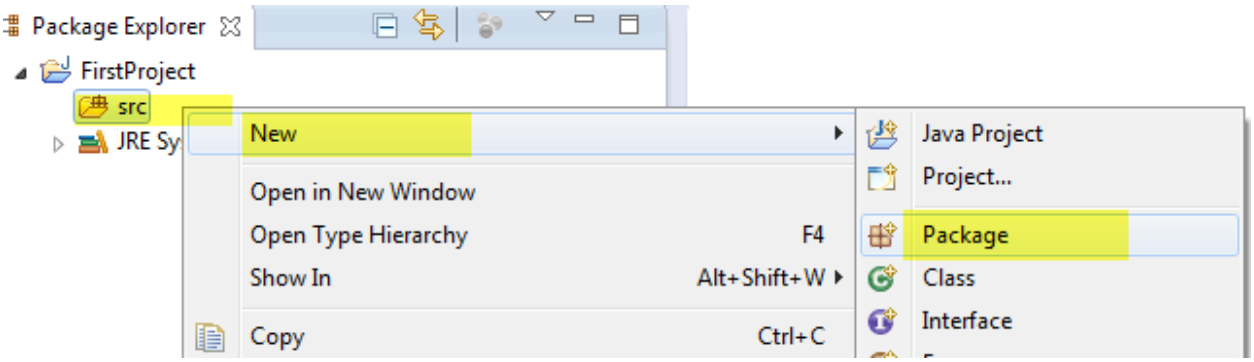


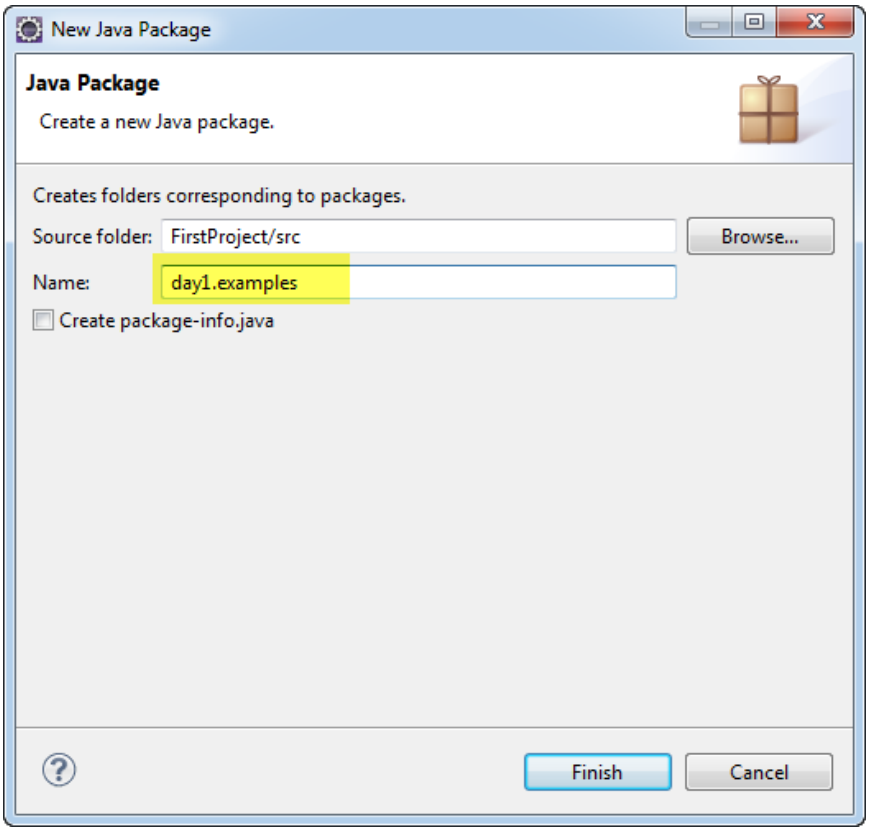
Give any name for the Java project



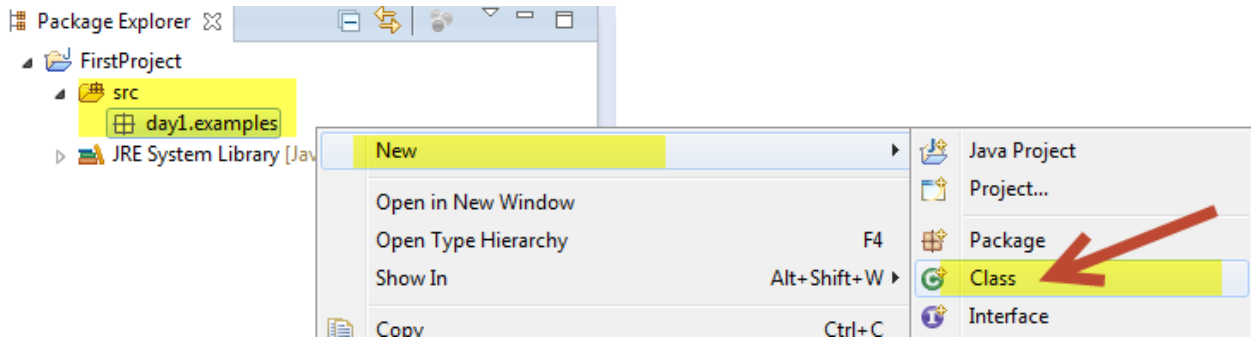


Create a package under the source folder. Packages are like folders in Java they are used to organize your programs.

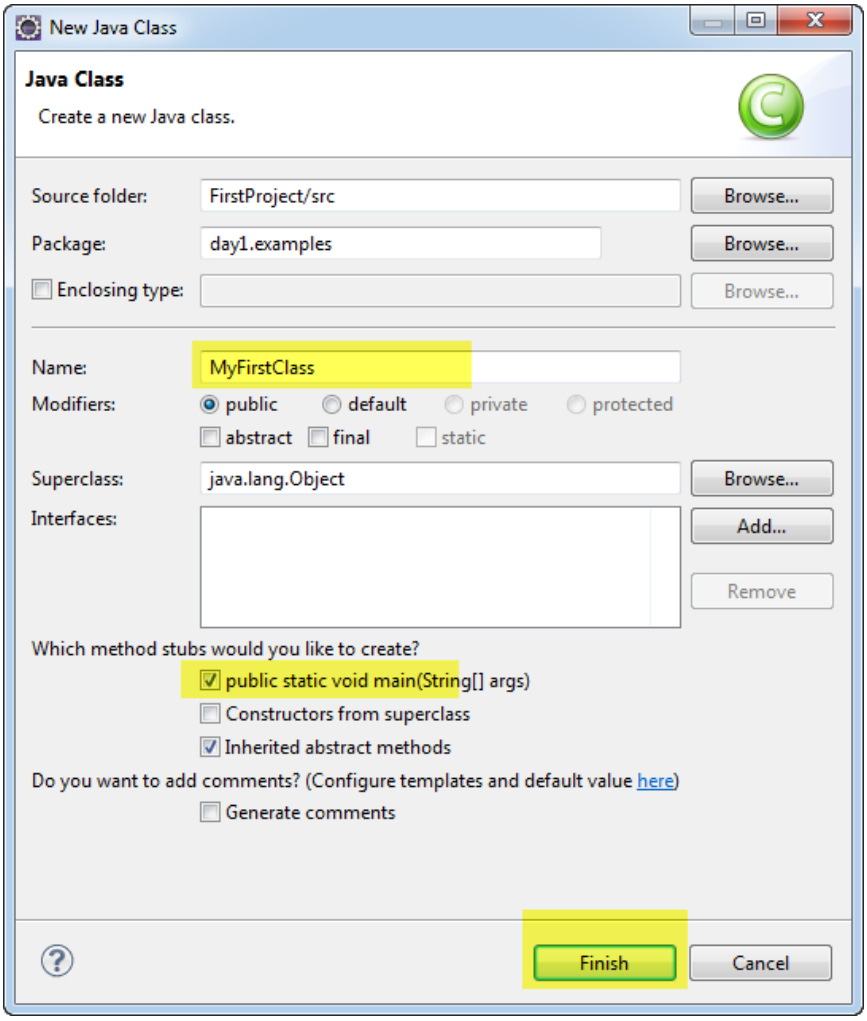




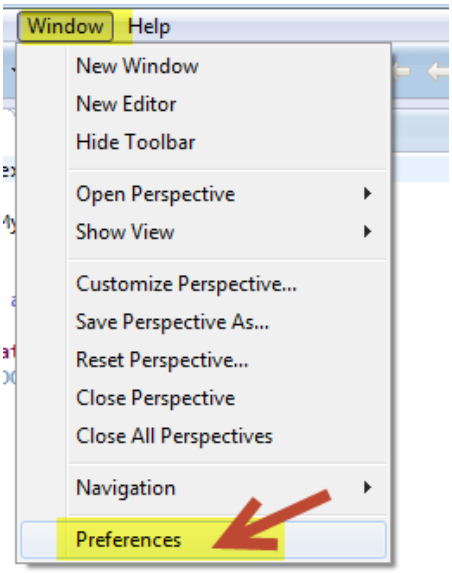
To write a program in Java you first need to create a class. Don't worry about what a class is, we will learn that later.

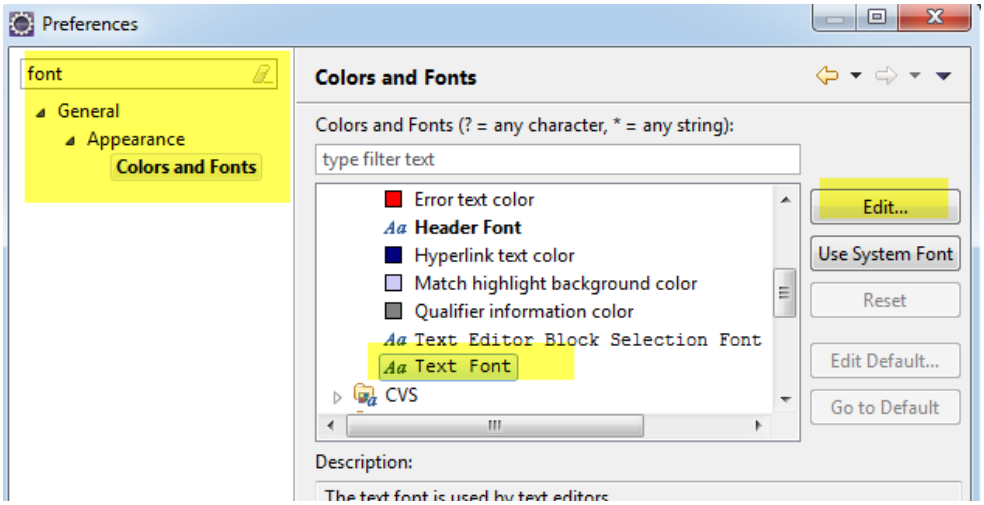


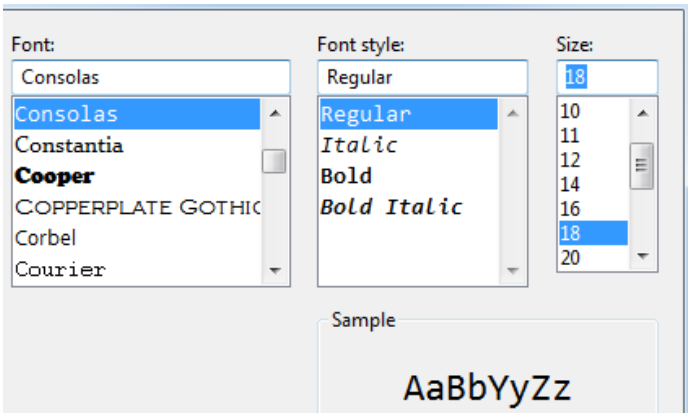
Make sure you check “public static void main(String[] args)”. We will see the use of this at a later stage.



To change the font size in eclipse.

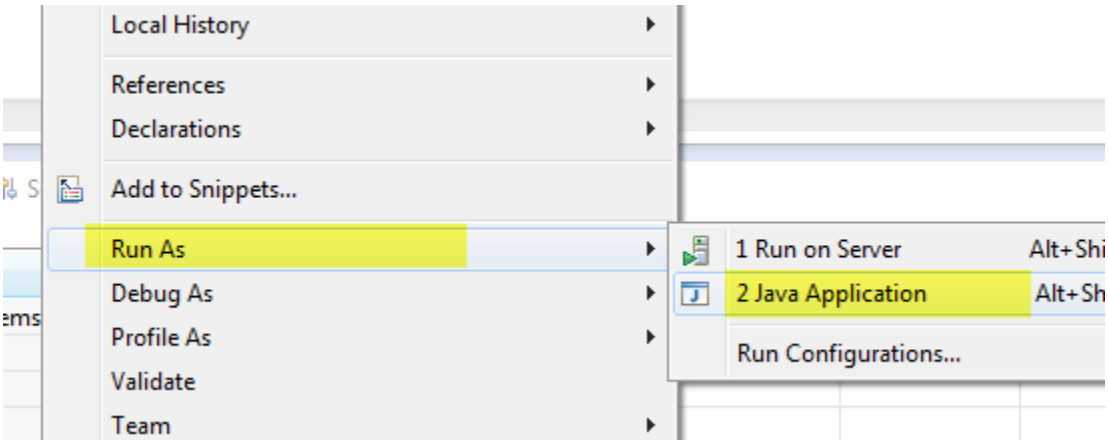


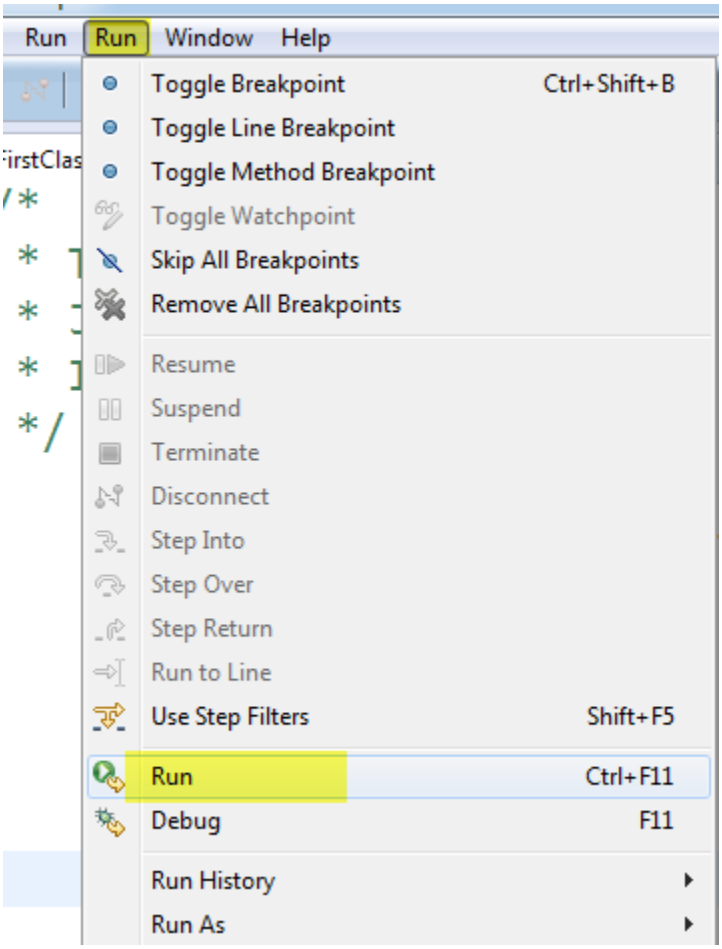




First Program - Print statement [00h:11m:25s]

```
package day1.examples;

public class MyFirstClass {

/*
 * This is my first
 * Java program.
 * I am new to Java.
 */
	public static void main(String[] args) {
		// My First statement
		System.out.println("Hello World");

		int x = 10;
		int y = 3;
		int result = x % y;
		System.out.println("result = " + result);

	}
}
```




Run
Run
Window
Help
Toggle Breakpoint
Ctrl+Shift+B
Toggle Line Breakpoint
Toggle Method Breakpoint
Toggle Watchpoint
Skip All Breakpoints
Remove All Breakpoints
Resume
Suspend
Terminate
Disconnect
Step Into
Step Over
Step Return
Run to Line
Use Step Filters
Shift+F5
Run
Ctrl+F11
Debug
F11
Run History
Run As

if condition [00h:18m:00s]

```
package day1.examples;

public class ExampleIf {

	public static void main(String[] args) {

		int x = 20;
		int y = 20;

		if(x < y){
			System.out.println("X is less than Y");
		}else if(x == y){
			System.out.println("X is equal to Y");
		}else{
			System.out.println("X is greater than Y");
		}
	}
}
```

for loop [00h:24m:24s]

```
package day1.examples;

public class ExampleFor {

	public static void main(String[] args) {
		// i++   i = i + 1
	//  for(initial value; condition; increment/decrement)
		for(int i = 10; i > 0; i--){
			System.out.println("i = " + i);
		}
	}
}
```

while loop [00h:28m:08s]

```
package day1.examples;

public class ExampleWhile {

	public static void main(String[] args) {

		int x = -10;

		while(x <= 0){
			System.out.println("x = " + x);
			//x++;
			//x = x + 1;
			x += 2;
		}
	}
}
```

do while loop [00h:31m:20s]

```
package day1.examples;

public class ExampleDoWhile {

	public static void main(String[] args) {

		int x = -10;

		do{
			System.out.println("x = " + x);
			x--;
		}while(x > 0);
	}
}
```

&& and || [00:34m:15s]

```
package day1.examples;

public class ExampleAndOr {

	public static void main(String[] args) {

		int x, y;
		x = -10;
		y = 10;
		// && and
		// || or
		if(x > 0 && y > 0){
			System.out.println(" Both nums are +ve");
		}else if(x > 0 || y > 0){
			System.out.println(" atleast one num is +ve");
		}else{
			System.out.println(" Both nums are -ve");
		}
	}
}
```

switch case [00h:37m:52s]

```
package day1.examples;

public class ExampleSwitchCase {

	public static void main(String[] args) {

		String j = "Two";

		switch(j){

			case "Zero":
				System.out.println("Value is 0");
				break;
			case "One":
				System.out.println("Value is 1");
				break;
			case "Two":
				System.out.println("Value is 2");
				break;
			case "Three":
				System.out.println("Value is 3");
				break;
			default:
				System.out.println("No Value");
				break;
		}
	}
}
```

Arrays and enhanced for [00h:40m:53s]

```
package day1.examples;

public class ExampleArrays {

	public static void main(String[] args) {

		int[] a = {10, 20, 30, 40, 50};

		System.out.println(a[2]  + "  " + a[4]);

		//enhanced for
		for(int temp : a){
			System.out.println(temp);
		}

		System.out.println("---------");
		int[] x = new int[5];

		x[3] = 25;
		x[0] = 12;

		for(int temp2 : x){
			System.out.println(temp2);
		}
		System.out.println("---------");
		String[] st = {"one", "two", "three"};

		for(String temp : st){
			System.out.println(temp);
		}

		System.out.println("---------");
		String[] stx = new String[5];

		stx[3] = "25";
		stx[0] = "12";
		for(String temp2 : stx){
			System.out.println(temp2);
		}
	}
}
```

Two Dimensional arrays [00h:48m:41s]

```
package day1.examples;

public class ExampleTwoDimArray {

	public static void main(String[] args) {

		int[][] TwoDim = new int[4][3];

		//TwoDim[2][1] = 10;

		int temp = 10;

		for(int i = 0; i < 4; i++){
			for(int j = 0; j < 3; j++){

				TwoDim[i][j] = temp;
				temp += 10;
			}
		}

		for(int i = 0; i < 4; i++){
			for(int j = 0; j < 3; j++){

				System.out.print(TwoDim[i][j] + " ");

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

String manipulation [00h:54m:30s]

```
package day1.examples;

public class ExampleString {

	public static void main(String[] args) {

		String x = "James Dean";

		System.out.println("Hello " + x);

		System.out.println(x.toUpperCase());

		System.out.println(x.substring(2));

		System.out.println(x.substring(2,7));

		System.out.println(x.charAt(3));

		String age = "37";
		String salary = "78965.83";
		//Wrapper classes

		int a = Integer.parseInt(age) + 2;
		System.out.println(a);

		double s = Double.parseDouble(salary) * .15;
		System.out.println(s);
	}
}
```

Exercise [01h:00m:00s]

```
package day1.exercises;

public class ExerciseA {

	public static void main(String[] args) {

		double salary = 20000;
		double tax = 0.0;

		if(salary <= 15000){
			tax = salary * .10;
		}else if(salary <= 40000){
			tax = salary * .20;
		}else{
			tax = salary * .30;
		}
		System.out.println("Tax = " + tax);
	}
}
```

```
package day1.exercises;

public class ExerciseB {

	public static void main(String[] args) {

		String[] nums = {"10", "20", "30", "40"};

		int total = 0;

		for(String temp : nums){
			total += Integer.parseInt(temp);
		}

		System.out.println("Total =" + total);

	}
}
```

```
package day1.exercises;

public class ExerciseC {

	public static void main(String[] args) {

		String a = "Hello World";
		System.out.println(a.substring(6).toLowerCase());

	}
}
```

Class and Object [01h:12m:45s]

```
package day2.classandobject;

public class Employee {

	//Class : Class has data and methods.
	//	Template for objects.

	double salary;
	double bonus;

	void calculateTotalPay(){
		double totalPay = salary + bonus;
		System.out.println("Total Pay = " + totalPay);
	}
}
```

```
package day2.classandobject;

public class TestEmployee {

	public static void main(String[] args) {
		//Object : Copy / Instance of a class.

		Employee alex = new Employee();
		Employee linda = new Employee();
		Employee john = new Employee();

		alex.salary = 90000;
		alex.bonus = 20000;
		alex.calculateTotalPay();

		linda.salary = 100000.78;
		linda.bonus = 23456.89;
		linda.calculateTotalPay();

	}
}
```

Return a value/ passing arguments/ this keyword [01h:27m:27s]

```
package day2.classandobject;

public class Box {

	int length;// class level data
	int width;
//returntype methodname(passing arguments)
	int calculateArea(int length, int width){
		int area = this.length * this.width ;
		System.out.println("Area = "+ area);
		return area;
	}
}
```

```
package day2.classandobject;

public class TestBox {

	public static void main(String[] args) {

		Box ups = new Box();
		Box fedEx = new Box();

		ups.length = 5;
		ups.width = 10;
		int area1 = ups.calculateArea(4, 3);

		fedEx.length = 6;
		fedEx.width = 7;
		int area2 = fedEx.calculateArea(2, 5);

		System.out.println("Total of ups & fedEx = " + (area1 + area2));

	}
}
```

Constructors [01h:36m:30s]

```
package day2.constructors;

public class SmallBox {

	int length;
	int width;

	//Constructor : Is a method that has the same name as the class.
	// It is executed when an object is created.
	// It is used to set default values.
	// does not return anything including void.
	SmallBox(){
		this.length = 5;
		this.width = 6;
		System.out.println("Constructor fired");
	}

	SmallBox(int length, int width){
		this.length = length;
		this.width = width;
	}

	void calculateArea(){
		System.out.println("Area = " + (length * width));
	}
}
```

```
package day2.constructors;

public class TestBox {
	public static void main(String[] args) {
		SmallBox obj = new SmallBox();
		obj.calculateArea();

		SmallBox ups = new SmallBox(3,4);
		ups.calculateArea();
	}
}
```

Polymorphism (Overloading) [01h:45m:10s]

```
package day2.oop.overloading;

public class Box {

	//Polymorphism
	//Overloading: when methods of the same name are
	//		differentiated by their passing arguments;
	void calculateArea(int length){
		System.out.println("Area = " + (length*length));
	}

	void calculateArea(double length){
		System.out.println("Double Area = " + (length*length));
	}

	void calculateArea(int length, int width){
		System.out.println("Area = " + (length*width));
	}

}
```

```
package day2.oop.overloading;

public class TestBox {

	public static void main(String[] args) {

		Box obj = new Box();
		obj.calculateArea(4);
		obj.calculateArea(3.7);
		obj.calculateArea(2,4);

	}
}
```

Encapsulation & Data hiding [01h:48m:36s]

```
package day2.oop.datahiding;

public class Employee {

	private double salary;
	private double bonus;

	public double getBonus() {
		return bonus;
	}

	public void setBonus(double bonus) {
		this.bonus = bonus;
	}

	public void setSalary(double salary){

		if(salary >= 40000 && salary <=200000){
			this.salary = salary;
		}else{
			this.salary = 0;
			System.out.println("Please check salary");
		}
	}

	public double getSalary(){
		return salary;
	}

	public void calculateTotalPay(){
		double totalPay = salary + bonus;
		System.out.println("Total Pay = " + totalPay);
	}
}
```

```
package day2.oop.datahiding;

public class TestEmployee {

	public static void main(String[] args) {

		Employee alex = new Employee();
		//alex.salary = 100000;
		alex.setSalary(100000);
		alex.setBonus(20000);
		alex.calculateTotalPay();

		//System.out.println(alex.getSalary());
	}
}
```

Sample Project [02h:05m:13s]

```
package day2.sampleproject;

public class Employee {

	private String empName;
	private int grade;

	public int getGrade() {
		return grade;
	}

	public Employee(String empName, int grade){
		this.empName = empName;
		this.grade = grade;
	}

/*	public String getEmpInfo(){
		return empName + "(" + grade + ")";
	}*/

	public String toString(){
		return empName + "(" + grade + ")";
	}

}
```

```
package day2.sampleproject;

import java.util.ArrayList;
import java.util.List;

public class Department {

	private String deptName;
	private double budget;
	//private Employee[] emps = new Employee[5];
	private List<Employee> emps = new ArrayList<>();
	//private int counter = 0;

	public Department(String deptName){
		this.deptName = deptName;
		this.budget = 50000;
	}

	public void addEmployee(Employee obj){
		//emps[counter] = obj;
		emps.add(obj);
		//counter++;

		if(obj.getGrade() >= 5){
			this.budget += 150000;
		}else{
			this.budget += 100000;
		}
	}

	public void describe(){
			String temp = "Dept Name: " + this.deptName
				+ "\nBudget : " + this.budget
				+ "\nEmployees : ";
		for(Employee x : emps){
			//if(x != null){
				temp += x + " ";
			//}
		}
		System.out.println(temp);
	}
}
```

```
package day2.sampleproject;

public class TestCompany {

	public static void main(String[] args) {

		Employee alex = new Employee("Alex Rod", 6);
		Employee linda = new Employee("Linda Berry", 7);
		Employee john = new Employee("John Doe", 3);

		Employee sara = new Employee("Sara Time", 7);
		Employee james = new Employee("James Doe", 4);

		Department sales = new Department("XYZ Sales");
		Department IT = new Department("XYZ IT");

		sales.addEmployee(alex);
		sales.addEmployee(linda);
		sales.addEmployee(john);

		IT.addEmployee(sara);
		IT.addEmployee(james);

		sales.describe();
		System.out.println("--------------");
		IT.describe();

	}
}
```

Inheritance [02h:32m:16s]

```
package day3.inheritance;

// base class, parent class, super class
public class Box {

	public void calculateArea(int length, int width){

		System.out.println("Area = " + (length * width));

	}
}
```

```
package day3.inheritance;

// inheritance : when a class acquires the properties
// of another class

// subclass
public class NewBox extends Box{

	public void calculateVolume(int length, int width, int height){

		System.out.println("Volume= "+ (length*width*height));

		}

	public void calculateArea(int length, int width){

		System.out.println("Sub Area = " + (length / width));

	}

}
```

```
package day3.inheritance;

public class TestBox {

	public static void main(String[] args) {

		Box fedEx = new Box();

		fedEx.calculateArea(3, 4);

		NewBox ups = new NewBox();
		ups.calculateArea(4, 2);
		ups.calculateVolume(3, 4, 5);

	}
}
```

Abstract Class [02h:37m:35s]

```
package day3.abstractex;

public abstract class Container {

	public void calculateVolume(int length, int height){
		double volume = calculateAreaOfBase(length) * height;
		System.out.println("Volume = " + volume);
	}

	public abstract double  calculateAreaOfBase(int length);
}
```

```
package day3.abstractex;

public class CircleContainer extends Container{

	public double calculateAreaOfBase(int length) {

		double area = Math.PI * (length/2)*(length/2);
		System.out.println("Circle Area = " + area);
		return area;

	}
}
```

```
package day3.abstractex;

public class SquareContainer extends Container{

	public double calculateAreaOfBase(int length) {

		double area = length * length;
		System.out.println("square Area = " + area);
		return area;
	}
}
```

```
package day3.abstractex;

public class TestContainer {

	public static void main(String[] args) {

		Container c1 = new CircleContainer();
		c1.calculateVolume(10, 5);

		c1 = new SquareContainer();
		c1.calculateVolume(10, 5);

	}
}
```

Interface [02h:47m:35s]

```
package day3.interfaceex;

public interface Office {

	public void New();
	public void open();
	public void save();
}
```

```
package day3.interfaceex;

public class Word implements Office{

	public void New(){
		System.out.println("Word New .doc");
	}

	public void open(){
		System.out.println("Word Open .doc");
	}

	public void save(){
		System.out.println("Word Save .doc");
	}

}
```

```
package day3.interfaceex;

public class Excel implements Office{

	public void New(){
		System.out.println("Excel New .xls");
	}

	public void open(){
		System.out.println("Excel Open .xls");
	}

	public void save(){
		System.out.println("Excel Save .xls");
	}
}
```

```
package day3.interfaceex;

public class TestOffice {

	public static void main(String[] args) {

		Office obj = new Word();
		obj.open();
		obj.save();

		obj = new Excel();
		obj.open();
		obj.save();

	}
}
```

super keyword [02h:53m:39s]

```
package day3.superex;

public class Box {

	public double calculateArea(double length, double width){
		System.out.println("Area = " + (length * width));
		return (length * width);
	}
}
```

```
package day3.superex;

public class SubBox extends Box{

	public void calculateVolume(int length, int width, int height){
		double volume = super.calculateArea(length, width) * height;
		System.out.println("Volume = "+ volume);
	}
}
```

```
package day3.superex;

public class TestBox {

	public static void main(String[] args) {
		SubBox obj = new SubBox();
		obj.calculateVolume(4, 5, 6);

	}
}
```

static keyword [02h:56m:19s]

```
package day3.staticex;

// static is one per class / Not one per object
public class ExampleStatic {
	//static data
	static int staticVariable;

	int nonStaticVariable;

	//static method
	static public void staticMethod()
	{
		System.out.println("staticVariable = " + staticVariable);
		staticVariable++;
		//static methods can only access other static data & methods
	}

	 public void nonStaticMethod()
	{
		System.out.println("nonStaticVariable = " + nonStaticVariable);
		nonStaticVariable++;
	}

	 static {
		 System.out.println("Static block of code");
	 }

	 ExampleStatic(){
		 System.out.println("Constructor");
	 }
}
```

```
package day3.staticex;

public class TestStatic {

	public static void main(String[] args) {
		ExampleStatic obj = new ExampleStatic();
		obj.staticMethod();
		obj.staticMethod();

		ExampleStatic.staticMethod();
		obj.nonStaticMethod();
		obj.nonStaticMethod();

		System.out.println("------------");

		ExampleStatic obj2 = new ExampleStatic();
		obj2.staticMethod();
		obj2.staticMethod();

		obj2.nonStaticMethod();
		obj2.nonStaticMethod();

	}
}
```

```
Static block of code
Constructor
staticVariable = 0
staticVariable = 1
staticVariable = 2
nonStaticVariable = 0
nonStaticVariable = 1
------------
Constructor
staticVariable = 3
staticVariable = 4
nonStaticVariable = 0
nonStaticVariable = 1
```

Collections

ArrayList [03h:08m:38s]

```
package day4.collections;

import java.util.ArrayList;
import java.util.List;
public class ExampleList {

	public static void main(String[] args) {
		String arr[] = new String[10];
		arr[0] = "zero";
		arr[9] = "nine";
		//arr[10] = "onemore";
		//non generic list, can add anything
		List list = new ArrayList();
		list.add("zero");
		list.add("one");
		list.add(2);
		list.add(true);

		for(Object temp : list){
			System.out.println(temp);
		}
		System.out.println("---------");
		//generic list can only add specific
		List<String> gList = new ArrayList<>();
		gList.add("zero");
		gList.add("one");
		gList.add("Two");
		gList.add("Three");
		gList.add("Four");
		//gList.add(5); //cannot add a number
		gList.remove(2);
		gList.remove("Four");
		gList.add(0, "Start");

		for(String temp : gList){
			System.out.println(temp);
		}
	}
}
```

HashMap [03h:18m:21s]

```
package day4.collections;

import java.util.HashMap;

public class HashMapExample {

	public static void main(String[] args) {

		HashMap<String, Double> hm = new HashMap<>();

		hm.put("Alex", 98765.45);
		hm.put("Linda", 108765.45);
		hm.put("John", 88765.45);

		System.out.println(hm.get("Linda"));

	}
}
```

Error Handling

try catch finally Exceptions [03h:27m:02s]

```
package day4.errorhandling;

public class ExampleException {

	public static void main(String[] args) {

		int d[] = new int[3];
		int a = 10;
		int b = 1;
		int c = 0;

		try{
			d[1] = 10;
			c = a/b;
			return;
		} catch (ArrayIndexOutOfBoundsException | ArithmeticException e) {
			System.out.println("Some message = " + e);
		} catch (Exception e) {
			System.out.println("Exception= " + e);
		} finally{
			System.out.println("Will always run");
		}

		System.out.println("c = " + c);

	}
}
```

throw and throws [03h:35m:20s]

```
package day4.errorhandling.throwandthrows;

public class ProgA {

	public int divideNums(int a, int b) throws Exception {
		int c = 0;

		try{
		c = a / b;
		}catch(Exception e){
			System.out.println("ProgA exception");
			throw new Exception("Please check your numbers");
		}

		return c;
	}
}
```

```
package day4.errorhandling.throwandthrows;

public class ProgB {

	public static void main(String[] args) {

		ProgA obj = new ProgA();

		try{
			System.out.println(obj.divideNums(10, 0));
		}catch(Exception e){
			System.out.println("Prog B = " + e.getMessage());
		}
	}
}
```

Miscellaneous

final keyword [03h:45m:29s]

```
package day5.finalkeyword;
public  class ExampleFinal {
	//final keyword can be applied to
	//data, methods & classes.
	//final data is a constant. Cannot be changed.
	//final methods cannot be overridden.
	//final classes cannot be sub classed.
	final double pi = 3.14;
	public  void methodA(){
		//pi = 4.0; // cannot be changed
		System.out.println("pi = " + pi);
	}
}
```

```
package day5.finalkeyword;

public class Example2 extends ExampleFinal{

	public void methodA(){
			System.out.println("example 2");
		}
}
```

```
package day5.finalkeyword;

public class TestFinal {

	public static void main(String[] args) {
		ExampleFinal obj1 = new ExampleFinal();

	}
}
```

Variable Arguments [03h:40m:43s]

```
package day5.varargs;

public class ExampleVarArgs {
	// variable arguments

	public static void addNumbers(String b, int ... a){

		int sum = 0;

		for(int x : a){
			sum += x;
		}

		System.out.println("sum = " + sum + " b = " + b);
	}

	public static void main(String args[]) {
		/*ExampleVarArgs obj = new ExampleVarArgs();
		obj.addNumbers(3, 4, 5);*/

		ExampleVarArgs.addNumbers("Some text" , 3, 4, 5, 78, 92, 34);

	}
}
```

Java definitions

Class: Data and methods that act on the Data. It's a template for an object.

Object: Instance or copy of a class.

Primitive types: byte, short, char, int, long, float , double, boolean.
Java has 8 primitive data types.

Name	Bits	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2,147,483,648 to 2,147,483,647
long	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	'y'
boolean	1	true or false

String is not primitive data type. Use String to store text.
String x = "Hello Word";

Wrappers: Integer, Double, Float, Long, Boolean, Character, Short, Byte.
Can be used to convert String to int (eg) int a = Integer.parseInt("10"); etc.

OOPS concepts: Encapsulation, Polymorphism and Inheritance.

Encapsulation: The concept of having data and methods that act on the data.

Data hiding. When a class hides its attributes (data) and behavior (methods) to other classes.

Access Modifier	class	package	subclass	project
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

Polymorphism:
Overloading: When methods of the same name are differentiated by their passing arguments.
Overriding: When a method in the subclass has the same signature as the method in the super class, the method in the subclass overrides (takes precedence) over the method in the super class.

Inheritance: when a class acquires the properties of another class.

Abstract class: when a class has method declarations (empty methods) and fully defined methods.

Interface: when a class has only method declarations (empty methods), it's called an interface.
Variables declared in an interface are by default public static final.

A class extends another class. A class can extend only one class

A class implements an interface. A class can implement many interfaces.

An interface extends an interface.

static: keyword that can be applied to a variable, method or block of code. It has only one instance per class (where as other variables will be one for every object). It will be executed first when a class in created and it's done only once.
A static method can only call other static methods or variables.

Constructor: method with the same name as that of its class. It's executed when an object is created.

final: keyword that can be applied to class, method or variable. For variables, it denotes that it's a constant. For methods, it means it cannot be overridden. For classes, it means it cannot be subclassed.

finally: keyword used in a try-catch block. The code in a finally block will always be executed, if there was an exception or not.

List: An interface used to hold a collection of objects.

ArrayList: implements the List interface. List a = new ArrayList();
Difference between an ArrayList and Vector
Vector is thread safe and ArrayList is not. (Thread safe in other words synchronized. Only one thread can access at a time. This will ensure other objects don't alter values).

HaspMap and HashTable: Used to hold Key value pair. HashTable is synchronized, HashMap is not.

Package: used like a folder to hold all related classes.

import: used to import other packages.