JAVA FOR BEGINNERS A COMPLETE GUIDE

By
Salman Abdul Rahim
(Bachelors in Computer Science from BBSUL, Karachi, Pakistan)

INTRODUCTION TO JAVA:

Java is a well-known and a robust object-oriented programming language, developed in the year 1995.It was originated by James Gosling. It's a programming language from which we can develop a wide range of applications that can be executed on a single personal computer as well as on the servers.
This guide focuses the necessary areas and objectives that are mandatory to be learned by the beginners of Java programming language. From the Installation of the Java Development Kit (JDK) to some of its useful and essential packages for developing applications as a beginner, including some interesting topics that you might find awesome and knowledgeable for making a perfect and strong basics of Java Language.

FEATURES OF JAVA:

The following are some features of Java Programming Language:

- **Portability** – Java Programming Language is platform independent and architecture neutral.

- **Robustness –** Java Programming Language is a robust language as it has its own built-in Garbage Collection feature.

- **Multi-Threaded –** Java Programming language has two threads. One for execution of code, and the second for eliminating garbage to save the memory space.

- **Secure –** Java has its own runtime environment namely “Java Runtime Environment” due to which it is more secure**.**

TOPIC#1:

JAVA DEVELOPMENT KIT (JDK)

1.1 What is Java Development Kit (JDK)?

The Java Development Kit (JDK) is a software development environment used for developing Java Applications and Applets. It includes the Java Runtime Environment (JRE) and the Java Virtual Machine (JVM). The JDK allows the developers to create Java programs that can be executed and run by the JVM and JRE.

1.2 Why use Java Development Kit (JDK)?

The JDK contains various components and tools. The uses of JDK may be divided according to its components. Each of these components and tools have their specific use such as, the compilers and debuggers are necessary for developing Java applets and applications, the libraries provided by JRE are essential for developing Java programs, JVM and other components are mandatory to run applets and applications written in the Java programming language. Thus, JDK is a basic tool to program and develop using Java language

1.3 Features of Java Development Kit (JDK)

Java includes a number of features. However, the features of Java are constantly updated time to time. Nowadays the version of Java contains the features given below:

- JEP 309: Dynamic class-file constants
- JEP 318: Epsilon: A no-op garbage collector
- JEP 323: Local-variable syntax for lambda parameters
- JEP 331 Low over-head heap profiling
- JEP 321: HTTP client (standard)
- JEP 332: Transport Layer Security (TLS) 1.3
- JEP 328: Flight recorder

- JEP 333: ZGC: A scalable low latency garbage collector
- Unicode 11.0.0 support

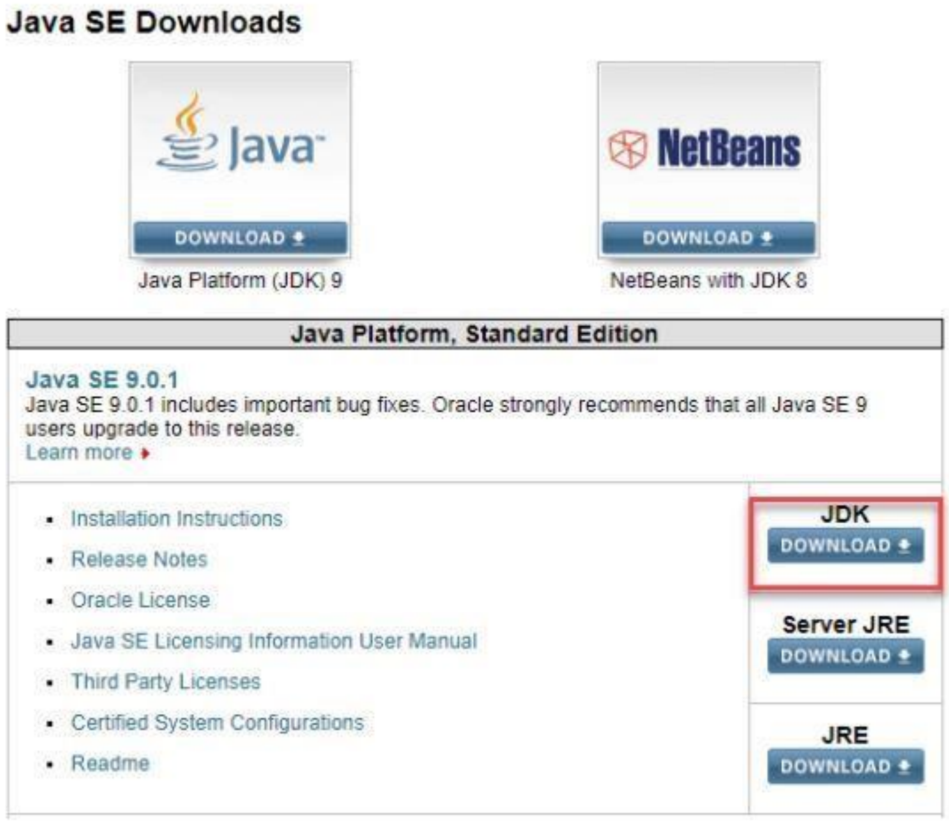
1.4 Installation process of Java Development Kit (JDK)

Following are steps to install Java in Windows:

Step 1) Go to web browser and search for JDK. Click on Download JDK.

For java latest version.

Fig 1.1:



Java SE Downloads

Java Platform (JDK) 9

NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 9.0.1

Java SE 9.0.1 includes important bug fixes. Oracle strongly recommends that all Java SE 9 users upgrade to this release.

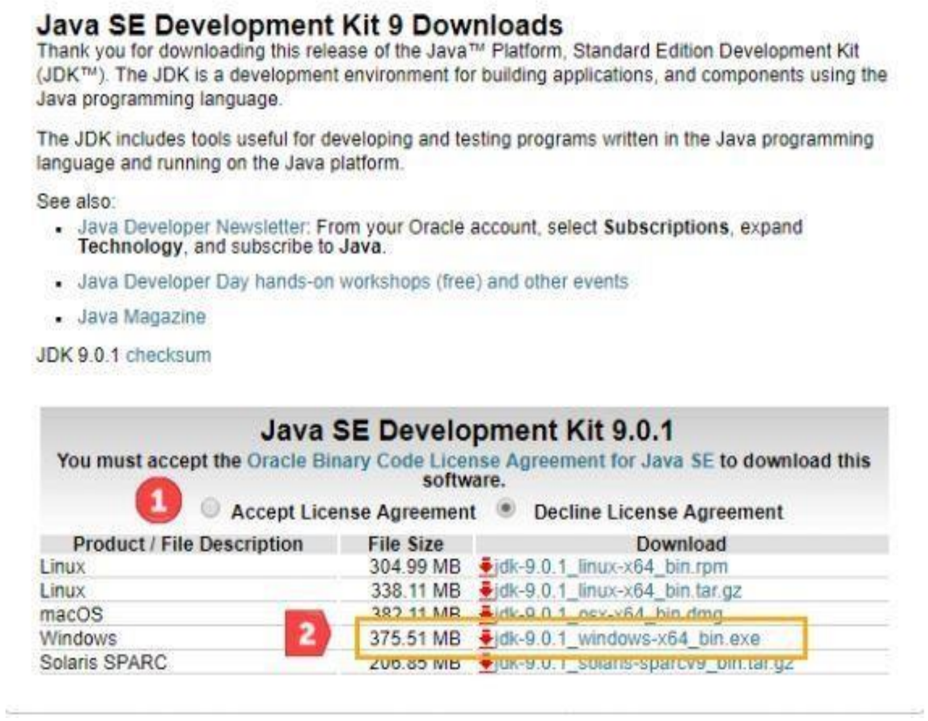
Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
- Third Party Licenses
- Certified System Configurations
- Readme

Step 2) Next,

1. Accept License Agreement
2. Download latest Java JDK for your version (32 or 64 bit) of java for Windows.

<u>Fig 1.2:</u>



Java SE Development Kit 9 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 9.0.1 checksum

Java SE Development Kit 9.0.1

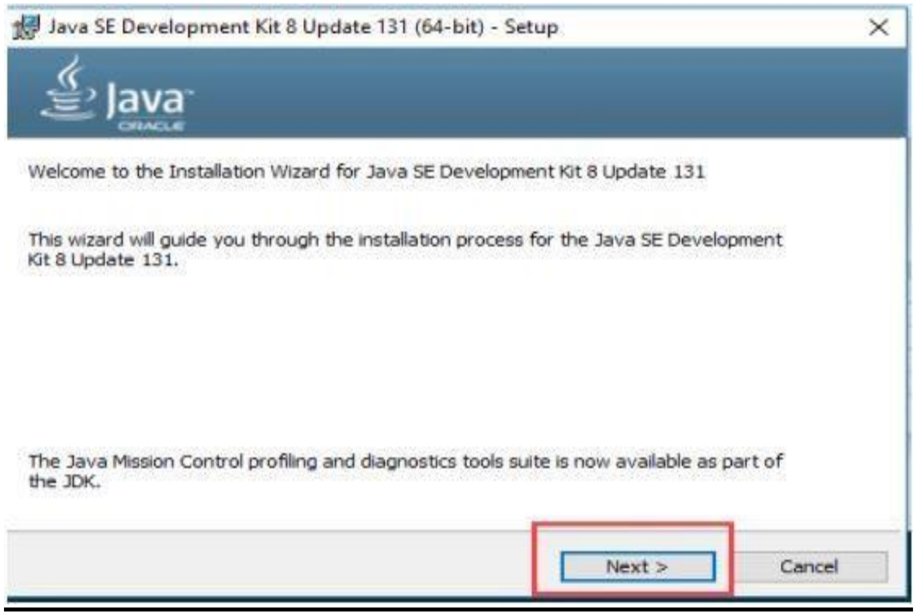
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement [✓ ● Decline License Agreement]

Product / File Description	File Size	Download
Linux	304.99 MB	jdk-9.0.1_linux-x64_bin.rpm
Linux	338.11 MB	jdk-9.0.1_linux-x64_bin.tar.gz
macOS	382.11 MB	jdk-9.0.1_osx-x64_bin.dmg
Windows	375.51 MB	jdk-9.0.1_windows-x64_bin.exe
Solaris SPARC	206.85 MB	jdk-9.0.1_solaris-sparcv9_bin.tar.gz

Step 3) Once the download is complete, run the exe for install JDK. Click Next.

<u>Fig 1.3:</u>



Java SE Development Kit 8 Update 131 (64-bit) - Setup

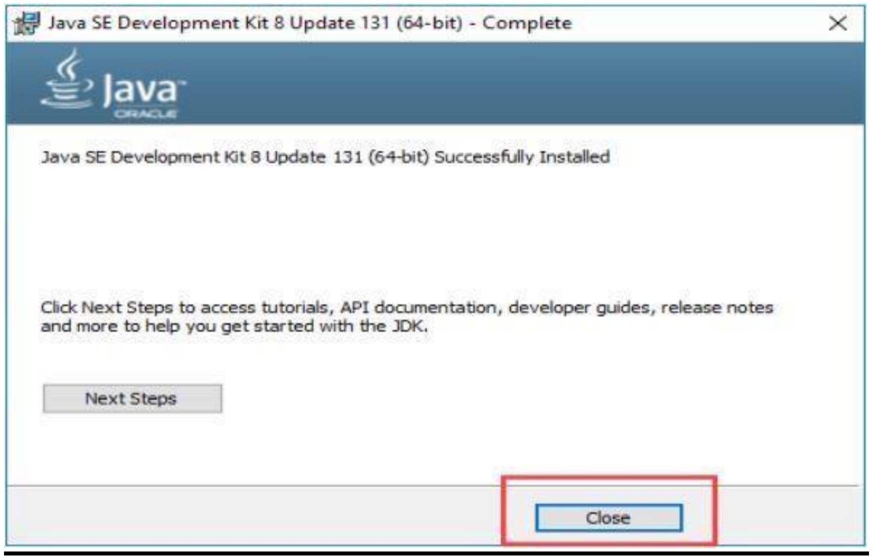
Welcome to the Installation Wizard for Java SE Development Kit 8 Update 131

This wizard will guide you through the installation process for the Java SE Development Kit 8 Update 131.

The Java Mission Control profiling and diagnostics tools suite is now available as part of the JDK.

Step 4) Once installation is complete click Close.

<u>Fig 1.4:</u>



Java SE Development Kit 8 Update 131 (64-bit) - Complete

Java SE Development Kit 8 Update 131 (64-bit) Successfully Installed

Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

TOPIC#02:

NETBEANS

2.1 What is Netbeans?

Netbeans is an Integrated Development Environment (IDE). Netbeans allows applications to be developed from a set of modular software components called as modules. Netbeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5 and Javascript.

2.2 Why use NetBeans?

NetBeans IDE is the official IDE for Java. With its editors, code analyzers, and converters, you can quickly and smoothly upgrade your applications to use new Java language constructs, such as lambdas, functional operations, and method references.

2.3 Features of Netbeans

NetBeans have a number of important features that are given as under:

- **Out of the Box -** A key feature of NetBeans is the short time difference between installing it and beginning to create meaningful applications in it. Despite its significant plugin ecosystem, not much is needed to be installed or configured, since everything is available "out of the box" as soon as you start it up.

- **Java Editor** - The language-aware NetBeans editor detects errors while you type and assists you with documentation popups and smart code completion – all with the speed and simplicity of your favorite lightweight text editor. Of course, the Java editor in NetBeans is much more than a text editor – it indents lines, matches words and brackets, and highlights source code syntactically and semantically.

- **Internet of Things** - Directly from NetBeans, you can create, test, debug, deploy, and profile applications that will run on the Raspberry Pi, mobile phones, PDAs, set-top boxes, and other mobile and embedded systems.

- **Profiler** - The NetBeans Profiler provides expert assistance for optimizing your application's speed and memory usage, while making it easier to build reliable and scalable Java SE, JavaFX and Java EE applications.

- **Git and Mercurial** - Without requiring you to install any plugins, NetBeans automatically lets you work easily and intuitively with a wide range of popular versioning systems, specifically Git, Mercurial, and Subversion.

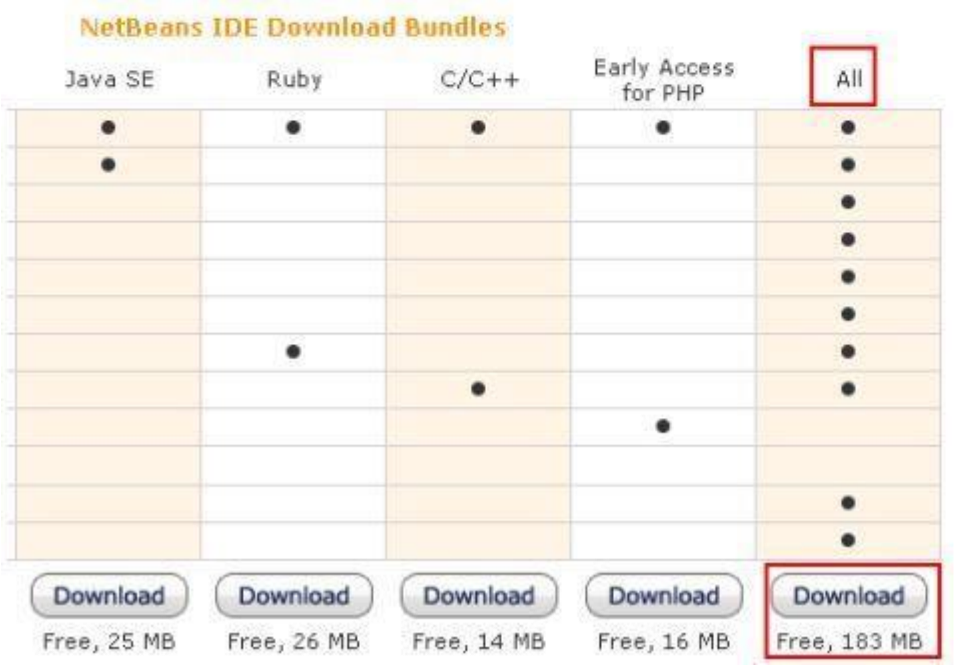
- **Configurability -** The NetBeans workspace can easily be modified. Customize the buttons in the toolbar or drag and reposition tabs in the application frame to suit your individual workflow. Undock tabs and drag them outside the application frame, even onto a different monitor and change keyboard shortcuts to match your own preferences.

2.4 Installation process of Netbeans

The steps to install NetBeans in Windows is given as under:

Step 1) As usual, go toNetbeans.org to download the latest version of NetBeans. We downloaded the full version that is the full bundled 'package'.

<u>Fig 2.1:</u>

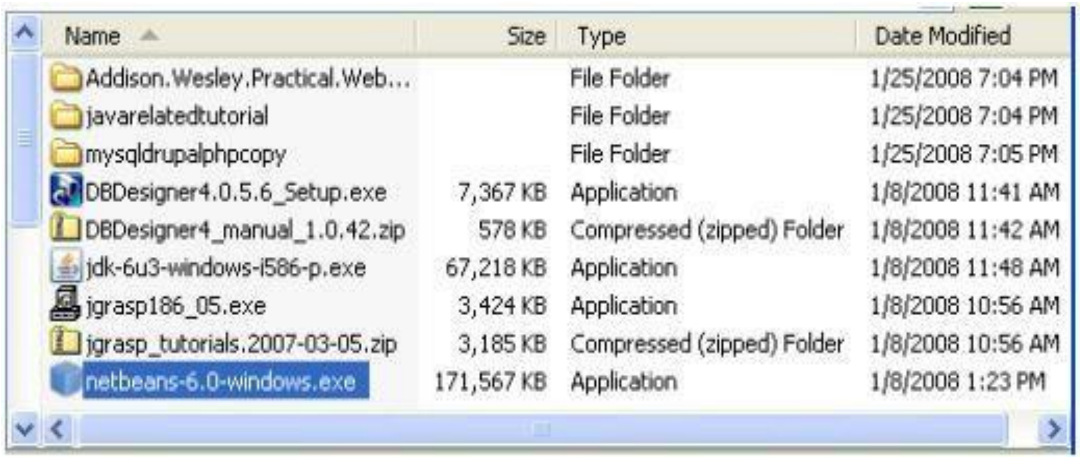


NetBeans IDE Download Bundles

Java SE	Ruby	C/C++	Early Access for PHP	All
●	●	●	●	●
●				●
				●
				●
				●
				●
	●			●
		●		●
			●	
				●
				●
Download	Download	Download	Download	Download
Free, 25 MB	Free, 26 MB	Free, 14 MB	Free, 16 MB	Free, 183 MB

Step 2) Double click the Windows exe self-extracting file.

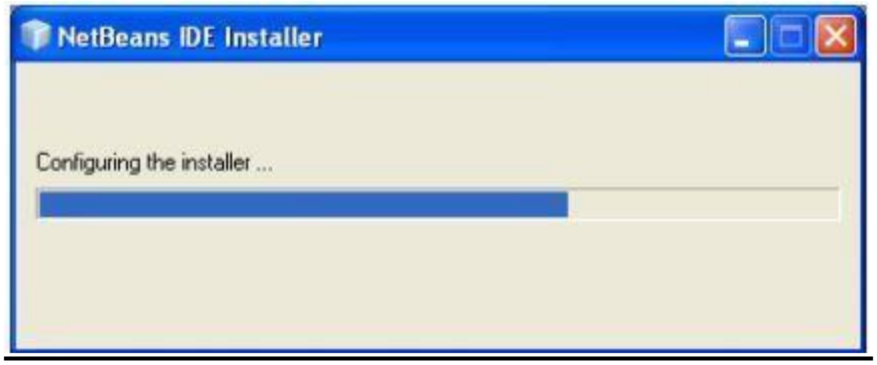
<u>Fig 2.2:</u>



Name	Size	Type	Date Modified
Addison.Wesley.Practical.Web...		File Folder	1/25/2008 7:04 PM
javarelatedtutorial		File Folder	1/25/2008 7:04 PM
mysqldrupalphpcopy		File Folder	1/25/2008 7:05 PM
DBDesigner4.0.5.6_Setup.exe	7,367 KB	Application	1/8/2008 11:41 AM
DBDesigner4_manual_1.0.42.zip	578 KB	Compressed (zipped) Folder	1/8/2008 11:42 AM
jdk-6u3-windows-i586-p.exe	67,218 KB	Application	1/8/2008 11:48 AM
jgrasp186_05.exe	3,424 KB	Application	1/8/2008 10:56 AM
jgrasp_tutorials.2007-03-05.zip	3,185 KB	Compressed (zipped) Folder	1/8/2008 10:56 AM
netbeans-6.0-windows.exe	171,567 KB	Application	1/8/2008 1:23 PM

Step 3) The NetBeans IDE installer will be launched.

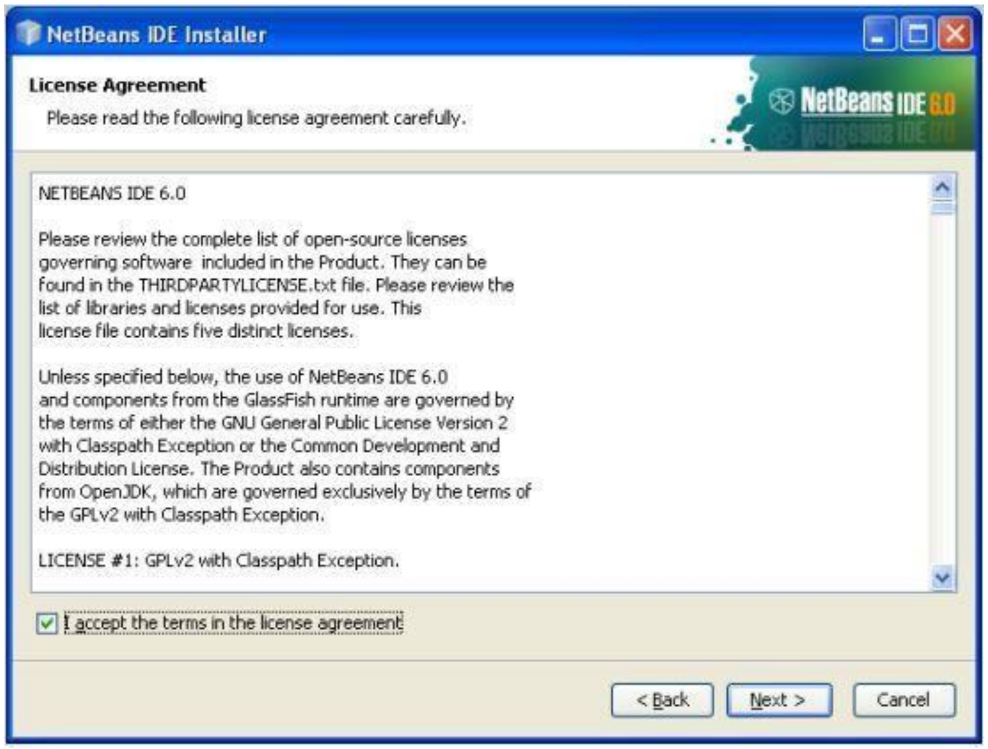
Fig 2.3:



Configuring the installer ...

Step 4) Accept the License Agreement and click Next.

Fig 2.4:



License Agreement

Please read the following license agreement carefully.

NETBEANS IDE 6.0

Please review the complete list of open-source licenses governing software included in the Product. They can be found in the THIRDPARTYLICENSE.txt file. Please review the list of libraries and licenses provided for use. This license file contains five distinct licenses.

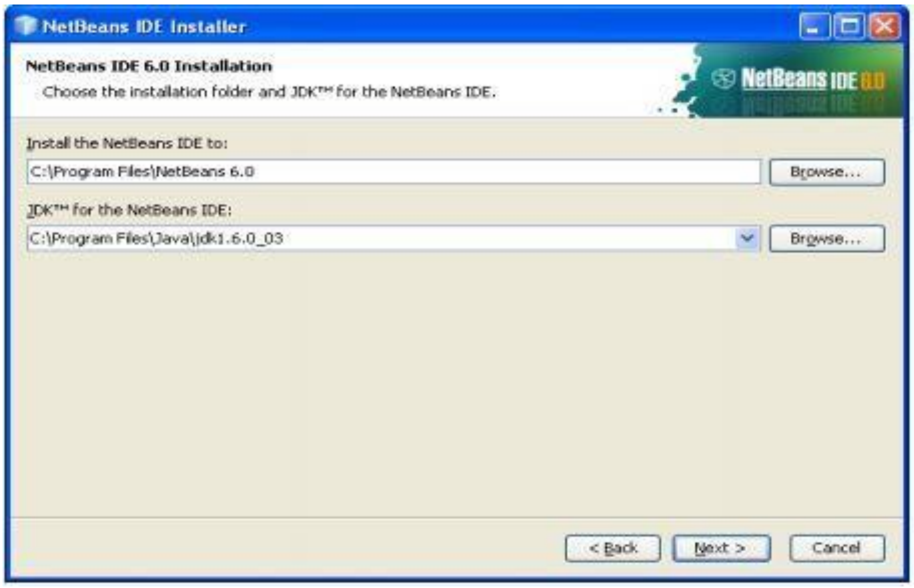
Unless specified below, the use of NetBeans IDE 6.0 and components from the GlassFish runtime are governed by the terms of either the GNU General Public License Version 2 with Classpath Exception or the Common Development and Distribution License. The Product also contains components from OpenJDK, which are governed exclusively by the terms of the GPLv2 with Classpath Exception.

LICENSE #1: GPLv2 with Classpath Exception.

[✓] I accept the terms in the license agreement

Step 5) Choose the installation path. Click Browse if you want to change. The Java JDK properly set to the Click Next.

Fig 2.5:



NetBeans IDE Installer

NetBeans IDE 6.0 Installation

Choose the installation folder and JDK™ for the NetBeans IDE.

Install the NetBeans IDE to:

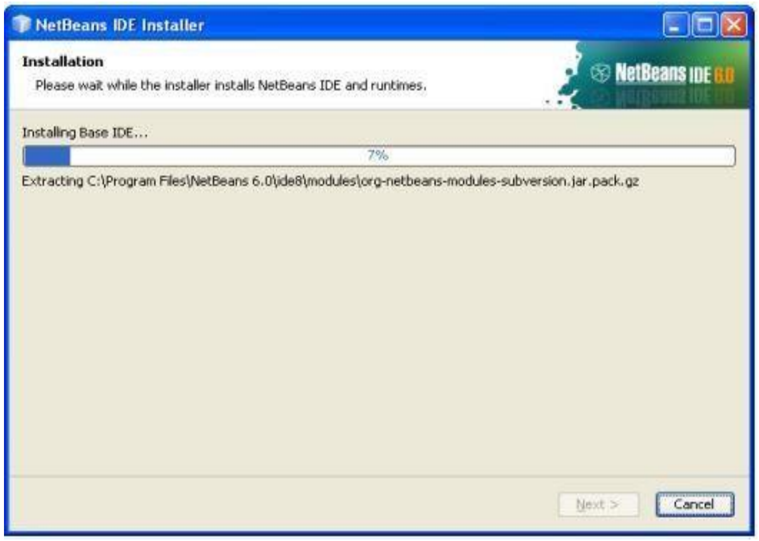
C:\Program Files\NetBeans 6.0

JDK™ for the NetBeans IDE:

C:\Program Files\Java\jdk1.6.0_03

Step 6) The installation begins. Wait and relax.

Fig 2.6:



NetBeans IDE Installer

Installation

Please wait while the installer installs NetBeans IDE and runtimes.

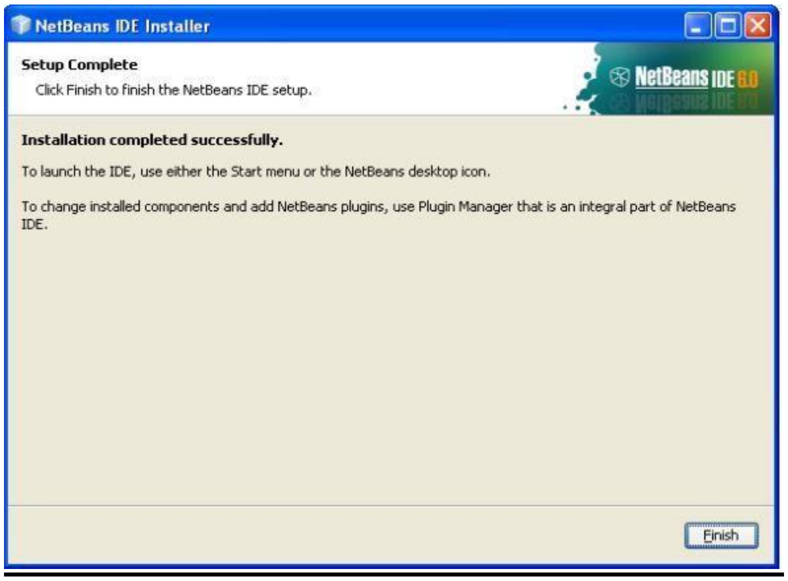
Installing Base IDE...

7%

Extracting C:\Program Files\NetBeans 6.0\ide8\modules\org-netbeans-modules-subversion.jar.pack.gz

Step 7) The installation completed successfully as shown in the following Figure. Click Finish.

<u>Fig 2.7:</u>


NetBeans IDE Installer

Setup Complete

Click Finish to finish the NetBeans IDE setup.

Installation completed successfully.

To launch the IDE, use either the Start menu or the NetBeans desktop icon.

To change installed components and add NetBeans plugins, use Plugin Manager that is an integral part of NetBeans IDE.

TOPIC#03:

DATA TYPES IN JAVA

DATA TYPES IN JAVA:

Data type specifies the size and type of values that can be stored in an identifier. The Java language is rich in its data types. Data Types in Java are classified into two types:

- ✓ Primitive Data Types
- ✓ Non-Primitive Data Types

3.1 Primitive Data Types

The primitive data types in Java include the following:

3.1.1 Integer:

Integer types can hold whole numbers, such as 123 and −96. The size of the values that can be stored depends on the integer type that we choose. These Data types are written as:

- byte (having size 1 byte and range of values -128 to 127)
- short (having size 2 bytes and range of values -32768 to 32767)
- int (having size 4 bytes and range of values −2,147,483,648 to 2,147,483,647)
- long (having size 8 bytes and range of values -9,223,372,036,854,775,808 to 9,223,372,036,854,755,807)

3.1.2 Floating Point:

Floating point data types are used to represent numbers with a fractional part. Single precision floating point numbers occupy 4 bytes and Double precision floating point numbers occupy 8 bytes such that:

- float (having size 4 bytes and range of values 3.4e–038 to 3.4e+038)
- double (having size 8 bytes and range of values 1.7e–308 to 1.7e+038)

3.1.3 Character:

It stores character constants in the memory. It assumes a size of 2 bytes, but basically it can hold only a single character because char stores Unicode character sets. It has a minimum value of ‘u0000’ (or 0) and a maximum value of ‘uffff’ (or 65,535, inclusive).

3.1.4 Boolean:

Boolean data types are used to store values with two states: true or false.

3.2 Non-Primitive Data Types

The Non-primitive data types in Java include the following:

3.2.1 String:

A Java String is a sequence of single characters. The “char” data type represents a single character. A string is also a class, which means that it has its own methods for working on strings.

3.2.2 Arrays:

An array is a group of variables that share the same data type, and are referred to by a common name. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index. The array index ranges from 0 to n−1; therefore, in an array of size 10, the first element is stored at index 0 and the last or the 10th element at index 9.

3.2.3 Interface:

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods. Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements. Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

TOPIC#04:

LOOPS IN JAVA

LOOPS IN JAVA:

A loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. There are four kinds of loops in Java language which are given as under:

- ✓ For Loop
- ✓ While Loop
- ✓ Do-While Loop
- ✓ Enhanced For Loop

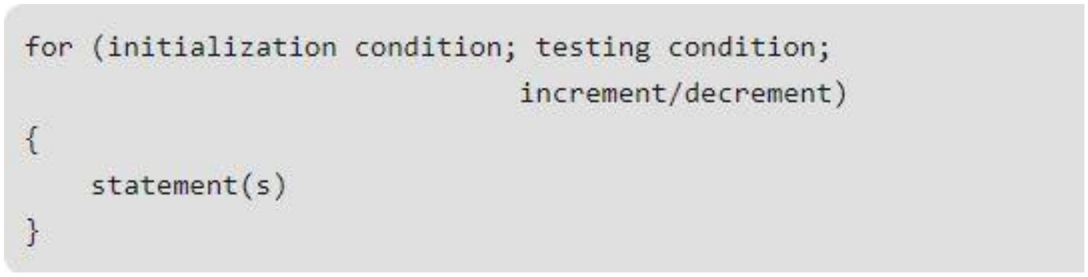
4.1 For Loop

The "for loop" provides a concise way of writing the loop structure. A "for" statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping. The "for loop" includes following parts:

- **Initialization condition:** Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.
- **Testing Condition:** It is used for testing the exit condition for a loop. It must return a Boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.
- **Statement execution:** Once the condition is evaluated to true, the statements in the loop body are executed.
- **Increment/ Decrement:** It is used for updating the variable for next iteration.
- **Loop termination:** When the condition becomes false, the loop terminates marking the end of its life cycle.

Syntax:

<u>Fig 4.1</u>



```
for (initialization condition; testing condition;
                              increment/decrement)
{
    statement(s)
}
```

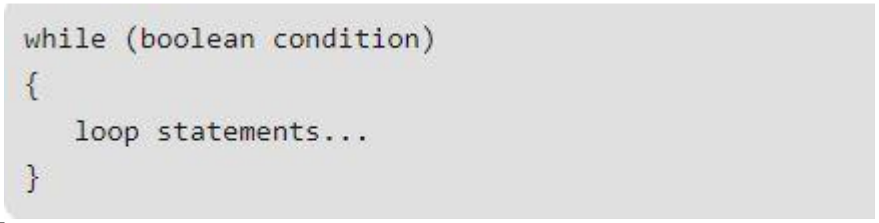
4.2 While Loop:

A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

- While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called Entry control loop.
- Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.
- When the condition becomes false, the loop terminates which marks the end of its life cycle.

Syntax:

<u>Fig 4.2:</u>



```
while (boolean condition)
{
   loop statements...
}
```

-

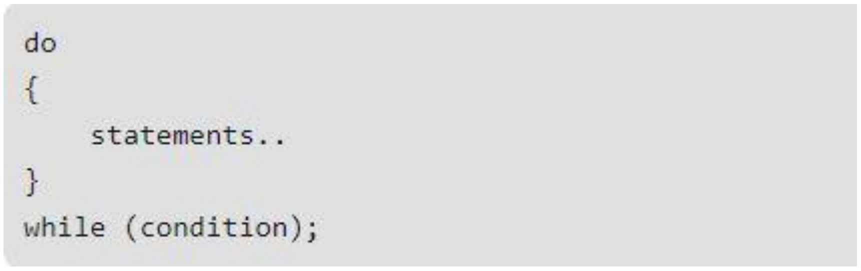
4.3 Do-While Loop:

“Do-while” loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of Exit Control Loop.

1) Do-while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.
2) After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
3) When the condition becomes false, the loop terminates which marks the end of its life cycle.
4) It is important to note that the do-while loop will execute its statements at least once before any condition is checked, and therefore is an example of exit control loop.

Syntax:

<u>Fig 4.3:</u>



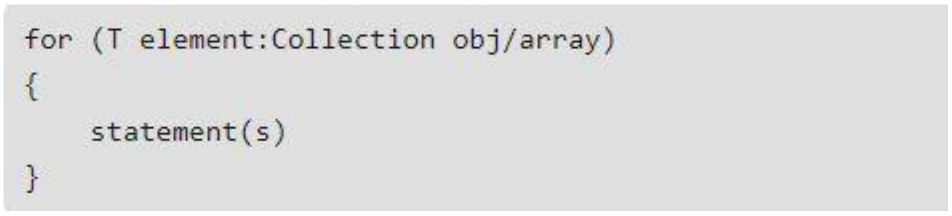
```
do
{
    statements..
}
while (condition);
```

4.4 Enhanced For Loop:

Java also includes another version of for loop introduced in Java 5. Enhanced for loop provides a simpler way to iterate through the elements of a collection or array. It is inflexible and should be used only when there is a need to iterate through the elements in sequential manner without knowing the index of currently processed element. Also note that the object/variable is immutable when enhanced for loop is used i.e it ensures that the values in the array cannot be modified, so it can be said as read only loop where you can't update the values as opposite to other loops where values can be modified. We recommend using this form of the for statement instead of the general form whenever possible.(as per JAVA doc.)

Syntax:

Fig 4.4:



```
for (T element:Collection obj/array)
{
    statement(s)
}
```

TOPIC#05:

CONTROL STRUCTURES IN JAVA

CONTROL STRUCTURES IN JAVA:

A program executes from top to bottom except when we use control statements, we can control the order of execution of the program, based on logic and values.

Following are some control structures used in Java:

- If statement
- Else statement
- Else-if statement
- Switch

5.1 **If Statement:**

The first contained statement (that can be a block) of an “if “statement only executes when the specified condition is true. If the condition is false and there is not else keyword then the first contained statement will be skipped and execution continues with the rest of the program. The condition is an expression that returns a Boolean value.

5.2 **Else Statement:**

The Else statement executes when the “if” and “else-if” statements are false. The Else statement doesn’t need any condition to be executed. It just provides a block in which the last and the least condition of the ifelse flow is executed.

5.3 **Else-if Statement:**

The Else-if statement is used when we have more than one conditions in our program. It is executed when the “if” condition is false and if the condition provided to the “else-if” is true, the block of code of the condition is executed. Otherwise, control is shifted towards next “else-if” or “else” condition given.

5.4 **Switch Statement:**

The switch statement is a multi-way branch statement. The switch statement of Java is another selection statement that defines multiple paths of execution of a program. It provides a better alternative than a large series of if-else-if statements. An expression must be of a type of byte, short, int or char. Each of the values specified in the case statement must be of a type compatible with the expression. Duplicate case values are not allowed. The break statement is used inside the switch to terminate a statement sequence. The break statement is optional in the switch statement.

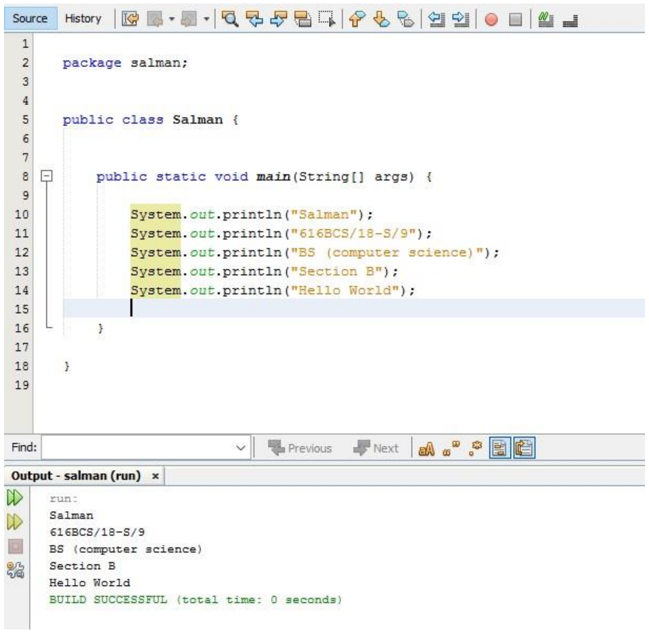
TOPIC#06:

JAVA PROGRAMS

EXERCISE # 01

6.1 PROGRAM # 01:

Fig 6.1:

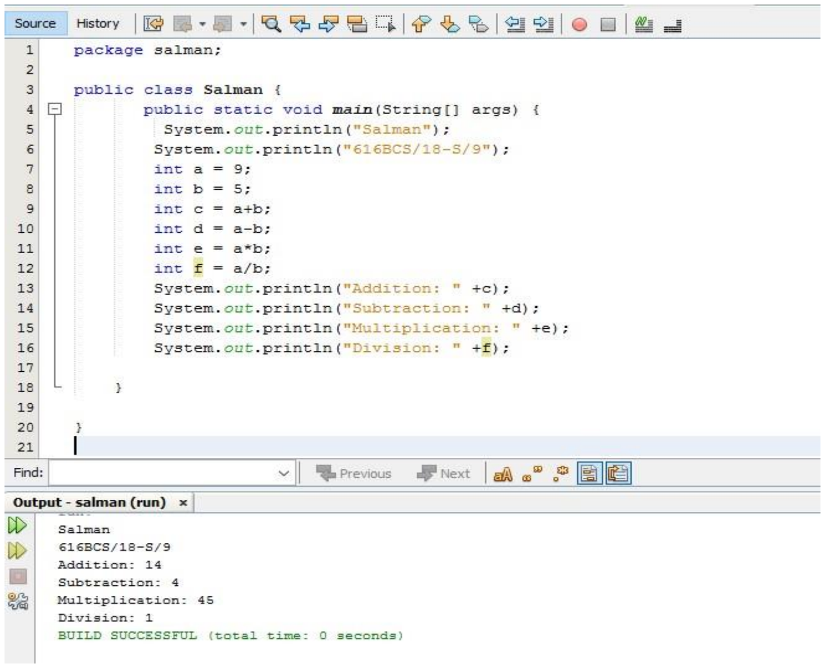


```
package salman;


public class Salman {


    public static void main(String[] args) {

        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        System.out.println("BS (computer science)");
        System.out.println("Section B");
        System.out.println("Hello World");

    }

}
```

```
run:
Salman
616BCS/18-S/9
BS (computer science)
Section B
Hello World
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.2 PROGRAM # 02:

Fig 6.2:

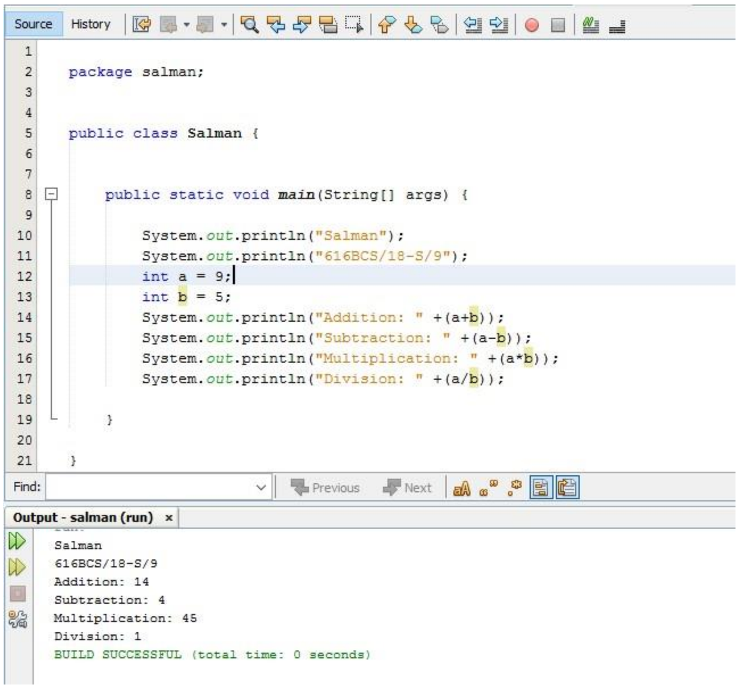


```
package salman;

public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");
         int a = 9;
         int b = 5;
         int c = a+b;
         int d = a-b;
         int e = a*b;
         int f = a/b;
         System.out.println("Addition: " +c);
         System.out.println("Subtraction: " +d);
         System.out.println("Multiplication: " +e);
         System.out.println("Division: " +f);

    }

}
```

```
Salman
616BCS/18-S/9
Addition: 14
Subtraction: 4
Multiplication: 45
Division: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.3 PROGRAM # 03:

Fig 6.3:

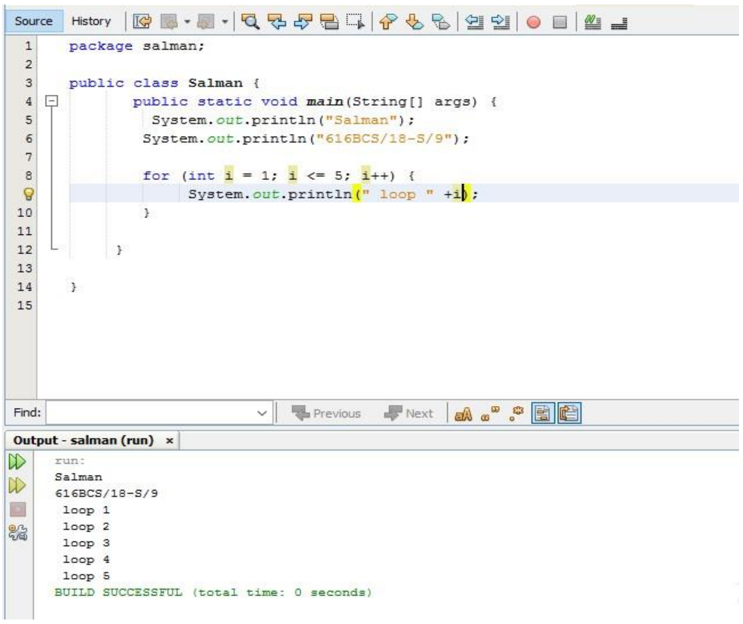


```
package salman;


public class Salman {


    public static void main(String[] args) {

        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        int a = 9;
        int b = 5;
        System.out.println("Addition: " +(a+b));
        System.out.println("Subtraction: " +(a-b));
        System.out.println("Multiplication: " +(a*b));
        System.out.println("Division: " +(a/b));

    }

}
```

```
Salman
616BCS/18-S/9
Addition: 14
Subtraction: 4
Multiplication: 45
Division: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.4 PROGRAM # 04:

Fig 6.4:

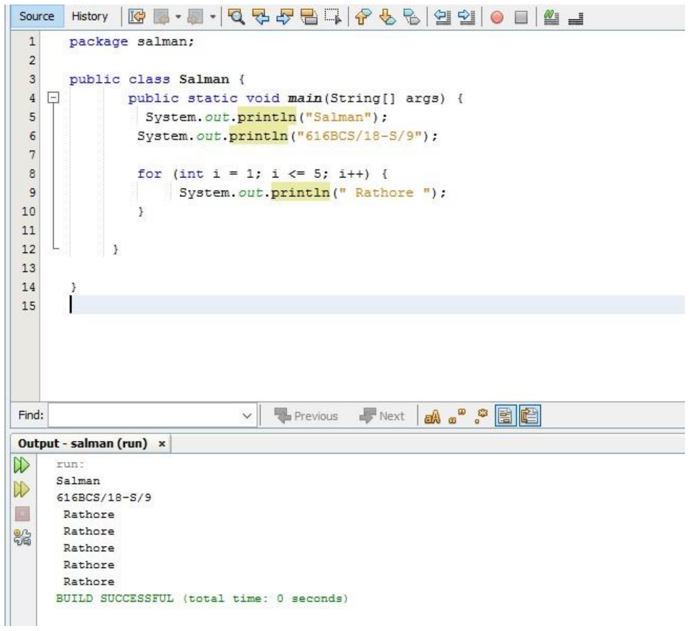


```
package salman;

public class Salman {
        public static void main(String[] args) {
         System.out.println("Salman");
        System.out.println("616BCS/18-S/9");

        for (int i = 1; i <= 5; i++) {
            System.out.println(" loop " +i);
        }

    }

}
```

```
run:
Salman
616BCS/18-S/9
 loop 1
 loop 2
 loop 3
 loop 4
 loop 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.5 PROGRAM # 05:

Fig 6.5:



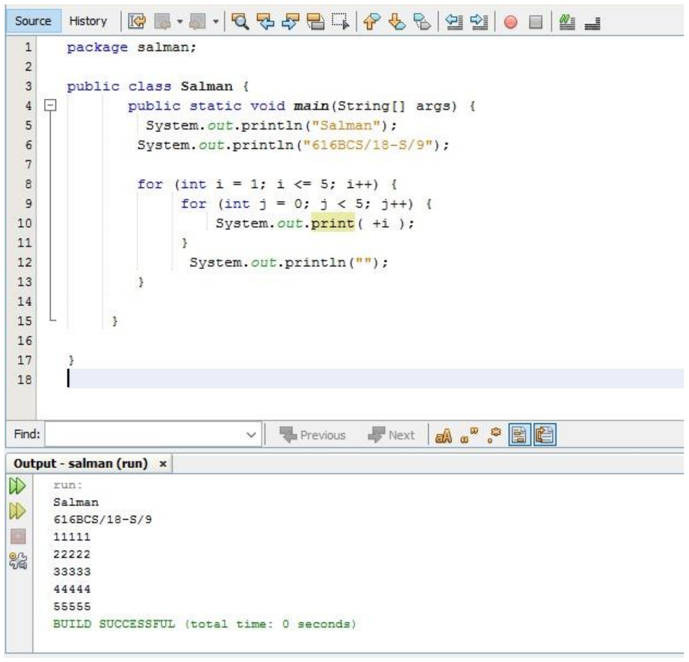
```
package salman;

public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");

         for (int i = 1; i <= 5; i++) {
              System.out.println(" Rathore ");
         }

    }

}

```

Output - salman (run)

```
run:
Salman
616BCS/18-S/9
 Rathore
 Rathore
 Rathore
 Rathore
 Rathore
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.6 PROGRAM # 06:

Fig 6.6:



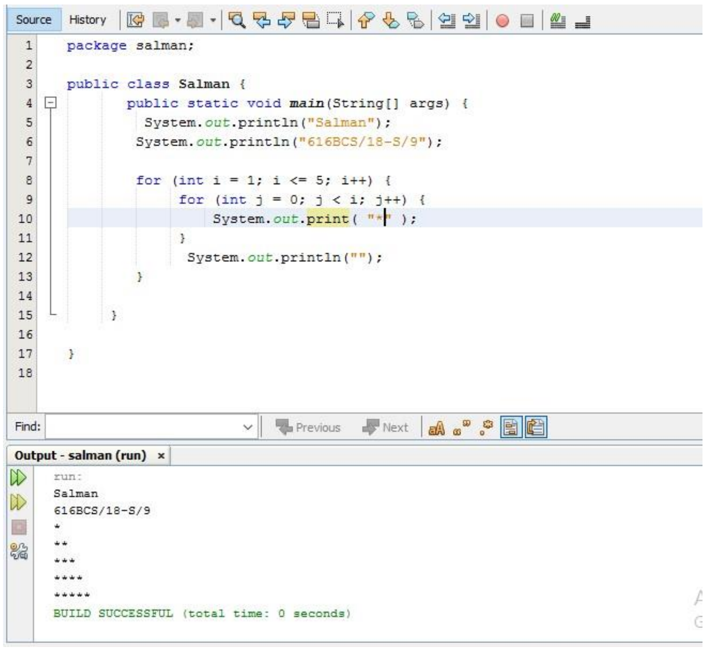
```
package salman;

public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");

         for (int i = 1; i <= 5; i++) {
              for (int j = 0; j < 5; j++) {
                  System.out.print( +i );
              }
               System.out.println("");
         }

    }

}

```

Output - salman (run)

```
run:
Salman
616BCS/18-S/9
11111
22222
33333
44444
55555
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.7 PROGRAM # 07:

Fig 6.7:

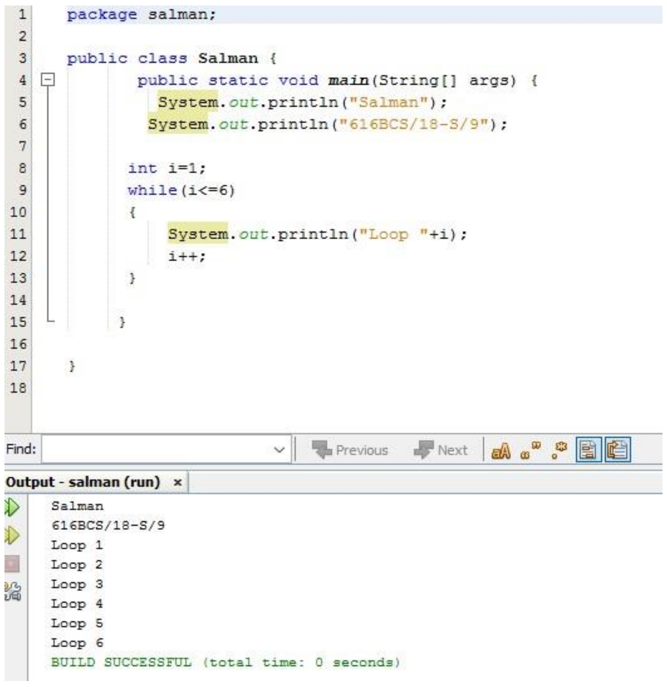


```
package salman;

public class Salman {
        public static void main(String[] args) {
         System.out.println("Salman");
        System.out.println("616BCS/18-S/9");

        for (int i = 1; i <= 5; i++) {
              for (int j = 0; j < i; j++) {
                  System.out.print( "*" );
              }
               System.out.println("");
        }

    }

}
```

```
run:
Salman
616BCS/18-S/9
*
**
***
****
*****
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.8 PROGRAM # 08:

Fig 6.8:

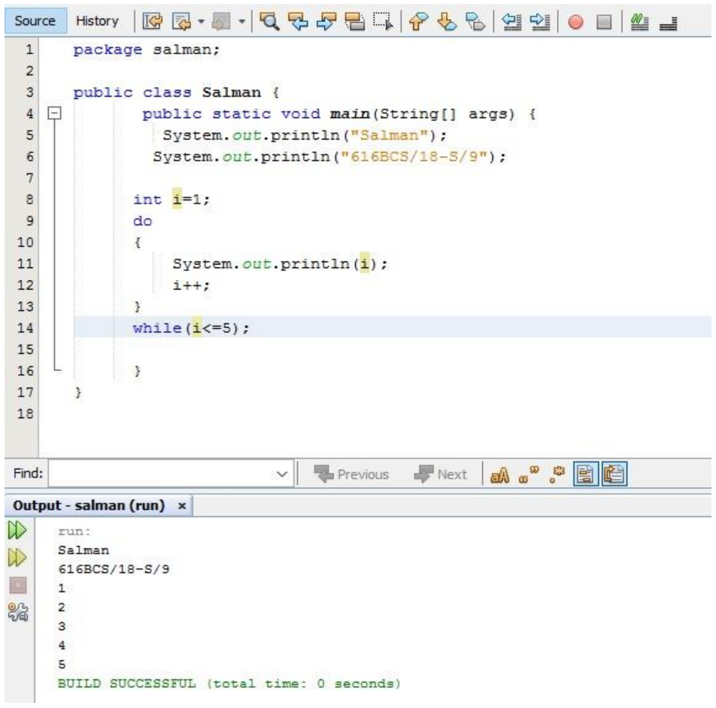


```
package salman;

public class Salman {
        public static void main(String[] args) {
         System.out.println("Salman");
        System.out.println("616BCS/18-S/9");

      int i=1;
      while(i<=6)
      {
          System.out.println("Loop "+i);
          i++;
      }

    }

}
```

```
Salman
616BCS/18-S/9
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5
Loop 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.9 PROGRAM # 09:

Fig 6.9:

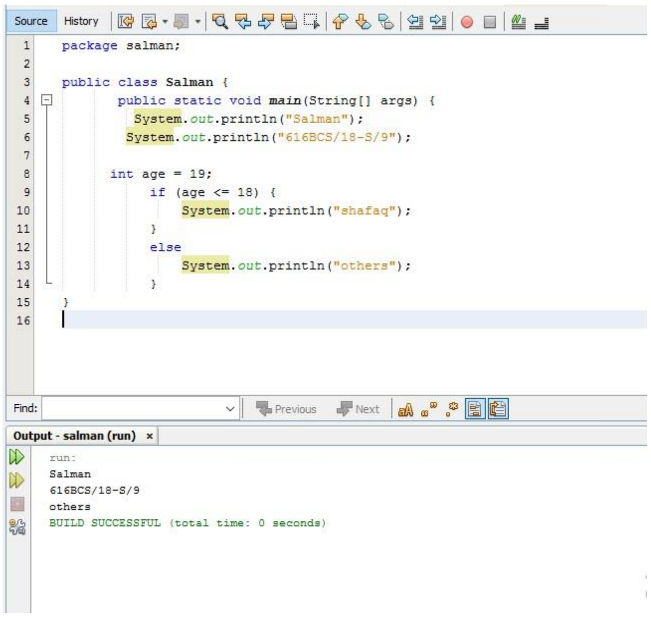


```
package salman;

public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");

      int i=1;
      do
      {
          System.out.println(i);
          i++;
      }
      while(i<=5);

      }
}
```

```
run:
Salman
616BCS/18-S/9
1
2
3
4
5
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.10 PROGRAM # 10:

Fig 6.10:

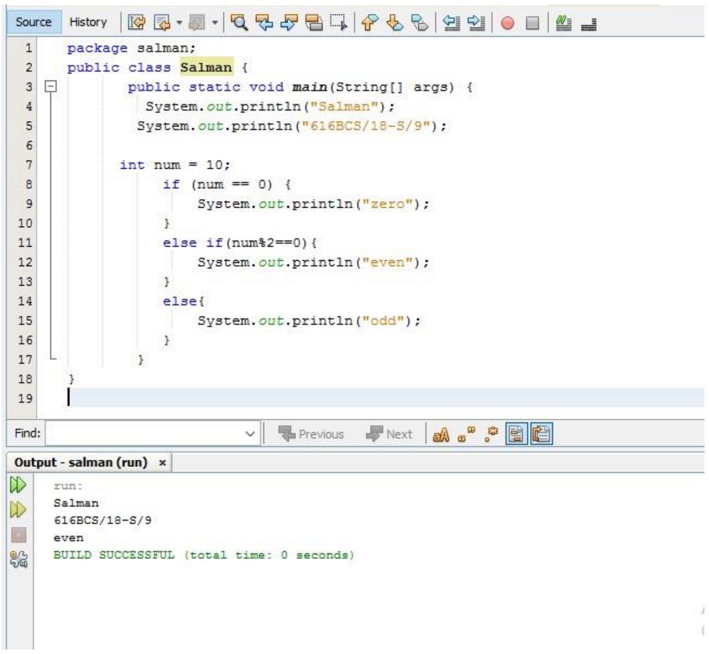


```
package salman;

public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");

     int age = 19;
          if (age <= 18) {
              System.out.println("shafaq");
          }
          else
              System.out.println("others");
          }
  }
```

```
run:
Salman
616BCS/18-S/9
others
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.11 PROGRAM # 11:

<u>Fig 6.11</u>:

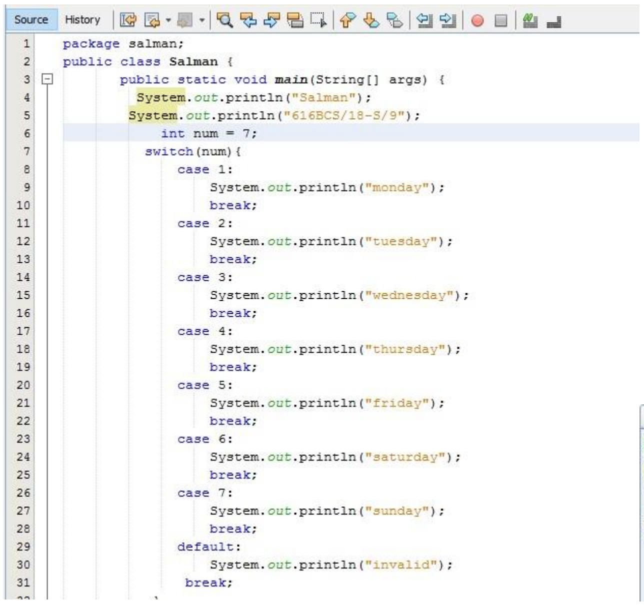


```
package salman;
public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");

       int num = 10;
            if (num == 0) {
                System.out.println("zero");
            }
            else if(num%2==0){
                System.out.println("even");
            }
            else{
                System.out.println("odd");
            }
        }
}
```

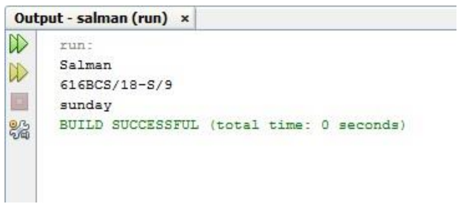
```
run:
Salman
616BCS/18-S/9
even
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.12 PROGRAM # 12:

<u>Fig 6.12</u>:



```
package salman;
public class Salman {
        public static void main(String[] args) {
          System.out.println("Salman");
         System.out.println("616BCS/18-S/9");
             int num = 7;
           switch(num){
               case 1:
                   System.out.println("monday");
                   break;
               case 2:
                   System.out.println("tuesday");
                   break;
               case 3:
                   System.out.println("wednesday");
                   break;
               case 4:
                   System.out.println("thursday");
                   break;
               case 5:
                   System.out.println("friday");
                   break;
               case 6:
                   System.out.println("saturday");
                   break;
               case 7:
                   System.out.println("sunday");
                   break;
               default:
                   System.out.println("invalid");
                break;
```



```
run:
Salman
616BCS/18-S/9
sunday
BUILD SUCCESSFUL (total time: 0 seconds)
```

TOPIC#07:

CLASSES AND OBJECTS

7.1 CLASS:

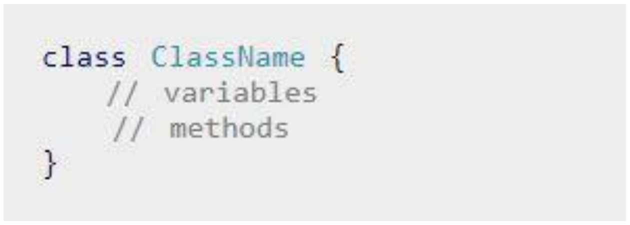
A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

- **Modifiers:** A class can be public or has default access.
- **Class name:** The name should begin with a initial letter (capitalized by convention).
- **Superclass (if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- **Interfaces (if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- **Body:** The class body surrounded by braces, { }.

There are various types of classes that are used in real time applications such as Nested Classes, Anonymous Classes, and Lambda Expressions.

Here is how a class is defined in Java:

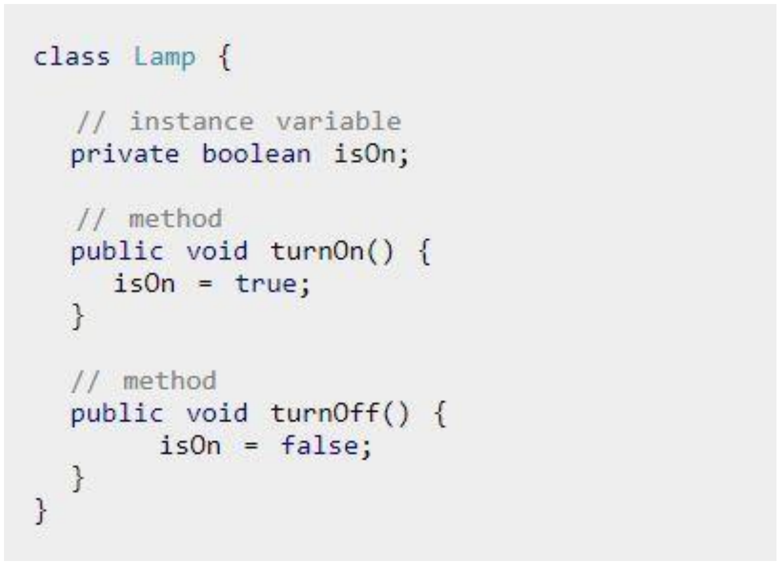
Fig 7.1:



```
class ClassName {
    // variables
    // methods
}
```

Here is an example of class in Java:

Fig 7.2:



```
class Lamp {

   // instance variable
   private boolean isOn;

   // method
   public void turnOn() {
      isOn = true;
   }

   // method
   public void turnOff() {
         isOn = false;
   }
}
```

Here, we defined a class named “Lamp”. The class has one instance variable (variable defined inside class) “isOn” and two methods “turnOn()” and “turnOff()”. These variables and methods defined within a class are called Members of the class.

7.2 OBJECT:

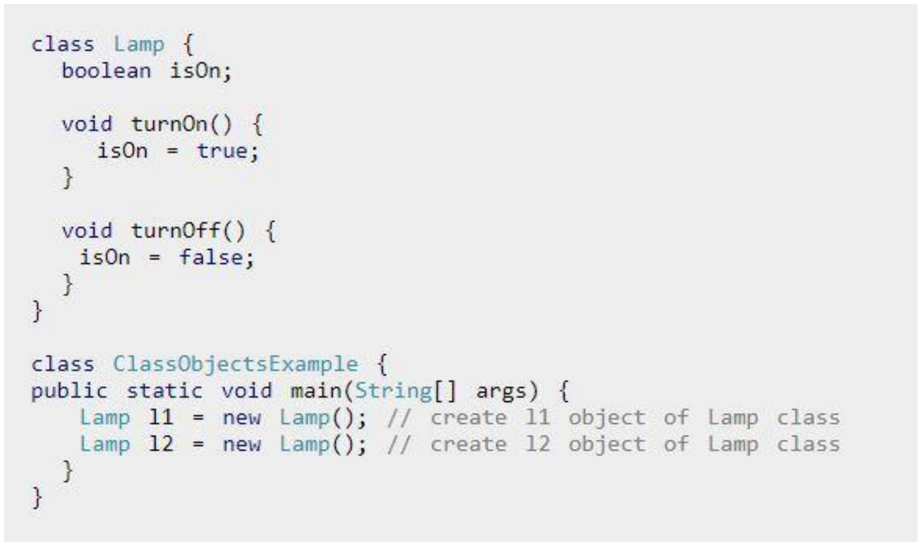
Instance of a class is called an Object. It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.

2. **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.

3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.
   Here is an example of Object in Java:

   Fig 7.3:



```java
class Lamp {
  boolean isOn;

  void turnOn() {
    isOn = true;
  }

  void turnOff() {
   isOn = false;
  }
}

class ClassObjectsExample {
public static void main(String[] args) {
    Lamp l1 = new Lamp(); // create l1 object of Lamp class
    Lamp l2 = new Lamp(); // create l2 object of Lamp class
  }
}
```

This program creates two objects “l1” and “l2” of class “Lamp”.

.

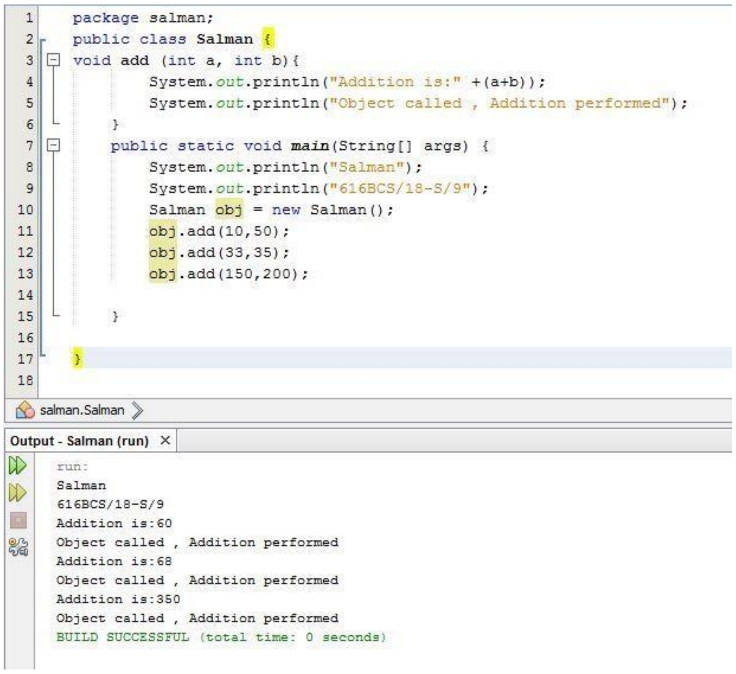
TOPIC#08:

JAVA PROGRAMS

EXERCISE #02

8.1 PROGRAM#01:

<u>Fig 8.1</u>:

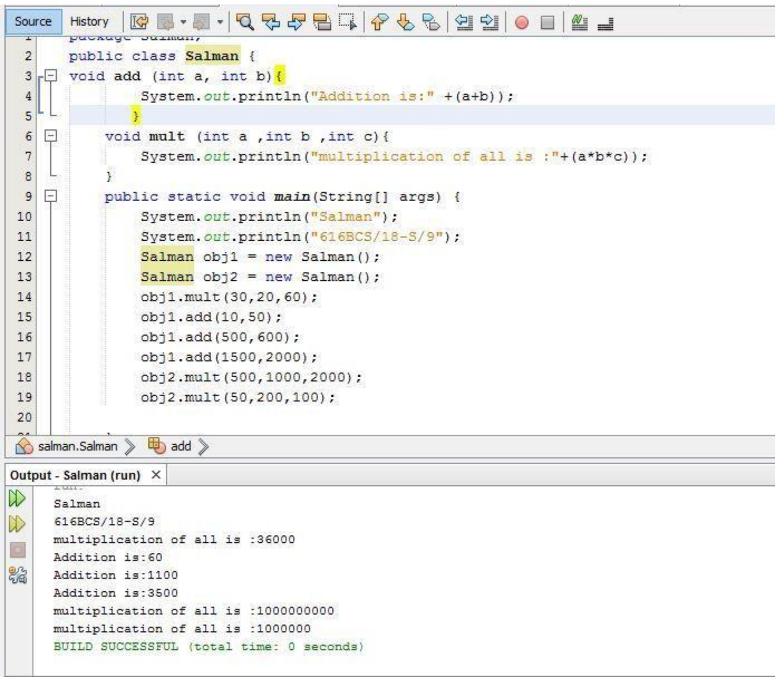


```
package salman;
public class Salman {
void add (int a, int b){
        System.out.println("Addition is:" +(a+b));
        System.out.println("Object called , Addition performed");
    }
    public static void main(String[] args) {
        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        Salman obj = new Salman();
        obj.add(10,50);
        obj.add(33,35);
        obj.add(150,200);

    }

}

```

```
run:
Salman
616BCS/18-S/9
Addition is:60
Object called , Addition performed
Addition is:68
Object called , Addition performed
Addition is:350
Object called , Addition performed
BUILD SUCCESSFUL (total time: 0 seconds)
```

8.2 PROGRAM#02:

<u>Fig 8.2</u>:

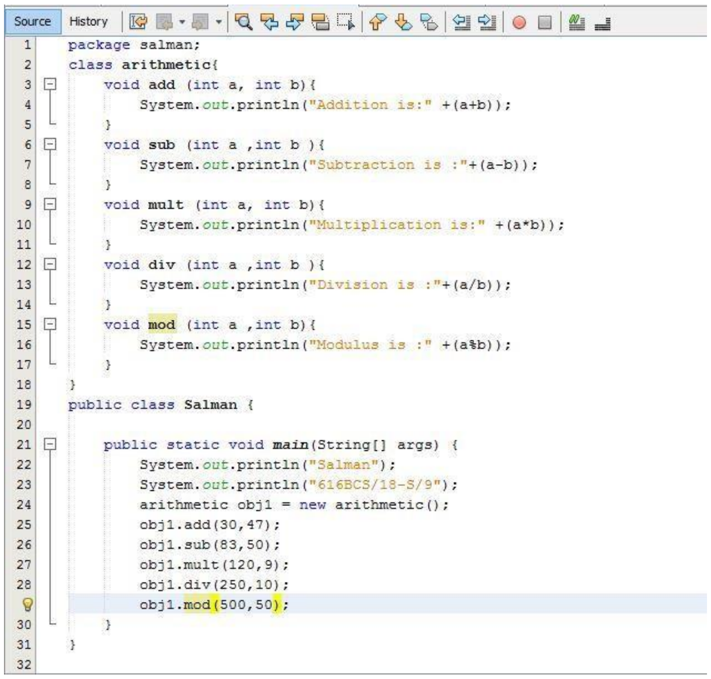


```
package salman;
public class Salman {
void add (int a, int b){
        System.out.println("Addition is:" +(a+b));
        }
    void mult (int a ,int b ,int c){
        System.out.println("multiplication of all is :"+(a*b*c));
    }
    public static void main(String[] args) {
        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        Salman obj1 = new Salman();
        Salman obj2 = new Salman();
        obj1.mult(30,20,60);
        obj1.add(10,50);
        obj1.add(500,600);
        obj1.add(1500,2000);
        obj2.mult(500,1000,2000);
        obj2.mult(50,200,100);

```

```
Salman
616BCS/18-S/9
multiplication of all is :36000
Addition is:60
Addition is:1100
Addition is:3500
multiplication of all is :1000000000
multiplication of all is :1000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

8.3 PROGRAM # 03:

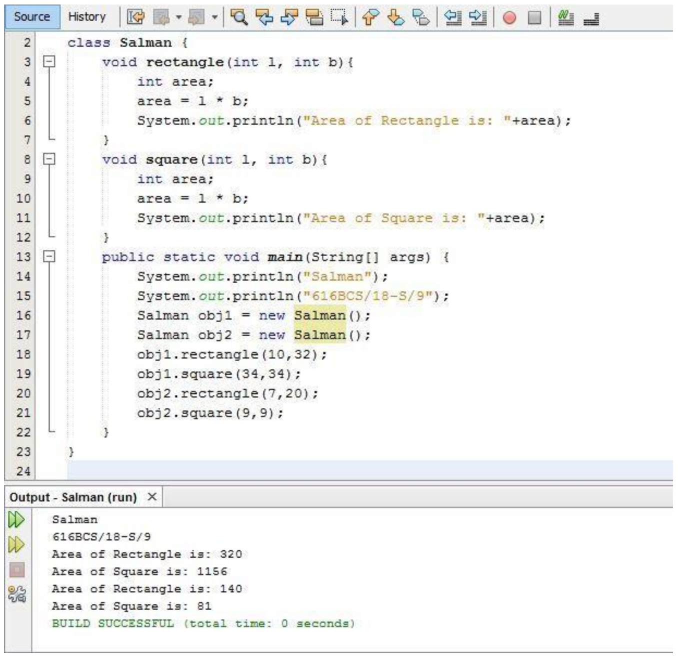
Fig 8.3:



```java
package salman;
class arithmetic{
    void add (int a, int b){
        System.out.println("Addition is:" +(a+b));
    }
    void sub (int a ,int b ){
        System.out.println("Subtraction is :"+(a-b));
    }
    void mult (int a, int b){
        System.out.println("Multiplication is:" +(a*b));
    }
    void div (int a ,int b ){
        System.out.println("Division is :"+(a/b));
    }
    void mod (int a ,int b){
        System.out.println("Modulus is :" +(a%b));
    }
}
public class Salman {

    public static void main(String[] args) {
        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        arithmetic obj1 = new arithmetic();
        obj1.add(30,47);
        obj1.sub(83,50);
        obj1.mult(120,9);
        obj1.div(250,10);
        obj1.mod(500,50);
    }
}
```

8.4 PROGRAM # 04:

Fig 8.4:



```java
class Salman {
    void rectangle(int l, int b){
        int area;
        area = l * b;
        System.out.println("Area of Rectangle is: "+area);
    }
    void square(int l, int b){
        int area;
        area = l * b;
        System.out.println("Area of Square is: "+area);
    }
    public static void main(String[] args) {
        System.out.println("Salman");
        System.out.println("616BCS/18-S/9");
        Salman obj1 = new Salman();
        Salman obj2 = new Salman();
        obj1.rectangle(10,32);
        obj1.square(34,34);
        obj2.rectangle(7,20);
        obj2.square(9,9);
    }
}
```

Output - Salman (run)

```
Salman
616BCS/18-S/9
Area of Rectangle is: 320
Area of Square is: 1156
Area of Rectangle is: 140
Area of Square is: 81
BUILD SUCCESSFUL (total time: 0 seconds)
```

TOPIC#09:

SWING AND JOPTION PANE

9.1 SWING:

Swing is a part of JFC (Java Foundation Classes). Building Graphical User Interface in Java requires the use of Swings. Swing Framework contain a large set of components which allow high level of customization and provide rich functionalities, and is used to create window based applications. Java swing components are lightweight, platform independent.

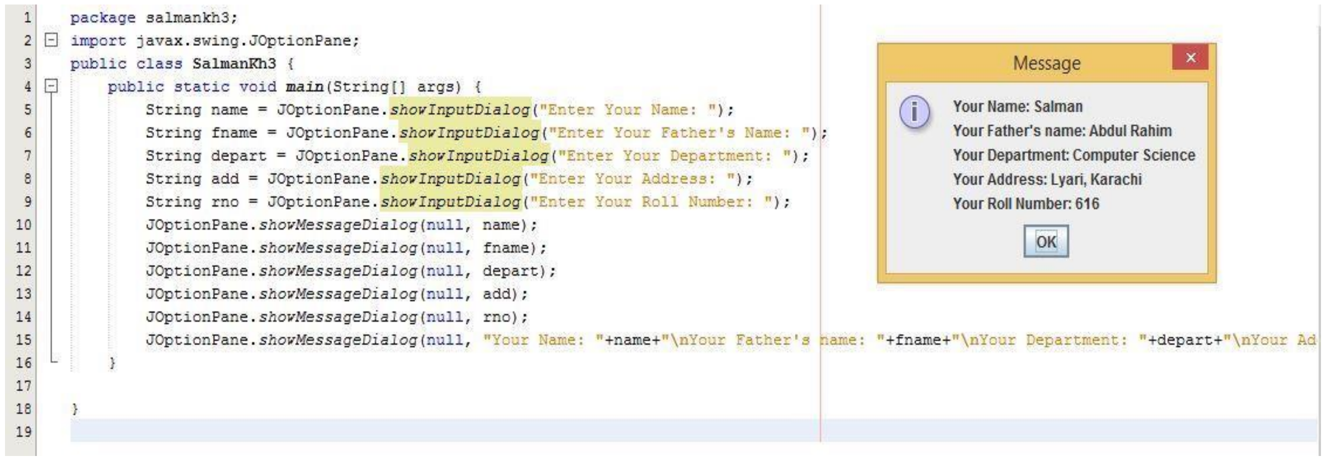
9.2 JOptionPane:

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user.

9.3 EXAMPLE PROGRAMS:

9.3.1 Example 01:

Fig 9.1:

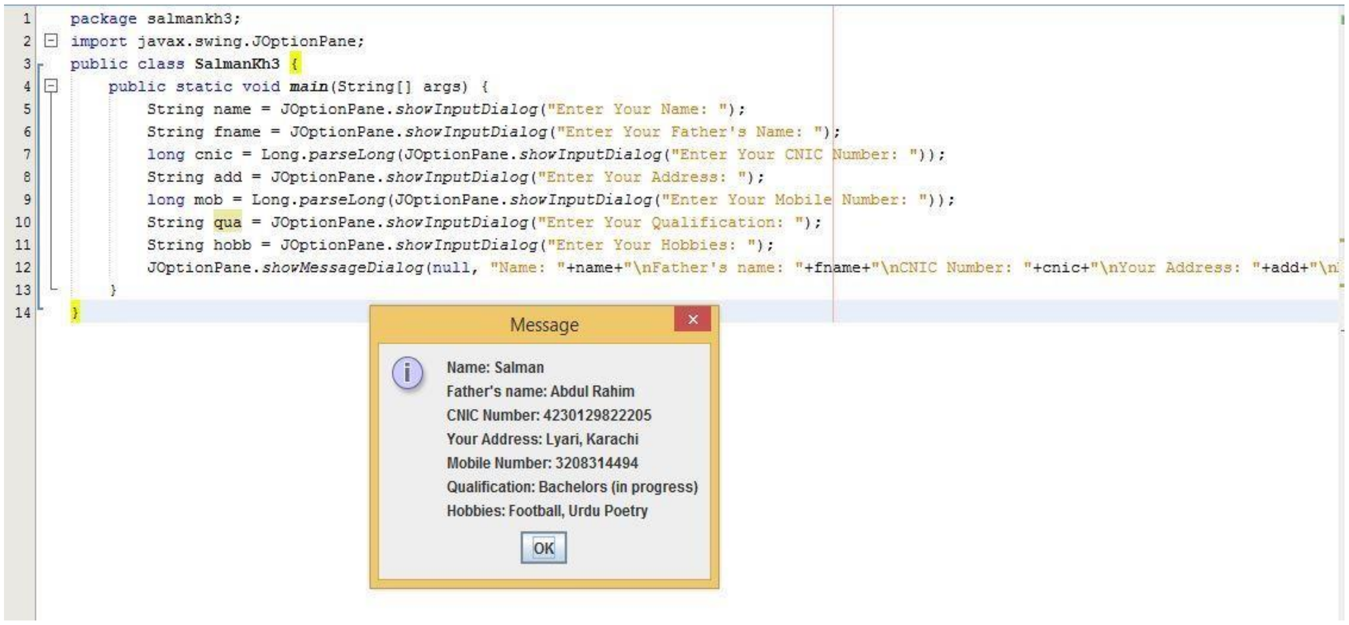


```
package salmankh3;
import javax.swing.JOptionPane;
public class SalmanKh3 {
    public static void main(String[] args) {
        String name = JOptionPane.showInputDialog("Enter Your Name: ");
        String fname = JOptionPane.showInputDialog("Enter Your Father's Name: ");
        String depart = JOptionPane.showInputDialog("Enter Your Department: ");
        String add = JOptionPane.showInputDialog("Enter Your Address: ");
        String rno = JOptionPane.showInputDialog("Enter Your Roll Number: ");
        JOptionPane.showMessageDialog(null, name);
        JOptionPane.showMessageDialog(null, fname);
        JOptionPane.showMessageDialog(null, depart);
        JOptionPane.showMessageDialog(null, add);
        JOptionPane.showMessageDialog(null, rno);
        JOptionPane.showMessageDialog(null, "Your Name: "+name+"\nYour Father's name: "+fname+"\nYour Department: "+depart+"\nYour Ad
    }

}
```

```
Your Name: Salman
Your Father's name: Abdul Rahim
Your Department: Computer Science
Your Address: Lyari, Karachi
Your Roll Number: 616
```

9.3.2 Example 02:

Fig 9.2:

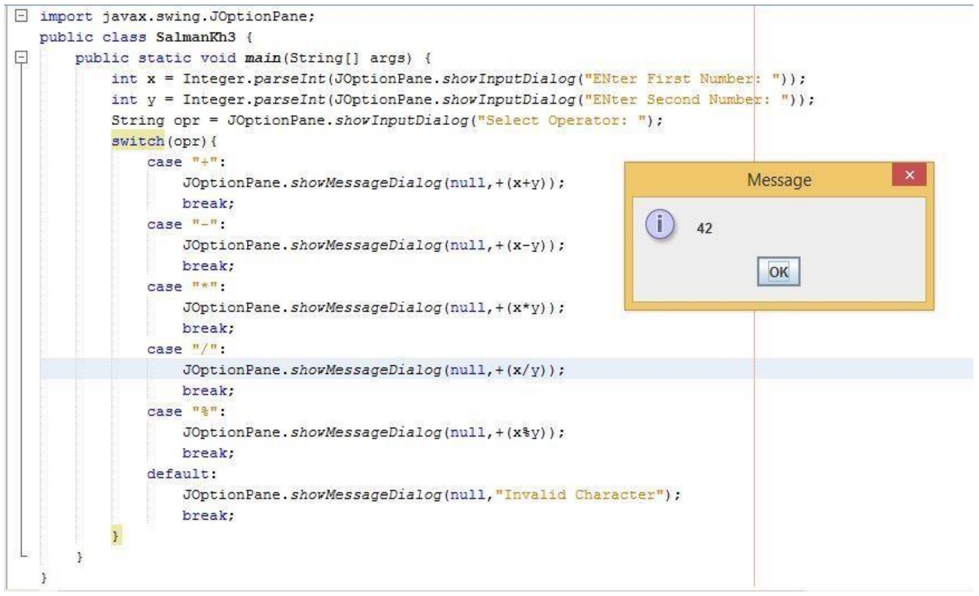


```
package salmankh3;
import javax.swing.JOptionPane;
public class SalmanKh3 {
    public static void main(String[] args) {
        String name = JOptionPane.showInputDialog("Enter Your Name: ");
        String fname = JOptionPane.showInputDialog("Enter Your Father's Name: ");
        long cnic = Long.parseLong(JOptionPane.showInputDialog("Enter Your CNIC Number: "));
        String add = JOptionPane.showInputDialog("Enter Your Address: ");
        long mob = Long.parseLong(JOptionPane.showInputDialog("Enter Your Mobile Number: "));
        String qua = JOptionPane.showInputDialog("Enter Your Qualification: ");
        String hobb = JOptionPane.showInputDialog("Enter Your Hobbies: ");
        JOptionPane.showMessageDialog(null, "Name: "+name+"\nFather's name: "+fname+"\nCNIC Number: "+cnic+"\nYour Address: "+add+"\n
    }
}
```

```
Name: Salman
Father's name: Abdul Rahim
CNIC Number: 4230129822205
Your Address: Lyari, Karachi
Mobile Number: 3208314494
Qualification: Bachelors (in progress)
Hobbies: Football, Urdu Poetry
```

9.3.3 Example 03:

Fig 9.3:



```
import javax.swing.JOptionPane;
public class SalmanKh3 {
    public static void main(String[] args) {
        int x = Integer.parseInt(JOptionPane.showInputDialog("ENter First Number: "));
        int y = Integer.parseInt(JOptionPane.showInputDialog("ENter Second Number: "));
        String opr = JOptionPane.showInputDialog("Select Operator: ");
        switch(opr){
            case "+":
                JOptionPane.showMessageDialog(null,+(x+y));
                break;
            case "-":
                JOptionPane.showMessageDialog(null,+(x-y));
                break;
            case "*":
                JOptionPane.showMessageDialog(null,+(x*y));
                break;
            case "/":
                JOptionPane.showMessageDialog(null,+(x/y));
                break;
            case "%":
                JOptionPane.showMessageDialog(null,+(x%y));
                break;
            default:
                JOptionPane.showMessageDialog(null,"Invalid Character");
                break;
        }
    }
}
```

```
42
```

TOPIC#10:

SCANNER IN JAVA

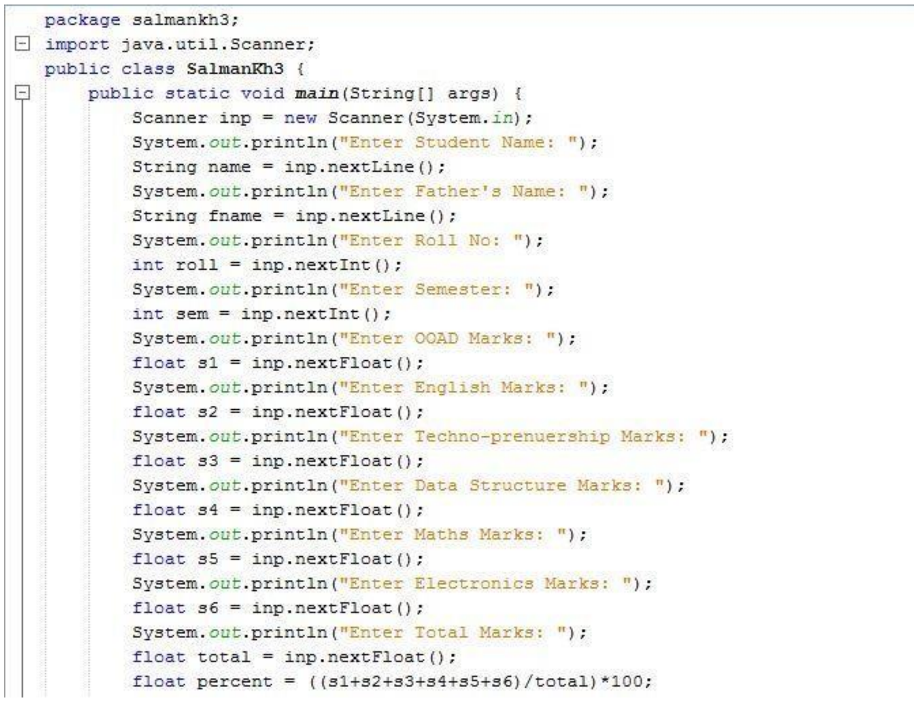
10.1 SCANNER:

The scanner class is used to get user input, and it is found in the Java.util package. To use the scanner class, create an object of the class and use any of the available methods found in the scanner class documentation.

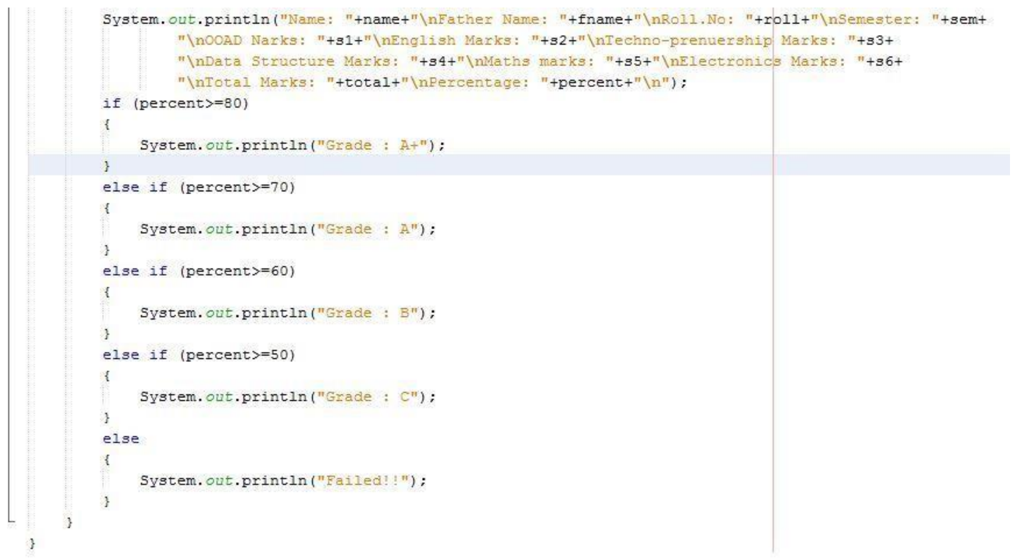
10.2 EXAMPLE PROGRAMS:

10.2.1 **Example 01**:

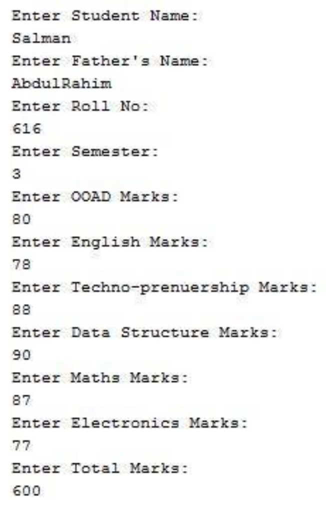
Fig 10.1:



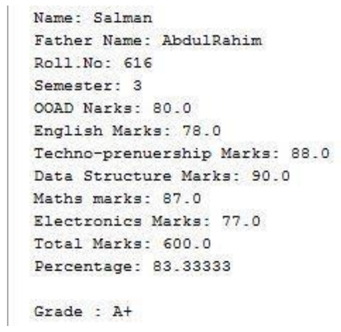
```
package salmankh3;
import java.util.Scanner;
public class SalmanKh3 {
    public static void main(String[] args) {
        Scanner inp = new Scanner(System.in);
        System.out.println("Enter Student Name: ");
        String name = inp.nextLine();
        System.out.println("Enter Father's Name: ");
        String fname = inp.nextLine();
        System.out.println("Enter Roll No: ");
        int roll = inp.nextInt();
        System.out.println("Enter Semester: ");
        int sem = inp.nextInt();
        System.out.println("Enter OOAD Marks: ");
        float s1 = inp.nextFloat();
        System.out.println("Enter English Marks: ");
        float s2 = inp.nextFloat();
        System.out.println("Enter Techno-prenuership Marks: ");
        float s3 = inp.nextFloat();
        System.out.println("Enter Data Structure Marks: ");
        float s4 = inp.nextFloat();
        System.out.println("Enter Maths Marks: ");
        float s5 = inp.nextFloat();
        System.out.println("Enter Electronics Marks: ");
        float s6 = inp.nextFloat();
        System.out.println("Enter Total Marks: ");
        float total = inp.nextFloat();
        float percent = ((s1+s2+s3+s4+s5+s6)/total)*100;
```



```java
        System.out.println("Name: "+name+"\nFather Name: "+fname+"\nRoll.No: "+roll+"\nSemester: "+sem+
                "\nOOAD Narks: "+s1+"\nEnglish Marks: "+s2+"\nTechno-prenuership Marks: "+s3+
                "\nData Structure Marks: "+s4+"\nMaths marks: "+s5+"\nElectronics Marks: "+s6+
                "\nTotal Marks: "+total+"\nPercentage: "+percent+"\n");
        if (percent>=80)
        {
            System.out.println("Grade : A+");
        }
        else if (percent>=70)
        {
            System.out.println("Grade : A");
        }
        else if (percent>=60)
        {
            System.out.println("Grade : B");
        }
        else if (percent>=50)
        {
            System.out.println("Grade : C");
        }
        else
        {
            System.out.println("Failed!!");
        }
    }
}
```



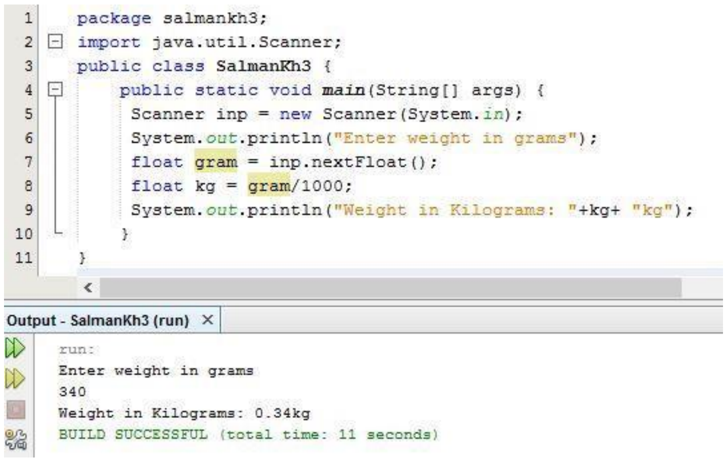
```
Enter Student Name:
Salman
Enter Father's Name:
AbdulRahim
Enter Roll No:
616
Enter Semester:
3
Enter OOAD Marks:
80
Enter English Marks:
78
Enter Techno-prenuership Marks:
88
Enter Data Structure Marks:
90
Enter Maths Marks:
87
Enter Electronics Marks:
77
Enter Total Marks:
600
```



```
Name: Salman
Father Name: AbdulRahim
Roll.No: 616
Semester: 3
OOAD Narks: 80.0
English Marks: 78.0
Techno-prenuership Marks: 88.0
Data Structure Marks: 90.0
Maths marks: 87.0
Electronics Marks: 77.0
Total Marks: 600.0
Percentage: 83.33333

Grade : A+
```

10.2.2 <u>Example 02</u>:

<u>Fig 10.2</u>:



```java
package salmankh3;
import java.util.Scanner;
public class SalmanKh3 {
    public static void main(String[] args) {
     Scanner inp = new Scanner(System.in);
     System.out.println("Enter weight in grams");
     float gram = inp.nextFloat();
     float kg = gram/1000;
     System.out.println("Weight in Kilograms: "+kg+ "kg");
    }
}
```

Output - SalmanKh3 (run)

```
run:
Enter weight in grams
340
Weight in Kilograms: 0.34kg
BUILD SUCCESSFUL (total time: 11 seconds)
```

TOPIC#11:

UML AND ITS TECHNIQUES

11.1 Unified Modeling Language (UML):

Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved in object-oriented software development. It uses graphic notation to create visual models of software systems.

11.2 Elements Of UML Architecture:

The UML architecture is based on the meta-object facility, which defines the foundation for creating modeling language. They are precise enough to generate the entire application. It supports high level development concepts such as frameworks, patterns and collaborations. UML includes a collection of elements such as:

- Programming Language Statements
- Actors: specify a role played by a user or any other system interacting with the subject.
- Activities: These are tasks, which must take place in order to fulfill an operation contract. They are represented in activity diagrams.
- Business Process: includes a collection of tasks producing a specific service for customers and is visualized with a flowchart as a sequence of activities.
- Logical and Reusable Software Components.

11.3 UML Diagrams:

UML diagrams can be divided into two categories. The first type includes six diagram types representing structural information. The second includes the remaining seven representing general types of behavior.
Structure diagrams are used in documenting the architecture of software systems and are involved in the system being modeled. Different structure diagrams are:

- Class Diagram: represents system class, attributes and relationships among the classes.
- Component Diagram: represents how components are split in a software system and dependencies among the components.
- Deployment Diagram: describes the hardware used in system implementations.
- Composite Structure Diagram: describes internal structure of classes.
- Object Diagram: represents a complete or partial view of the structure of a modeled system.
- Package Diagram: represents splitting of a system into logical groupings and dependency among the grouping.

Behavior diagrams represent functionality of software system and emphasize on what must happen in the system being modeled. The different behavior diagrams are:

- Activity Diagram: represents step by step workflow of business and operational components.
- Use Case Diagram: describes functionality of a system in terms of actors, goals as use cases and dependencies among the use cases.
- UML State Machine Diagram: represents states and state transition.
- Communication Diagram: represents interaction between objects in terms of sequenced messages.
- Timing Diagrams: focuses on timing constraints.
- Interaction Overview Diagram: provides an overview and nodes representing communication diagrams.
- Sequence Diagram: represents communication between objects in terms of a sequence of messages.

TOPIC#12:

JFrame, JTextField,

JTextArea, JButton, JLabel

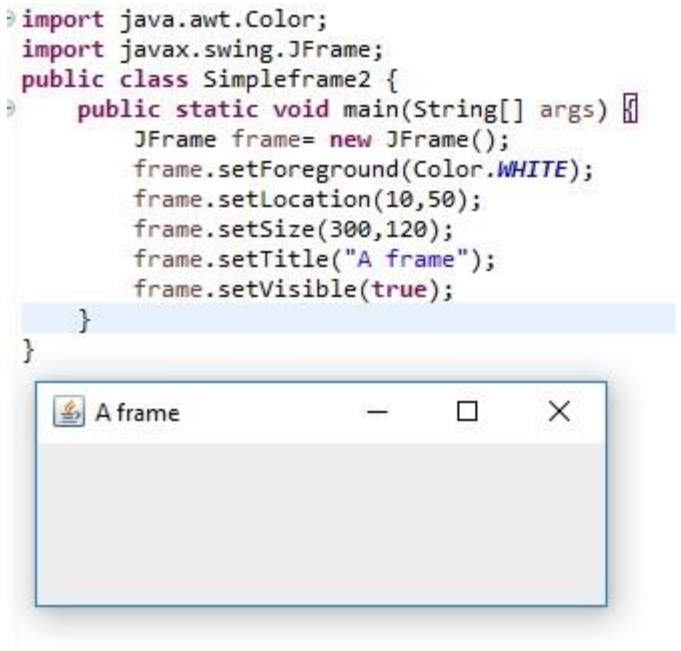
12.1 What is JFrame?

JFrame is a class of javax.swing package extended by java.awt.frame, it adds support for JFC/SWING component architecture. It is the top level window, with border and a title bar. JFrame class has many methods which can be used to customize it. Some of them are given as under:

- setForeground
- setSize
- setTitle
- setLocation

Program Example:

Fig 12.1:



```java
import java.awt.Color;
import javax.swing.JFrame;
public class Simpleframe2 {
    public static void main(String[] args) {
        JFrame frame= new JFrame();
        frame.setForeground(Color.WHITE);
        frame.setLocation(10,50);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setVisible(true);
    }
}
```

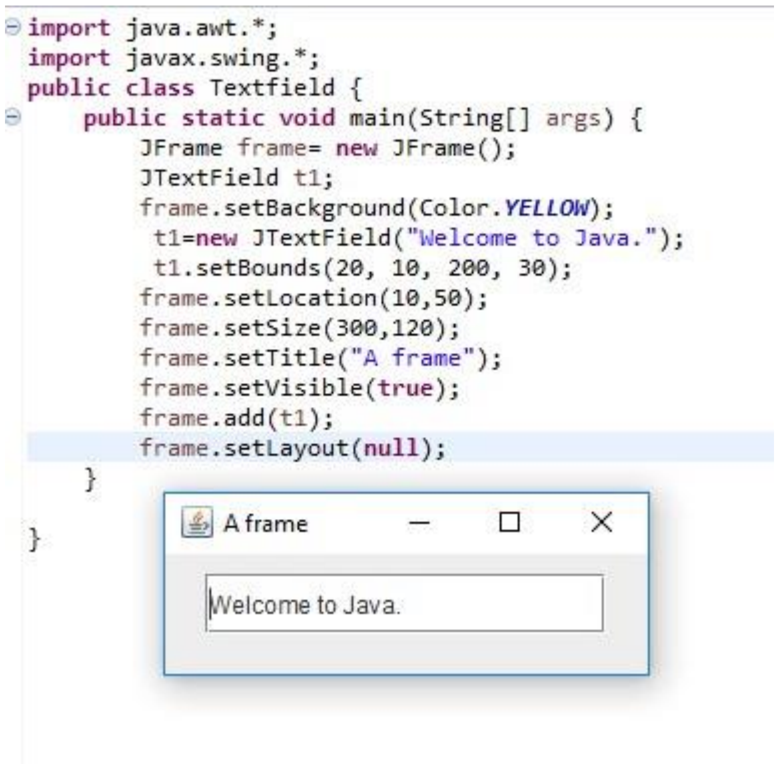
12.2 What Is JTextField?

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class. This object uses many methods for its customization. Some of the methods are given below:

- addActionListener()
- setFont()
- getAction()

Program Example:

<u>Fig 12.2</u>:



```
import java.awt.*;
import javax.swing.*;
public class Textfield {
    public static void main(String[] args) {
        JFrame frame= new JFrame();
        JTextField t1;
        frame.setBackground(Color.YELLOW);
         t1=new JTextField("Welcome to Java.");
         t1.setBounds(20, 10, 200, 30);
        frame.setLocation(10,50);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setVisible(true);
        frame.add(t1);
        frame.setLayout(null);
    }

}
```

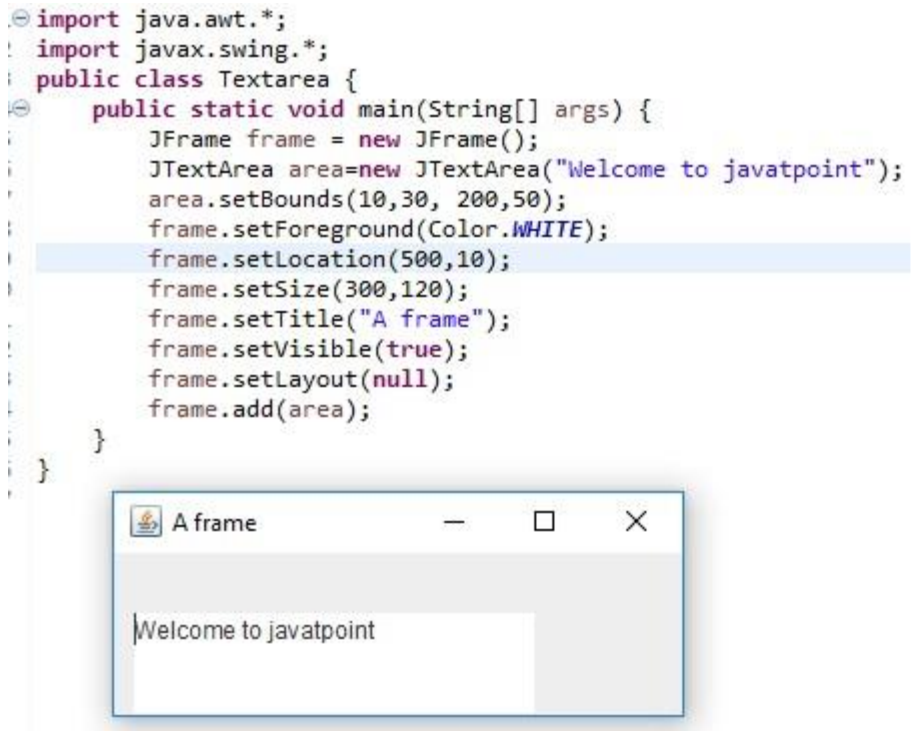
12.3 What is JTextArea?

The object of a JTextArea class is a multi-line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class. There are many methods used for the customization of JTextField. Some of them are given as under:

- setRows()
- setColumns()
- setFont()
- append()

Program Example:

Fig 12.3:



```
import java.awt.*;
import javax.swing.*;
public class Textarea {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        JTextArea area=new JTextArea("Welcome to javatpoint");
        area.setBounds(10,30, 200,50);
        frame.setForeground(Color.WHITE);
        frame.setLocation(500,10);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setVisible(true);
        frame.setLayout(null);
        frame.add(area);
    }
}
```

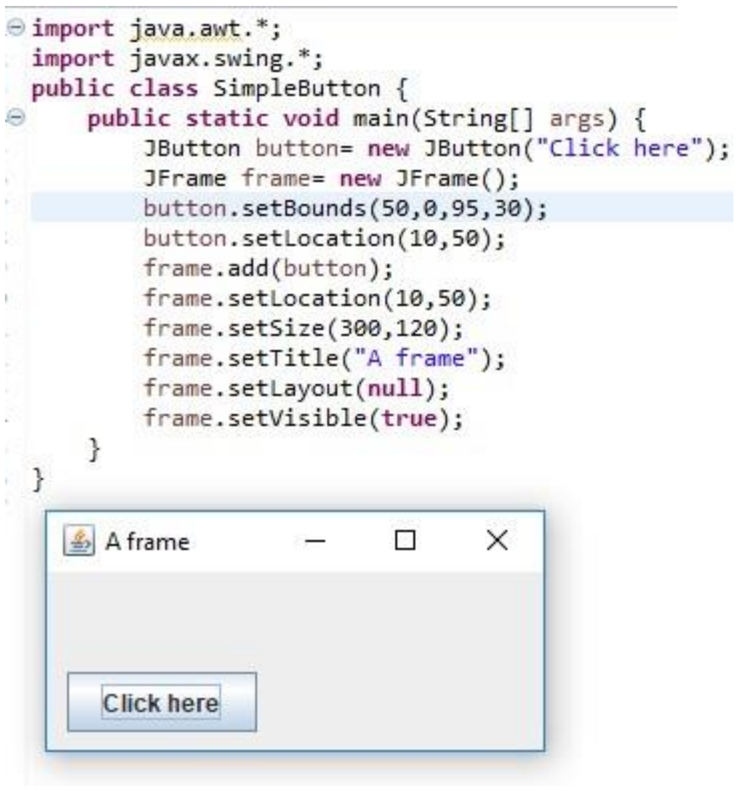
12.4 What is JButton?

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class. JButton includes many methods for the purpose of accommodation. Some of them are given as under:

- void setText(String s)
- String getText()
- void setIcon(Icon b)
- Icon getIcon()

Program Example:

Fig 12.4:



```
import java.awt.*;
import javax.swing.*;
public class SimpleButton {
    public static void main(String[] args) {
        JButton button= new JButton("Click here");
        JFrame frame= new JFrame();
        button.setBounds(50,0,95,30);
        button.setLocation(10,50);
        frame.add(button);
        frame.setLocation(10,50);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```

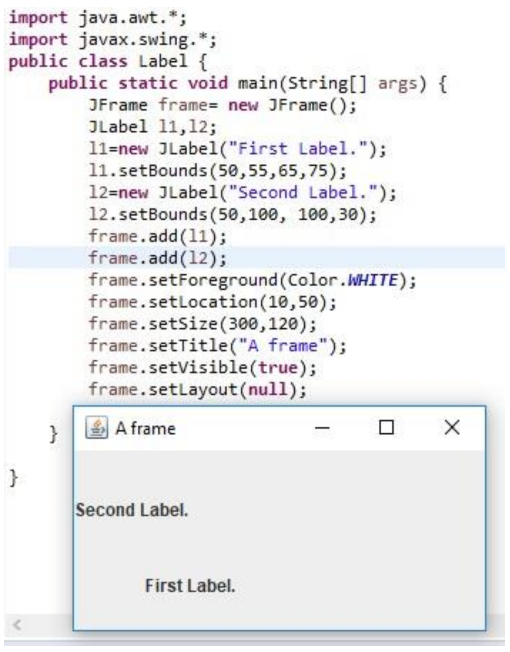
12.5 What is JLabel?

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class. JLabel has several methods used for accommodation. Some of them are given below:

- void setText(String a)
- String getText()
- void setHorizontalAlignment(int align)
- int getHorizontalAlignment()

Program Example:

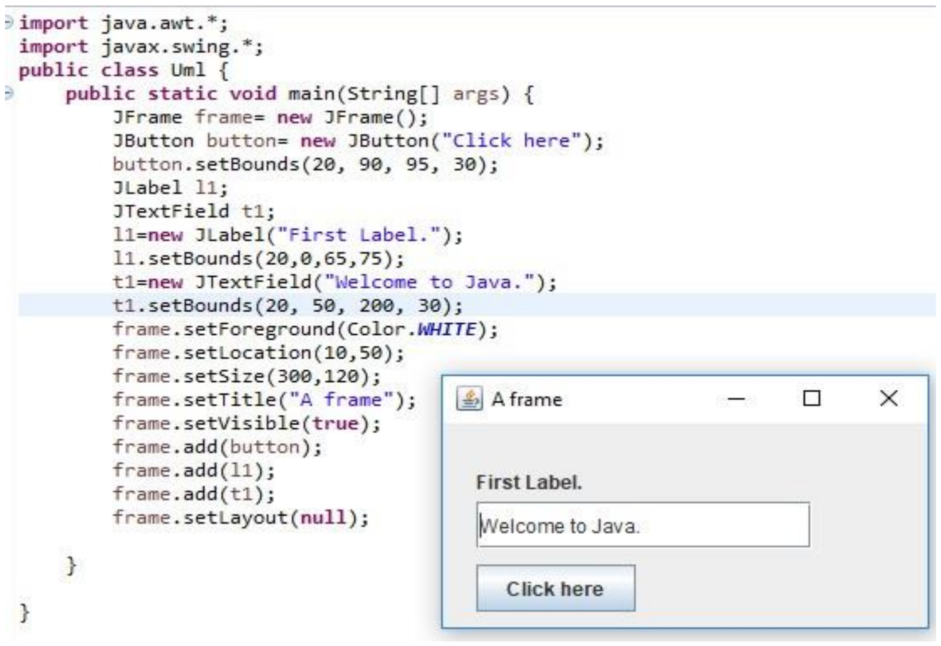
<u>Fig 12.5</u>:



```java
import java.awt.*;
import javax.swing.*;
public class Label {
    public static void main(String[] args) {
        JFrame frame= new JFrame();
        JLabel l1,l2;
        l1=new JLabel("First Label.");
        l1.setBounds(50,55,65,75);
        l2=new JLabel("Second Label.");
        l2.setBounds(50,100, 100,30);
        frame.add(l1);
        frame.add(l2);
        frame.setForeground(Color.WHITE);
        frame.setLocation(10,50);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setVisible(true);
        frame.setLayout(null);
    }
}
```

12.6 Combined Program:

<u>Fig 12.6</u>:



```java
import java.awt.*;
import javax.swing.*;
public class Uml {
    public static void main(String[] args) {
        JFrame frame= new JFrame();
        JButton button= new JButton("Click here");
        button.setBounds(20, 90, 95, 30);
        JLabel l1;
        JTextField t1;
        l1=new JLabel("First Label.");
        l1.setBounds(20,0,65,75);
        t1=new JTextField("Welcome to Java.");
        t1.setBounds(20, 50, 200, 30);
        frame.setForeground(Color.WHITE);
        frame.setLocation(10,50);
        frame.setSize(300,120);
        frame.setTitle("A frame");
        frame.setVisible(true);
        frame.add(button);
        frame.add(l1);
        frame.add(t1);
        frame.setLayout(null);

    }

}
```

TOPIC#13:

JScrollPane

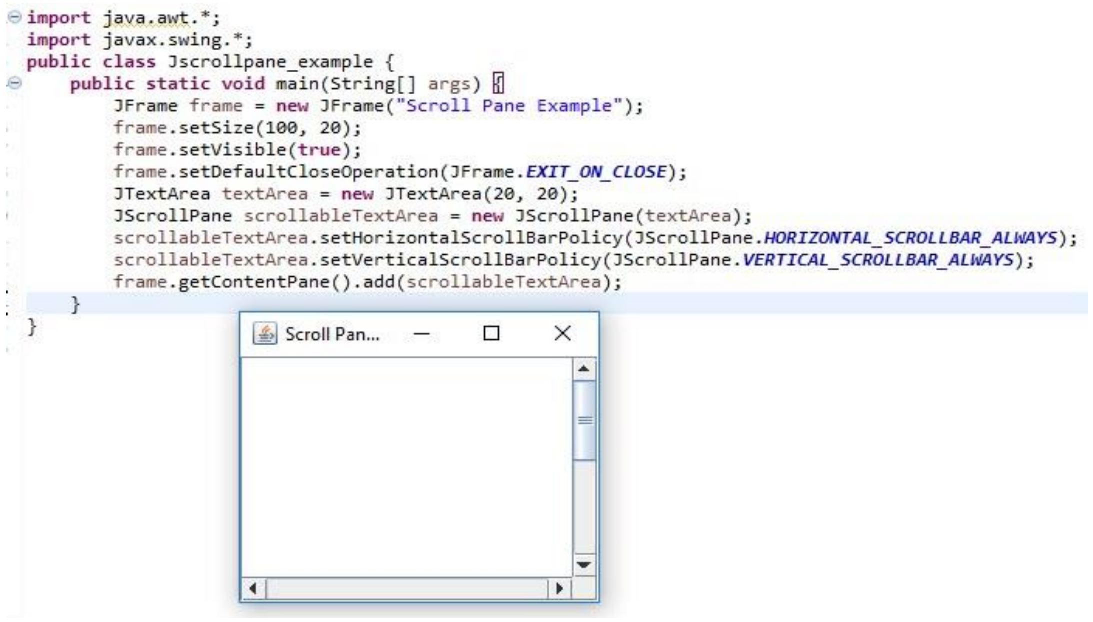
13.1 What is JScrollPane?

A JScrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically. It includes variety of methods which are used for the purpose of customizing a scroll bar. Some of the methods are given as under:

- void setColimnHeaderView (Component)
- void setRowHeaderView (Component)
- void setCorner (String, Component)
- Component getCorner (String)

Program Example:

Fig 13.1:



```
import java.awt.*;
import javax.swing.*;
public class Jscrollpane_example {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Scroll Pane Example");
        frame.setSize(100, 20);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JTextArea textArea = new JTextArea(20, 20);
        JScrollPane scrollableTextArea = new JScrollPane(textArea);
        scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        frame.getContentPane().add(scrollableTextArea);
    }
}
```

-----x----x----x-----

So, that’s all in the “Java For Beginners – A Complete Guide”. We hope so, that it is really a knowledgeable and profitable guide for the people who are interested in learning Java Programming Language.
Thank You very much!

Best Regards,
Salman Abdul Rahim
(Author)

[illegible]