

# LOURDES MATHA

## COLLEGE OF SCIENCE AND TECHNOLOGY

Bureau Veritas Certified ISO 9001-2015 Institution

Managed by Archdiocese of Changanacherry



### OBJECT ORIENTED PROGRAMMING LAB(20MCA132)

Name : .....

University Reg No:LMC21MCA-.....

Class: ...MCA.....Semester.....S2.....

Year: .....2021-2023.....

From Page No: .....1.....To.....

*Certified Bonafide Record of Work Done By*

.....

External Examiner

Head of the Department

Faculty-in-Charge

Prof. Bismi K Charleys

Prof. Sherin Joseph

Place: Kuttichal

Date: .....

# **INDEX**

SL.NO	NAME OF THE EXPERIMENT	DATE	PAGE.NO
1	Program to find product having lowest price	17/03/2022	30
2	Matrix addition	21/03/2022	33
3	Add complex numbers	24/03/2022	37
4	Program to implement inner class and static nested class	31/03/2022	40
5	Program to sort strings	31/03/2022	43
6	String manipulations	01/04/2022	46
7	Program to implement array of objects	04/04/2022	49
8	Area of different shapes using method overloading	07/04/2022	53
9	Program to implement multilevel inheritance	18/04/2022	55
10	Program to implement interface	25/04/2022	61
11	Program to implement packages	28/04/2022	66
12	Authenticate username and password using user defined exception	28/04/2022	70
13	Find average of N numbers and raise user defined exception for negative input	10/05/2022	75
14	Program to implement Runnable Interface	16/05/2022	78
15	Program to find maximum of three numbers using AWT	18/05/2022	82
16	Simple calculator using AWT components	19/05/2022	86
17	Mouse events using event handling	30/05/2022	91
18	Key events using event handling	09/06/2022	96
19	Write, Read and Display contents of a file	09/06/2022	99
20	Program to copy one file to another	10/06/2022	101

# **INTRODUCTION**

## **JAVA**

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James Gosling is known as the father of Java. Before Java, its name was Oak. Since Oak was already a registered company, James Gosling and his team renamed Oak as Java.

Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

## **Java terminologies**

Let us get familiarised with some common Java terms.

## **Java Virtual Machine (JVM)**

This is generally referred as JVM. Before, we discuss about JVM let's see the phases of program execution. Phases are as follows: we write the program, then we compile the program and at last we run the program.

1. Writing of the program is of course done by Java programmer.
2. Compilation of program is done by Javac compiler, Javac is the primary Java compiler included in Java development kit (JDK). It takes Java program as input and generates Java bytecode as output.
3. In third phase, JVM executes the bytecode generated by compiler. This is called program run phase. So, now that we understood that the primary function of JVM is to execute the bytecode produced by compiler. Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems. That is why we call Java as platform independent language.

## **bytecode**

As discussed above, Javac compiler of JDK compiles the Java source code into bytecode so that it can be executed by JVM. The bytecode is saved in a .class file by compiler.

## **Java Development Kit(JDK)**

As the name suggests this is complete Java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc. In order to create, compile and run Java program you would need JDK installed on your computer.

## **Java Runtime Environment(JRE)**

JRE is a part of JDK which means that JDK includes JRE. When you have JRE installed on your system, you can run a Java program however you won't be able to compile it. JRE includes JVM, browser plugins and applets support. When you only need to run a Java program on your computer, you would only need JRE.

## **Features of JAVA**

A list of most important features of Java language is given below.

1. **Simple:** Java is considered as one of simple language because it does not have complex features like Operator overloading, Multiple inheritance, pointers and Explicit memory allocation.
2. **Object-Oriented:** Java is an Object Oriented language. Object oriented programming is a way of organizing programs as collection of objects, each of which represents an instance of a class. Basic concepts of OOPs are:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

3. **Portable:** As discussed above, Java code that is written on one machine can run on another machine. The platform independent byte code can be carried to any platform for execution that makes Java code portable.

4. **Platform independent:** Java is a platform independent language. Compiler(Javac) converts source code (.java file) to the byte code(.class file). As mentioned above, JVM executes the bytecode produced by compiler. This byte code can run on any platform such as Windows, Linux, Mac OS etc. Which means a program that is compiled on windows can run on Linux and vice-versa. Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems. That is why we call Java as platform independent language.

5. **Secure:** We don't have pointers and we cannot access out of bound arrays (you get `ArrayIndexOutOfBoundsException` if you try to do so) in Java. That's why several security flaws like stack corruption or buffer overflow is impossible to exploit in Java.

6. **Robust:** Robust means reliable. Java programming language is developed in a way that puts a lot of emphasis on early checking for possible errors. The main features of Java that makes it robust are garbage collection, Exception Handling and memory allocation.

7. **Architecture neutral:** Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed. In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

8. **High-performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

9. **Multithreading:** Java supports multithreading. Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilisation of CPU.

10. **Distributed:** Using Java programming language we can create distributed applications. RMI (Remote Method Invocation) and EJB (Enterprise Java Beans) are used for creating distributed applications in Java. In simple words: The Java programs can be distributed on more than one systems that are connected to each other using internet connection. Objects on one JVM (Java virtual machine) can execute procedures on a remote JVM.

11. **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++. Java supports dynamic compilation and automatic memory management (garbage collection).

12. **Strongly typed:** Java is a strongly typed language. The Java compiler enforces type checking of each assignment made in a program at compile time. If the type checking fails, then a compile-time error is issued.

## **JAVA IDE**

IDE: Integrated Development Environment. There are various free IDEs available for JAVA. A few of them are listed below.

- Eclipse
- NetBeans
- IntelliJ IDEA
- BlueJ
- (Oracle) JDeveloper
- Greenfoot

## **OBJECT ORIENTED PROGRAMMING (OOPs)**

Object-Oriented Programming or OOPs refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code. Objects are seen by the viewer or user, performing tasks assigned by you. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

### **Benefits of Object-Oriented Programming**

- Improved productivity during software development
- Improved software maintainability
- Faster development sprints
- Lower cost of development
- Higher quality software

### **Basic Concepts of OOPs are:**

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

### **CLASSES**

- A Class is a collection of data members and member functions. Data members are used to store information; member functions are used to perform operations on data.
- A class is a group of objects which have common properties.
- It is a template or blueprint from which objects are created.

A class in Java can contains:

- Fields
- Methods
- Constructors
- Blocks
- Nested Classes and Interface.

Each field and method has an access level:

- **private:** accessible only in this class.
- **protected:** accessible only in this package and in all subclasses of this class.
- **public:** accessible everywhere this class is available.

**Syntax to declare a class:**

```
class ClassName  
{  
    Fields or Data members;  
    Methods or member functions;  
}
```

**Field declarations Syntax:**

```
Class classname  
{  
    Datatype variablename1;  
    Datatype variablename2;  
    .  
    .  
    .  
    Datatype variablenamen;  
}
```



## **Method**

A method in Java is a collection of statements that perform some specific task. The method returns the result of the statements inside it. A method can also perform some specific task without returning anything.

### **Method declaration Syntax:**

```
Returntype methodname(ArgumentsList)  
  
{  
  
Method body;  
  
}
```

## **OBJECT**

An entity that has state and behaviour is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical.

An object has three characteristics:

- **State** : Represents the data (value) of an object.
- **Behavior**: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity**: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

### **Syntax to declare an object:**

```
Classname objectname=new Classname( );
```

## **ABSTRACTION**

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

## **Ways to achieve Abstraction:**

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

### **Abstract class**

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods.

### **Syntax:**

```
abstract class classname  
  
{  
  
Body of the abstract class  
  
}
```

### **Abstract Method**

A method which is declared as abstract and does not have implementation is known as an abstract method.

### **Syntax:**

```
Abstract returntype methodname (ArgumentsList); //no method body
```

## INHERITANCE

Inheritance is a feature of Object-Oriented Programming in Java that allows programmers to create new(child) classes that share some of the attributes of existing(parent) classes. It is an object-oriented process by which one class acquires or inherits the properties and functionalities of another class.

Inheritance provides the reusability of code. Each child class defines only those features that are unique to it, and the child class inherits the rest of the features from the parent class.

### Syntax

```
class Subclass-name extends Super_class-name
{
    //methods and fields
}
```

## TYPES OF INHERITANCE:

### • **Single Level Inheritance:**

When one base class is being inherited by one sub class then that kind of inheritance is known as single level inheritance.

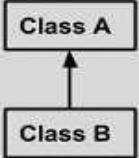
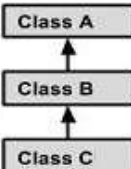
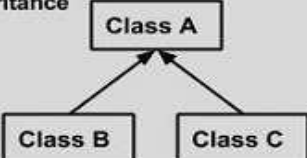
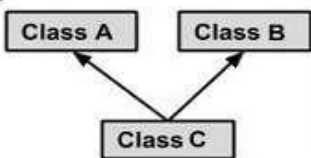
### • **Multi Level Inheritance:**

When a sub class is in turn being inherited then that kind of inheritance is known as multi-level inheritance.

### • **Hierarchical Inheritance:**

When a base class is being inherited by one or more sub class then that kind of inheritance is known as hierarchical inheritance.

A sub class uses the keyword —extends‖ to inherit a base class.

<b>Single Inheritance</b>  <pre> graph BT     B[Class B] --&gt; A[Class A] </pre>	<pre> public class A {     ..... } public class B extends A {     ..... } </pre>
<b>Multi Level Inheritance</b>  <pre> graph BT     C[Class C] --&gt; B[Class B]     B --&gt; A[Class A] </pre>	<pre> public class A { ..... } public class B extends A { ..... } public class C extends B { ..... } </pre>
<b>Hierarchical Inheritance</b>  <pre> graph BT     B[Class B] --&gt; A[Class A]     C[Class C] --&gt; A </pre>	<pre> public class A { ..... } public class B extends A { ..... } public class C extends A { ..... } </pre>
<b>Multiple Inheritance</b>  <pre> graph BT     C[Class C] --&gt; A[Class A]     C --&gt; B[Class B] </pre>	<pre> public class A { ..... } public class B { ..... } public class C extends A,B {     ..... } // Java does not support multiple inheritance </pre>

## ENCAPSULATION

- It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together the code and the data it manipulates. Another way to think about encapsulation is that it is a protective shield that prevents the data from being accessed by the code outside this shield.
- Technically, in encapsulation, the variables or the data in a class is hidden from any other class and can be accessed only through any member function of the class in which they are declared.
- In encapsulation, the data in a class is hidden from other classes, which is similar to what data-hiding does. So, the terms “encapsulation” and “data-hiding” are used interchangeably.
- Encapsulation can be achieved by declaring all the variables in a class as private and writing public methods in the class to set and get the values of the variables.

## **POLYMORPHISM**

It refers to the ability of object-oriented programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities.

### **Types of polymorphism**

In Java polymorphism is mainly divided into two types:

1. Compile-time Polymorphism
2. Runtime Polymorphism

#### **Compile-time polymorphism**

It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

But Java doesn't support the Operator Overloading.

**Method Overloading:** When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in the number of arguments or/and a change in the type of arguments.

#### **Run-time polymorphism**

It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

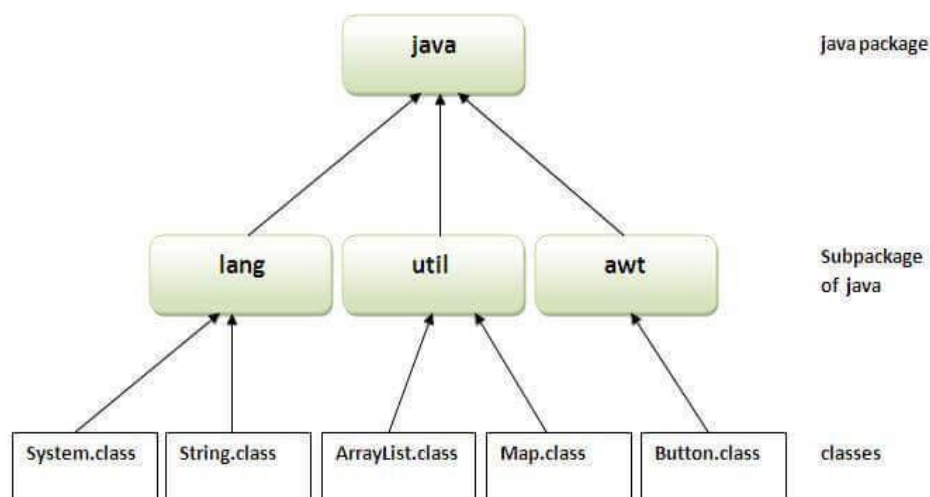
**Method overriding**, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

## JAVA PACKAGES

- A package in Java is used to group related classes.
- A java package is a group of similar types of classes, interfaces and sub-packages.
- Think of it as a folder in a file directory.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Packages are divided into two categories:

1. Built-in Packages (packages from the Java API)
2. User-defined Packages (create your own packages)



### Built-in Packages

- The Java API is a library of prewritten classes that are free to use, included in the Java Development Environment.
- The library is divided into packages and classes.
- We can either import a single class (along with its methods and attributes), or a whole package that contain all the classes that belong to the specified package.
- To use a class or a package from the library, you need to use the import keyword:

### **Syntax**

*import package.name.Class; // Import a single class*

*import package.name.\*; // Import the whole package*

### **User Defined Packages**

The users of the Java language can also create their own packages. They are called user-defined packages. User defined packages can also be imported into other classes & used exactly in the same way as the Built in packages.

### **Creating User Defined Packages**

#### **Syntax**

*package packageName;*

*public class className*

*{*

*// Body of className*

*}*

We must first declare the name of the package using the package keyword followed by the package name. This must be the first statement in a Java source file. Then define a classes as normally as define a class.

**Steps For Creating Package :** To create a user defined package the following steps should be involved :-

1: Declare the package at the beginning of a file.

#### **Syntax**

*package packageName;*

2: Define the class that is to be put in the package & declare it public.

Java also supports the concept of package hierarchy. This is done by specifying multiple names in a package statement, seprated by dots (.).

Ex :- *package firstPackage.secondPackage;*

**Accessing A Package:** Java package can be accessed either using a fully qualified class name or using a shortcut approach through the import statement.

### Syntax

```
import package1[.package2][.package3].classname;
```

### Example

*MyPackageClass.java*

```
package mypack;
```

```
class MyPackageClass
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
    System.out.println("This is my package!");
```

```
}
```

```
}
```

Save the file as MyPackageClass.java, and compile it:

- Then **compile** the package:

```
C:\Users\Your Name>javac -d . MyPackageClass.java
```

This forces the compiler to create the "mypack" package.

- The -d keyword specifies the destination for where to save the class file. You can use any directory name, like c:/user (windows), or, if you want to keep the package within the same directory, you can use the dot sign ".",
- To **run** the MyPackageClass.java file, write the following:

```
C:\Users\Your Name>java mypack.MyPackageClass
```



## **INTERFACE IN JAVA**

- The interface in Java is a mechanism to achieve abstraction.
- There can be only abstract methods in the Java interface, not method body.
- It is used to achieve abstraction and multiple inheritance in Java.
- An interface in Java is a blueprint of a class.
- It has static constants and abstract methods.
- Like abstract classes, interfaces cannot be used to create objects
- Interface methods do not have a body - the body is provided by the "implement" class
- On implementation of an interface, you must override all of its methods
- Interface methods are by default abstract and public
- Interface attributes are by default public, static and final
- An interface cannot contain a constructor (as it cannot be used to create objects)

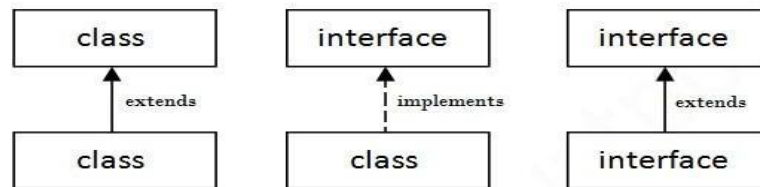
### **Why use Java interface?**

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.

### **How to declare an interface?**

- An interface is declared by using the interface keyword.
- It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default.
- A class that implements an interface must implement all the methods declared in the interface.

## The relationship between classes and interfaces

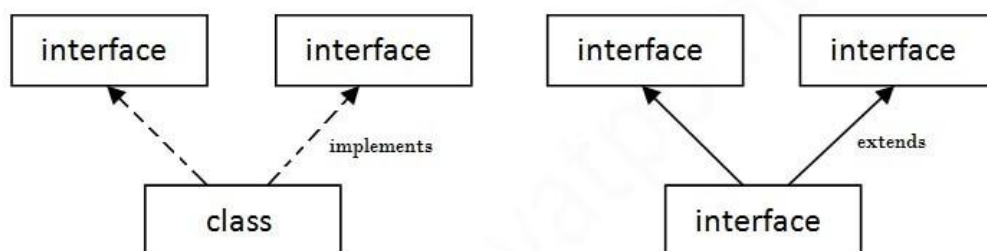


### Syntax:

```
interface <interface_name>
{
    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

### Multiple Inheritance In Java By Interface

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



Multiple Inheritance in Java

## **MULTITHREADING**

- Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.
- Each part of such program is called a thread. So, threads are light-weight processes within a process.

### **Advantages of Java Multithreading**

- 1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- 2) You can perform many operations together, so it saves time.
- 3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

## **THREAD**

- A thread is a lightweight subprocess, the smallest unit of processing.
- It is a separate path of execution.
- Threads are independent.
- If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.

### **Thread creation by extending the Thread class**

We create a class that extends the **java.lang.Thread** class.

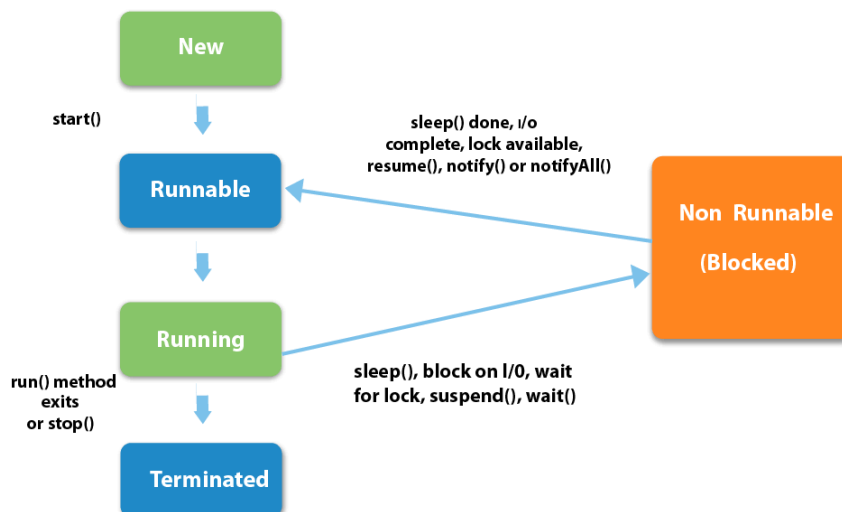
- This class overrides the run() method available in the Thread class.
- A thread begins its life inside run() method.
- We create an object of our new class and call start() method to start the execution of a thread.
- Start() invokes the run() method on the Thread object.

### **Thread lifecycle**

The life cycle of the thread in java is controlled by JVM.

The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



## New

The thread is in new state if you create an instance of Thread class but before the invocation of `start()` method.

## Runnable

The thread is in runnable state after invocation of `start()` method, but the thread scheduler has not selected it to be the running thread.

## Running

The thread is in running state if the thread scheduler has selected it.

## Terminated

A thread is in terminated or dead state when its `run()` method exits

## Thread Creation

There are two ways to create a thread:

- By extending Thread class
- By implementing Runnable interface.

## Thread class

- Thread class provide constructors and methods to create and perform operations on a thread.
- Thread class extends Object class and implements Runnable interface.

Commonly used methods of Thread class:

- **public void run():** is used to perform action for a thread.
- **public void start():** starts the execution of the thread. JVM calls the run() method on the thread.
- **public void sleep(long milliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
- **public void join():** waits for a thread to die.

## Runnable interface

- The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
- Runnable interface have only one method named run().
- **public void run():** is used to perform action for a thread.
- **Starting a thread:** start() method of Thread class is used to start a newly created thread.

It performs following tasks:

- A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

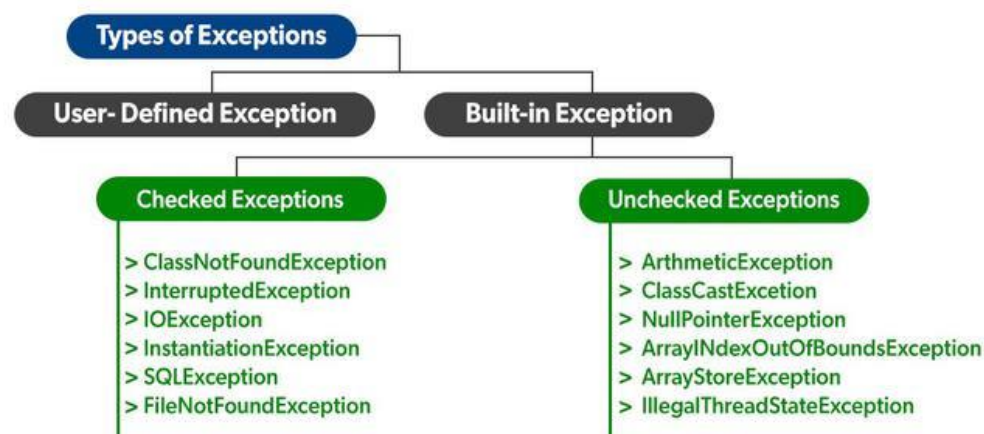
## EXCEPTION HANDLING

Exception Handling in Java is one of the effective means to handle the runtime errors so that the regular flow of the application can be preserved. Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program's instructions. Exceptions can be caught and handled by the program. When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

## Types of Exceptions

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



## Built-in Exceptions

Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

1. **Checked Exceptions:** Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.
2. **Unchecked Exceptions:** The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

### **User-Defined Exception.**

- We can create our own exceptions that are derived classes of the Exception class.
- Creating our own Exception is known as custom exception or user-defined exception.
- Basically, Java custom exceptions are used to customize the exception according to user need.
- In order to create custom exception, we need to extend Exception class that belongs to java.lang package.

### **APPLET IN JAVA**

- Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as a part of a web document.
- After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.
- Any applet in Java is a class that extends the java.applet.Applet class.
- An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.
- JVM creates an instance of the applet class and invokes init() method to initialize an Applet.

### **Life Cycle of an Applet**

Four methods in the Applet class gives you the framework on which you build any serious applet –

The method execution sequence when an applet is executed is:

- `init()`– This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- `start()` – This method is automatically called after the browser calls the `init` method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- `paint()` – Invoked immediately after the `start()` method, and also any time the applet needs to repaint itself in the browser. The `paint()` method is actually inherited from the `java.awt`.

The method execution sequence when an applet is closed is:

- `stop()` – This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- `destroy()` – This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.

### **Applet Class**

Every applet is an extension of the `java.applet.Applet` class. The base Applet class provides methods that a derived Applet class may call to obtain information and services from the browser context.

These include methods that do the following –

- Get applet parameters
- Get the network location of the HTML file that contains the applet
- Get the network location of the applet class directory
- Print a status message in the browser
- Fetch an image
- Fetch an audio clip
- Play an audio clip
- Resize the applet



Additionally, the Applet class provides an interface by which the viewer or browser obtains information about the applet and controls the applet's execution. The viewer may –

- Request information about the author, version, and copyright of the applet
- Request a description of the parameters the applet recognizes
- Initialize the applet
- Destroy the applet
- Start the applet's execution
- Stop the applet's execution

### Invoking an Applet

An applet may be invoked by embedding directives in an HTML file and viewing the file through an applet viewer or Java-enabled browser.

The **<applet> tag** is the basis for embedding an applet in an HTML file.

```
<applet code = "Appletclassname.class" width = "320" height =  
"120"></applet>
```

The code attribute of the <applet> tag is required. It specifies the Applet class to run. Width and height are also required to specify the initial size of the panel in which an applet runs. The applet directive must be closed with an </applet> tag.

To **run an applet** we require one of the following tools :

1. Java enabled web browser (such as HotJava or Netscape)
2. Java appletviewer

Appletviewer is available with the JDK(java development kit) .we run the applet

as follows:

```
C:\> appletviewer demo.html
```

## EVENTS

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

## EVENT HANDLING

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

**Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.

**Listener** - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

### Steps involved in event handling

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- The method is now get executed and returns.

### Important Event Classe and Interface

Event Classe	Description	Listener Interface
<b>ActionEvent</b>	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
<b>MouseEvent</b>	generated when mouse is dragged, moved, clicked, pressed or released also when the enters or exit a component	MouseListener
<b>KeyEvent</b>	generated when input is received from keyboard	KeyListener
<b>ItemEvent</b>	generated when check-box or list item is clicked	ItemListener
<b>TextEvent</b>	generated when value of textarea or textfield is changed	TextListener
<b>MouseWheelEvent</b>	generated when mouse wheel is moved	MouseWheelListener
<b>WindowEvent</b>	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
<b>ComponentEvent</b>	generated when component is hidden, moved, resized or set visible	ComponentEventListener
<b>ContainerEvent</b>	generated when component is added or removed from container	ContainerListener
<b>AdjustmentEvent</b>	generated when scroll bar is manipulated	AdjustmentListener
<b>FocusEvent</b>	generated when component gains or loses keyboard focus	FocusListener

#### Listener Interface

#### Methods

##### ActionListener

- actionPerformed()

##### KeyListener

- keyTyped()
- keyPressed()
- keyReleased()

##### MouseListener

- mousePressed()
- mouseClicked()
- mouseEntered()
- mouseExited()
- mouseReleased()

##### MouseMotionListener

- mouseMoved()
- mouseDragged()

##### WindowListener

- windowActivated()
- windowDeactivated()
- windowOpened()
- windowClosed()
- windowClosing()
- windowIconified()
- windowDeiconified()

## JAVA IO STREAM

Java performs I/O through Streams. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical.

Java encapsulates Stream under java.io package. Java defines two types of streams. They are,

1. **Byte Stream** : It provides a convenient means for handling input and output of byte.
2. **Character Stream** : It provides a convenient means for handling input and output of characters. Character stream uses Unicode and therefore can be internationalized.

### The predefined streams

The System class of Java.lang package contains three predefined stream variables: in, out and err.

- **System.out** refers to standard output stream which is the console.
- **System.in** refers to standard input, which is the keyboard by default.
- **System.err** refers to the standard error stream, which also is the console by default.

### Byte Streams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, **FileInputStream** and **FileOutputStream**.

### FileInputStream

This stream is used for reading data from the files. Objects can be created using the keyword new and there are several types of constructors available.

Following constructor takes a file name as a string to create an input stream object to read the file –

```
InputStream f = new FileInputStream("C:/java/hello");
```

Following constructor takes a file object to create an input stream object to read the file. First we create a file object using File() method as follows –

```
File f = new File("C:/java/hello");
```

```
InputStream f = new FileInputStream(f);
```

### **FileInputStream Method & Description**

1. **public void close() throws IOException{}:** This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.

2. **protected void finalize()throws IOException {}:** This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.

3. **public int read(int r)throws IOException{}:** This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's the end of the file.

4. **public int read(byte[] r) throws IOException{}:** This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If it is the end of the file, -1 will be returned.

5. **public int available() throws IOException{}:** Gives the number of bytes that can be read from this file input stream. Returns an int.

### **FileOutputStream**

FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.

Here are two constructors which can be used to create a FileOutputStream object.

Following constructor takes a file name as a string to create an input stream object to write the file –

```
OutputStream f = new FileOutputStream("C:/java/hello")
```

Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using File() method as follows –

```
File f = new File("C:/java/hello");
```

```
OutputStream f = new FileOutputStream(f);
```

### **FileOutputStream Method & Description**

1. **public void close() throws IOException{}**: This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.

2. **protected void finalize()throws IOException {}**: This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.

3. **public void write(int w)throws IOException{}**: This methods writes the specified byte to the output stream.

4. **public void write(byte[] w)**: Writes w.length bytes from the mentioned byte array to the OutputStream.

DATE: 17/03/2022

PROGRAM NO: 1

AIM: Define a class 'product' with data members pcode, pname and price. Create 3 objects of the class and find the product having the lowest price.

ALGORITHM:

step 1: start.

step 2: import Scanner class.

step 3: create class name 'Product'.

step 4: create constructor for class 'Product' and read parameters (product code, product name, product cost).

step 5: create method to get least cost product.

step 6: create main class.

step 7: create three objects for three products.

step 8: call least cost product method.

step 9: stop.

PROGRAM:

```
import java.util.Scanner;

class Product
{
    int pcode,price;

    Product()
    {
        String pname;
        Scanner in=new Scanner(System.in);
```

```

        System.out.println("Enter the pcode");
        pcode=in.nextInt();
        System.out.println("Enter the pname");
        pname=in.next();
        System.out.println("Enter the price");
        price=in.nextInt();
    }

    void min_price(Product p1,Product p2,Product p3)
    {
        if(p1.price<p2.price)
        {
            if(p1.price<p3.price)
                System.out.println(p1.price);
            else
                System.out.println(p3.price);
        }
        else
        {
            if(p2.price<p3.price)
                System.out.println(p2.price);
            else
                System.out.println(p3.price);
        }
    }
}

```



```

    }
    public class Product_sale
    {
        public static void main(String args[])
        {
            Product p1=new Product();
            Product p2=new Product();
            Product p3=new Product();
            System.out.println("Minimum price:");
            p1.min_price(p1,p2,p3);
        }
    }

```

### OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C01> javac Product_sale.java
PS C:\Users\athul\Desktop\java\record\C01> java Product_sale
Enter the pcode
201
Enter the pname
mobile
Enter the price
10000
Enter the pcode
301
Enter the pname
car
Enter the price
200000
Enter the pcode
101
Enter the pname
watch
Enter the price
3000
Minimum price:
3000

```

DATE: 21/03/2022

PROGRAM NO: 2

AIM : Read 2 matrices from the console and perform matrix addition.

### ALGORITHM

step 1: start

step 2: import Scanner class

step 3: create class 'Matrix' for matrix addition

step 4: create method for performing matrix addition and return result.

step 5: create main class

step 6: read require parameters for addition of matrices

step 7: if rows and columns of two matrix are possible perform matrix addition by calling method declared and return result

step 8: else terminate the program with message

step 9: stop

### PROGRAM

```
import java.util.Scanner;

class Matrix
{
    Scanner in=new Scanner(System.in);

    int[][] insert(int m,int n)
    {
        int[][] arr=new int[10][10];
```

```

        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                arr[i][j]=in.nextInt();
            }
        }
        return arr;
    }

    int[][] add(int a[][],int b[][],int m,int n)
    {
        int[][] sum=new int[10][10];
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                sum[i][j]=a[i][j]+b[i][j];
            }
        }
        return sum;
    }
}

public class Matrix_addition
{

```

```

public static void main(String args[])
{
    int a[][]=new int[10][10];
    int b[][]=new int[10][10];
    int c[][]=new int[10][10];
    int i,j;

    Matrix m=new Matrix();
    Scanner in=new Scanner(System.in);

    System.out.println("Enter no of rows for matrix 1");
    int m1=in.nextInt();
    System.out.println("Enter no of columns for matrix 1");
    int n1=in.nextInt();

    System.out.println("Enter no of rows for matrix 2");
    int m2=in.nextInt();
    System.out.println("Enter no of columns for matrix 2");
    int n2=in.nextInt();

    if(m1==m2 && n1==n2)
    {
        System.out.println("Enter the elements of first matrix");
        a=m.insert(m1,n1);
        System.out.println("Enter the elements of second matrix");
        b=m.insert(m1,n2);
    }
}

```

```

        c=m.add(a,b,m1,n1);

        System.out.println("Sum of two matrices are:");
        for(i=0;i<m1;i++)
        {
            System.out.println();
            for(j=0;j<n1;j++)
            {
                System.out.print(c[i][j]+"\\t");
            }
        }
    }
    else
        System.out.println("Matrix addition is not possible");
}
}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C01> javac Matrix_Addition.java
PS C:\Users\athul\Desktop\java\record\C01> java Matrix_Addition
Enter no of rows for matrix 1
2
Enter no of columns for matrix 1
2
Enter no of rows for matrix 2
2
Enter no of columns for matrix 2
2
Enter the elements of first matrix
5
4
3
2
Enter the elements of second matrix
1
2
6
3
Sum of two matrices are:

6      6
9      5

```

DATE: 24/03/2022

PROGRAM NO: 3

AIM : Add complex numbers.

### ALGORITHM

step 1: start

step 2: import Scanner class

step 3: create class 'Complex'

step 4: create constructor for getting real and imaginary part of a complex number.

step 5: create method to perform complex numbers addition

step 6: create main class

step 7: read two complex numbers and give to corresponding objects as parameters.

step 8: call method for addition of two complex numbers.

step 9: stop

### PROGRAM

```
import java.util.Scanner;
```

```
class Complex
{
    int rp,ip;
    Complex(int r,int i)
    {
        rp=r;
```

```

        ip=i;
    }

    void sum(Complex c1,Complex c2)
    {
        int rsum=c1.rp+c2.rp;
        int isum=c1.ip+c2.ip;
        System.out.println(rsum+" "+isum+"i");
    }
}

public class Complex_Program
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter the first complex number");
        System.out.println("Enter Real Part");
        int frp=in.nextInt();
        System.out.println("Enter Imaginary Part");
        int fip=in.nextInt();
        System.out.println("The First complex number");
        System.out.println(frp+" "+fip+"i");

        System.out.println("Enter the second complex number");
        System.out.println("Enter Real Part");
    }
}

```

```

int srp=in.nextInt();

    System.out.println("Enter Imaginary Part");
    int sip=in.nextInt();
    System.out.println("The second complex number");
    System.out.println(srp+" "+sip+"i");

    Complex c1=new Complex(frp,fip);
    Complex c2=new Complex(srp,sip);
    System.out.println("Addition of two Complex Numbers are");
    c1.sum(c1,c2);
}
}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C01> javac Complex_Program.java
PS C:\Users\athul\Desktop\java\record\C01> java Complex_Program
Enter the first complex number
Enter Real Part
2
Enter Imaginary Part
4
The First complex number
2+4i
Enter the second complex number
Enter Real Part
3
Enter Imaginary Part
2
The second complex number
3+2i
Addition of two Complex Numbers are
5+6i

```



DATE: 31/03/2022

PROGRAM NO: 4

AIM: Create CPU with attribute price. Create inner class Processor (no. of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of Processor and RAM.

### ALGORITHM

step 1: start

step 2: import Scanner class

step 3: create class CPU and initialize variable price

step 4: create public inner class Processor and initialize variables no of cores and manufacturer

step 5: create public static nested class RAM and initialize variables memory and manufacturer

step 6: create main class

step 7: create object for class CPU

step 8: create object for inner class Processor

step 9: create object for static nested class RAM

step 10: print all variables

step 11: stop

### PROGRAM

```
import java.util.Scanner;

class CPU
{
    double price=39999.89;
```

```

public class Processor
{
    int ncores=8;
    String pmanufact="Intel";
}

public static class RAM
{
    int memory=4;
    String rmanufact="Kingston";
}
}

public class SystemInfo
{
    public static void main(String args[])
    {
        CPU c1=new CPU();
        CPU.Processor p1=new CPU().new Processor();
        CPU.RAM r1=new CPU.RAM();

        System.out.println("CPU Price");
        System.out.println(c1.price);
        System.out.println("No. of Cores");
        System.out.println(p1.ncores);
        System.out.println("Processor Manufacture");
        System.out.println(p1.pmanufact);
    }
}

```

```
System.out.println("RAM Memory");

System.out.println(r1.memory);

System.out.println("RAM Manufacture");

System.out.println(r1.rmanufact);

}

}
```

### OUTPUT

```
PS C:\Users\athul\Desktop\java\record\C01> javac SystemInfo.java
PS C:\Users\athul\Desktop\java\record\C01> java SystemInfo
CPU Price
39999.89
No. of Cores
8
Processor Manufacture
Intel
RAM Memory
4
RAM Manufacture
Kingston
```

DATE: 31/03/2022

PROGRAM NO: 5

AIM : Program to Sort strings.

### ALGORITHM

step 1: start

step 2: import Scanner and Arrays class

step 3: create class for sorting string and its corresponding method

step 4: create main class

step 5: create object

step 6: read set of strings as array from user

step 7: call method for sorting strings

step 8: print sorted strings

step 9: stop

### PROGRAM

```
import java.util.Scanner;
import java.util.Arrays;
class String_Sort
{
    String[] sort(String a[],int n)
    {
        String[] arr=a;
        for(int i=0;i<n;i++)
```

```

    {
        for(int j=i+1;j<n;j++)
        {
            if(arr[i].compareTo(arr[j])>0)
            {
                String temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    return arr;
}
}

public class String_program
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        String arr[]=new String[10];
        int i,n;
        String_Sort s = new String_Sort();

        System.out.println("Enter number of string inputs");
        n=in.nextInt();
        System.out.println("Enter "+n+" strings");
    }
}

```

```

        for(i=0;i<n;i++)
            arr[i]=in.next();

        System.out.println("Given String");
        for(i=0;i<n;i++)
            System.out.print(arr[i]+"\\t");
        System.out.println();

        arr=s.sort(arr,n);

        System.out.println("After String Sort");
        for(i=0;i<n;i++)
            System.out.print(arr[i]+"\\t");

    }
}

```

### OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C02> javac String_program.java
PS C:\Users\athul\Desktop\java\record\C02> java String_program
Enter number of string inputs
3
Enter 3 strings
veena alvin divya
Given String
veena  alvin  divya
After String Sort
alvin  divya  veena

```

DATE: 01/04/2022

PROGRAM NO: 6

AIM: Perform string manipulations.

### ALGORITHM

step 1: start  
step 2: import Scanner class  
step 3: create main class  
step 4: read string  
step 5: implement toLowerCase method  
step 6: implement toUpperCase method  
step 7: implement length method  
step 8: implement trim method  
step 9: implement replace method  
step 10: implement toString method  
step 11: stop

### PROGRAM

```
import java.util.Scanner;  
  
public class String_Manipulation  
{  
    public static void main(String args[])  
    {  
        Scanner in=new Scanner(System.in);
```

```

System.out.println("Enter the string");
String txt1=in.nextLine();
System.out.println("The string to lower case");
System.out.println(txt1.toLowerCase());
System.out.println("The string to upper case");
System.out.println(txt1.toUpperCase());
System.out.println("The length of given string");
System.out.println(txt1.length());
System.out.println("The trimmed string");
System.out.println(txt1.trim());
System.out.println("The char to be replaced");
// in.skip("[\r\n]+");
String a=in.nextLine();
System.out.println("The char replaced with");
// in.skip("[\r\n]+");
String b=in.nextLine();
System.out.println("The replaced string");
String txt2=txt1.replace(a,b);
System.out.println(txt2);

System.out.println("Enter an integer value to convert it into String");
Integer num=in.nextInt();
System.out.println("Value converted to String");
System.out.println(num.toString());
}
}

```



## OUTPUT

```
PS C:\Users\athul\Desktop\java\record\C02> javac String_Manipulation.java
PS C:\Users\athul\Desktop\java\record\C02> java String_Manipulation
Enter the string
Welcome to JAVA
The string to lower case
welcome to java
The string to upper case
WELCOME TO JAVA
The length of given string
15
The trimmed string
Welcome to JAVA
The char to be replaced
A
The char replaced with
0
The replaced string
Welcome to JOVO
Enter an integer value to convert it into String
123
Value converted to String
123
```

DATE: 04/04/2022

PROGRAM NO: 7

AIM: Program to create a class for Employee having attributes eNo, eName eSalary. Read n employ information and Search for an employee given eNo, using the concept of Array of Objects.

### ALGORITHM

step 1: start

step 2: import Scanner class

step 3: create class Employee

step 4: create constructor Employee to hold employee id,employee name and employee salary

step 5: create main class

step 6: create array object to hold multiple employees and their corresponding details.

step 7: create a menu driven system for implementing inserting an employee and searching for an employee.

step 8: stop

### PROGRAM

```
import java.util.Scanner;
```

```
class Employee
```

```
{
```

```
    int eNo;
```

```
    String eName;
```

```
    float eSalary;
```

```

Employee(int eid,String en,float es)
{
    eNo=eid;
    eName=en;
    eSalary=es;
}
}

public class Emp_list
{
    public static void main(String args[])
    {
        int choice,i=-1,flag=0;
        Employee[] emp=new Employee[10];
        Scanner in=new Scanner(System.in);

        while(true)
        {
            System.out.println("1.Insert new employee");
            System.out.println("2.Search for an employee");
            System.out.println("3.Exit");
            System.out.println("Enter your choice");
            choice=in.nextInt();

            switch(choice)
            {

```

```

case 1: System.out.println("Enter employee Number");
        int empid=in.nextInt();
        System.out.println("Enter employee Name");
        String empn=in.next();
        System.out.println("Enter employee Salary");
        float empf=in.nextFloat();
        emp[++i]=new Employee(empid,empn,empf);
        break;

case 2: System.out.println("Enter employee Number to be
searched");
        flag=0;
        empid=in.nextInt();
        for(int j=0;j<i+1;j++)
        {
            if(emp[j].eNo==empid)
            {
                System.out.println("Employee Number: "+emp[j].eNo);
                System.out.println("Employee Name: "+emp[j].eName);
                System.out.println("Salary: "+emp[j].eSalary);
                flag=1;
            }
        }
        if(flag==0)
        {
            System.out.println("Employee not found!!!");
        }
        break;

```

```

        case 3: System.exit(0);

        default: System.out.println("Invalid Choice");

    }

}

}

}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C02> javac Emp_list.java
PS C:\Users\athul\Desktop\java\record\C02> java Emp_list
1.Insert new employee
2.Search for an employee
3.Exit
Enter your choice
1
Enter employee Number
101
Enter employee Name
joseph
Enter employee Salary
15000
1.Insert new employee
2.Search for an employee
3.Exit
Enter your choice
1
Enter employee Number
201
Enter employee Name
Meera
Enter employee Salary
50000
1.Insert new employee
2.Search for an employee
3.Exit
Enter your choice
2
Enter employee Number to be searched
201
Employee Number: 201
Employee Name: Meera
Salary: 50000.0
1.Insert new employee
2.Search for an employee
3.Exit
Enter your choice
3

```

DATE: 07/04/2022

PROGRAM NO: 8

AIM : Area of different shapes using overloaded functions.

### ALGORITHM

step 1: start

step 2: create class Shapes

step 3: create method named area for square

step 4: create method named area for rectangle

step 5: create method named area for circle

step 6: create main class

step 7: create object

step 8: call corresponding methods

step 9: stop

### PROGRAM

```
class Shapes
{
    int area(int a)
    {
        return a*a;
    }
    int area(int a,int b)
    {
        return a*b;
```

```

    }

    double area(double r)
    {
        return 3.14*r*r;
    }
}

public class Area
{
    public static void main(String args[])
    {
        Shapes sh= new Shapes();
        System.out.println("Area Of Sqaure of length 5: "+sh.area(5));
        System.out.println("Area Of Rectangle of length 5 and breadth 4: "+sh.area(5,4));
        System.out.println("Area Of Circle of radius 2: "+sh.area(2.00));
    }
}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C03> javac Area.java
PS C:\Users\athul\Desktop\java\record\C03> java Area
Area Of Sqaure of length 5: 25
Area Of Rectangle of length 5 and breadth 4: 20
Area Of Circle of radius 2: 12.56

```

DATE: 18/04/2022

PROGRAM NO: 9

AIM: Create a class 'Person' with data members Name, Gender, Address, Age and a constructor to initialize the data members and another class 'Employee' that inherits the properties of class Person and also contains its own data members like Empid, Company\_name, Qualification, Salary and its own constructor. Create another class 'Teacher' that inherits the properties of class Employee and contains its own data members like Subject, Department, Teacherid and also contain constructors and methods to display the data members. Use array of objects to display details of N teachers.

ALGORITHM

- step 1: start
- step 2: import Scanner class
- step 3: create class Person
- step 4: create constructor to read values (name, gender, address, age)
- step 5: create method named display for displaying read values
- step 6: create class Employee
- step 7: create constructor to read values (empid, company name, qualification, salary)
- step 8: create method named display for displaying read values
- step 9: create class Teacher
- step 10: create constructor to read values (teacher id, department, subject)
- step 11: create method named display for displaying read values
- step 12: create main class
- step 13: create array for object Teacher
- step 14: create a menu driven system for insert teachers, search for a teacher
- step 15: stop



## PROGRAM

```
import java.util.Scanner;

class Person
{
    Scanner in=new Scanner(System.in);

    String name,gender,address;

    int age;

    Person()
    {
        System.out.println("Enter name");
        name=in.nextLine();

        System.out.println("Enter gender if male M if female F others O");
        gender=in.nextLine();

        System.out.println("Enter address");
        address=in.nextLine();

        System.out.println("Enter age");
        age=in.nextInt();
    }

    void display()
    {
        System.out.println("Name: "+name);
        System.out.println("Gender: "+gender);
        System.out.println("Address: "+address);
        System.out.println("Age: "+age);
    }
}
```

```

}
class Employee extends Person
{
    int empid;
    float salary;
    String cname,quali;
    Employee()
    {
        System.out.println("Enter Employee id");
        empid=in.nextInt();
        System.out.println("Enter Company Name");
        cname=in.next();
        System.out.println("Enter Education Qualification");
        in.skip("[\r\n]+");
        quali=in.next();
        System.out.println("Enter Salary");
        salary=in.nextFloat();
    }
    void display()
    {
        super.display();
        System.out.println("Employee id: "+empid);
        System.out.println("Company Name: "+cname);
        System.out.println("Education Qualification: "+quali);
        System.out.println("Salary: "+salary);
    }
}

```

```

    }
class Teacher extends Employee
{
    int tid;
    String sub,dept;
    Teacher()
    {
        System.out.println("Enter Teacher id");
        tid=in.nextInt();
        System.out.println("Enter Subject");
        sub=in.next();
        System.out.println("Enter Department");
        in.skip("[\r\n]+");
        dept=in.next();
    }
    void display()
    {
        super.display();
        System.out.println("Teacher id: "+tid);
        System.out.println("Subject: "+sub);
        System.out.println("Department: "+dept);
    }
}
public class Details
{
    public static void main(String args[])

```

```

{
    int i=-1;
    Teacher[] t=new Teacher[10];
    Scanner in=new Scanner(System.in);
    while(true)
    {
        System.out.println("1.Insert Teacher");
        System.out.println("2.Display Teachers");
        System.out.println("3.Exit");
        System.out.println("Enter your choice!!!");
        int choice = in.nextInt();
        switch(choice)
        {
            case 1:t[++i]=new Teacher();
                break;
            case 2:for(int j=0;j<=i;j++)
                t[j].display();
                break;
            case 3:System.exit(1);
                break;
            default:System.out.println("Invalid Choice");
        }
    }
}

```

## OUTPUT

```
PS C:\Users\athul\Desktop\java\record\C03> javac Details.java
PS C:\Users\athul\Desktop\java\record\C03> java Details
1.Insert Teacher
2.Display Teachers
3.Exit
Enter your choice!!!
1
Enter name
Jijo
Enter gender if male M if female F others O
M
Enter address
TVM
Enter age
35
Enter Employee id
101
Enter Company Name
TCS
Enter Education Qualification
Mtech
Enter Salary
75000
Enter Teacher id
201
Enter Subject
Matha
Enter Department
CSE
1.Insert Teacher
2.Display Teachers
3.Exit
Enter your choice!!!
2
Name: Jijo
Gender: M
Address: TVM
Age: 35
Employee id: 101
Company Name: TCS
Education Qualification: Mtech
Salary: 75000.0
Teacher id: 201
Subject: Matha
Department: CSE
1.Insert Teacher
2.Display Teachers
3.Exit
Enter your choice!!!
3
```

DATE: 25/04/2022

PROGRAM NO: 10

AIM: Create an interface having prototypes of functions area() and perimeter(). Create two classes Circle and Rectangle which implements the above interface. Create a menu driven program to find area and perimeter of objects.

### ALGORITHM

- step 1: start
- step 2: import Scanner class
- step 3: create interface Shapes and initialize area and perimeter
- step 4: create class Circle implements from class Shapes
- step 5: create corresponding area and perimeter for class Circle
- step 6: create class Rectangle implements from class Shapes
- step 7: create corresponding area and perimeter for class Rectangle
- step 8: create main class
- step 9: create menu driven system for finding area of circle, perimeter of circle, area of rectangle and perimeter of rectangle
- step 10: stop

### PROGRAM

```
import java.util.Scanner;

interface Shapes
{
    Scanner in=new Scanner(System.in);
    static double PI=3.14;
    double area();
```

```

        double perimeter();
    }

class Circle implements Shapes
{
    public double area()
    {
        System.out.println("Enter the radius");
        int radius=in.nextInt();
        return PI*radius*radius;
    }

    public double perimeter()
    {
        System.out.println("Enter the radius");
        int radius=in.nextInt();
        return 2*PI*radius;
    }
}

class Rectangle implements Shapes
{
    public double area()
    {
        System.out.println("Enter the length");
        int length=in.nextInt();

```

```

        System.out.println("Enter the breadth");
        int breadth=in.nextInt();
        return length*breadth;
    }

    public double perimeter()
    {
        System.out.println("Enter the length");
        int length=in.nextInt();
        System.out.println("Enter the breadth");
        int breadth=in.nextInt();
        return 2*(length+breadth);
    }
}

public class Dimensions
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        int choice;
        Circle c=new Circle();
        Rectangle r=new Rectangle();
        while(true)
        {
            System.out.println("1.Find Area of a Circle");

```



```
System.out.println("2.Find Perimeter of a Circle");
System.out.println("3.Find Area of a Rectangle");
System.out.println("4.Find Perimeter of a Rectangle");
System.out.println("5.Exit");
System.out.println("Enter your choice");
choice=in.nextInt();
```

```
    switch(choice)
    {
        case 1:System.out.println(c.area());
                break;
        case 2:System.out.println(c.perimeter());
                break;
        case 3:System.out.println(r.area());
                break;
        case 4:System.out.println(r.perimeter());
                break;
        case 5:System.exit(0);
                break;
        default:System.out.println("Invalid Choice!!!");
    }

}

}

}
```

## OUTPUT

```
PS C:\Users\athul\Desktop\java\record\C03> javac Dimensions.java
PS C:\Users\athul\Desktop\java\record\C03> java Dimensions
1.Find Area of a Circle
2.Find Perimeter of a Circle
3.Find Area of a Rectangle
4.Find Perimeter of a Rectangle
5.Exit
Enter your choice
1
Enter the radius
2
12.56
1.Find Area of a Circle
2.Find Perimeter of a Circle
3.Find Area of a Rectangle
4.Find Perimeter of a Rectangle
5.Exit
Enter your choice
3
Enter the length
4
Enter the breadth
3
12.0
1.Find Area of a Circle
2.Find Perimeter of a Circle
3.Find Area of a Rectangle
4.Find Perimeter of a Rectangle
5.Exit
Enter your choice
5
```

DATE: 28/04/2022

PROGRAM NO: 11

AIM: Create a Graphics package that has classes and interfaces for figures Rectangle, Triangle, Square and Circle. Test the package by finding the area of these figures.

### ALGORITHM

#### Package program

- step 1: start
- step 2: create package pack;
- step 3: create public class named arp
- step 4: create interface arp and initialize methods for area and perimeter
- step 5: create public class rectangle implements from class arp
- step 6: create public methods area and perimeter for class rectangle
- step 7: create public class circle implements from class arp
- step 8: create public methods area and perimeter for class circle
- step 9: stop

#### Main program

- step 1: start
- step 2: import pack package
- step 3: create main class
- step 4: read length and breadth from user
- step 5: create object for class rectangle
- step 6: call method for finding area and perimeter for rectangle
- step 7: display corresponding results of rectangle
- step 8: read radius from user

step 9: create object for class circle

step 10: call method for finding area and perimeter for circle

step 11: display corresponding results of circle

step 12: stop

### PROGRAM

Package program: arp.java

```
package pack;
public class arp
{
interface arpe
{
    float area (float x, float y);
    float perimeter (float a,float b);
}
public class rectangle implements arpe
{
    public float area(float x,float y)
    {
        float sum=x*y;
        return sum;
    }
    public float perimeter (float a,float b)
    {
        float sum=a+a+b+b;
```

```

        return sum;
    }
}
public class circle implements arpe
{
    public float area(float x,float y)
    {
        float sum = x*x*y;
        return sum;
    }
    public float perimeter(float a,float b)
    {
        float sum=2*b*a;
        return sum;
    }
}
}

```

Main program: findarea.java

```

import java.util.Scanner;
import pack.arp;
public class findarea
{
    public static void main(String[] args)
    {

```

```

float area, perimeter;

Scanner in =new Scanner(System.in);

arp h=new arp();

System.out.println("Enter the length of the rectangle");

float a=in.nextFloat();

System.out.println("enter the breadth of the rectangle");

float b=in.nextFloat();

arp.rectangle r=h.new rectangle();

area=r.area(a,b);

perimeter=r.perimeter(a,b);

System.out.println("rectangle : area = "+area+" perimeter =
"+perimeter);

System.out.println("Enter the radius of the circle");

float c=in.nextFloat();

float pie=3.14f;

arp.circle cir=h.new circle();

area=cir.area(c,pie);

perimeter=cir.perimeter(c,pie);

System.out.println("circle : area = "+area+" perimeter = "+perimeter);

}

}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C04> javac -d . arp.java
PS C:\Users\athul\Desktop\java\record\C04> javac findarea.java
PS C:\Users\athul\Desktop\java\record\C04> java findarea
Enter the length of the rectangle
5
enter the breadth of the rectangle
2
rectangle : area = 10.0 perimeter = 14.0
Enter the radius of the circle
2
circle : area = 12.56 perimeter = 12.56

```

DATE: 28/04/2022

PROGRAM NO: 12

AIM: Write a user defined exception class to authenticate the user name and password.

### ALGORITHM

step 1: start

step 2: import Exception and Scanner class

step 3: create class Authenticate extends Exception

step 4: create constructor which inherits message from super class exception when an error is thrown

step 5: create constructor to hold multiple userid and password

step 6: create main class

step 7: create array of objects for class Authenticate

step 8: inside try block create a menu driven system for sign in and log in

step 9: throw Authenticate exception with message when userid and/or password miss match

step 10: catch Authenticate exception in catch block and display the thrown message

step 11: stop

### PROGRAM

```
import java.lang.Exception;
import java.util.Scanner;
class Authenticate extends Exception
{
    String uid,pswd;
```

```

Authenticate(String message)
{
    super(message);
}
Authenticate(String userid,String password)
{
    uid=userid;
    pswd=password;
}
}

```

```

public class Login
{
    public static void main(String args[])
    {

        Authenticate a[]=new Authenticate[20];
        Scanner in=new Scanner(System.in);
        try
        {
            int ch=0,i=-1,flag=0;

            while(true)
            {
                System.out.println("1.Sign Up");
                System.out.println("2.Login");
            }
        }
    }
}

```



```

System.out.println("3.Exit");
System.out.println("Enter your choice");
ch=in.nextInt();

switch(ch)
{
    case 1: System.out.println("Enter new login name");
        String user_id=in.next();
        System.out.println("Enter new login password");
        in.skip("[\r\n]+");
        String passwrđ=in.next();
        a[++i]=new Authenticate(user_id,passwrđ);
        System.out.println("New User added");
        break;

    case 2: System.out.println("Enter login name");
        user_id=in.next();
        System.out.println("Enter login password");
        in.skip("[\r\n]+");
        passwrđ=in.next();
        flag=0;
        for(int j=0;j<i+1;j++)
        {
            if(a[j].uid.matches(user_id))
            {
                if(a[j].pswd.matches(passwrđ))

```

```

        {
            System.out.println("Welcome "+a[j].uid+" Login
Success!!!");
            flag=1;
        }
    }
}
if(flag==0)
{
    throw new Authenticate("Invalid Credentials Please Check
before Login!!!");
}
break;

case 3: System.exit(1);
    break;

default: System.out.println("Invalid Option!!!");
}
}
}
catch(Authenticate e)
{
    System.out.println(e.getMessage());
}
}
}

```

## OUTPUT

```
PS C:\Users\athul\Desktop\java\record\C04\Exception> javac Login.java
PS C:\Users\athul\Desktop\java\record\C04\Exception> java Login
1.Sign Up
2.Login
3.Exit
Enter your choice
1
Enter new login name
Admin
Enter new login password
Admin123
New User added
1.Sign Up
2.Login
3.Exit
Enter your choice
2
Enter login name
Admin
Enter login password
Admin123
Welcome Admin Login Success!!!
1.Sign Up
2.Login
3.Exit
Enter your choice
2
Enter login name
user
Enter login password
user123
Invaild Credentials Please Check before Login!!!
```

DATE: 10/05/2022

PROGRAM NO: 13

AIM: Find the average of N positive integers, raising a user defined exception for each negative input.

### ALGORITHM

- step 1: start
- step 2: import Exception and Scanner class
- step 3: create class Check which extends from super class Exception
- step 4: create constructor which inherits message from super class exception when an error is thrown
- step 5: create main class
- step 6: on try block take values from user to find average
- step 7: if negative value is given throw Check Exception with message
- step 8: on catch block the message is displayed which was thrown
- step 9: at finally block the average of values before negative value is evaluated and result is displayed
- step 10: stop

### PROGRAM

```
import java.lang.Exception;  
import java.util.Scanner;  
class Check extends Exception  
{  
    Check(String message)  
    {
```

```

        super(message);
    }
}

public class Array_avg
{
    public static void main(String args[])
    {
        int a[]=new int[10];
        int sum=0,k=0;
        Scanner in=new Scanner(System.in);

        try
        {
            int i=-1;
            System.out.println("Enter values: ");
            while(true)
            {
                a[++i]=in.nextInt();
                k++;
                if(a[i]<0)
                {
                    k--;
                    throw new Check("Enter only positive values");
                }
            }
        }
    }
}

```

```

    }
    catch(Check c)
    {
        System.out.println(c.getMessage());
    }

    finally
    {
        for(int j=0;j<k;j++)
        {
            sum+=a[j];
        }
        float avg= sum/k;
        System.out.println("Average of given values in array is : "+avg);
    }
}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C04\Exception> javac Array_avg.java
PS C:\Users\athul\Desktop\java\record\C04\Exception> java Array_avg
Enter values:
10 20 -12 30
Enter only positive values
Average of given values in array is : 15.0

```

DATE: 16/05/2022

PROGRAM NO: 14

AIM: Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads.  
(Runnable Interface)

### ALGORITHM

step 1: start

step 2: import Scanner class

step 3: create class Fibonacci which is implemented from interface Runnable

step 4: create constructor to read range

step 5: inside run method program to display Fibonacci series with in range

step 6: create class Even which is implemented from interface Runnable

step 7: create constructor to read starting and ending range

step 8: inside run method program to display even numbers within range

step 9: create main class

step 10: create object for Fibonacci and create corresponding Thread object from Fibonacci object

step 11: create object for Even and create corresponding Thread object from Even object

step 12: start Fibonacci thread

step 13: start Even thread

step 14: stop

## PROGRAM

```
import java.util.Scanner;

class Fibonacci implements Runnable
{
    int n;

    Scanner in=new Scanner(System.in);

    Fibonacci()
    {
        System.out.println("Enter the range");
        n=in.nextInt();
    }

    public void run()
    {
        int a=0,b=1,c=0;

        System.out.println("The 1 fibonacci number is: "+a);
        System.out.println("The 2 fibonacci number is: "+b);
        for(int i=2;i<n;i++)
        {
            c=a+b;

            System.out.println("The "+(i+1)+" fibonacci number is: "+c);
            a=b;
            b=c;
        }
    }
}
```



```

class Even implements Runnable
{
    int n1,n2;
    Scanner in=new Scanner(System.in);
    Even()
    {
        System.out.println("Enter the starting range");
        n1=in.nextInt();
        System.out.println("Enter the ending range");
        n2=in.nextInt();
    }
    public void run()
    {
        for(int i=n1;i<=n2;i++)
        {
            if(i%2==0)
                System.out.println(i+" is even");
        }
    }
}

```

```

class RInterface
{
    public static void main(String args[])
    {
        Fibonacci runnable1=new Fibonacci();
    }
}

```

```

Thread threadf=new Thread(runnable1);

Even runnable2=new Even();
Thread threade=new Thread(runnable2);

threadf.start();
threade.start();
}
}

```

## OUTPUT

```

PS C:\Users\athul\Desktop\java\record\C04> javac RInterface.java
PS C:\Users\athul\Desktop\java\record\C04> java RInterface
Enter the range
5
Enter the starting range
1
Enter the ending range
10
The 1 fibonacci number is: 0
2 is even
4 is even
6 is even
The 2 fibonacci number is: 1
8 is even
10 is even
The 3 fibonacci number is: 1
The 4 fibonacci number is: 2
The 5 fibonacci number is: 3

```

DATE: 18/05/2022

PROGRAM NO: 15

AIM: Program to find maximum of three numbers using AWT.

ALGORITHM

step 1: start

step 2: import java.awt.\*

step 3: java.awt.event.\*

step 4: import java.applet.\*

step 5: create class Myapplet extends from subclass Applet and implements from interface ActionListener

step 6: initialize variables for finding maximum of three numbers

step 7: initialize TextFiled for three numbers and button to execute program

step 8: create method init and add Labels, TextFiled and Buttons which you liked to be displayed

step 9: add actionListener to corresponding Button

step 10: create method paint with object of class Graphics to display result

step 11: create method actionPerformed with object of class(ActionEvent)

step 12: get corresponding values from text filed and convert it into integers

step 13: implement program for maximum of three numbers

step 14: call repaint method to update string result

step 15: write applet HTML code as comment

step 16: stop

## PROGRAM

Myapplet.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class MyApplet extends Applet implements ActionListener
{
    int f,s,t,big;
    TextField fno=new TextField(15);
    TextField sno=new TextField(15);
    TextField tno=new TextField(15);
    Button b=new Button("Find");
    public void init()
    {
        add(new Label("Enter First Number:"));
        add(fno);
        add(new Label("Enter Second Number:"));
        add(sno);
        add(new Label("Enter Third Number:"));
        add(tno);
        add(b);
        b.addActionListener(this);
    }
    public void paint(Graphics g)
    {
```

```

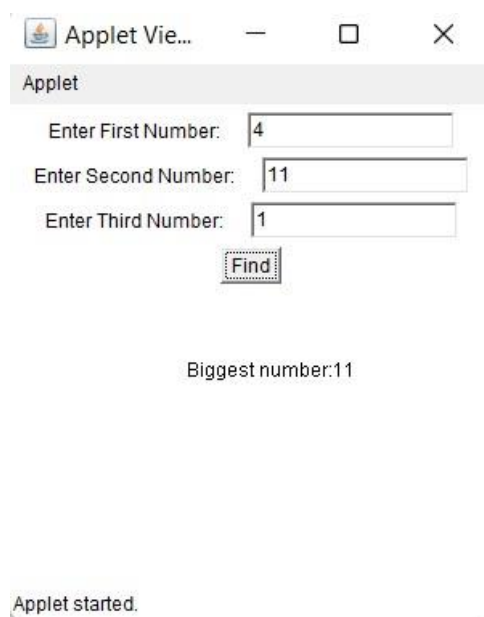
        g.drawString("Biggest number:" +big,110,170);
    }
    public void actionPerformed(ActionEvent e)
    {
        int f= Integer.parseInt(fno.getText());
        int s= Integer.parseInt(sno.getText());
        int t= Integer.parseInt(tno.getText());
        if(f > s)
        {
            if(f > t)
                big=f;
            else
                big=t;
        }
        else
        {
            if(s > t)
                big=s;
            else
                big=t;
        }
        repaint();
    }
}

/*<applet code="Myapplet.class" width=300 height=300>
</applet>*/

```

## OUTPUT

```
PS C:\Users\athul> cd Desktop\java\record\C05
PS C:\Users\athul\Desktop\java\record\C05> javac Myapplet.java
PS C:\Users\athul\Desktop\java\record\C05> appletviewer Myapplet.java
|
```



PROGRAM NO: 16

AIM: Implement a simple calculator using AWT components.

ALGORITHM

step 1: start

step 2: import java.awt.\*

step 3: java.awt.event.\*

step 4: import java.applet.\*

step 5: create class Calculator extends from subclass Applet and implements from interface ActionListener

step 6: initialize variables for getting two values and result

step 7: initialize TextFiled for two numbers and result

step 8: buttons to execute operations (add, mul, sub, div, moddiv)

step 9: create method init and add Labels, TextFiled and Buttons which you liked to be displayed

step 10: add actionListener to corresponding Operation Buttons

step 11: create method actionPerformed with object of class(ActionEvent)

step 12: get corresponding values from text filed and convert it into integers

step 13: getActionCommand for corresponding operation button

step 14: compare the operator and perform appropriate operation

step 15: write result on result TextField

step 16: call repaint method to update result on applet interface

step 17: write applet HTML code as comment

step 18: stop

## PROGRAM

Calculator.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class Calculator extends Applet implements ActionListener
{
    int a,b,res;
    TextField fno=new TextField(10);
    TextField sno=new TextField(10);
    TextField rno=new TextField(10);
    Button b1=new Button("+");
    Button b2=new Button("-");
    Button b3=new Button("*");
    Button b4=new Button("/");
    Button b5=new Button("%");
    Button b6=new Button("Clear");
    public void init()
    {
        add(new Label("Enter First Number:"));
        add(fno);
        add(new Label("Enter Second Number:"));
        add(sno);
        add(new Label("Result:"));
```



```

        add(rno);
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        b5.addActionListener(this);
        b6.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        int a= Integer.parseInt(fno.getText());
        int b= Integer.parseInt(sno.getText());
        String S=e.getActionCommand();
        if(S.equals("+"))
        {
            res=a+b;
        }
        else if(S.equals("-"))
        {
            res=a-b;
        }
    }

```

```

    }
    else if(S.equals("*"))
    {
        res=a*b;
    }
    else if(S.equals("/"))
    {
        res=a/b;
    }
    else if(S.equals("%"))
    {
        res=a%b;
    }
    rno.setText(res+"");
    if(S.equals("Clear"))
    {
        fno.setText("");
        sno.setText("");
        rno.setText("");
    }
    repaint();
}

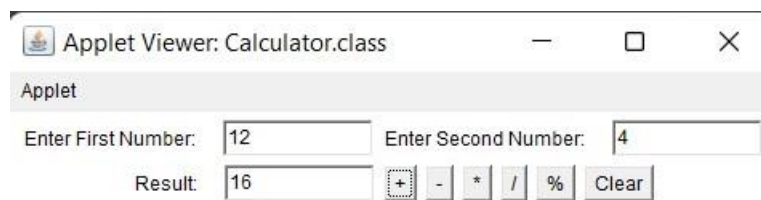
}

/*<applet code="Calculator.class" width=480 height=300>
</applet>*/

```

## OUTPUT

```
PS C:\Users\athul> cd Desktop\java\record\C05
PS C:\Users\athul\Desktop\java\record\C05> javac Calculator.java
PS C:\Users\athul\Desktop\java\record\C05> appletviewer Calculator.java
```



Applet started.

DATE: 30/05/2022

PROGRAM NO: 17

AIM: Develop a program to handle all mouse events.

### ALGORITHM

step 1: start

step 2: import java.awt.\*

step 3: java.awt.event.\*

step 4: import java.applet.\*

step 5: create class Mouse\_Event extends from subclass Applet and implements from interface MouseListener, MouseMotionListener

step 6: initialize variables for getting current coordinates x and y and string variable for getting status message

step 7: create method init and add MouseListener, MouseMotionListener

step 8: create public method mouseClicked with object of class MouseEvent

step 9: get message on string variable and repaint the display

step 10: create public method mouseEntered with object of class MouseEvent

step 11: get message on string variable and repaint the display

step 12: create public method mouseExited with object of class MouseEvent

step 13: get message on string variable and repaint the display

step 14: create public method mousePressed with object of class MouseEvent

step 15: get message on string variable and live coordinates on coordinate variables and repaint the display

step 16: create public method mouseReleased with object of class MouseEvent

step 17: get message on string variable and live coordinates on coordinate variables and repaint the display.

step 18: create public method mouseDragged with object of class MouseEvent

step 19: get message on string variable and live coordinates on coordinate variables, showStatus() then repaint the display

step 20: create public method mouseMoved with object of class MouseEvent

step 21: get live coordinates using showStatus()

step 22: create method paint with object of class Graphics to display status message

step 23: write applet HTML code as comment

step 24: stop

### PROGRAM

Mouse\_Event.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Mouse_Event extends Applet implements
MouseListener, MouseMotionListener
{
    int x=10;
    int y=10;
    String msg="";
    public void init()
    {
        addMouseListener(this);
```

```
        addMouseMotionListener(this);

    }

    public void mouseClicked(MouseEvent e)
    {
        msg="Mouse Clicked";
        repaint();
    }

    public void mouseEntered(MouseEvent e)
    {
        msg="Mouse Entered";
        repaint();
    }

    public void mouseExited(MouseEvent e)
    {
        msg="Mouse Exited";
        repaint();
    }

    public void mousePressed(MouseEvent e)
    {
        x=e.getX();
        y=e.getY();
        msg="Down";
        repaint();
    }
}
```

```

public void mouseReleased(MouseEvent e)
{
    x=e.getX();
    y=e.getY();
    msg="Up";
    repaint();
}

public void mouseDragged(MouseEvent e)
{
    x=e.getX();
    y=e.getY();
    msg="*";
    showStatus("Dragging mouse at "+x+" & "+y);
    repaint();
}

public void mouseMoved(MouseEvent e)
{
    showStatus("Moving mouse at "+e.getX()+" & "+e.getY());
}

public void paint(Graphics g)
{
    g.drawString(msg,x,y);
}

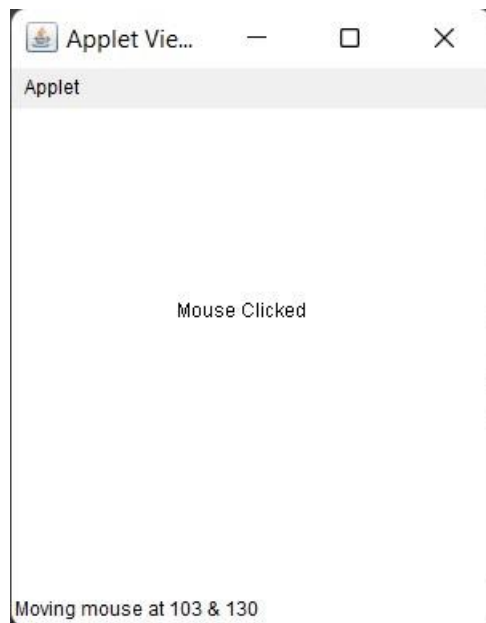
}

/*<applet code="Mouse_Event.class" width=300 height=300>
</applet>*/

```

## OUTPUT

```
PS C:\Users\athul> cd Desktop\java\record\C05
PS C:\Users\athul\Desktop\java\record\C05> javac Mouse_Event.java
PS C:\Users\athul\Desktop\java\record\C05> appletviewer Mouse_Event.java
```





PROGRAM NO: 18

AIM : Develop a program to handle Key events.

ALGORITHM

step 1: start

step 2: import java.awt.\*

step 3: java.awt.event.\*

step 4: import java.applet.\*

step 5: create class Key\_Event extends from subclass Applet and implements from interface KeyListener

step 6: initialize variables for fixing message coordinates x and y and string variable for getting status message

step 7: create method init and add KeyListener and requestFocus()

step 8: create public method keyPressed with object of class KeyEvent

step 9: showStatus() with message “Key Down”

step 10: create public method keyReleased with object of class KeyEvent

step 11: showStatus() with message “Key Up”

step 12: create public method keyTyped with object of class KeyEvent

step 13: get current pressed key using getKeyChar() and store it into message string variable and repaint the display

step 14: create method paint with object of class Graphics to display status message with fixed coordinates

step 15: write applet HTML code as comment

step 16: stop

## PROGRAM

Key\_Event.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class Key_Event extends Applet implements KeyListener
{
    int x=10;
    int y=20;
    String msg="";
    public void init()
    {
        addKeyListener(this);
        requestFocus();
    }
    public void keyPressed(KeyEvent e)
    {
        showStatus("Key Down");
    }
    public void keyReleased(KeyEvent e)
    {
        showStatus("Key Up");
    }
    public void keyTyped(KeyEvent e)
```

```

{
    msg+=e.getKeyChar();
    repaint();
}

public void paint(Graphics g)
{
    g.drawString(msg,x,y);
}
}

/*<applet code="Key_Event.class" width=300 height=300>
</applet>*/

```

## OUTPUT

```

PS C:\Users\athul> cd Desktop\java\record\C05
PS C:\Users\athul\Desktop\java\record\C05> javac Key_Event.java
PS C:\Users\athul\Desktop\java\record\C05> appletviewer Key_Event.java

```



Key Up

DATE: 09/06/2022

PROGRAM NO: 19

AIM: Write a program to write to a file, then read from the file and display the contents on the console.

### ALGORITHM

step 1: start

step 2: import java.io.\*

step 3: create main class by adding throws IOException and EOFException

step 4: create objects for FileInputStream and FileOutputStream

step 5: open file “data.txt” as the object for FileOutputStream

step 6: take a string and convert it into bytes

step 7: write converted byte string to “data.txt”

step 8: close output file

step 9: open “data.txt” as input file

step 5: while read file until -1.then,

step 6: do display of file as character of corresponding byte

step 7: close input file

### PROGRAM

FileBasicByte.java

```
import java.io.*;
class FileBasicByte
{
```

```

public static void main(String args[])throws IOException,EOFException

{
    FileOutputStream outs=null;
    FileInputStream ins=null;
    int b;
    outs = new FileOutputStream("data1.txt");
    String wr="This is the implementation of write and read operation";
    byte[] arr = wr.getBytes();
    outs.write(arr);
    outs.close();
    ins = new FileInputStream("data1.txt");
    while((b=ins.read())!=-1)
        System.out.print((char) b);
    ins.close();
}
}

```

## OUTPUT

```

PS C:\Users\athul> cd Desktop\java\record\C06
PS C:\Users\athul\Desktop\java\record\C06> javac FileBasicByte.java
PS C:\Users\athul\Desktop\java\record\C06> java FileBasicByte
This is the implementation of write and read operation

```

data1.txt

This is the implementation of write and read operation

PROGRAM NO: 20

AIM: Write a program to copy one file to another.

ALGORITHM

step 1: start

step 2: import java.io.\*

step 3: create main class and add throws IOException and EOFException

step 4: create objects for FileInputStream and FileOutputStream and give arguments as parameters

step 5: while read file until -1.then,

step 6: do write of file on outputfile corresponds to read of file using variable

step 7: close input file

step 8: close output file

step 9: stop

PROGRAM

data.txt

This is java file program for demonstrate  
copy the contents  
from  
one file  
to  
another file.

## FileCopy.java

```
import java.io.*;

class FileCopy
{
    public static void main(String args[])throws IOException,EOFException
    {

        FileOutputStream outs=null;
        FileInputStream ins=null;
        int b;
        outs = new FileOutputStream(args[1]);
        ins = new FileInputStream(args[0]);

        while((b=ins.read())!=-1)
            outs.write(b);
        System.out.println("Copy completed!!!");
        outs.close();
        ins.close();
    }
}
```

## OUTPUT

```
PS C:\Users\athul> cd Desktop\java\record\C06
PS C:\Users\athul\Desktop\java\record\C06> javac FileCopy.java
PS C:\Users\athul\Desktop\java\record\C06> java FileCopy data.txt newfile.txt
Copy completed!!!
```

newfile.txt

This is java file program for demonstrate  
copy the contents  
from  
one file  
to  
another file.