



**REVA**  
UNIVERSITY

Bengaluru, India



# **School of Electronics and Communication Engineering**

**Program : B.Tech. in ECM**

**Java Programming Lab**

**LABORATORY MANUAL**

**B20EP0507**

**V Semester**

**2020-24**

## **Vision of the School**

The School of Electronics and Communication Engineering is envisioned to be a leading centre of higher learning with academic excellence in the field of electronics and communication engineering blended by research and innovation in tune with changing technological and cultural challenges supported with leadership qualities, ethical and moral values.

## **Mission of the School**

**M1** Establish a unique learning environment to enable the students to face the challenges in the field of Electronics and Communication Engineering and explore multidisciplinary which serve the societal requirements.

**M2** Create state-of-the-art laboratories, resources, and exposure to the current industrial trends to enable students to develop skills for solving complex technological problems of current times and provide a framework for promoting collaborative and multidisciplinary activities.

**M3** Promote the establishment of Centres of Excellence in niche technology areas to nurture the spirit of innovation and creativity among faculty and students.

**M4** Offer ethical and moral value-based education by promoting activities which inculcate the leadership qualities, patriotism and set high benchmarks to serve the society

## **Program Educational Objectives**

The Program Educational Objectives of B. Tech in Electronics and Computer Engineering are as follows:

**PEO-1:** Have successful professional career in industry, government, and software organization as innovative engineers

**PEO-2:** Successfully solve engineering problems related to Electronics and Computer Engineering by communicating effectively either as a team or as a team member and lead the team

**PEO-3:** Pursue higher studies and have an attitude of lifelong learning through cultural, technical and outreach activities

**PEO-4:** Serve the society regionally, globally and will take up entrepreneurship for the growth of the economy and generate employment

## CONTENTS

Sl.No.	EXP.No.	EXPERIMENT NAME	PAGE NO.
01		Syllabus	04
02		Introduction to JAVA and procedure to run JAVA program using BlueJ IDE	09
02	Exp-01a	Introduction to Compiling and Executing a Java Program using command prompt.	24
03	Exp-01b	Program to illustrate Data Types and Variables.	25
04	Exp-01c	Write a program to create three variables and find the number of distinct values using branching statements.	26
05	Exp-02a	Write a program to find the zodiac sign for the entered date & month using a branching statement.	27
06	Exp-02b	Write a program to create the following pattern using the looping structures.	30
07	Exp-03a	Write a program to create a BMI calculator that reads the user's weight in kilograms and height in meters, then calculates and displays the user's body mass index. Formula: $BMI = \text{weight (kg)} / [\text{height(m)}]^2$	32
08	Exp-03b	Write a program to calculate the value of $\pi$ from the infinite series.	34
09	Exp-04a	Write a java program to find the area and perimeter of a rectangle using the concept of class and objects.	36
10	Exp-04b	Write a java program to demonstrate copy constructor and constructor overloading.	38
11	Exp-04c	Write a java program to demonstrate function overloading and overriding.	40
12	Exp-05	Write a java program to create a program to superclass <b>MotorVehicle</b> with instance variable.	42
13	Exp-06	Write a java program to create an abstract class <b>Shape</b> which contains the abstract method <b>number Of Sides()</b> .	44
14	Exp-07	Write a program to create an interface <b>Calculator</b> which contains <b>add(), sub(), multiply(), divide(), remainder ()</b> abstract methods with two-parameter x and y.	45
15	Exp-08	Write a program to demonstrate the function on string like <b>to Lower Case(), to Upper Case(), length(), starts With(), ends With(), substring()</b> , and string conversion using <b>String.value Of()</b>	47
16	Exp-09	Write a java program using the concept of packages. Create a class <b>Trigonometry</b> which contain static method <b>sine (), cos (), tan (), cosec (), tan (), cosec (), sec (), cot ()</b> .	49
17	Exp-10a	Write a java class to demonstrate Arithmetic Exception, Null Pointer Exception, Array Index Out Of Bounds Exception.	51

---

18	Exp-10b	Write a java program to demonstrate working with files. (Hint:Files & Exception).	52
19	Exp-11	Write a Java Program using <b>the Runnable</b> interface to demonstrate the concepts of thread priorities. (Hint: Threads).	53
20	Exp-12	Write a java program to demonstrate the multithread and threadsynchronization (Hint: Multithreading)	54

## SYLLABUS

Course Title	Java Programming Lab				Course Type	HC		
Course Code	B20EP0507	Credits	1		Class	V Semester		
Course Structure	LTP	Credits	Contact Hours	Work Load	Total Number of Classes Per Semester		Assessment Weightage	
	Lecture	-	-	-				
	Tutorial	-	-	-				
	Practical	1	22	2	Theory	Practical	CIE	SEE
	<b>Total</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>-</b>	<b>28</b>	<b>50%</b>	<b>50 %</b>

### COURSE OVERVIEW:

Java is an object-oriented language that enables learners to create real-world applications. Java technology- based software works just about everywhere from the smallest devices to super computers! Java technology components are not impacted by the kind of computer, phone, smart device, or operating systems they are running on. The architecture-neutral nature of Java technology is important in a networked world where one cannot predict the kind of devices that partners, suppliers and employees use to connect to their organizations. The Java Programming in course is the first step for developing such applications. This course introduces object-oriented concepts and its implementation in Java technology programs. In addition, it covers syntax and semantics of the Java programming language.

### COURSE OBJECTIVES:

The objectives of this course are to:

1. Illustrate the creation of classes and objects in Java
2. Demonstrate concept reusing of code using inheritance and interfaces
3. Use proper program handling mechanism to write robust programs
4. Familiarize advance java concepts like threads, JDBC, Servlets, JSP

**COURSE OUTCOMES (COs):**

On successful completion of this course; the student shall be able to:

CO#	Course Outcomes	POs	PSOs
CO1	Implement simple programs using Java language concepts such as variables, conditional, methods and constructure	1,3,5	1,2
CO2	Apply program structure like inheritance, interface to develop programs.	1,2,3	1,2
CO3	Build application using the concept of packages,	1,2,3,4	2,3
CO4	Demonstrate the programs using concepts exception handling and filehandling	1,2,3	1,2
CO5	Conduct the software coding for the given design parameters individually (and in a team) within the stipulated time	5,9,10,12	1,2
CO6	Analyze the results, make relevant observations and measurements, and document the results in a form of report/journal.	5,9,10,12	1,2

**BLOOM'S LEVEL OF THE COURSE OUTCOMES**



CO#	Bloom's Level					
	Remember (L1)	Understand (L2)	Apply (L3)	Analyze (L4)	Evaluate (L5)	Create (L6)
CO1			✓			
CO2				✓		
CO3			✓			
CO4			✓			
CO5			✓			
CO6		✓				

**COURSE ARTICULATION MATRIX:**

CO / POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1		2								1	2	
CO2	2	3	2		2								3	2	
CO3	2	2	3		2									2	1
CO4	2	3	2		2								3	2	
CO5					3				3	3		1			
CO6					3				3	3		1			

Note: 1-Low, 2-Medium, 3-High

**Couse Contents:**

Sl. No.	Name of the Practice Session	Tools and Techniques	Expected Skill/Ability
1	a) Introduction to Compiling and Executing a Java Program using command prompt. b) Program to illustrate Data Types and Variables c) Write a program to create three variables and find the number of distinct values using branching statements	Command Prompt/ Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
2	a) Write a program to find the zodiac sign for the entered date & month using a branching statement b) Write a program to create the following pattern using the looping structures. <div style="display: flex; justify-content: space-around; margin-top: 10px;">   </div>	Command Prompt/ Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
3	a) Write a program to create a BMI calculator that reads the user's weight in kilograms and height in meters, then calculates and displays the user's body mass index. Formula: $BMI = \text{weight (kg)} / [\text{height(m)}]^2$ [Reference Values: Underweight: less than 18.5, Normal: between 18.5 and 24.9, Overweight: between 25 and 29.9, Obese: 30 or greater] b) Write a program to calculate the value of $\pi$ from the infinite series $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots$ Print a table that shows the value of $\pi$ approximated by computing the first 200,000 terms of this series. How many terms do you have to use before you first get a value that begins with 3.14159?	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs

4	<p>a) Write a java program to find the area and perimeter of arectangle using the concept of class and objects.</p> <p>b) Write a java program to demonstrate copy constructor and constructor overloading</p> <p>c) Write a java program to demonstrate function overloading and overriding</p>	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
5	<p>Write a java program to create a program to superclass <b>MotorVehicle</b> with instance variable <b>modelName, modelNumber, modelPrice</b>, parameterized constructor, and <b>display ()</b> method. Create a sub-class that inherits the features of the superclass and has its instance variable <b>discountRate</b>, parameterized constructor, and <b>display ()</b>, <b>discount ()</b> methods. Create an object for the car class and invoke all the methods using the object of that class. (Hint: Single Inheritance)</p>	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
6	<p>Write a java program to create an abstract class Shape which contains the abstract method numberOfSides(). Create different sub-classes by the name Trapezoid, Triangle, and Hexagon, which extends the shape class. Develop a class ShapeDemo which contains the main method. Create the object for different subclasses with the main method and invoke the method numberOfSides() using the objects of classes. (Hint: Abstract Class)</p>	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
7	<p>Write a program to create an interface Calculator which contains add(), sub(), multiply(), divide(), remainder () abstract methods with two-parameter x and y. Develop a class CalculatorDemo which inherits the features of the interface. Create an object for the CalculatorDemo class and invoke all the methods of this class. (Hint: implementing interfaces/ multiple inheritances)</p>	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
8	<p>Write a program to demonstrate the function on string like toLowerCase(), toUpperCase(), length(), startsWith(), endsWith(), substring(), and string conversion using String.valueOf() (Hint: String Handling)</p>	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs



9	Write a java program using the concept of packages. Create a class Trigonometry which contain static method sine (), cos (), tan (), cosec (), tan (), cosec (), sec (), cot (). Print the value of a given angle in degree by calling these methods. (Hint: Implementing packages)	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
10	a) Write a java class to demonstrate ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException (Hint: Handling predefined exceptions) b) Write a java program to demonstrate working with files (Hint: Files & Exception)	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
11	Write a Java Program using the Runnable interface to demonstrate the concepts of thread priorities. (Hint: Threads)	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs
12	Write a java program to demonstrate the multithread and thread synchronization (Hint: Multithreading)	Eclipse IDE / BlueJ	Apply Java Programming concepts to write programs

**TEXTBOOKS:**

1. Patrick Naughton, "The Java Handbook", Tata McGraw-Hill, 2006
2. Herbert Schildt, Java™: The Complete Reference, McGraw-Hill, Tenth Edition, 2018.

**REFERENCE BOOKS:**

1. Bruce Eckel, "Thinking in Java", III Edition, Pearson 2004.
2. Y. Daniel Liang, Introduction to Java programming-comprehensive version-Tenth Edition, Pearsonltd 2015
3. Paul Deitel Harvey Deitel ,Java, How to Program, Prentice Hall; 9th edition , 2011
4. E Balagurusamy, Programming with Java A primer, Tata McGraw Hill companies.

## Introduction

Java is a programming language developed at Sun Microsystem in 1995. It was designed by James Gosling. The language derives much of its syntax from C++ (and its predecessor, C) but is simpler to use than C++. The reputation of Java has spread wide and far and has captured the imagination of most software professionals. Literally, thousands of applications are being built today in Java for different platforms, including desktop, web, and mobile. Possibly why java seems so popular is because it is reliable, portable, and easy to use; it has a modern construct that is required today to represent problems programmatically. Java, like C++, makes use of a principle called Object-Oriented Programming (OOP) to organize the program.

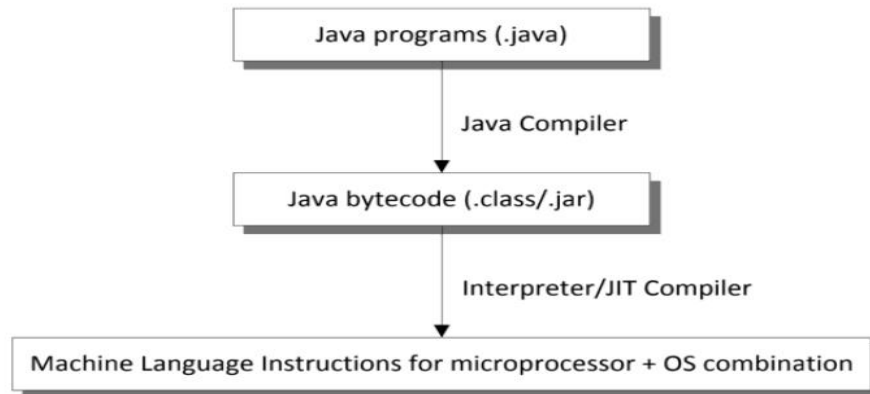
## How Java is Different

Java takes a different approach than the traditional approach taken by languages like c and C++. It lets application developers follow a “Compile Once, Run Anywhere” scenario. This means that once a java program compiles, it can get executed on different microprocessor + OS combinations without the need to recompile the program. This makes the Java program immensely portable across different microprocessor + OS combinations. The microprocessor + OS combinations are often called “Platform”. Hence Java is often called Platform-independent language or architecturally neutral language.

Java achieves this “Compile Once, Run Anywhere” and platforms independent magic through a program called Java Virtual Machine (JVM). When we compile java programs, they are not converted into machine language instruction for a specific microprocessor + OS combination. Instead, our Java Program has converted into bytecode instruction. These bytecode instructions are similar to machine code but are intended to be interpreted by JVM. A JVM provides an environment in which java bytecode can be executed.

During execution, the JVM runtime executes the bytecode by interpreting it using an interpreter program or compiling it using a just-in-time (JIT) compiler. JIT compilers are preferred as they work faster than interpreters. During interpretation or JIT compilation, the bytecode instruction is converted into machine language instruction for the microprocessor OS combination on which the program is being executed. This perfectly facilitates executing java programs on different machines connected to the internet.

A Java Program is typically stored in a .java file, and the bytecode is usually stored in a .class file. A complex program may consist of many .class files. For easier distribution, these multiple class files may be packaged together in a .jar file (short for java archive). The working of a java program discussed above is shown in the figure below.



### Features of Java

1. **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it will be easy to master.
2. **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
3. **Distributed:** Java is designed for the distributed environment of the internet.
4. **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 lightweight process.
5. **Robust:** Java makes an effort to eliminate error-prone situations by emphasizing mainly compile time error checking and runtime checking.
6. **Secure:** With Java's secure feature it enables to development of virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
7. **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.
8. **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.
9. **Portable:** Being architecture-neutral and having no implementation-dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
10. **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
11. **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.
12. **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 run-time information that can be used to verify and resolve access to objects at run-time.

## Tools

To create and run a java program, you need to install two software on your PC. These are

- a) Java Development Kit (JDK)
- b) Integrated Development Environment (IDE ) like BlueJ, Eclipse, NetBeans, Microsoft Visual Studio, etc.

## Executing Java Program using BlueJ IDE

```
//Ex: Let us consider a sample program

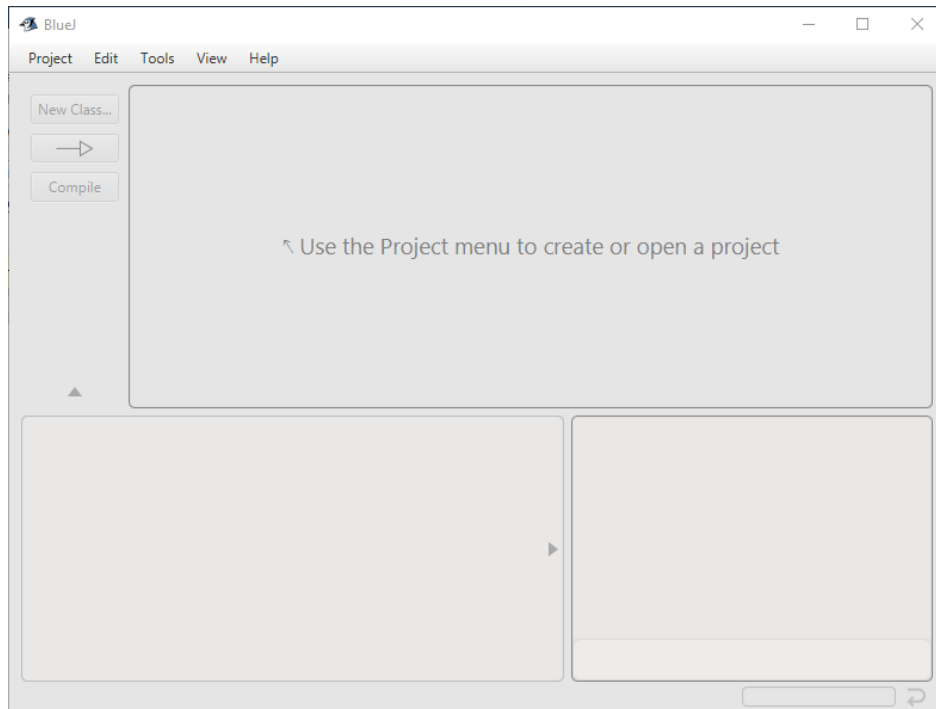
class Rectangle
{
    int l, b;
    Rectangle() {
        l = 10; b = 20;
    }

    Rectangle(int x, int y) {
        l = x; b = y;
    }
    int area() {
        return l*b;
    }
}

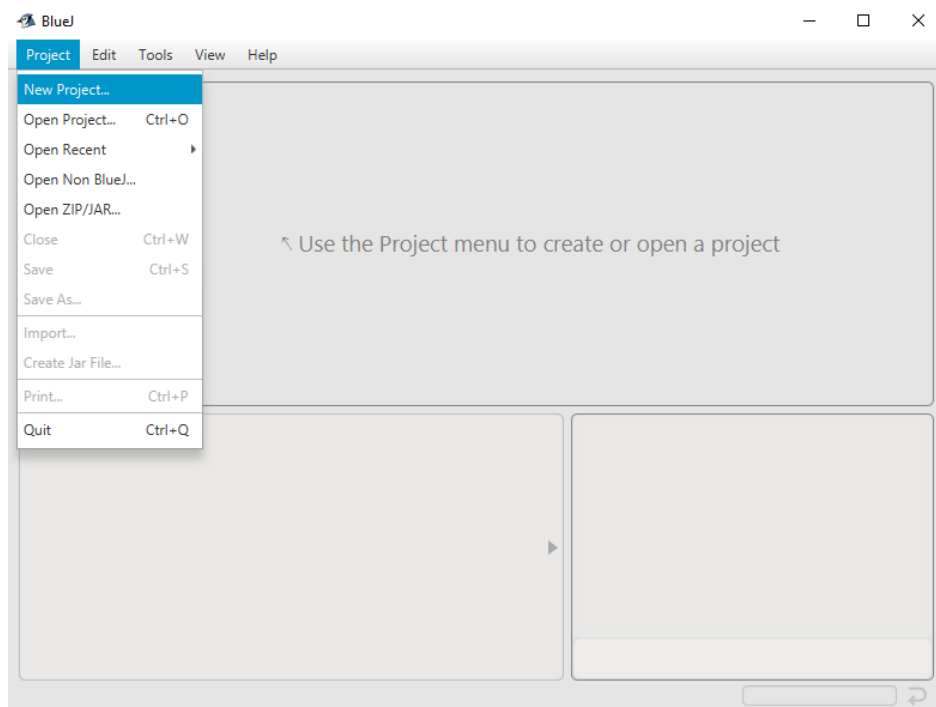
class TestClass {
    public static void main(String args[]) {
        Rectangle rectangle1 = new Rectangle();
        System.out.println("The area of a rectangle using
            first constructor is: "+rectangle1.area());
        Rectangle rectangle2 = new Rectangle(4,5);
        System.out.println("The area of a rectangle using
            second constructor is: "+rectangle2.area());
    }
}
```

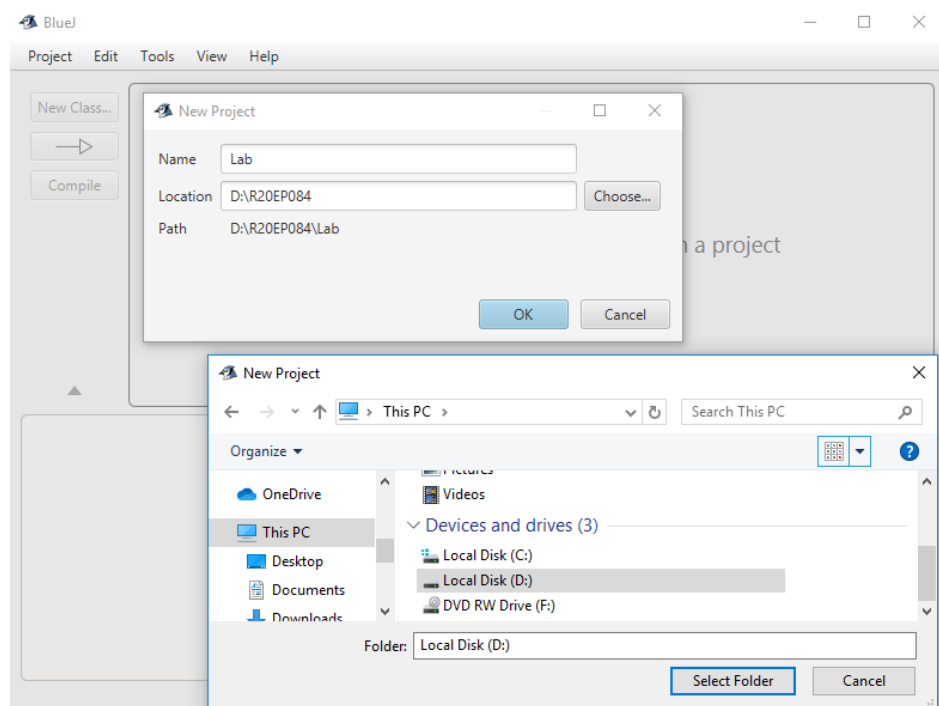
1. Download & Install JDK [https://download.oracle.com/java/18/latest/jdk-18\\_windows-x64\\_bin.exe](https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.exe)
2. Download & Install BlueJ <https://www.bluej.org/download/files/BlueJ-windows-503.msi>
3. Start BlueJ by either clicking on the icon you created when you downloaded and installed BlueJ

4. When BlueJ is up and running, you should see a window that looks something like this:

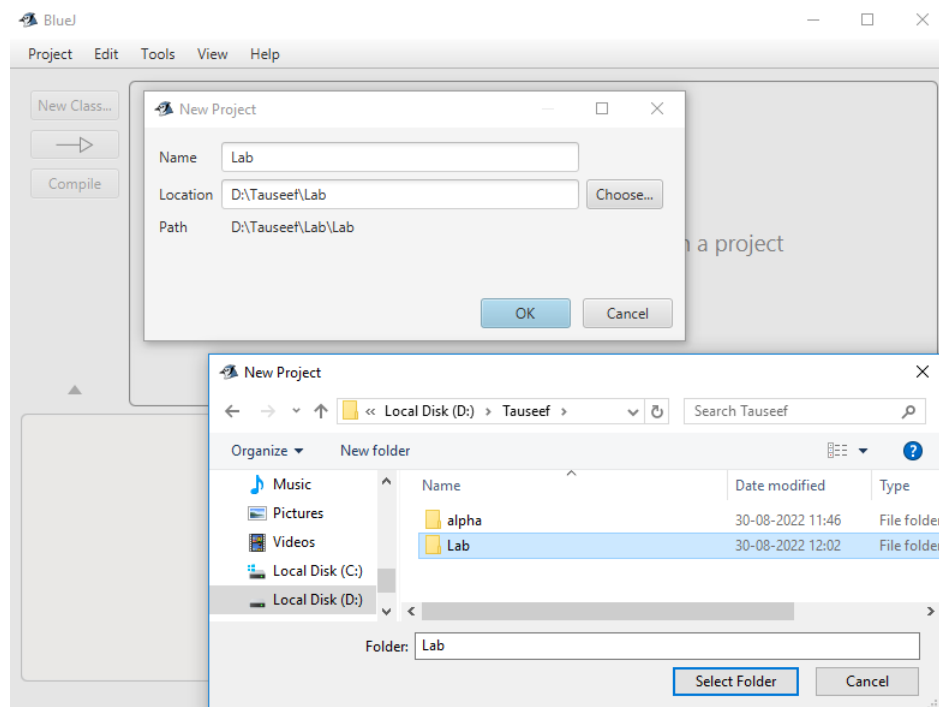


5. To create a new program, you first need to create a new project. Click on the Project menu and select New Project...



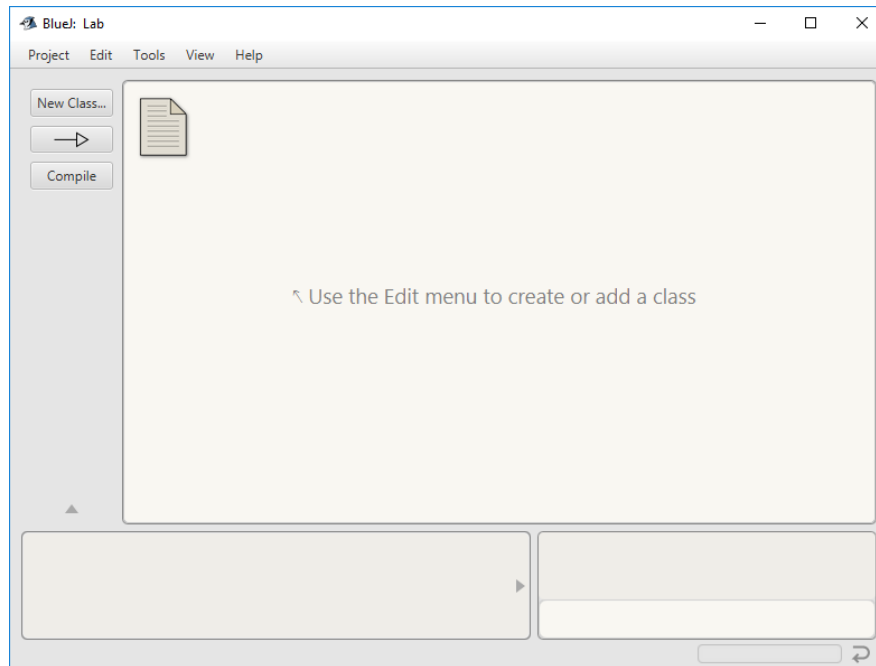


6. This opens a new window.



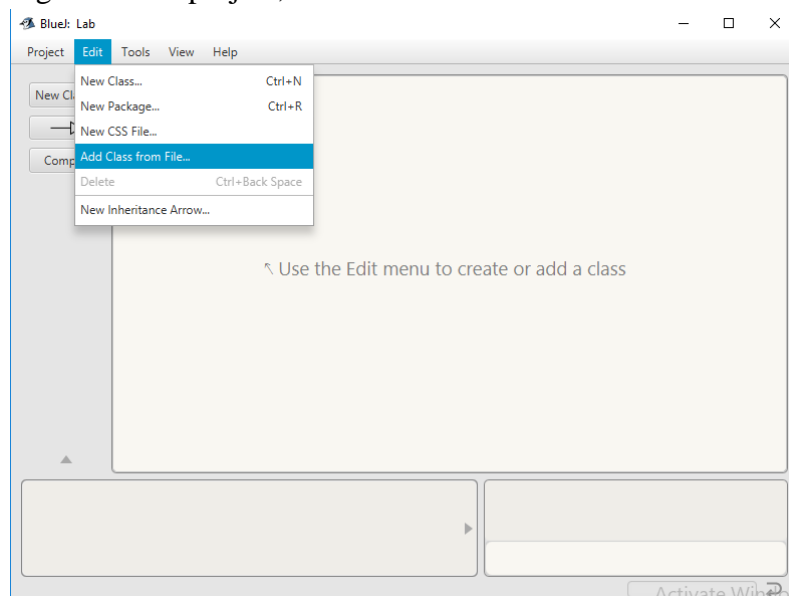
From this window, type in the new project name in the text field **File name**. The name of the project does not have to be the same as the name of the java file you will later add to the project. It can be anything you want.

7. After giving the new project and name and clicking Create, you see the new project.

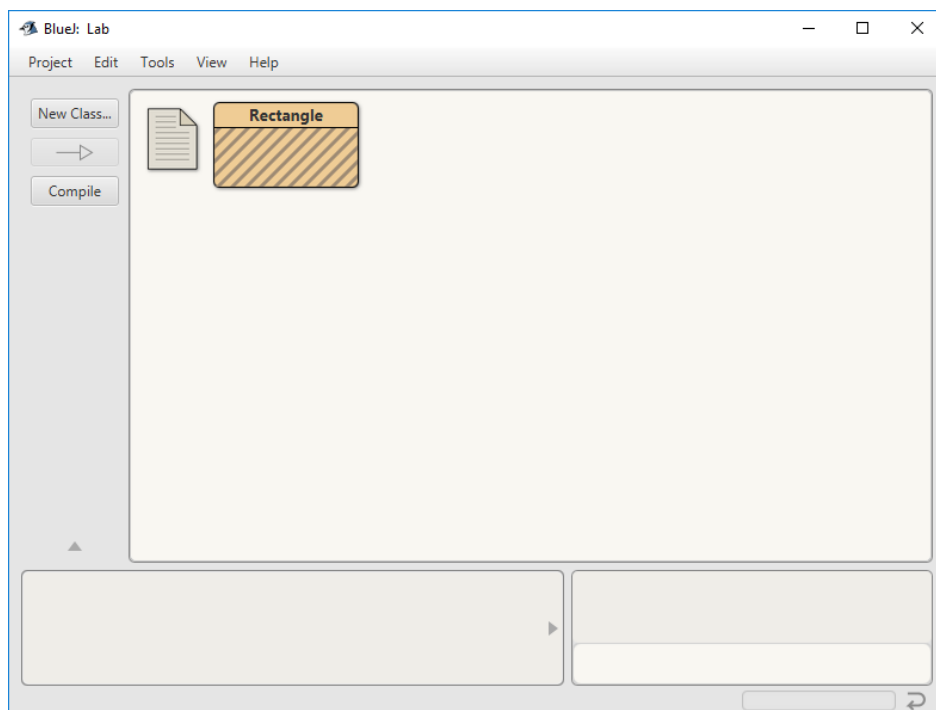
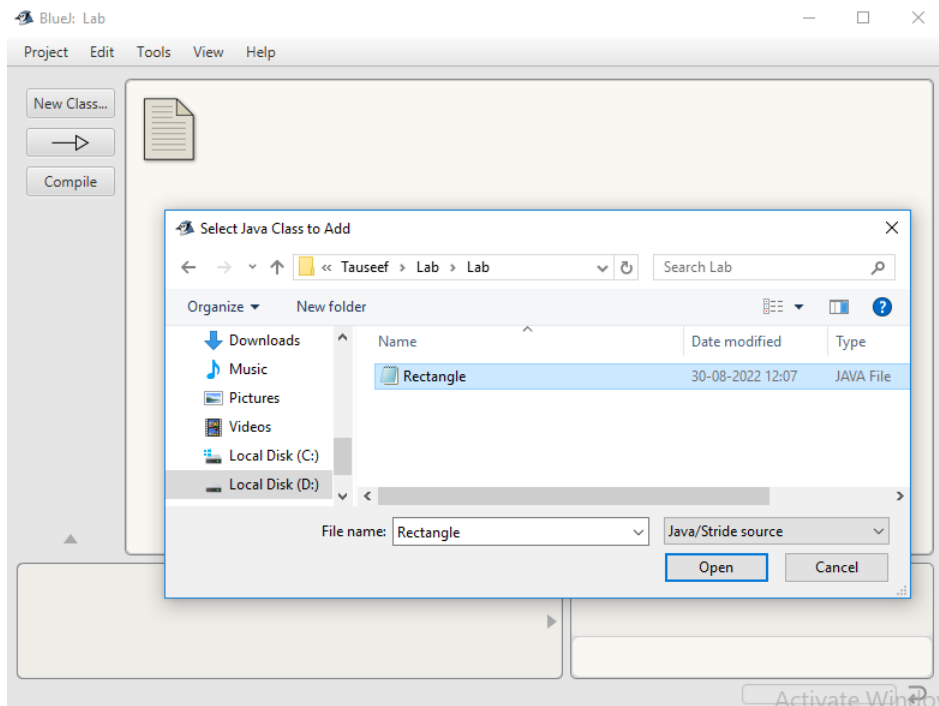


There aren't any classes, which are the basic building blocks of programs in Java, or code associated with the project yet. The icon that looks like a piece of paper is just a simple text file that acts as the project's readme file. You can double-click on it to add comments about the project as a whole or write notes to yourself on things that need to be done. You probably won't use it much at first.

8. If you need to add any pre-existing classes to the project, now is a good time to do that. To add the existing file to the project, click on the Edit menu and select Add Class from File...

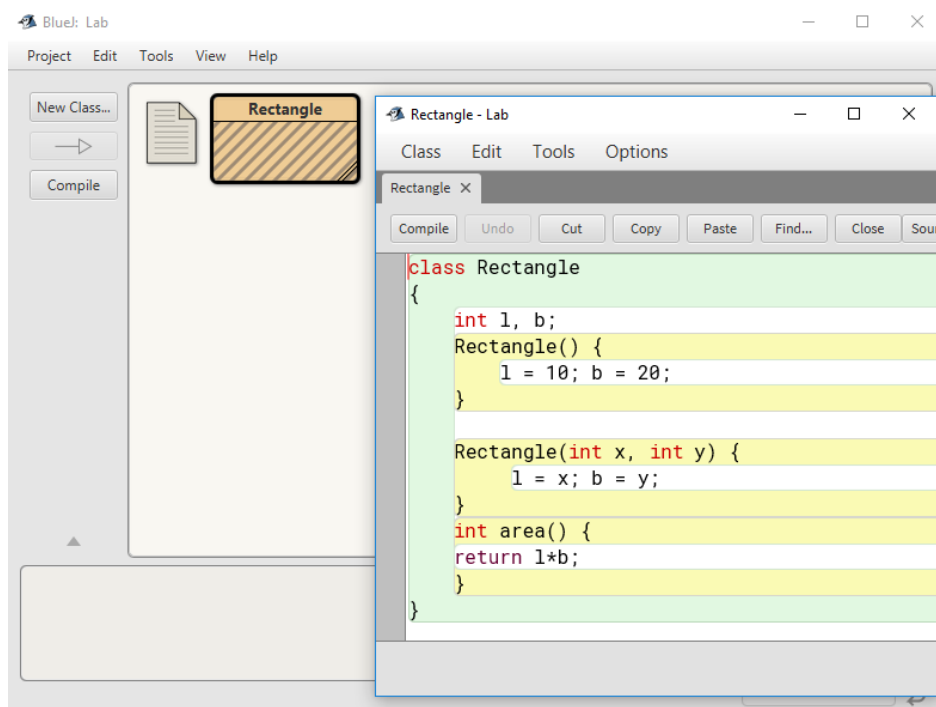


9. A new window pops up that allows you to navigate to the correct directory and select the file you want to add. Only files that end in **.java** are shown.

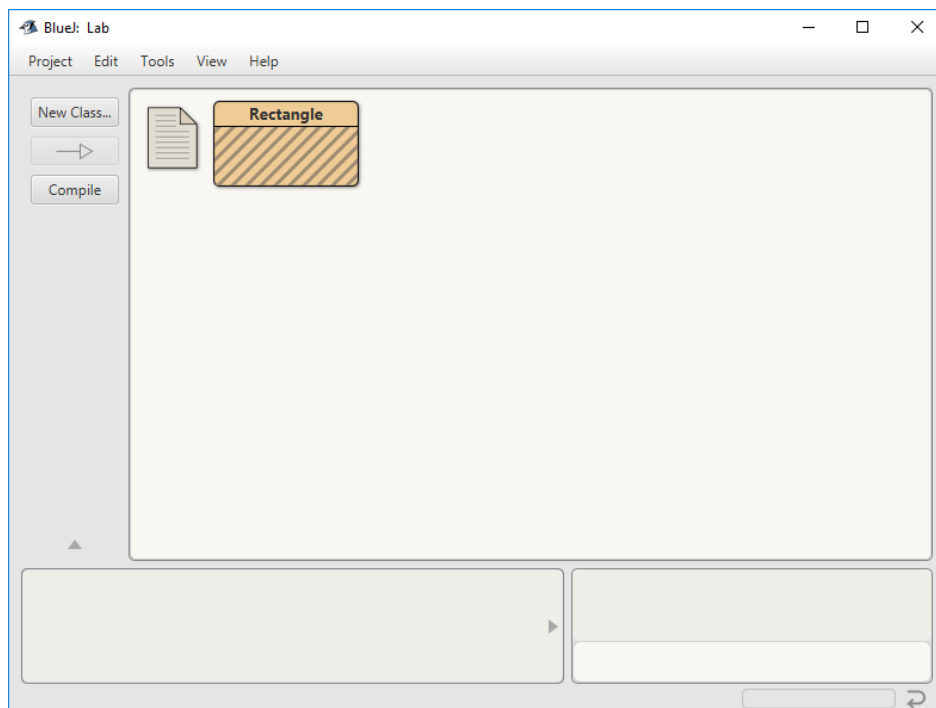


Navigate to the directory that contains the file or files you want to add. If you want to add a single file, click on it and then click on the **Add** button. If you're going to add two or more files from the same directory, hold down the **CTRL** key and click on all the files you want to add, then click on the **Add** button.

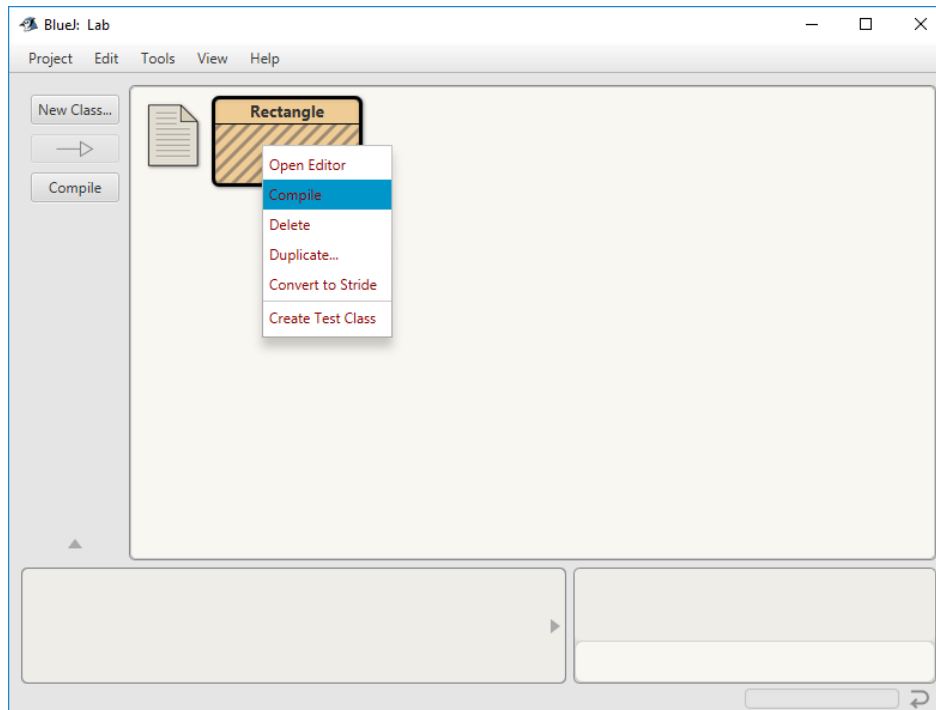




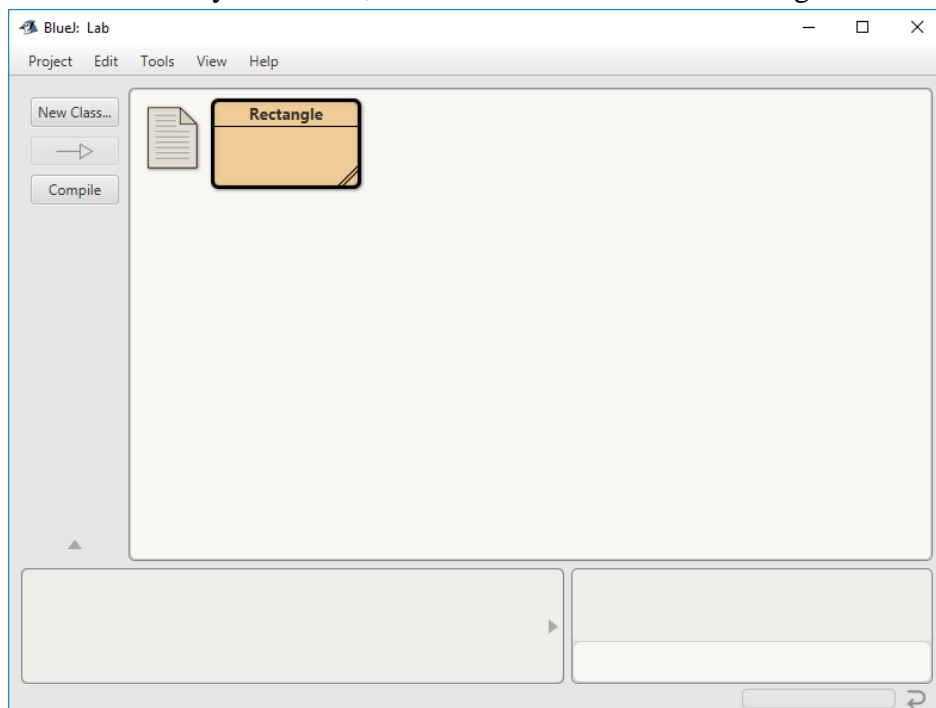
10. The new file/class has been added to the file. It shows up with diagonal lines because it has not been compiled yet.



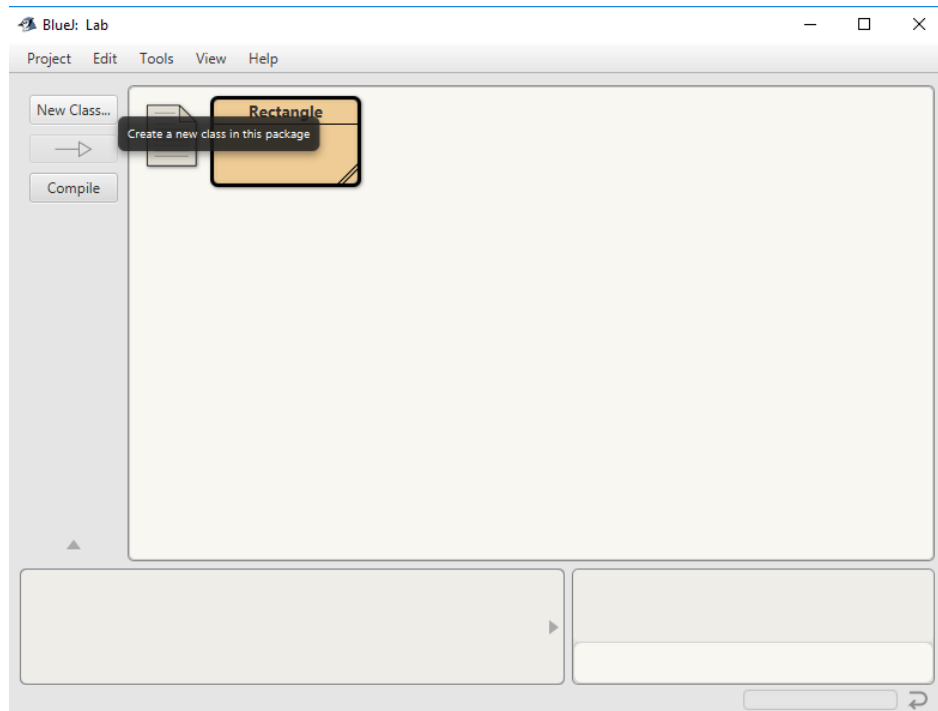
11. To compile a class that is part of a program, you can either click the **Compile** button on the left side of the window or right-click on the icon of the class or select the **Compile** option from the pop-up menu.



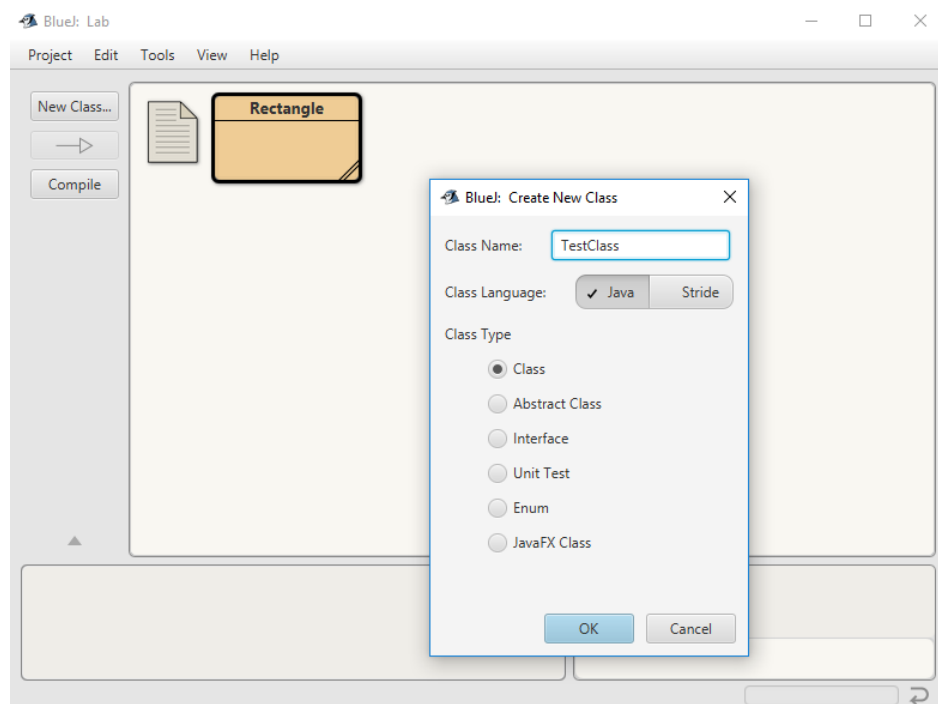
If the class does not have syntax errors, it will now be shown without diagonal lines on its icon.



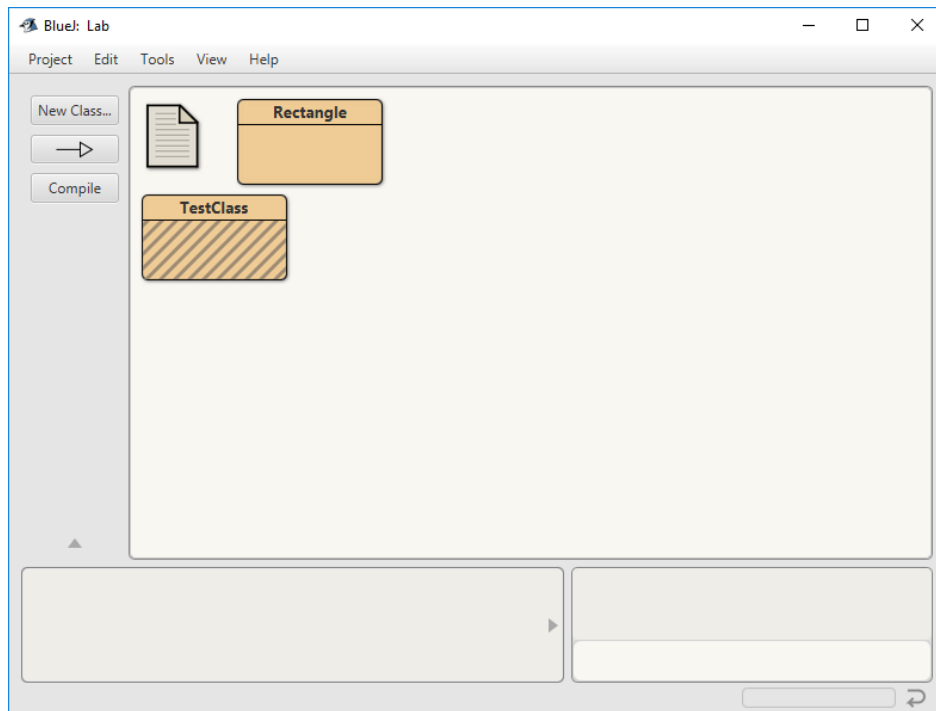
12. To add a new class to the project, click the **New Class** button.



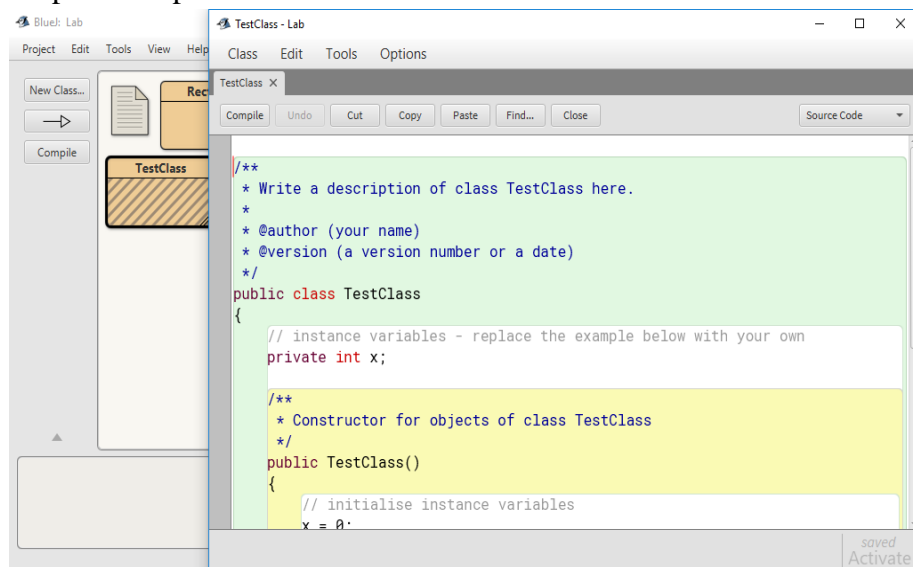
13. A menu pops up asking for the name of the class. In the example, let us name the class as Test Class



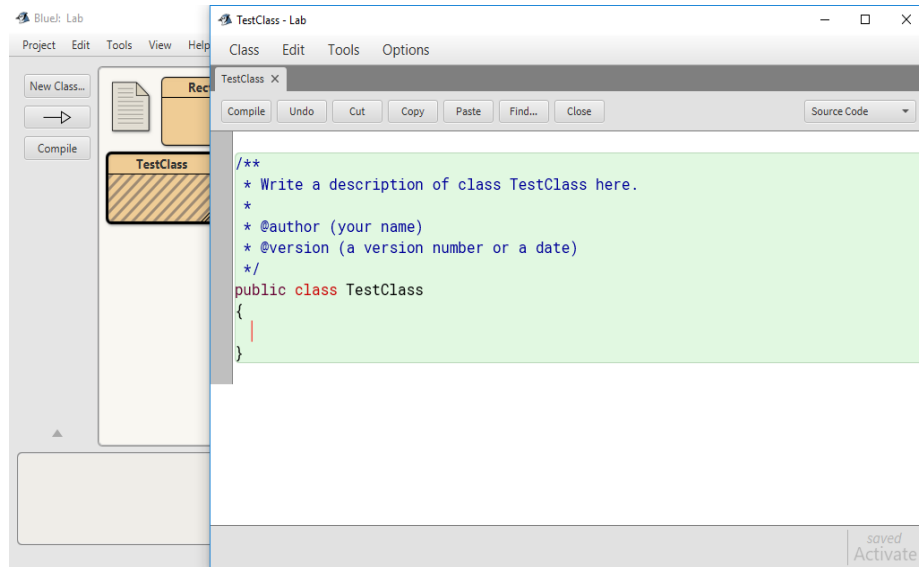
Ensure the Class radio button is selected, type in Box for the Class Name, and click the okay button. A new class has been added to our project.



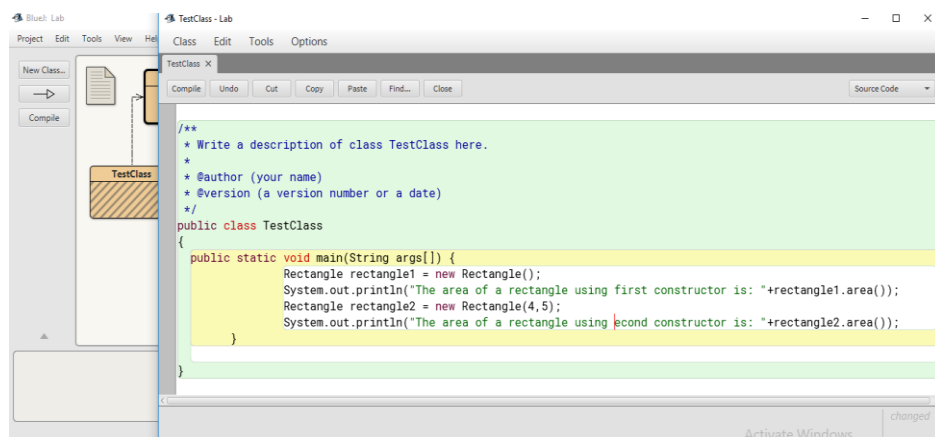
14. To edit the code in a class, double-click on its icon. This opens an editor window which is very like a simple word processor.

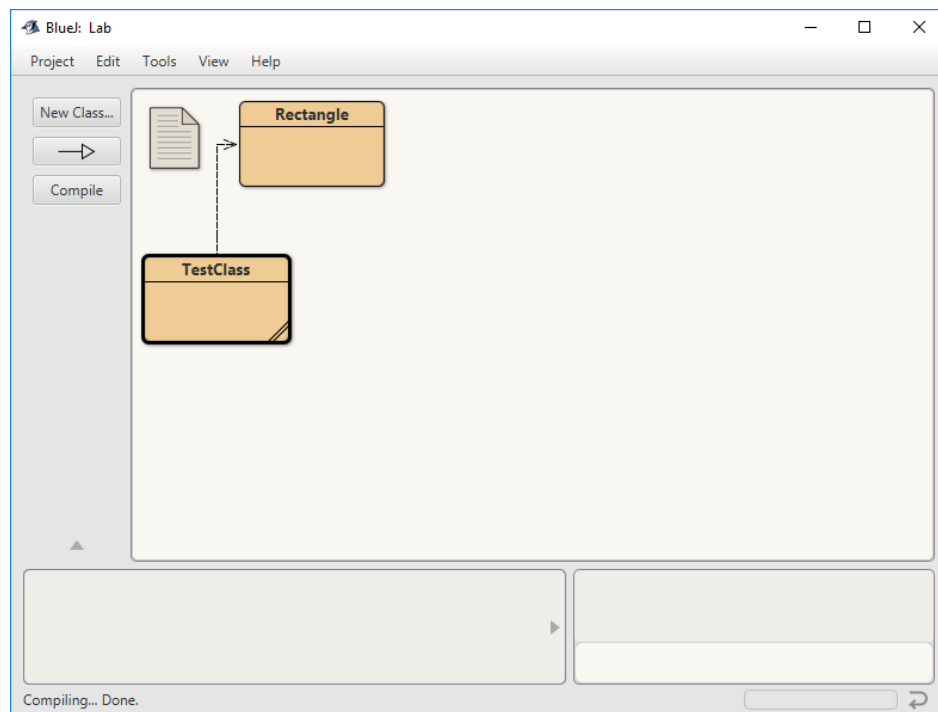
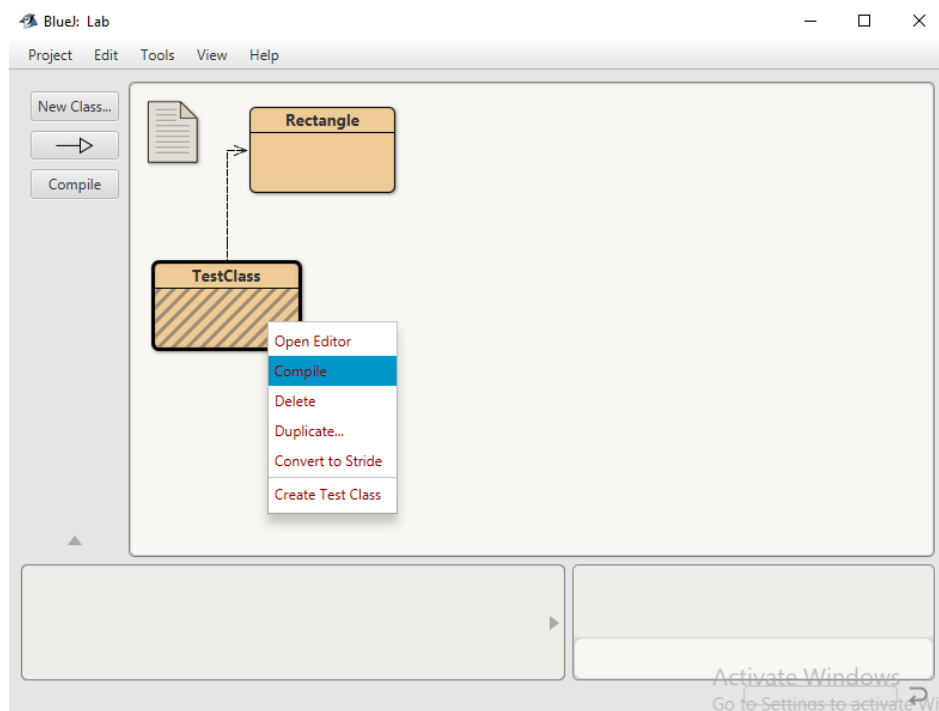


Select and delete the default code in the class.

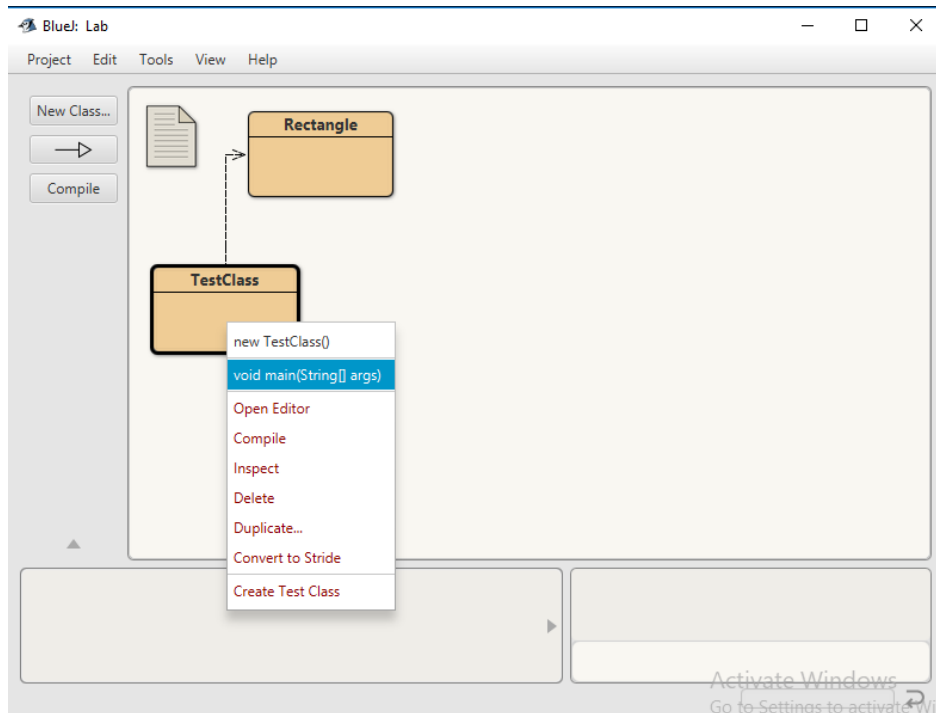


15. Type in the changes to the program. You save the program by clicking on the **Class** menu and selecting **Save**. You can attempt to compile the program by clicking the **Compile** button. Any errors will appear in the text area at the bottom of the edit window. If an error occurs, the line is highlighted in red. After compiling the program and getting it to work, close the editing window and go back to the project window. The program/class icon no longer has diagonal lines across it.

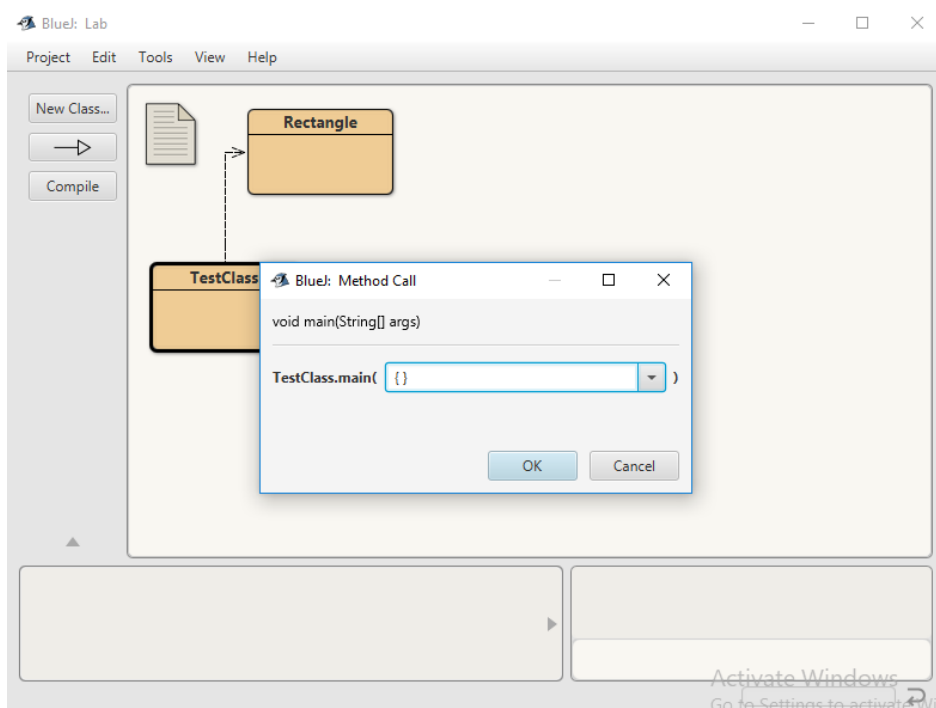




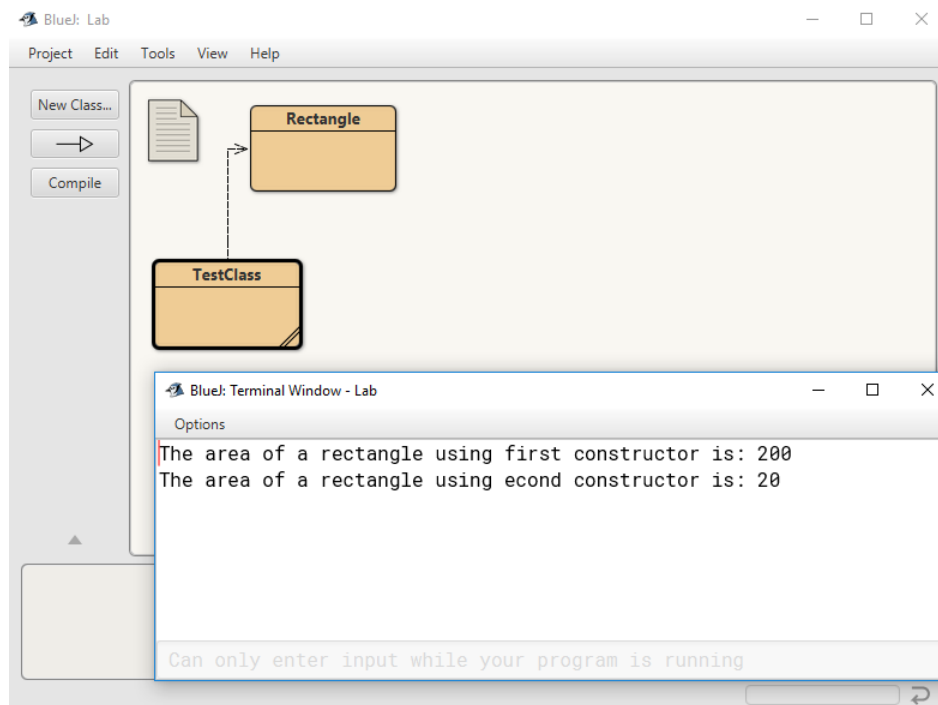
16. To run the program, click on the icon and select the main method. This brings up a menu that allows you to carry out various actions. (You could also compile classes here.) Select the menu option **void main(args)**. This causes the main method of the class to be executed.



17. Another menu pops up that asks what parameters or information you want to pass to the method, in this case, the main. In this class, we won't usually pass any info to method main, so click **Ok**.



18. This causes the program to run. In this example, a separate window opens, and you may have to click on that window to see the program run.





**EXPERIMENT-01a**

**AIM:** Introduction to Compiling and Executing a Java Program using command prompt.

***Program:***

```
public class GreetUser
{
    public static void main(String args[])
    {
        System.out.println("Hello, " + args[0] + " How are you today?");
    }
}
```

**Output:**

```
//Command line Arguments
C:\Lab> java GreetUser Tauseef
Hello, Tauseef How are you today?
```

**EXPERIMENT-01b**

**AIM:** Program to illustrate Data Types and Variables.

```
public class DataTypes {  
    public static void main(String[] args) {  
        byte b1 = 2, b2 = 3;  
        //byte b3 = b1 + b2; //Will not compile  
        int b3 = b1 + b2; //Will compile  
        System.out.println("Addition of two-byte variables is an int, Result = " + b3);  
        int c = 66;  
        System.out.println("Character at Ascii value: " + c + " is " + (char) c);  
        float f = 4.28f;  
        System.out.println("suffix F or f for a float variable " + f);  
        double d = 1e308;  
        System.out.println("double variable: " + d);  
    }  
}
```

***Output:***

Addition of two-byte variables is an int, Result = **5**  
Character at Ascii value: 66 is **B**  
suffix F or f for a float variable **4.28**  
double variable: **1.0E308**

**EXPERIMENT-01c**

**AIM:** Write a program to create three variables and find the number of distinct values using Branching statements.

**Program:**

```
public class Distinct {  
    public static void main(String args[]) {  
        int a = 1, b = 1, c = 2;  
        if ((a == b) && (a == c)) {  
            System.out.println("No of distinct value = 0");  
        } else if (((a == b) && (a != c)) || ((a == c) && (a != b)) || (b == c)) {  
            System.out.println("No of distinct value = 2");  
        } else  
            System.out.println("No of distinct value = 3");  
    }  
}
```

**Output:**

No of distinct value = 2

**EXPERIMENT-02a**

**AIM:** Write a program to find the zodiac sign for the entered date & month using a branching Statement.

**Note:**

*Capricorn (December 22 – January 19)*  
*Aquarius (January 20 – February 18)*  
*Pisces (February 19 – March 20)*  
*Aries (March 21 – April 19)*  
*Taurus (April 20 – May 20)*  
*Gemini (May 21 – June 20)*  
*Cancer (June 21 – July 22)*  
*Leo (July 23 – August 22)*  
*Virgo (August 23 – September 22)*  
*Libra (September 23 – October 22)*  
*Scorpio (October 23 – November 21)*  
*Sagittarius (November 22 – December 21)*

**Program:**

```
public class Zodiac {
    public static void main(String args[]) {
        int a, b;
        a = Integer.parseInt(args[0]); // Month
        b = Integer.parseInt(args[1]); // Date
        int z = 15;
        String z1[] =
{"Capricorn", "Aquarius", "Pisces", "Aries", "Taurus", "Gemini", "Cancer", "Leo", "Virgo", "Libra",
"Scorpio", "Sagittarius"};

        switch (a) {
            case 1:
                if ((b < 20)) {
                    z = 0;
                } else if (b > 19 && b <= 31) {
                    z = 1;
                }
                break;
            case 2:
                if (b < 18) {
                    z = 1;
                } else if (b > 17 && b < 30) {
                    z = 2;
                }
                break;
        }
    }
}
```

```
case 3:
    if (b < 20) {
        z = 2;
    } else if (b > 19 && b < 31) {
        z = 3;
    }
    break;
```

```
case 4:
    if (b < 20) {
        z = 3;
    } else if (b > 19 && b < 30) {
        z = 4;
    }
    break;
```

```
case 5:
    if (b < 21) {
        z = 4;
    } else if (b > 20 && b < 31) {
        z = 5;
    }
    break;
```

```
case 6:
    if (b < 21) {
        z = 5;
    } else if (b > 20 && b < 30) {
        z = 6;
    }
    break;
```

```
case 7:
    if (b < 23) {
        z = 6;
    } else if (b > 22 && b < 31) {
        z = 7;
    }
    break;
```

```
case 8:
    if (b < 23) {
        z = 7;
    } else if (b > 22 && b < 31) {
        z = 8;
    }
    break;
```

```
case 9:
    if (b < 23) {
```

```
        z = 8;
    } else if (b > 22 && b < 30) {
        z = 9;

    }
    break;

case 10:
    if (b < 23) {
        z = 9;
    } else if (b > 22 && b < 31) {
        z = 10;
    }
    break;

case 11:
    if (b < 22) {
        z = 10;
    } else if (b > 21 && b < 30) {
        z = 11;
    }
    break;

case 12:
    if (b < 22) {
        z = 11;
    } else if (b > 21 && b < 31) {
        z = 0;
    }
    break;

default:
    System.out.println("Month is not valid");
}
if ((z >= 0) && (z <= 11))
    System.out.println("Zodiac sign is: " + z1[z]);
else
    System.out.println("Date is not valid");
}
}
```

**Output:**

```
//Command Line Arguments
java Zodiac 6 7
Zodiac sign is: Cancer
```

**EXPERIMENT-02b**

**AIM:** Write a program to create the following pattern using the looping structures.

```

C:\ Command Prompt
D:\javaprg>java Pattern
  A
 ABA
ABCBA
ABCD CBA
ABCDEDCBA
ABCDEFEDCBA
ABCDEFGFEDCBA
D:\javaprg>

```

```

C:\ Command Prompt
D:\javaprg>java Pattern
  A
 AB
 ABC
ABCD
ABCDE
ABCDEF
ABCDEFG
D:\javaprg>_

```

**Program:**

```

public class Pattern {
    public static void main(String args[]) {
        int i, j, k, l;
        for (i = 0; i < 7; i++) {
            /* A for loop for printing blank spaces. First time 7 blank spaces will be printed and
            henceforth the number of spaces will reduce by 1 in every iteration */
            for (j = 7; j > i; j--)
                System.out.print(" ");
            /* ASCII value of A is taken and based value of i the loop is executed to print the
            alphabets. First time the only A will be printed as the value of i is 0. Second time AB
            will be printed by this loop and soon. */
            for (k = 65; k <= 65 + i; k++)
                System.out.print((char)(k));
            /*This loop prints the alphabets in reverse order*/
            for (l = k - 1; l > 65; l--)
                System.out.print((char)(l - 1));
            System.out.println();
        }
    }
}

```

**Output:**

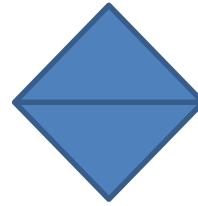
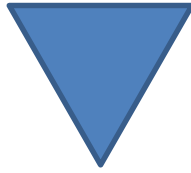
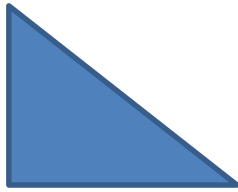
```

  A
 ABA
ABCBA
ABCD CBA
ABCDEDCBA
ABCDEFEDCBA
ABCDEFGFEDCBA

```

**Challenging experiment:**

1. Write a program for the given pattern using the looping structures.





**EXPERIMENT-03a**

**AIM:** Write a program to create a BMI calculator that reads the user's weight in kilograms and height in meters, then calculates and displays the user's body mass index. Formula:  $BMI = \text{weight (kg)} / [\text{height(m)}]^2$

[Reference Values: Underweight: less than 18.5, Normal: between 18.5 and 24.9, Overweight: between 25 and 29.9, Obese: 30 or greater]

**Program:**

```
import java.util.Scanner;

public class BMICalculator{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);

        double weight, height, bmi;

        System.out.printf("Input weight in kilograms: ");
        weight = input.nextDouble();

        System.out.printf("Input height in feet: ");
        height = 0.3048 * input.nextDouble();

        bmi = calculateMetric(weight, height);

        System.out.printf("Your BMI : %.1f\n", bmi);

        printBmiTable();
    }

    // calculate using metric measures
    private static double calculateMetric(double weight, double height){
        return weight / (height * height);
    }

    // National Institutes of Health.
    private static void printBmiTable() {
        System.out.println("\nBMI VALUES REFERENCE CHART");
        System.out.println("Underweight: less than 18.5");
        System.out.println("Normal: between 18.5 and 24.9");
        System.out.println("Overweight: between 25 and 29.9");
        System.out.println("Obese: 30 or greater");
    }
}
```

**Output:**

Input weight in kilograms: **62**

Input height in feet: **5.6**

Your BMI: **21.3**

**EXPERIMENT-03b**

**AIM:** Write a program to calculate the value of  $\pi$  from the infinite series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots\dots\dots$$

Print a table that shows the value of  $\pi$  approximated by computing the first 200,000 terms of this series. How many terms do you have to use before you first get a value that begins with 3.14159?

**Program:**

```
public class ValueOfPi {
    private static final long TERMS = 200000;

    public static void main(String[] args) {
        double infiniteSeries = 0.0f;
        boolean sign = true;
        long count = 0;

        for (int i = 1; i <= TERMS; i += 2) {
            // only compute odd numbers
            if (i % 2 == 0)
                continue;

            // check if addition or subtraction
            if (sign)
                infiniteSeries += (4.0 / (double) i);
            else
                infiniteSeries -= (4.0 / (double) i);

            System.out.printf("%d. i = %d, infiniteSeries = %f\n", ++count, i, infiniteSeries);

            // reverse the sign
            sign = !sign;
        }

        System.out.printf("PI calculated from the infinite series 20,000 times: %f\n",
            infiniteSeries);
    }
}
```

***Output:***

```
1. i = 1, infiniteSeries = 4.000000
2. i = 3, infiniteSeries = 2.666667
3. i = 5, infiniteSeries = 3.466667
4. i = 7, infiniteSeries = 2.895238
5. i = 9, infiniteSeries = 3.339683
6. i = 11, infiniteSeries = 2.976046
7. i = 13, infiniteSeries = 3.283738
8. i = 15, infiniteSeries = 3.017072
9. i = 17, infiniteSeries = 3.252366
10. i = 19, infiniteSeries = 3.041840
...
...
...
99998. i = 199995, infiniteSeries = 3.141583
99999. i = 199997, infiniteSeries = 3.141603
100000. i = 199999, infiniteSeries = 3.141583
PI calculated from the infinite series 20,000 times: 3.141583
```

**EXPERIMENT-04a**

**AIM:** Write a java program to find the area and perimeter of a rectangle using the concept of class and objects.

***Program:***

```
class Rectangle {
    float Length, Width;

    Rectangle () {
        Length = 1.0f;
        Width = 1.0f;
    }

    void setLength (float a) {
        if ((Length > 0) && (Length <= 20)) Length = a;
    }

    void setWidth (float a) {
        if ((Width > 0) && (Width <= 20)) Width = a;
    }

    float getLength () {
        return Length;
    }

    float getWidth () {
        return Width;
    }

    float perimeter () {
        float p;
        p = 2 * (getLength() + getWidth());
        return p;
    }

    float area () {
        float p;
        p = getLength() * getWidth();
        return p;
    }
}
```

```
public class Cal {  
    public static void main(String args[]) {  
        Rectangle rr = new Rectangle();  
        rr.setLength(15);  
        rr.setWidth(15);  
        System.out.println("Perimeter is:" + rr.perimeter());  
        System.out.println("Area is:" + rr.area());  
    }  
}
```

***Output:***

Perimeter is: **60.0**  
Area is: **225.0**

**EXPERIMENT-04b**

**AIM:** Write a java program to demonstrate copy constructor and constructor overloading.

**Program:**

```
public class CopyConDemo {
    int m;
    String n;
    public CopyConDemo(int m, String n) {
        this.m = m;
        this.n = n;
    }

    /*Copy constructor*/
    public CopyConDemo(CopyConDemo c) {
        this(c.getM(), c.getN());
    }

    int getM() {
        return m;
    }

    String getN() {
        return n;
    }

    void setM(int m) {
        this.m = m;
    }
    void setN(String n) {
        this.n = n;
    }

    public static void main(String args[]) {
        CopyConDemo c1 = new CopyConDemo(12, "Original");
        CopyConDemo c2 = new CopyConDemo(c1);
        System.out.println("Object c1: m = " + c1.getM() + " n = " + c1.getN());
        System.out.println("Object c2: m = " + c2.getM() + " n = " + c2.getN());
        c2.setN("Updated Copy");
        System.out.println("Object c2: m = " + c2.getM() + " n = " + c2.getN());
    }
}
```

***Output:***

Object c1: m = 12 n = Original

Object c2: m = 12 n = Original

Object c2: m = 12 n = Updated Copy



**EXPERIMENT-04c**

**AIM:** Write a java program to demonstrate function overloading and overriding

**Program:** //Overloading

```
class A {  
    public void p(double i) {  
        System.out.println(i * 2);  
    }  
}  
  
class B extends A {  
    public void p(int i) {  
        System.out.println(i);  
    }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        A a = new A();  
        a.p(10);  
        B b = new B();  
        b.p(10);  
        b.p(10.0);  
    }  
}
```

**Output:**

20.0  
10  
20.0

***Program: //Over riding***

```
class A {  
    public void p(double i) {  
        System.out.println(i * 2);  
    }  
}  
  
class B extends A {  
    public void p(double i) {  
        System.out.println(i);  
    }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        A a = new A();  
        a.p(10);  
        B b = new B();  
        b.p(10);  
        b.p(10.0);  
    }  
}
```

**Output:**

20.0

10.0

10.0

---

**EXPERIMENT-05**

**AIM:** Write a java program to create a program to superclass **Motor Vehicle** with instance variable **model Name, model Number, model Price**, parameterized constructor, and **display ()** method. Create a sub-class that inherits the features of the superclass and has its instance variable **discountRate**, parameterized constructor, and **display (), discount ()** methods. Create an object for the car class and invoke all the methods using the object of that class. (Hint: Single Inheritance)

**Program:**

```
class MotorVehicle {
    String modelName;
    int modelNumber;
    float modelPrice;

    MotorVehicle(String mname, int mnumber, float mprice) {
        modelName = mname;
        modelNumber = mnumber;
        modelPrice = mprice;
    }

    void display() {
        System.out.println("Model Name is : " + modelName);
        System.out.println("Model Number is : " + modelNumber);
        System.out.println("Model Price is : " + modelPrice);
    }
}

public class Car extends MotorVehicle {
    int discountRate;

    Car(String mname, int mnumber, float mprice, int dr) {
        super(mname, mnumber, mprice);
        discountRate = dr;
    }

    //implementing Polymorphism: Method Overriding
    void display()
    {
        super.display();
        System.out.println("The discount rate is : " + discountRate);
    }

    void discount() {
        float discount = modelPrice * discountRate / 100;
        float priceAfterDiscount = modelPrice - discount;
        System.out.println("The discount is : " + discount);
        System.out.println("The Price after discount rate is : " + priceAfterDiscount);
    }
}
```

```
public static void main(String args[]) {  
    Car c = new Car("Mercedes-Benz E-Class", 200, 8500000f, 10);  
    c.display();  
    c.discount();  
}  
}
```

***Output:***

Model Name is : **Mercedes-Benz E-Class**  
Model Number is : **200**  
Model Price is : **8500000.0**  
The discount rate is : **10**  
The discount is : **850000.0**  
The Price after discount rate is : **7650000.0**

---

**EXPERIMENT-06**

**AIM:** Write a java program to create an abstract class **Shape** which contains the abstract method **number Of Sides()**. Create different sub-classes by the name **Trapezoid, Triangle, and Hexagon**, which extends the shape class. Develop a class **Shape Demo** which contains the main method. Create the object for different subclasses with the main method and invoke the method **number Of Sides()** using the objects of classes. (**Hint: Abstract Class**)

***Program:***

```
abstract class Shape {
    abstract void numberOfSides();
}

class Trapezoid extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides in a Trapezoid is four");
    }
}

class Triangle extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides in a Traingle is three");
    }
}

class Hexagon extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides in a Hexagon is six");
    }
}

public class ShapeDemo {
    public static void main(String args[]) {
        Trapezoid tp = new Trapezoid();
        tp.numberOfSides();
        Triangle tr = new Triangle();
        tr.numberOfSides();
        Hexagon h = new Hexagon();
        h.numberOfSides();
    }
}
```

***Output:***

```
The number of sides in a Trapezoid is four
The number of sides in a Traingle is three
The number of sides in a Hexagon is six
```

**EXPERIMENT-07**

**AIM:** Write a program to create an interface **Calculator** which contains **add()**, **sub()**, **multiply()**, **divide()**, **remainder ()** abstract methods with two-parameter x and y. Develop a class **Calculator Demo** which inherits the features of the interface. Create an object for the Calculator Demo class and invoke all the methods of this class.

**(Hint: implementing interfaces / multiple inheritances)**

**Program:**

```
interface Calculator {
    public int add(int x, int y);
    public int sub(int x, int y);
    public int multiply(int x, int y);
    public int divide(int x, int y);
    public int remainder(int x, int y);
}

public class CalculatorDemo implements Calculator {
    public int add(int x, int y) {
        return (x + y);
    }
    public int sub(int x, int y) {
        return (x - y);
    }
    public int multiply(int x, int y) {
        return (x * y);
    }
    public int divide(int x, int y) {
        return (x / y);
    }

    public int remainder(int x, int y) {
        return (x % y);
    }

    public static void main(String args[]) {
        CalculatorDemo cd = new CalculatorDemo();
        System.out.println("ADD of two no is: " + cd.add(10, 10));
        System.out.println("SUB of two no is: " + cd.sub(1, 10));
        System.out.println("MULTIPLICATION of two no is: " + cd.multiply(5, 3));
        System.out.println("Divide of two no is: " + cd.divide(50, 10));
        System.out.println("REMAINDER of two no is: " + cd.remainder(50, 7));
    }
}
```

***Output:***

ADD of two no is: **20**

SUB of two no is: **-9**

MULTIPLICATION of two no is: **15**

Divide of two no is: **5**

REMAINDER of two no is: **1**

**EXPERIMENT-08**

**AIM:** Write a program to demonstrate the function on string like to Lower Case(), to Upper Case(), length(), starts With(), ends With(), substring(), and string conversion using String. Value Of()

**(Hint: String Handling)**

**Program:**

```
public class StringDemo {
    public static void main(String args[]) {
        String s1 = "Java Programming";
        System.out.println("The String is : " + s1.toLowerCase());
        System.out.println("The String is : " + s1.toUpperCase());
        System.out.println("The length of String is : " + s1.length());
        System.out.println("The String starts with j:" + s1.startsWith("j"));
        System.out.println("The String ends with G: " + s1.endsWith("G"));
        System.out.println("The SubString starting from index 3 is:" + s1.substring(3));
        System.out.println("The SubString starting from index 3 till index 6 is: " +
s1.substring(3, 6));

        // converting boolean to String
        boolean b = true;
        String s2 = String.valueOf(b);
        System.out.println("The converted String is : " + s2);
        // converting char to String
        char c = 'a';
        s2 = String.valueOf(c);
        System.out.println("The converted String is : " + s2);
        // converting double to String
        double d = 2.4d;
        s2 = String.valueOf(d);
        System.out.println("The converted String is : " + s2);
        // converting int to String
        int i = 24;
        s2 = String.valueOf(i);
        System.out.println("The converted String is : " + s2);
    }
}
```



***Output:***

The String is : **java programming**

The String is : **JAVA PROGRAMMING**

The length of String is : **16**

The String starts with j: **false**

The String ends with G: **false**

The SubString starting from index 3 is: **a Programming**

The SubString starting from index 3 till index 6 is: **a P**

The converted String is : **true**

The converted String is : **a**

The converted String is : **2.4**

The converted String is : **24**

**EXPERIMENT-09**

**AIM:** Write a java program using the concept of packages. Create a class Trigonometry which contain static method **sine ()**, **cos ()**, **tan ()**, **cosec ()**, **tan ()**, **cosec ()**, **sec ()**, **cot ()**. Print the value of a given angle in degree by calling these methods.

**(Hint: Implementing packages)**

**Program:**

```
package mypack;

public class Trigonometry {
    static float sine(double degrees) {
        double s;
        s = (Math.sin(Math.toRadians(degrees)));
        return (float) s;
    }

    static float cos(double degrees) {
        double s;
        s = (Math.cos(Math.toRadians(degrees)));
        return (float) s;
    }

    static float tan(double degrees) {
        double s;
        s = (Math.tan(Math.toRadians(degrees)));
        return (float) s;
    }

    static double cosec(double degrees) {
        double s;
        s = (Math.sin(Math.toRadians(degrees)));
        s = 1 / s;
        return s;
    }

    static double sec(double degrees) {
        double s;
        s = (Math.cos(Math.toRadians(degrees)));
        s = 1 / s;
        return s;
    }

    static double cot(double degrees) {
        double s;
        s = (Math.tan(Math.toRadians(degrees)));
        s = 1 / s;
        return s;
    }
}
```

```
public static void main(String args[]) {  
    System.out.println("Sin value of given angle in degree is:" + sine(30));  
    System.out.println("Cos value of given angle in degree is:" + cos(30));  
    System.out.println("Tan value of given angle in degree is:" + tan(30));  
    System.out.println("Cosec value of given angle in degree is:" + cosec(30));  
    System.out.println("Sec value of given angle in degree is:" + sec(30));  
    System.out.println("Cot value of given angle in degree is:" + cot(30));  
}  
}
```

***Output:***

Sin value of given angle in degree is: **0.5**

Cos value of given angle in degree is: **0.8660254**

Tan value of given angle in degree is: **0.57735026**

Cosec value of given angle in degree is: **2.000000000000000004**

Sec value of given angle in degree is: **1.1547005383792515**

Cot value of given angle in degree is: **1.7320508075688774**

**EXPERIMENT-10a**

**AIM:** Write a java class to demonstrate Arithmetic Exception, Null Pointer Exception, Array Index Out Of Bounds Exception.

**(Hint: Handling predefined exceptions)**

**Program:**

```
public class ExceptionsDemo
{
    public static void main (String args[])
    {
        try {
            int num1=30, num2=0; //num2 = 2;
            int output=num1/num2;
            System.out.println ("Result: "+output);

            int a[]=new int[10];
            a[11] = 9; // a[1] = 9;
            System.out.println ("Result: "+ a[1]);

            String str=null; //str= "Hello";
            System.out.println (str.length());
        }

        catch (Exception e) {
            System.out.println (e);
        }
    }
}
```

**Output 1:**

java.lang.ArithmeticException: / by zero

**Output 2:**

Result: 15

java.lang.Array Index Out Of Bounds Exception: Index 11 out of bounds for length 10

**Output 3:**

Result: 15

Result: 9 java.lang.NullPointerException

**Output 4:**

Result: 15

Result: 9

Result: 5

**EXPERIMENT-10b**

**AIM:** Write a java program to demonstrate working with files.  
**(Hint: Files & Exception)**

***Program:***

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ReadFile {
    public static void main(String[] args) {
        try {
            File myObj = new File("SampleFile.txt");
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                System.out.println(data);
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("File Not Found");
            e.printStackTrace();
        }
    }
}
```

**Output 1:**

File Not Found

**Output 2:**

Files in Java might be tricky, but it is fun enough!

**EXPERIMENT-11**

**AIM:** Write a Java Program using **the Runnable** interface to demonstrate the concepts of thread priorities.  
**(Hint: Threads)**

***Program:***

```
class Clicker implements Runnable {
    int click = 0;
    private Thread t;
    private boolean running = true;
    public Clicker(int p) {
        t = new Thread(this);
        t.setPriority(p);
    }
    public void run() {
        while (running) {
            click++;
        }
    }
    public void stop() {
        running = false;
    }
    public void start() {
        t.start();
    }
}

public class HiLoPri {
    public static void main(String args[ ]) {
        Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
        Clicker hi = new Clicker(Thread.NORM_PRIORITY + 2);
        Clicker lo = new Clicker(Thread.NORM_PRIORITY - 2);
        lo.start();
        hi.start();
        try {
            Thread.sleep(10000);
        }
        catch (Exception e) {
        }
        lo.stop();
        hi.stop();
        System.out.println(lo.click + " vs " + hi.click);
    }
}
```

**Output:**

804077335 vs 847304063

**EXPERIMENT-12**

**AIM:** Write a java program to demonstrate the multithread and thread synchronization.  
**(Hint: Multithreading)**

**Program:**

```
class Callme {
    //synchronized void call(String msg)
    void call(String msg)
    {
        System.out.print "[" + msg);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Interrupted");
        }
        System.out.println("]");
    }
}

class Caller implements Runnable {
    String msg;
    Callme target;
    Thread t;
    public Caller(Callme targ, String s) {
        target = targ;
        msg = s;
        t = new Thread(this);
    }
    public void run() {
        target.call(msg);
    }
}

class Synch {
    public static void main(String[] args) {
        Callme target = new Callme();
        Caller ob1 = new Caller(target, "Hello");
        Caller ob2 = new Caller(target, "Synchronized");
        Caller ob3 = new Caller(target, "World");
        // Start the threads.
        ob1.t.start();
        ob2.t.start();
        ob3.t.start();
        // wait for threads to end
        try {
            ob1.t.join();
            ob2.t.join();
            ob3.t.join();
        }
    }
}
```

```
    } catch(InterruptedException e) {  
        System.out.println("Interrupted");  
    }  
}  
}
```

**Output 1:**

using void call(String msg) [World[Synchronized[Hello]]]

**Output 2:**

using synchronized void call(String msg)

[Hello]

[World]

[Synchronized]





**REVA**  
UNIVERSITY

Bengaluru, India

Rukmini Knowledge Park, Kattigenahalli  
Yelahanka, Bengaluru - 560 064  
Karnataka, India.

Ph: +91- 90211 90211, +91 80 4696 6966  
E-mail: [admissions@reva.edu.in](mailto:admissions@reva.edu.in)

Follow us on

