Lecture No 1:

**· Introduction of Java Platform**

**· Install Java JDK and JRE**

**· Why did Google choose Java over other programming languages?**

**· Android OS Structure**

**· Install Android Studio**

**.** **Creating an Android Project (Java Project) Using Android Studio**

**Introduction of JAVA Platform**

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform.

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

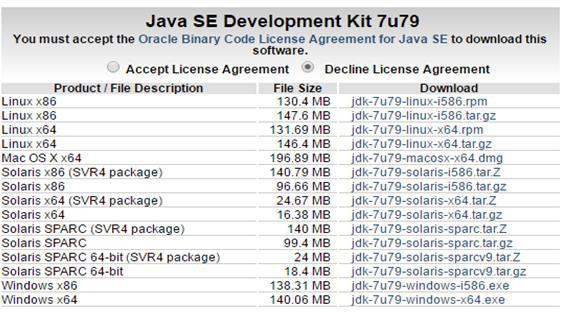
Java is −

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance** − with the use of Just-In-Time compilers, Java enables high performance.
* **Distributed** − Java is designed for the distributed environment of the internet.
* **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

**Install Java JDK and JRE**

To start creating the Java program, we must install the Android Studio (IDE) and to install it, we need to install the prerequisites of it which are the JDK and JRE. You may download the JDK and JRE for free from the official website of Oracle, Below is the download link:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

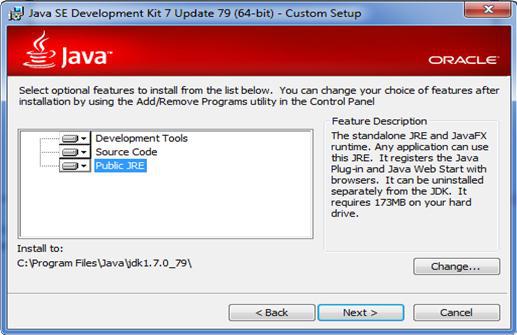


This Java SE kit which you will select includes the JDK & JRE thus your selection will depend on your operating system which you have on your computer.

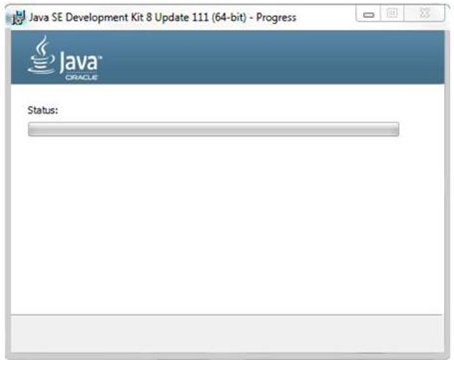
Select “**Accept License Agreement**” and click the JDK file type which is most suitable for your operating system. The selected file will be downloaded on your system. Once the file is downloaded on your system, double click on it and you will see the following screen: Then, click “**Next**” on the following dialog box:



On the following dialog box, click “**Next**”.



The installation will start like following: wait for finish

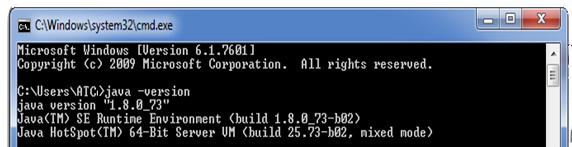


Once the installation is complete, you will see the following screen. Click “**Close**” to finish the installation procedure.



To verify if you have successfully installed JDK on your Windows machine, follow the steps below:

1. Open a command prompt by clicking Start Run, then type “**cmd**” , then click OK
2. In the window that opens, type java –version then press Enter if you are using Microsoft Windows.
3. You should see the following message in the console if your installation was completes successfully.



**Why did Google choose Java over other programming languages?**

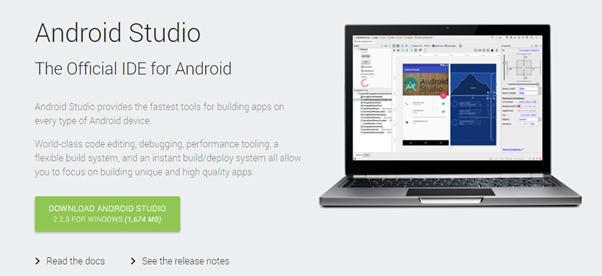
The basic advantages of having Java programming language for Android SDKs (software development kits) are given below:

1. Java is a known programming language; developers know it. Java has yet again emerged as one of the world’s most popular programming language. Also, there are many engineers who specialize in Java making a vast developers’ community which collaborates with each other.
2. It’s harder to shoot yourself with Java than with C/C++ code since it has no pointer arithmetic.
3. It runs in a VM, so no need to recompile it for every phone out there and Java is easy to secure. This is Java’s very important feature. Running on a VM (thus no recompiling) is a huge plus. Also, it easily separates processes from each other, preventing a rogue application from destroying your phone or interfering with other applications. Every App has assigned its own address.
4. As said in point number 1 above, since Java is the most popular programming language, a large number of development tools are available for developers. Java has huge open source support, with many libraries and tools that are available to make developers life easier.
5. Several mobile phones already use Java ME, so Java is known in the mobile industry and the engineering industry.
6. Also, Android as an operating system runs on many different hardware platforms including smart TVs, Android wear etc. Given that almost all VMs JIT compile down to native code, raw code speed is often comparable with native speed. A lot of delays attributed to higher-level languages are less to do with the VM overhead than other factors (a complex object runtime, ‘safety’ checking memory access by doing bounds checking, etc.).
7. Java allows developers to create sandbox applications, and create a better security model so that one bad App can’t take down your entire OS.

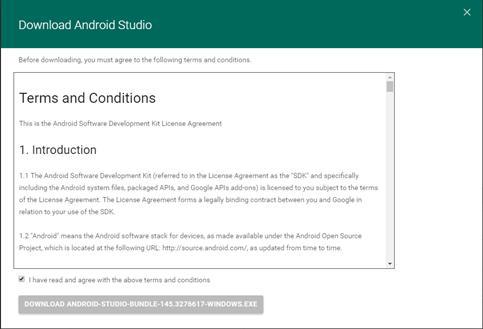
**Install Android Studio**

Following are the steps to install Android Studio on your system

1. Open the following link to download Android Studio <https://developer.android.com/studio/index.html>

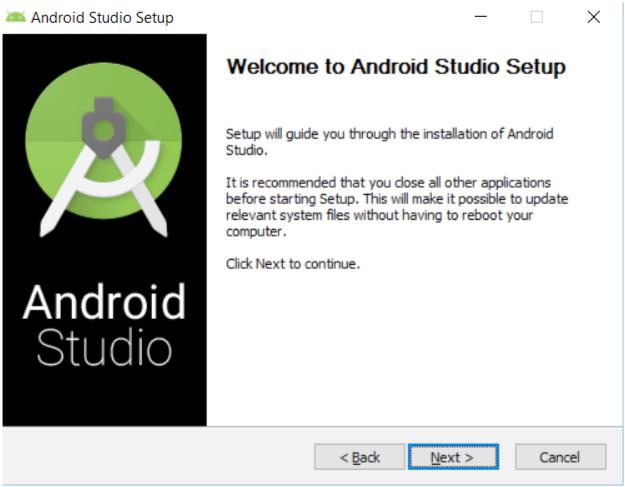


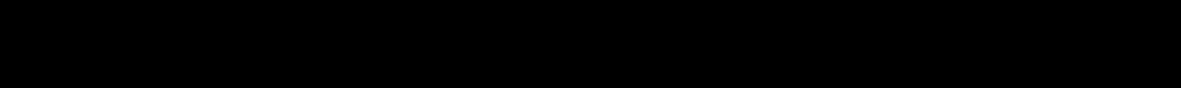
2. Click the “**Download Android Studio**” button.

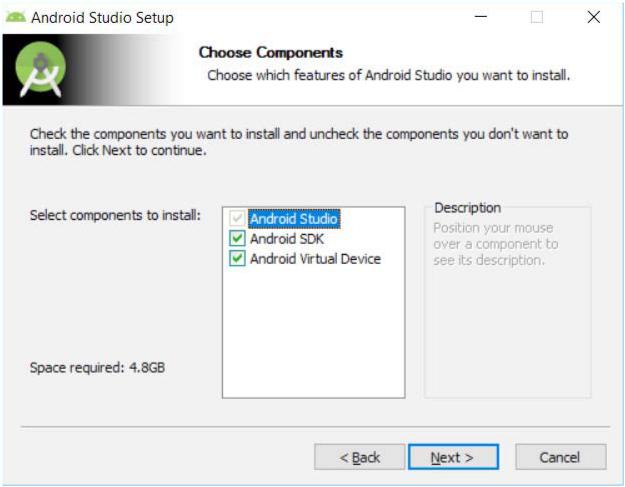


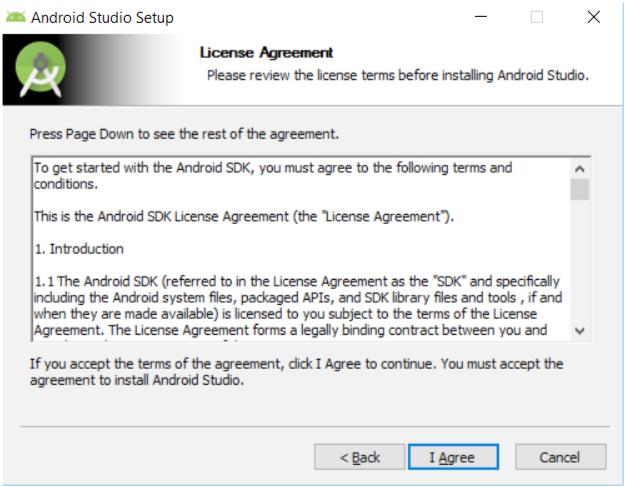
3. Accept the terms and conditions then click on “**Download Android Studio for Window**” buttons. This will start download of an executable file called android-studio-bundle-xxx.xxxxxxx.exe, where xxx.xxxxxxx refers the build number.

1. Run the Windows installer file to start the installation wizard.

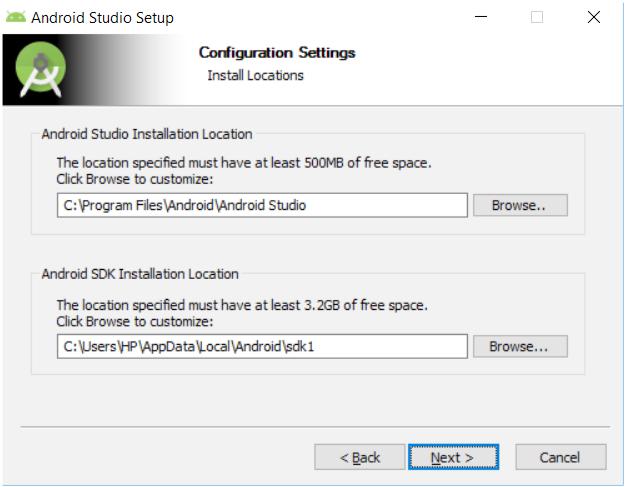


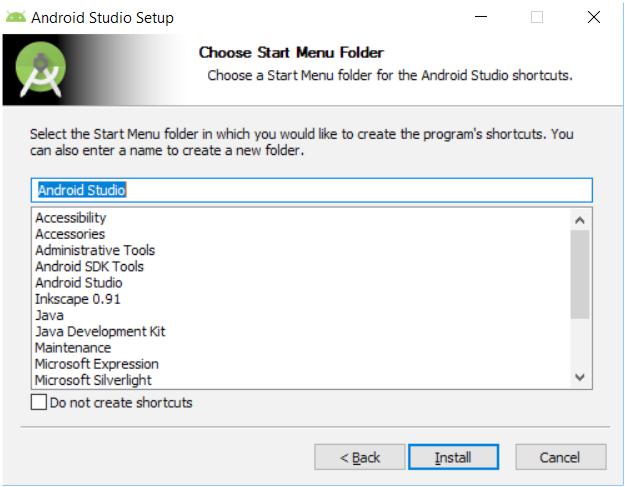
**Note:** Make sure that you have Java Development Kit (JDK) 6 or higher installed on your machine to run Android studio. Android Studio now comes bundled with Open JDK 8. Existing projects still use the JDK specified File > Project Structure > SDK Location. You can switch to use the new bundled JDK by clicking File > Project Structure > SDK Location and checking the Use embedded JDK checkbox.

1. Select the components you have to install and click, “**Next**”.
2. Click “**I Agree**” to move to next step.



1. Next, you should specify the location of the Android SDK folder if you already have the SDK installed on your machine. This is an important step to follow if you want to use the already installed Android SDK tools, platforms, and system images etc. instead of using the single platform and system image that come with Android Studio installation bundle.



1. Next, click “**Install**” to start the installation process.
2. Once installation is complete. Open the Android Studio and it will show following loading screen.



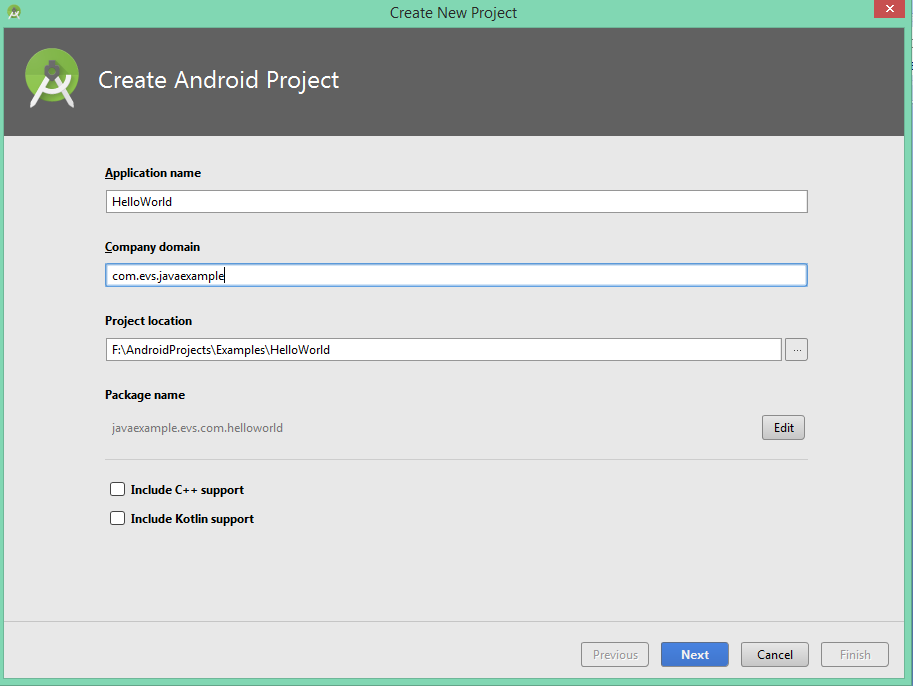
Now, Android Studio is installed on your machine and ready to be used for Android Development.

**Creating an Android Project (Java Project) Using Android Studio**

It’s important to know that Android studio isn’t built for Java development; it is specialized for building Android applications, so you will not be able to create “New Java Project” in Android Studio. Instead, you will be creating an Android project containing a Java library by which you will be testing your Java Code.

**1.** Open Android Studio

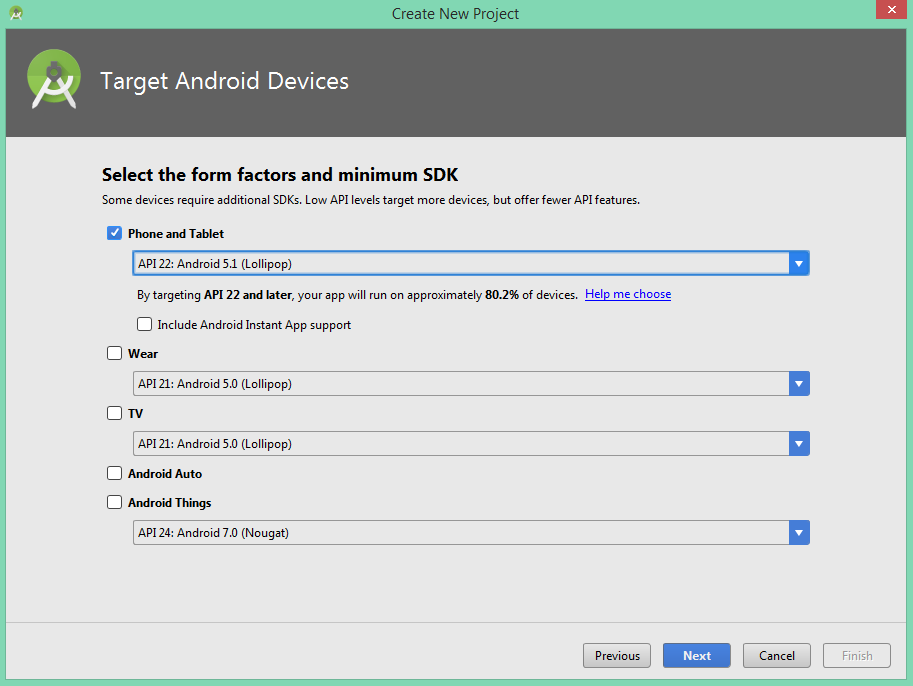
**2.** To create an Android project Click: “**Start a new Android Studio project**”, to get the following dialog box:



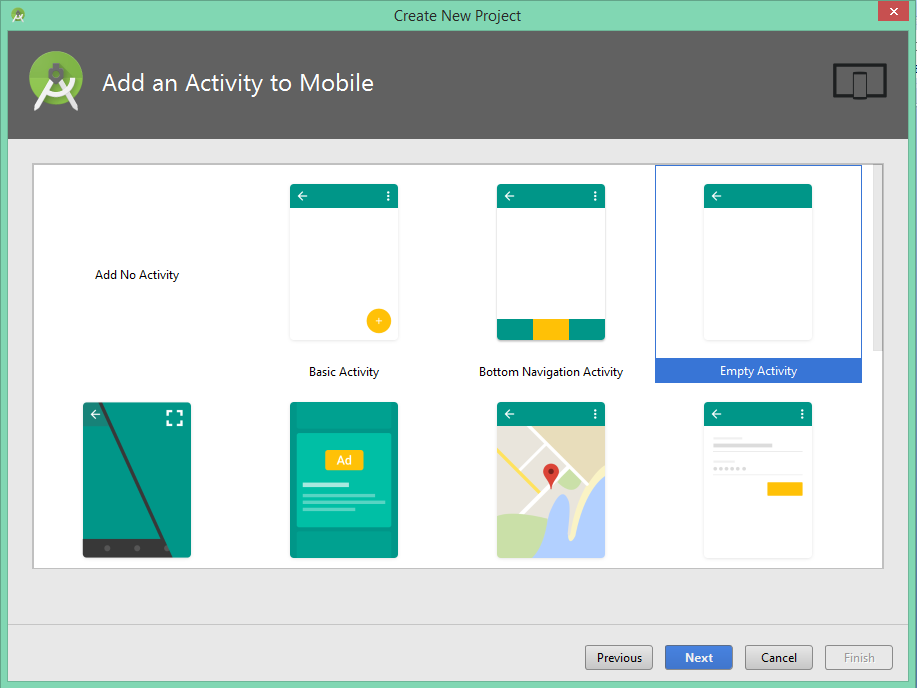
**3.** Write the Application Name: HelloWorld, Company Name here consists of three words separated by dot and in this example project write java.androidatc.com and the Project location you should select where the project will be saved, then click “**Next**”.

**Note:** The package name uniquely identifies the app on the device it is also unique in the Google Play store. This means that once you have published an app with this package name, you can never change it; doing so would cause your app to be treated as a brand new app, and existing users of your app will not see the newly packaged app as an update

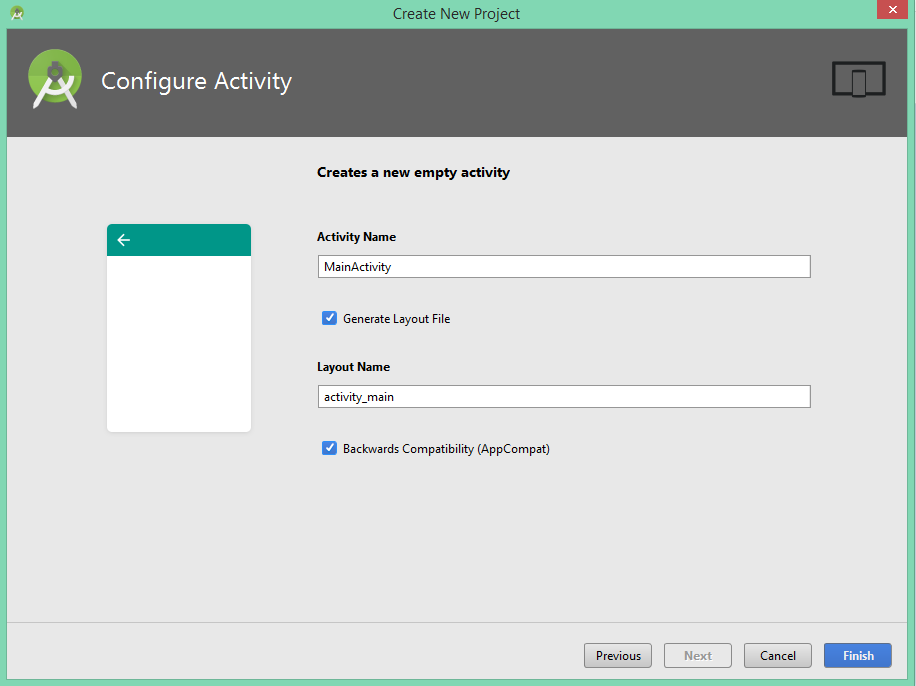
**4.** On the next dialog box, there are some details about the description of the Android device which this application will run on. You will learn about this dialog box in the Android Application Development course. Keep the default “**Next**”.



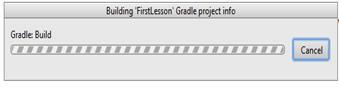
**5.** In the following dialog box, keep the default configuration “**Blank Activity**” and Click “**Next**”



**6.** In the following dialog box, keep the default configuration. This configuration will be explained in details in the Android Application Development course- click **Finish**.

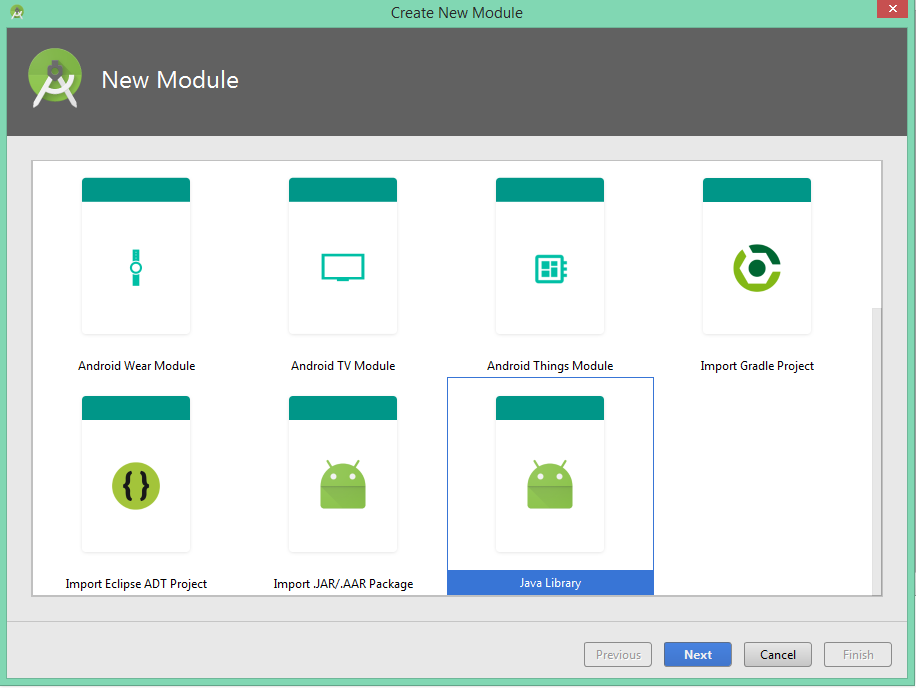


**7.** After clicking the Finish button, creating the application process starts and you will see the following process:

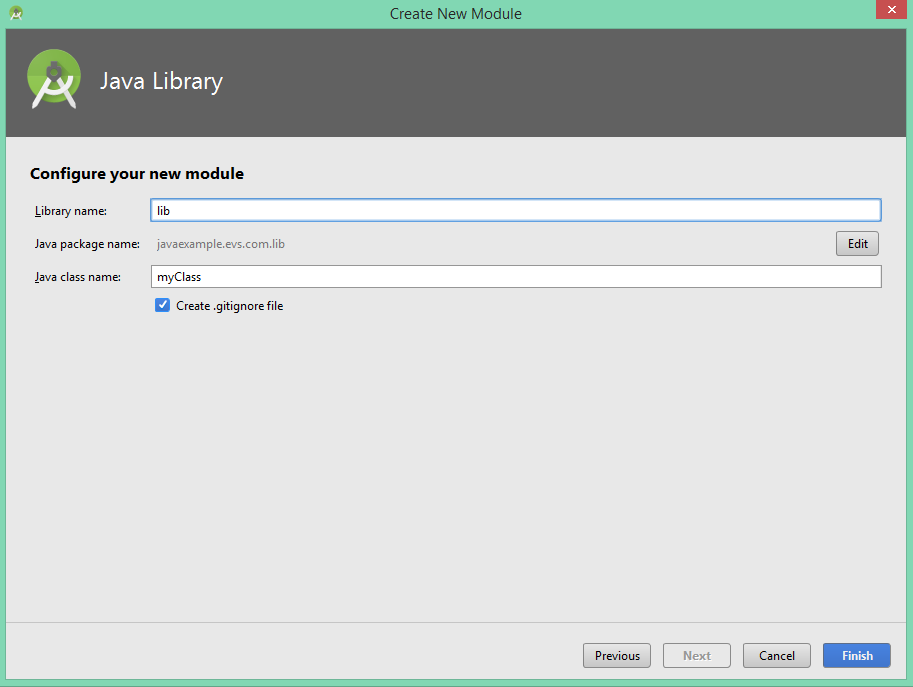


**8.** To create Java Library, click on File menu  **New**  **New Module**.

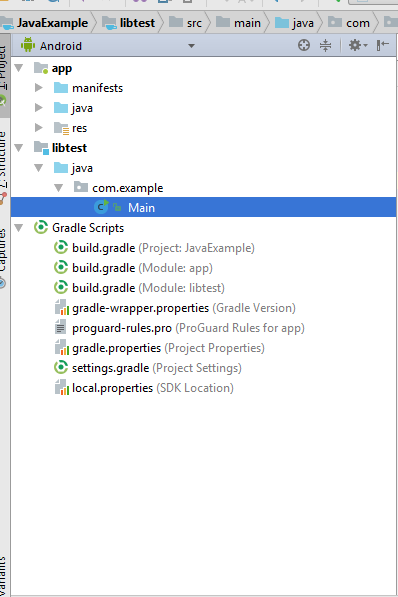
**9.** in the image below, we will get the below dialog box. Select “**Java Library**” and click “**Next”**.



**10.** In the following dialog box, enter the “**Library name**”: libTest and the “**Java class name**”: Main and then click **Finish**.



You will get the following illustrator on the left side of the Android Studio:



In the left side, there are three main elements where the solution explorer resides:

* **App**
* **Libtest (Library Name)**
* **Gradle Scripts**

**Note**: You will be dealing with only the Libtest module, where you will be writing the Java Code.

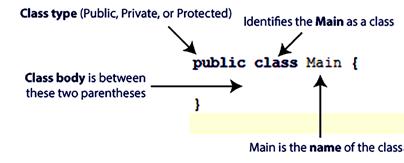
**Writing a Java Program**

Java program or Java project consists of group of classes; each class will achieve part of a Java program.

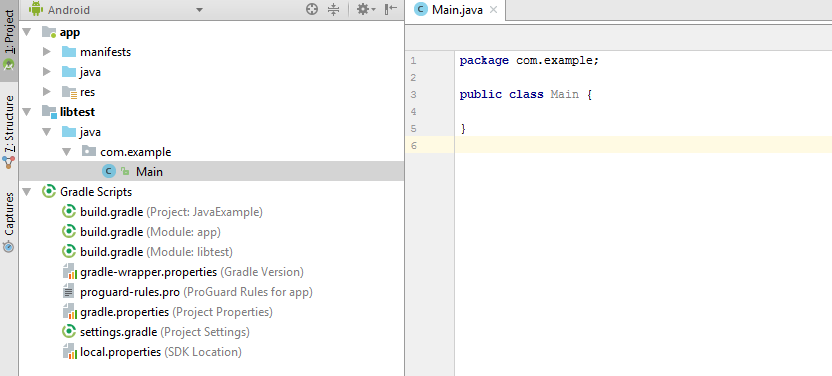
At the start of each project you should start with write class, to create a class you must consider the following three items:

1. Class type (Public, Private, or Protected).
2. Write “class” keyword to declare the class.
3. Next to “class” keyword is the name of your java class.

The following image may display what I mean more:



The following screenshot shows the Main class you have created.

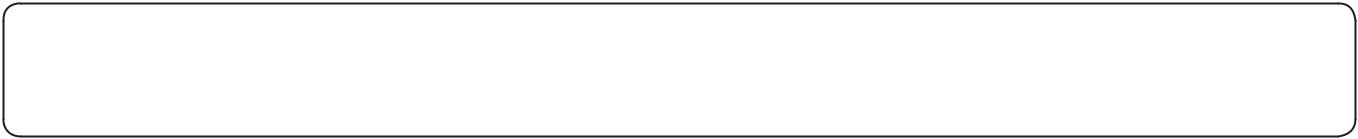


Each Java project consists of group of classes, and there are three types of classes:

1. **Public** is the default class type and it exposes the class to other classes outside the package which means any class can refer to the field or call the method of the public class.
2. **Private** hides from other classes within the package, which means that only the current class will have access to the field and methods.
3. **Protected** is a version of public restricted only to subclasses, which means that only the current class and subclasses (and sometimes also same-package classes) of the class will have access to the field or method of the protected class.

**Note** Before starting Java with Android Studio keep in mind the following:

1. The red marks in Android Studio refer to compile-time errors in your Java code. A compile-time error (also known as a compiler error) is an error that prevents the computer from translating your code.
2. Java is case-sensitive, which means that “**system.out.printLn**” isn’t the same as “**System.Out.Println”**

Java class starts with the following **function**:

**Public static void** main (String args[])

The following table displays what’s the meaning of each part:

|  |  |  |
| --- | --- | --- |
| **Public** | means that main ( ) can be called from anywhere. |  |
| **Static** | means that main () doest belong to the specific object. |  |
| **Void** | means that main( ) returns no value |  |
| **Main** | main is the name of a function. Main ( ) is special because it is the start |  |
| of the program. |  |
|  |  |
| **String** | means the data type |  |
| **Args** | args is the argument passed to the function. “args” is not special; you |  |
| could name it anything else and the program would work the same. |  |
|  |  |

**Java Methods**

If you run the above code it will not give you any result, you have to add the Java configuration, which will help building a Java program by adding methods.

A Java method is a collection of statements that are grouped together to perform an operation.

When you call the “**System.out.println()**” method, for example, the system actually executes several statements in order to display a message on the console.

**Print out on screen method:**

The following example will show how you can use the “**System.out.println()**” method to print out whatever is written between the two parentheses:

**public class** Main {

**public static void** main (String args[]){

System.out.println (“Welcome to Android Development Course”);

}}

When you will run the above class you will get “**Welcome to Android Development Course**” as output on the screen.

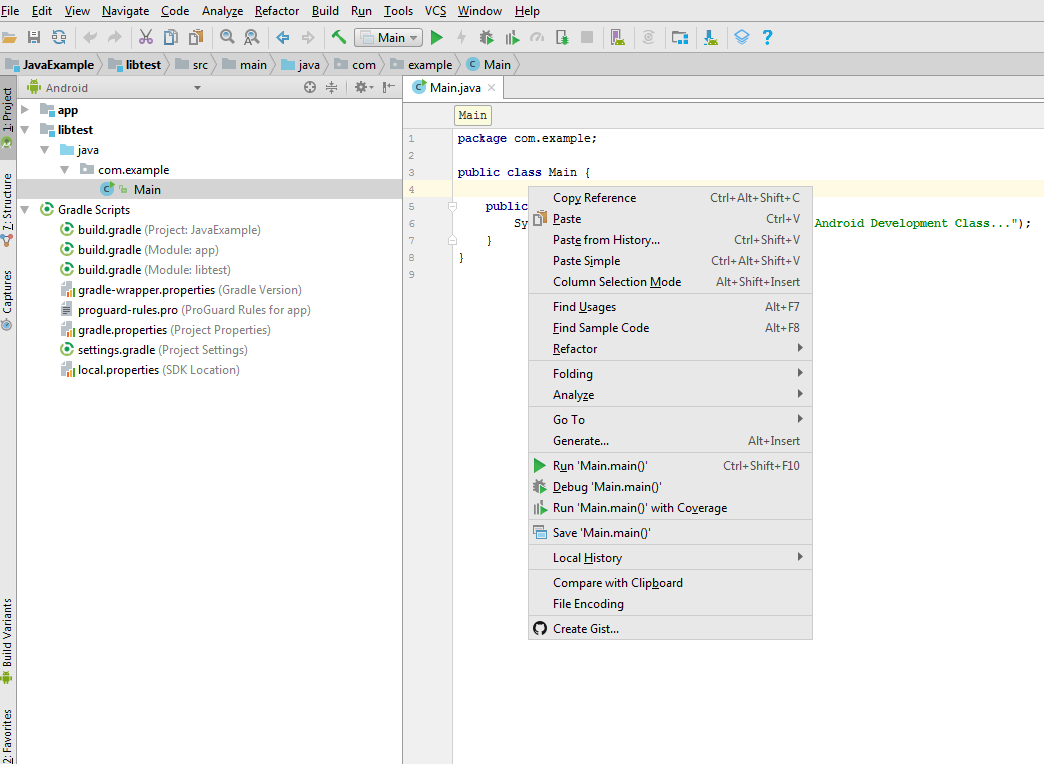
**Running a Java Program**

You can run a Java class by clicking the run button  on the Android Studio tool bar.



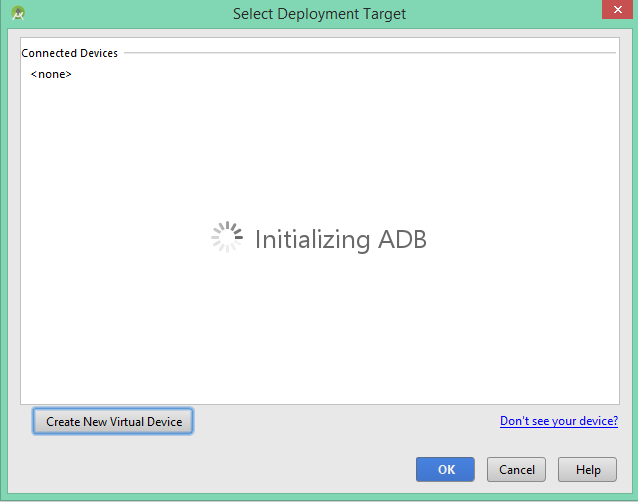
Or by press **Shift + F10** keys.

Also, you can run the class by right clicking anywhere on the class and then select runs on the shortcut menu as the following figure illustrates:



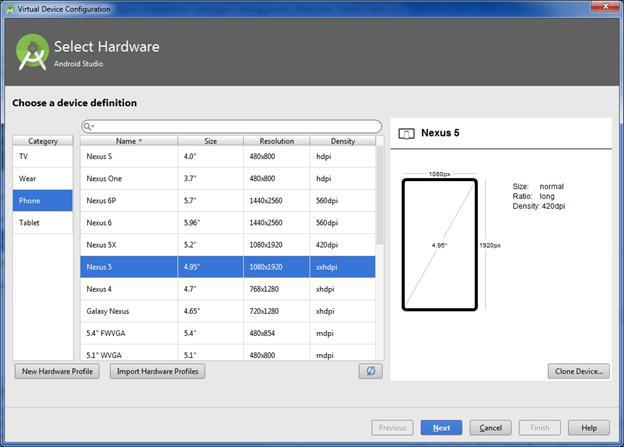
When you click Run option first time only, you will get the following dialog box which is asking you about the device details that the Java code will run on. The Java code in Android Studio can be tested on phone, tablet, wear or TV.

Click the wizard button which is beside the **Android virtual device**

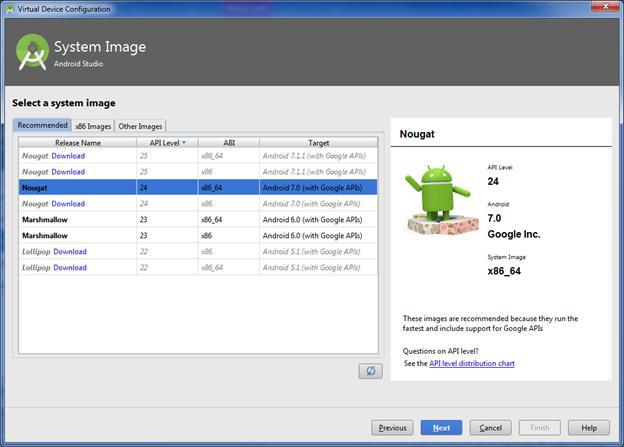


Then, click **“Create Virtual Device...”** button.

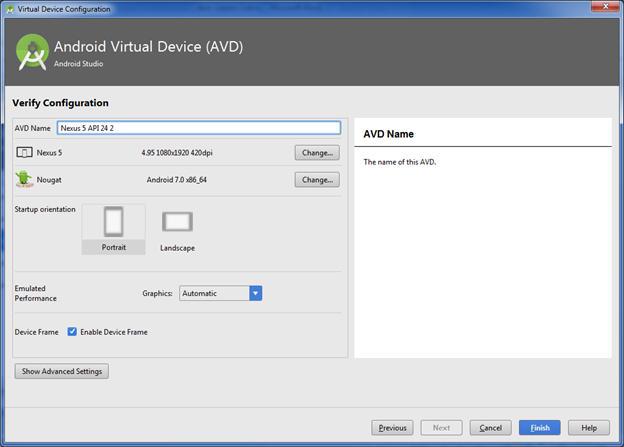
Select **“Nexus 5”** under **“Phone”** category as displayed on the hardware device, then click **Next.**

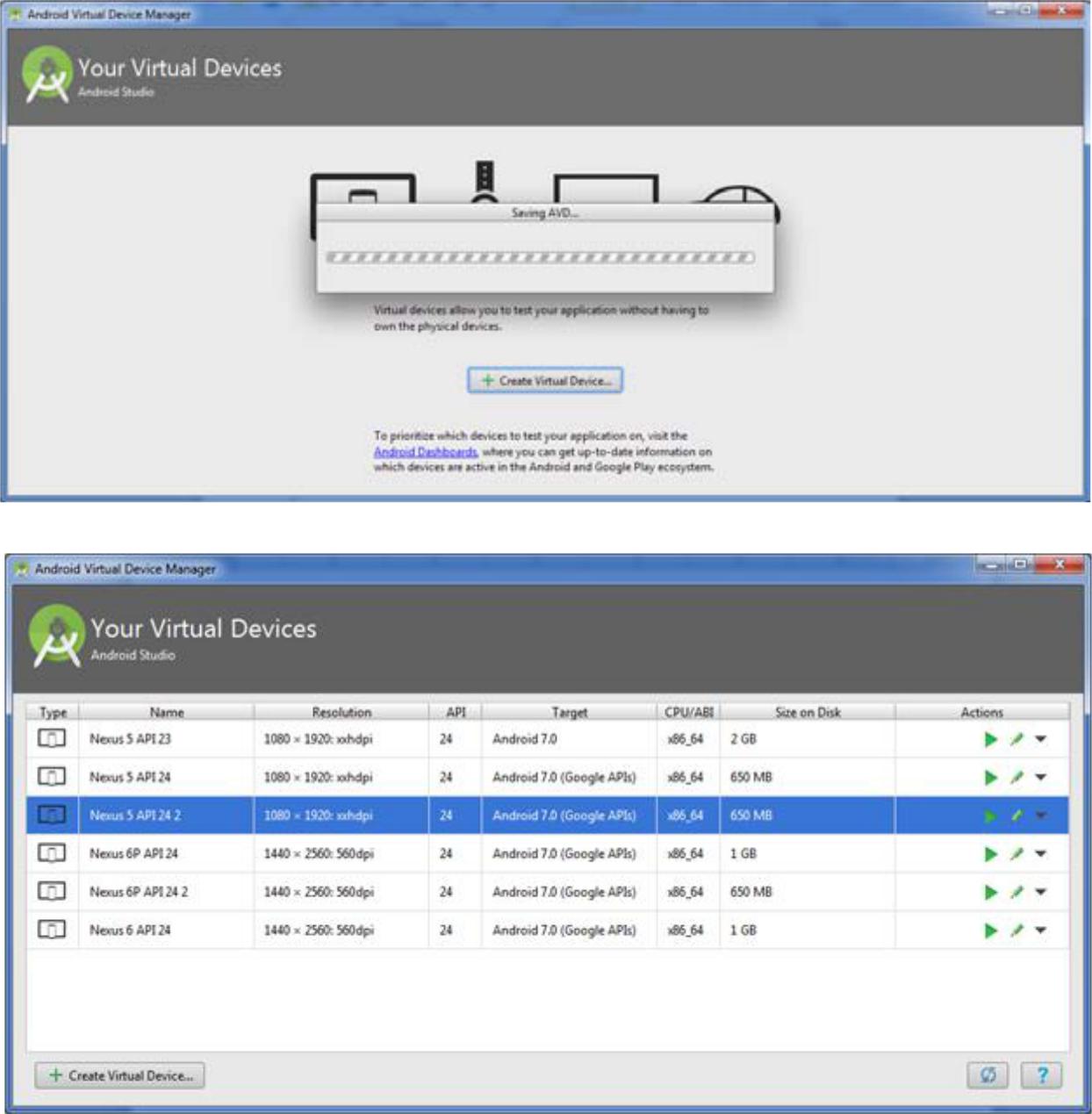


In the following screen, click “**Next”**

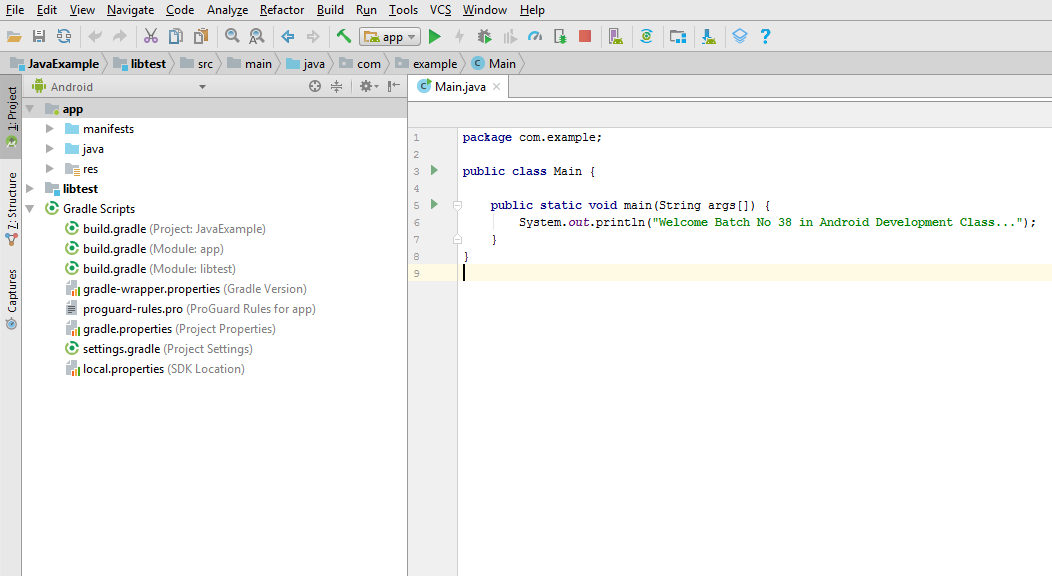


Keep the default configuration and click Finish in the following screen:

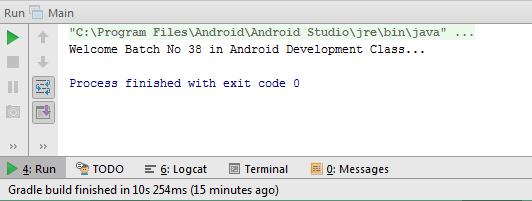


You will get the following screen:

Now when you will run the below class code again, it will ask again about the virtual device, this time select **“Nexus 5 API 24”** as Android virtual device and click “Ok” to see the output of your code.



You will see the following output, where you will get the “Hello Android ATC” text printed. This text can be anything you wrote above between parentheses of **System.out.println**.



Add more text using the System.out.print as shown below:

public class Main {

public static void main ( String[] args ) {

System.out.print(“Welcome Batch No 38 in Android Development Class…”);

System.out.print(“Java Fundamentals for Android™ Development”);

System.out.print(“Android™ Application Development”);

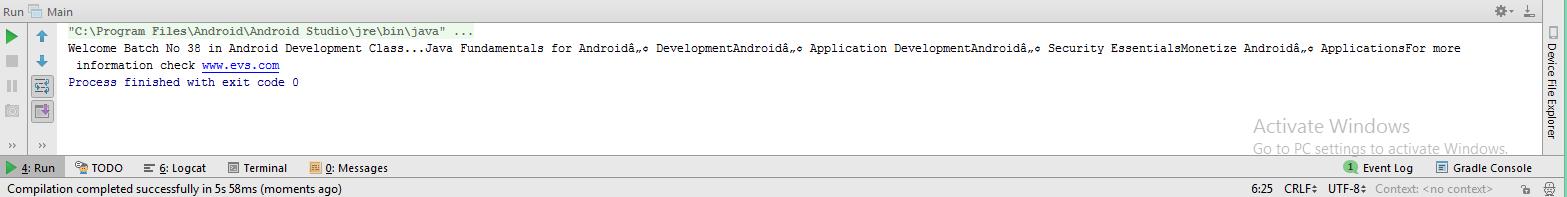
System.out.print(“Android™ Security Essentials”);

System.out.print(“Monetize Android™ Applications”);

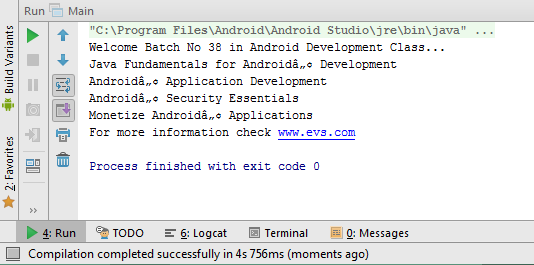
System.out.print(“For more information check www.evs.com”);

}}

When you run the Java program you will get the following result:



All the output of the **System.out.print** methods will be on the same line; however, if you replace **System.out.print** method with **System.out.println** in the previous code, we will get the following result:



**Write a Comment**

We have two ways to write the Java comment: you can define line comments or block comments.

**Line comments:**

Any line starting with double forward slash the Java compiler will consider it as comment, which means that this part will not run or appear to those who use this application, it will remain internal. Comments are used to write short description about different parts of the Java program.

public class Main {

public static void main (String args[]){

// this is my first line comment

System.out.println(“Weclome Batch No 38 in Android Class”);

}}

**Block comments:**

It is like a line comment but it includes more than one line and it starts with /\* and ends with \*/. This is used to add multiple lines as comments in the code without adding // in start of each line.

You can comment or uncomment any line or multiple lines of code in addition to adding description about code.

**Java Variables and Their Data Type**

A Java variable is a piece of memory that can contain a data value. Variables are typically used to store information which your Java program needs to do its job. A variable thus has a data type. All variables in Java must be declared before they can be used

**For example:**

Int x=1;

Doing so tells your program that a field (variable) named “x” exists, hold numerical data, having an initial value of “1” and the data type of this field is integer. A variable’s data type determines the values it may contain, plus the operations that may be performed on it. We have two categories of data types as follow:

1. Primitive data types.
2. Composite data types.

A primitive data type uses a small amount of memory to represent a single item of data. It is preserved by the programming language and reserved keywords are used for naming the primitive data types. In addition to int, the Java programming language supports seven other data types. The following table displays the Primitive data types:

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Description** | **Default** |  |
| **Type** |  | **Value** |  |
| **byte** | The byte data type is an 8-bit signed integer. | 0 |  |
| **short** | The short data type is a 16-bit signed integer. | 0 |  |
| **int** | The integer data type is a 32-bit signed integer. It has a | 0 |  |
| maximum value of 2,147,483,647. |  |
|  |  |  |
| **long** | The long data type is a 64-bit signed integer. | 0L |  |
| **float** | The float data type is a single-precision 32-bit floating point. | 0.0f |  |
| **double** | The double data type is a double-precision 64-bit floating point. | 0.0d |  |
| **Boolean** | The Boolean data type has only two possible values: true and false | ‘\u0000’ |  |
| **char** | The char data type is a single 16-bit Unicode character. | null |  |

Composite data types will be explained in the next lessons.

**Java has the following rules and conventions for naming variables:**

* Variable names are case-sensitive and white space is not permitted.
* Beginning with a letter, the dollar sign “$”, or the underscore character “\_” is allowed
* Subsequent characters may be letters, digits, dollar signs, or underscore characters.
* By convention, you should name your variables using “camel case”, i.e. if the name consists of only one word, it is all lowercase letters. If it consists of more than one word, the first letter of each subsequent word is capitalized.
* Also by convention, constants are all capitalized and contain underscore.

The following is an example which will show how to declare two integer variables (a & b) and declare the another variable c which will be the sum of a & b

public class Main {

public static void main (String args[]){

int a=1;

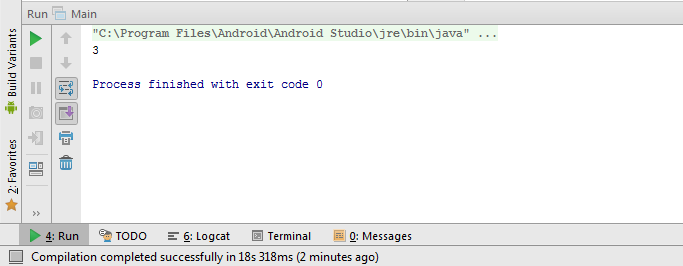
int b=2;

int c=a+b;

System.out.println(c);

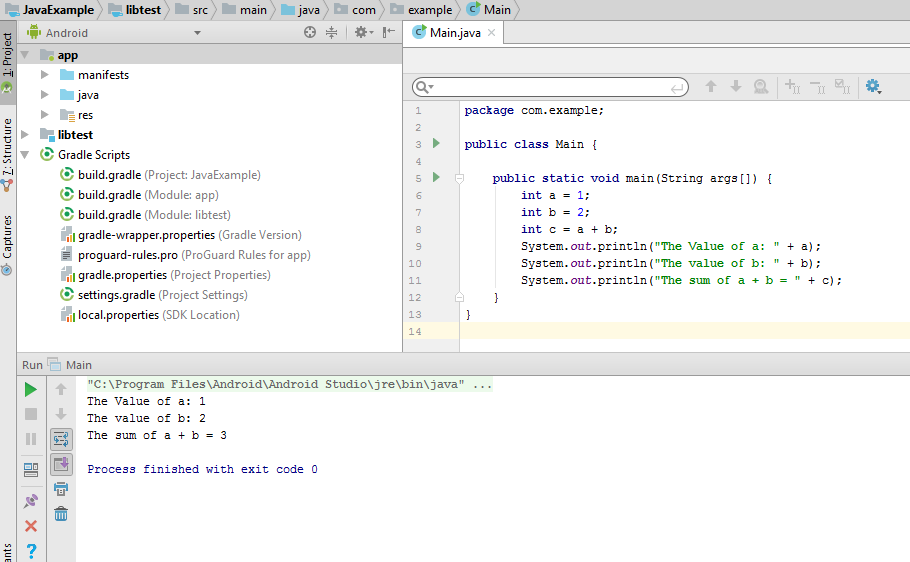
}}

When you will run this Java program, you will get the following result:

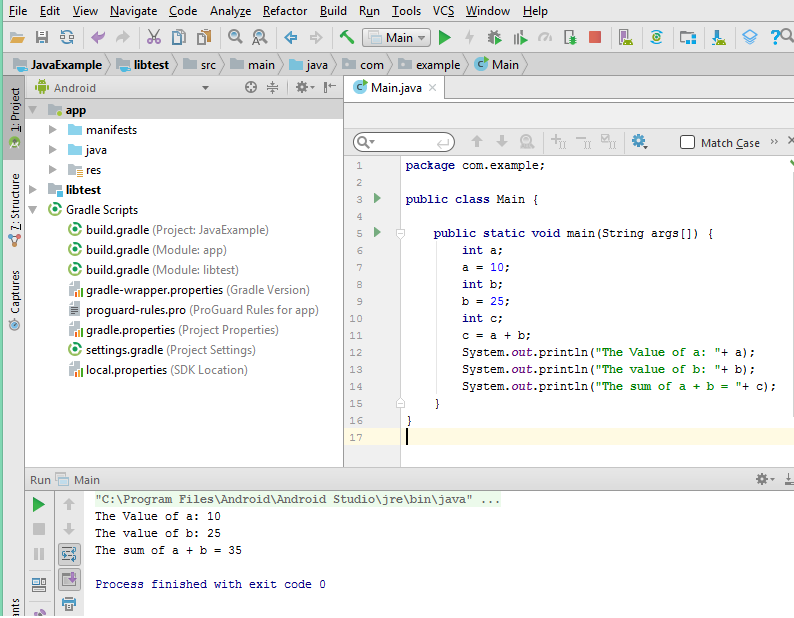


We can modify the previous code by modifying the print method as follows:

The output of run the previous code will be as follows:



The following code is another way to write the same previous Java code and it will give the same previous run result:



Unlike the previous examples, in the code above you declared the variables and then initialized them in the next line, whereas in the previous examples you declared and initialized the variables in the same line.

One of the main points to know while dealing with variables and objects is to know their life cycle.

The life cycle of an object starts with the declaration, in this phase you will be allocating the necessary space, second phase is to initialize your variable. After that, the object will reside in the memory until you destroy your object or when it reaches the end of its life cycle. Your program consists of many scopes and your object will remain accessible within its scope.

**Assignment statement and Assignment Expressions**

An assignment statement in Java uses the assignment operator (=) to assign the result of an expression to variable. In its simplest form, you can find below how the assignment is done in Java:

Variable = expression;

Example: int a = (b \* c) / 4;

A compound assignment operator is an operator that performs a calculation and an assignment at the same time. All Java binary arithmetic operators (that is, the ones that work on two operands) have equivalent compound assignment operators:

**+=** Addition and assignment

**-=**  Subtraction and assignment

**\*=** Multiplication and assignment

**/=** Division and assignment

**%=** Remainder and assignment

**For example**, the statement

a += 10; is equivalent to a = a +10;

More about these operators will be explained later

**The following are more examples about the data types Java:**

**Boolean data type:**

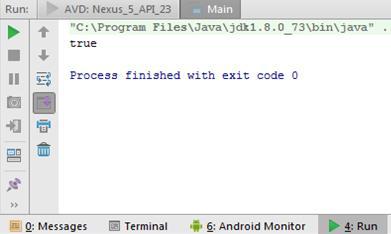
A boolean data type can hold only two values- true or false. Example:

Public class Main {

public static void main ( String[] args ) { boolean a=true; boolean b=false;

System.out.println(a);

}}

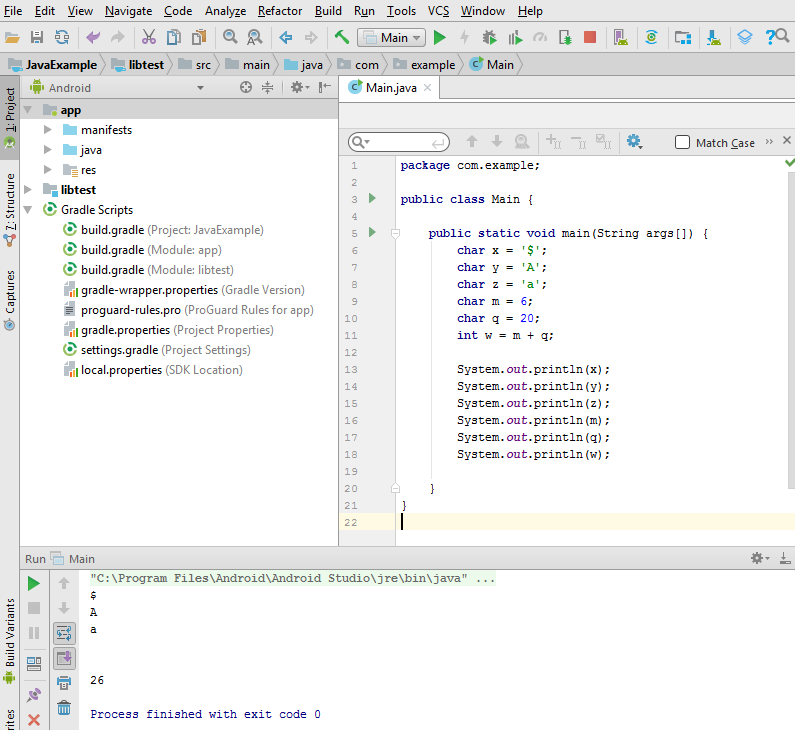
The run result of this code is as follows:

**char data type:**

The char data type is used to store a single character such as ‘a’, ‘Z’, ‘9’, ‘&’, ‘$’ and so on. Characters may be letters, digits or special symbols. A character can also be a non-graphic symbol such as a new line or a tab. Characters are always enclosed in single quotes.

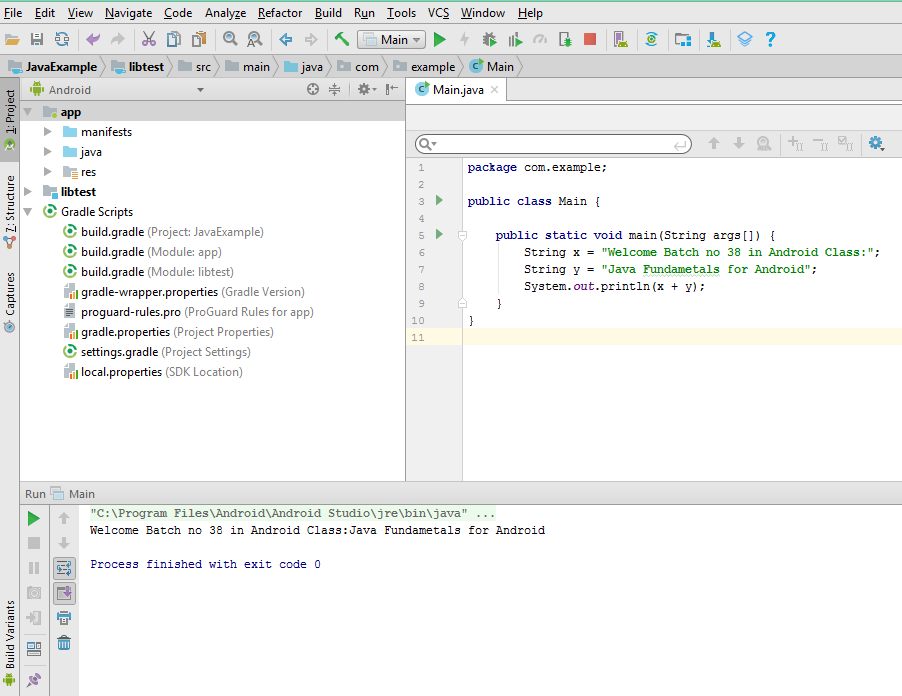
If you wish to store the value ‘$’ in a variable x, you would write the following statement:

char x = ‘$’; Example:



**String data type:**

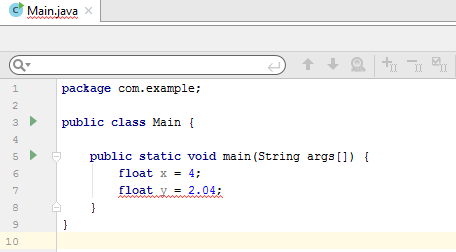
String data type is used to store words or sentences. The following is the output of the code written above.



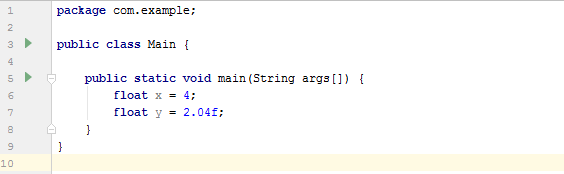
**Float data type:**

The type float specifies a single-precision value that uses 32 bits of storage. Single precision is faster on some processors and takes half as much space as double precision, but will become imprecise when the values are either very large or very small. Variables of type float are useful when you need a fractional component, but don’t require a large degree of precision. For example, float can be useful when representing dollars and cents.

The following image shows that when you declare variable x as float without fractional components you will not get any compile time error, however; if you declare another variable with fractional component you will get red underline as shown in the image below, that can be solved by adding character “f “ at the end of the number.



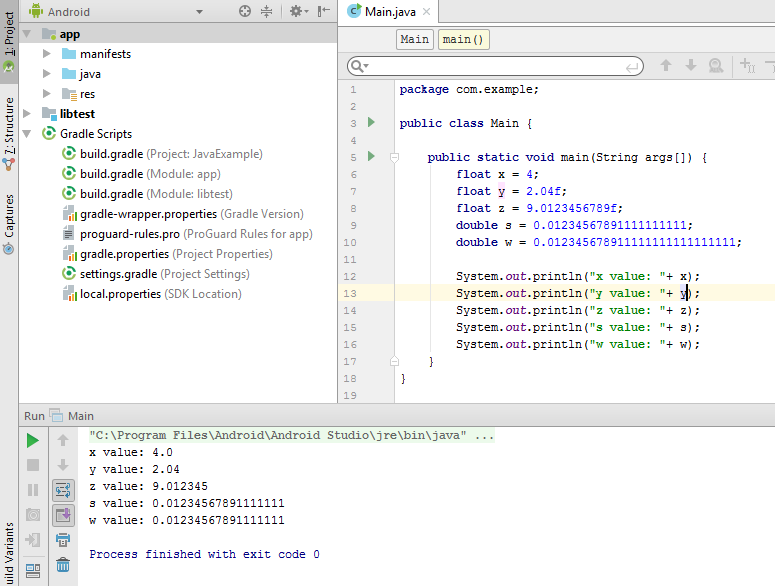
The next screenshot displays that red underline has been removed after adding the character “f” at the end of the float number:



**Double data type:**

Double precision, as denoted by the double keyword, uses 64 bits to store a value. Double precision is actually faster than single precision on some modern processors that have been optimized for high-speed mathematical calculations. All transcendental math functions, such as sin( ), cos( ), and sqrt( ), return double values. When you need to maintain accuracy over many iterative calculations, or are manipulating large-valued numbers, double is the best choice.

The following example displays how you can declare some variables as float and double data type, and how the Java program will consider the first 32 bits of the float number and first 64 bits of the double number in its output.



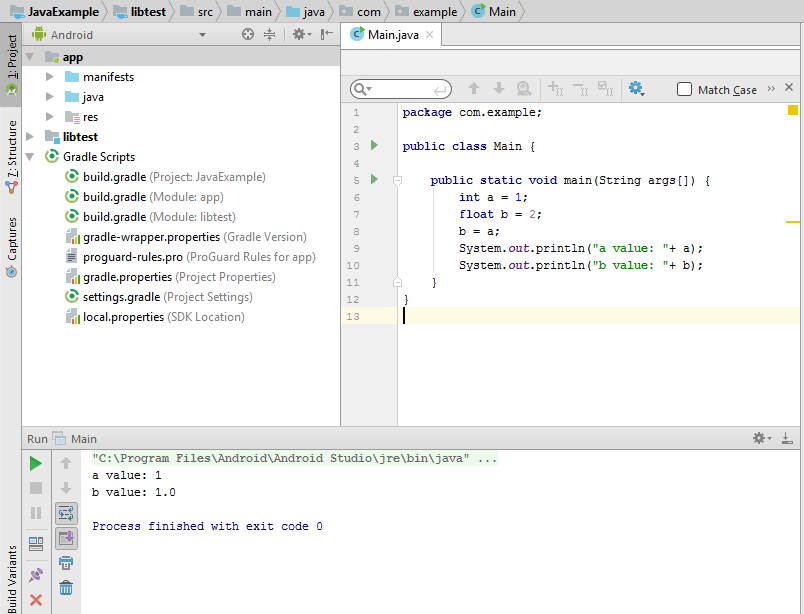
**Numeric type Conversions**

Casting between primitive types enables you to convert the value of one type to another primitive type. It most commonly occurs with the numeric types, and there’s one primitive type (Boolean) that can never be used in a cast. Boolean values must be either true or false and cannot be used in a casting operation.

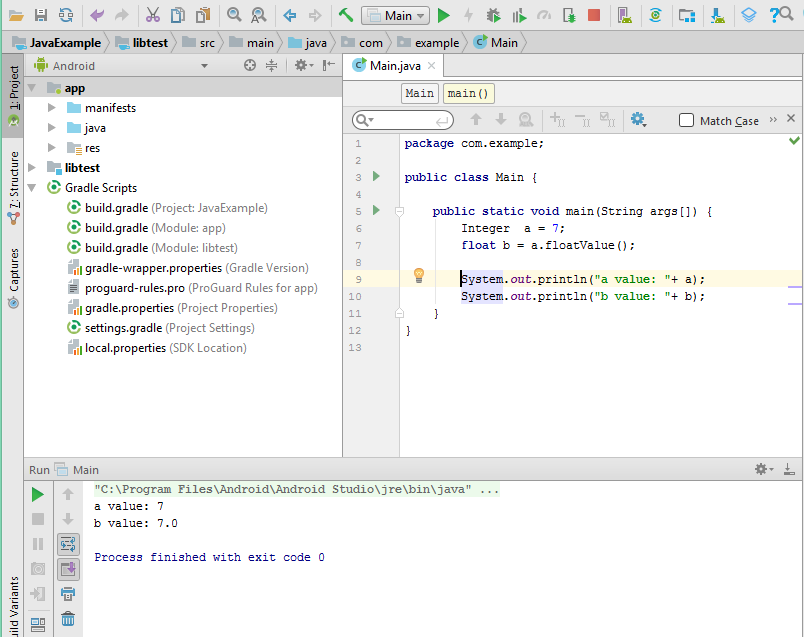
In many casts between primitive types, the destination can hold larger values than the source, so the value is converted easily. An example would be casting a byte into an int. Because a byte holds values from –128 to 127 and an int holds from -2.1 billion to 2.1 billion, there’s more than enough room to cast a byte into an int.

You can often automatically use a byte or a char as an int; you can use an int as a long, an int as a float, or anything as a double. In most cases, because the larger type provides more precision than the smaller, no loss of information occurs as a result. The exception is casting integers to floating-point values; casting an int or a long to a float, or a long to a double, can cause some loss of precision.

The following example displays how you can work with different data types in the same Java program without writing any extra command to convert part of them to another data type:



We can use a Java method to establish relationship between two different data types as is illustrated in the following example:



**Converting Numbers to Strings:**

Sometimes you need to convert a number to a string because you need to operate on the value in its string form. The following is an easy way to convert an integer variable to a string:

The following screenshot displays how the Android Studio as compiler will allow us to convert the value of one type to another primitive type:

