

# OBJECT ORIENTED PROGRAMMING with JAVA

BY

HINA FARYAL

BSIT 2<sup>ND</sup> EVENING

## OOP With JAVA

---

S.NO	List of Contents
1	Use Eclipse or Net bean platform and acquaint with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.
2	Write a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result. Handle any possible exceptions like divide by zero.
3	a) Write an applet program that displays a simple message b) Develop an applet that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the <b>button named “Compute” is clicked.</b>
4	Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.
5	Write a Java program that implements a multi-threaded program has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd the third thread will print the value of cube of the number.
6	Write a Java program for the following: i) Create a doubly linked list of elements. ii) Delete a given element from the above list. iii) Display the contents of the list after deletion.
7	Write a java program to simulate a traffic light. The program lets the user select one of the three lights: red, yellow or green. On selecting a <b>button, an appropriate message with ”Stop” or “Ready” or “ Go” should</b> appear above the buttons selected color. Initially, there is no message shown.
8	Write a java program to create an abstract class named shape that contains two integers and an empty method named print Area(). Provide three classes named Rectangle,, Triangle and Circle such that each one of the classes extends the class shape. Each one of the class contains

## OOP With JAVA

---

	only the method print Area() that print the area of the given shape
9	Suppose that table named Table.txt is stored in a text file. The first line in the file is the header, and the remaining lines correspond to rows in the table. The elements are separated by commas. Write a java program to display the table using in Grid Layout
10	Write a java program that handles all mouse events and shows the event name at the center of the window when mouse event is fired(Use Adapter classes).
11	Write a java program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by tab(\t).It takes a name or phone number as input and prints the corresponding other value from the hash table (hint : use hash tables)
12	a Java program that correctly implements the producer – consumer problem using the concept of inter-thread communication.
13	a Java program to list all the files in a directory including the files present in all its subdirectories.
14	Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending order

## **PROGRAM -1**

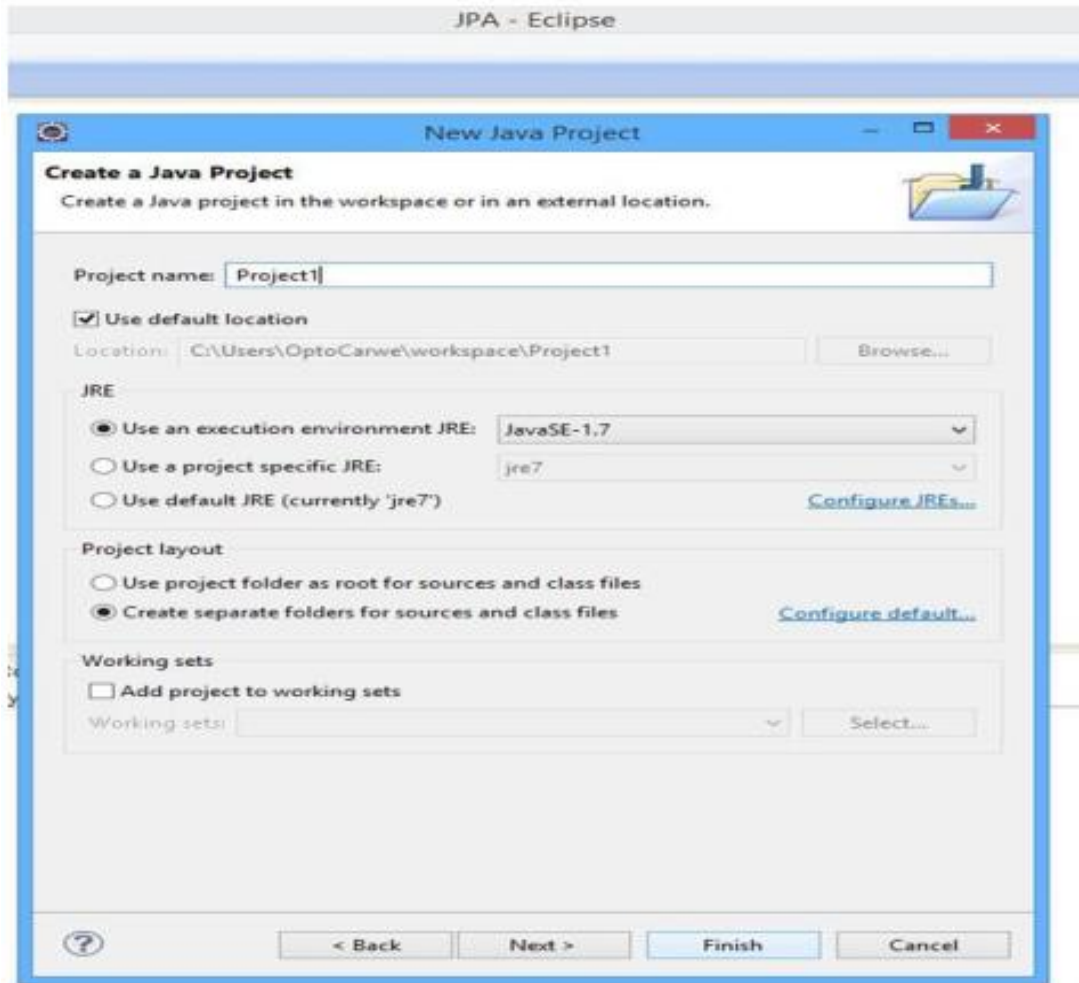
### **Steps to Execute Simple Java Program Using Eclipse**

**Step1: Begin by creating a new Java project.**

There are few different ways of accomplishing this. Click the arrow next to the left-most icon on the toolbar and select “Project” from the drop-down menu. Alternately start a new Java Project by choosing “File” then “New” followed by “Java Project”. .

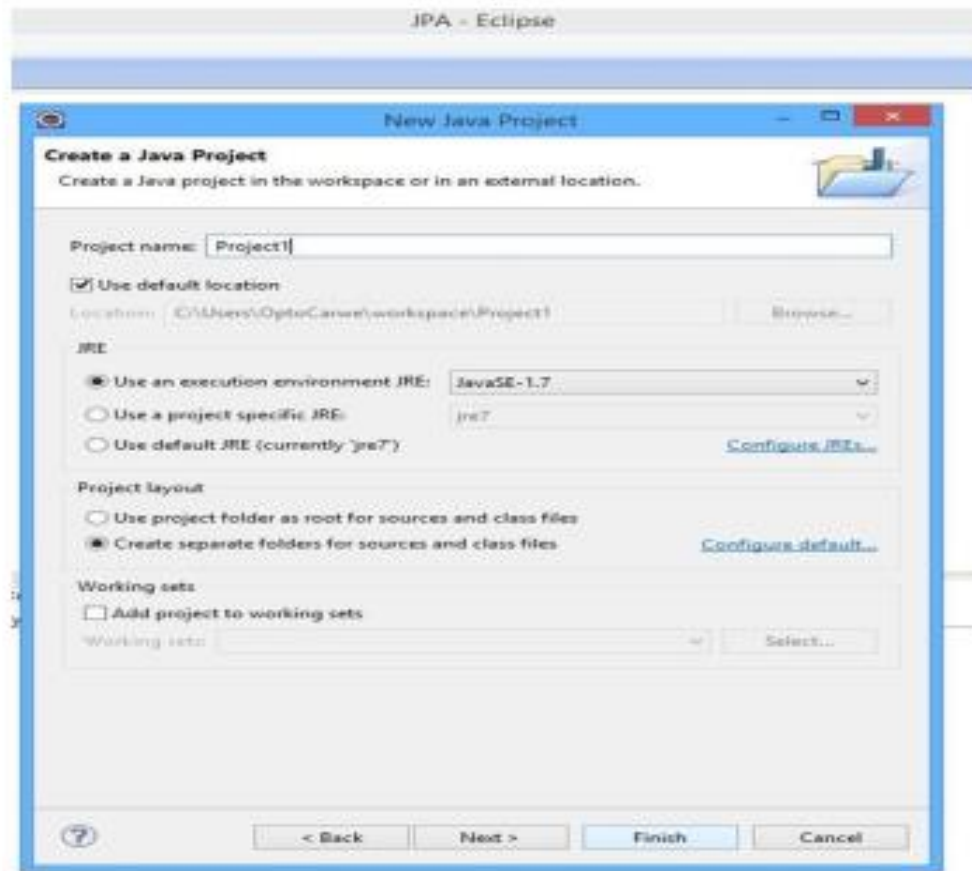
# OOP With JAVA

---



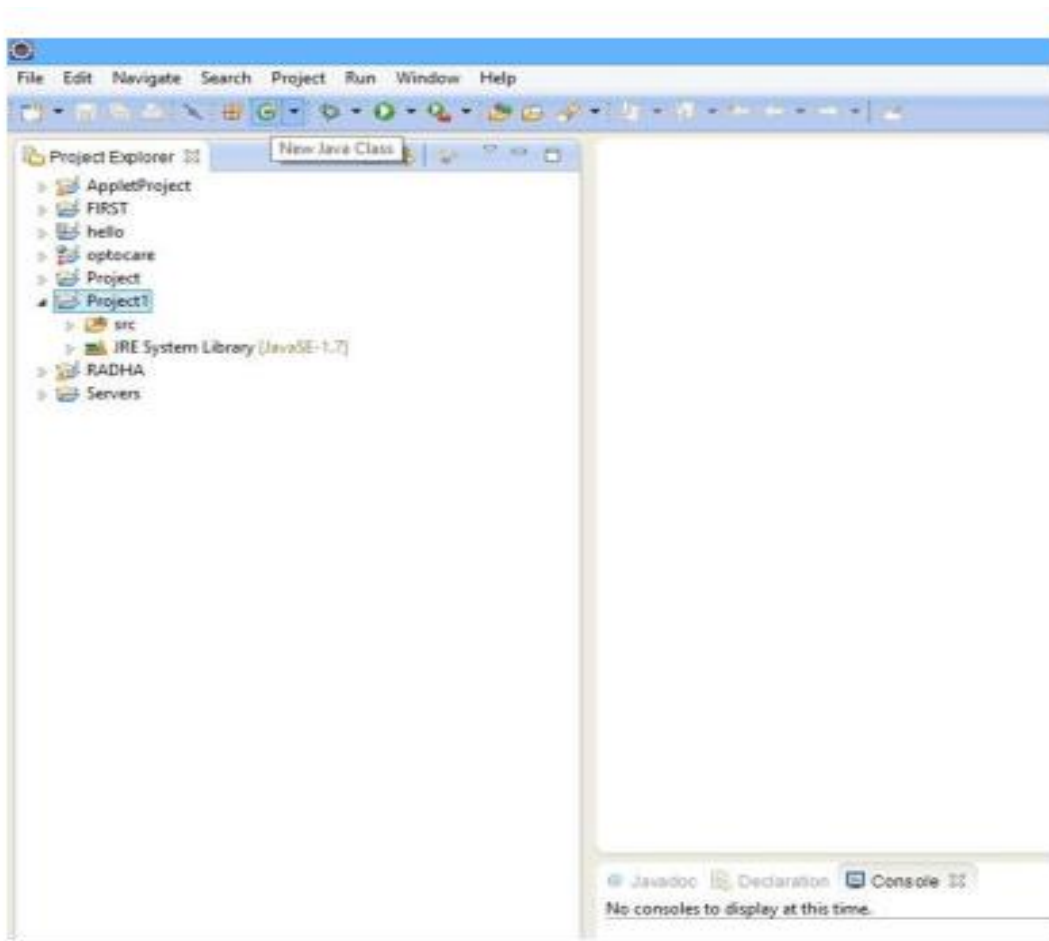
## Step2: Enter a Project Name

You will see a window titled "Create a Java Project". The buttons "Next" and "Finish" at the bottom of the window will be grayed out until a project name is entered in the first field. To proceed, give project name and enter it into this field then click "Finish". New project will appear on the left-hand side of the screen under "Package Explorer" among existing projects. Projects are listed in alphabetical order.



## Step3: Start a new java class.

Before begin writing code, need to create a new Java class. A class is a blueprint for an object. It defines the data stored in the object as well as its actions. Create a class by **clicking the “New Java”**.

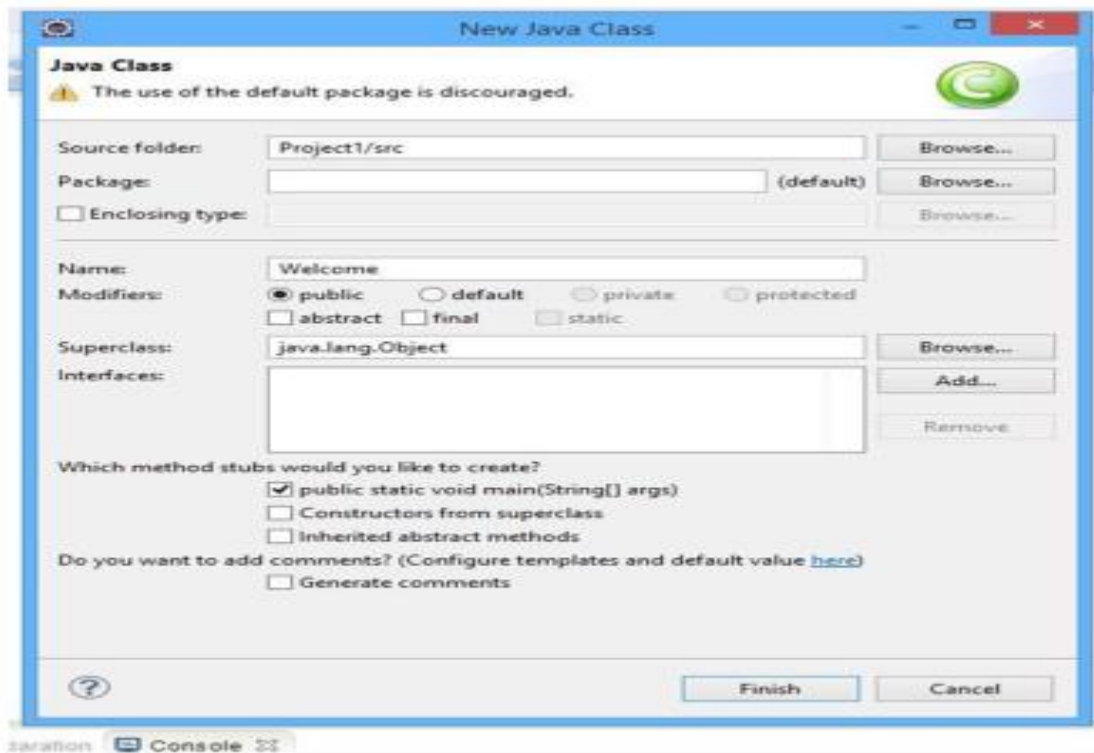


## OOP With JAVA

---

**STEP 4: Enter the name of your class.**

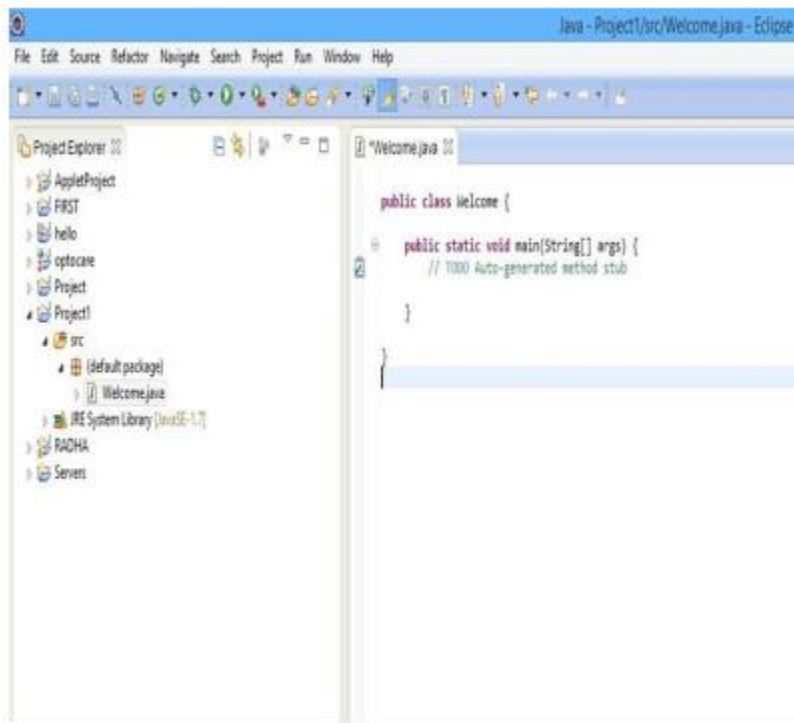
**You will see a window titled “Java Class.” To proceed, enter the name of class into the field “Name”. Since the class will be main class of the simple project, check the selection box labeled “public static void main(String[] args)” to include the method stub. Afterwards, click “Finish”.**





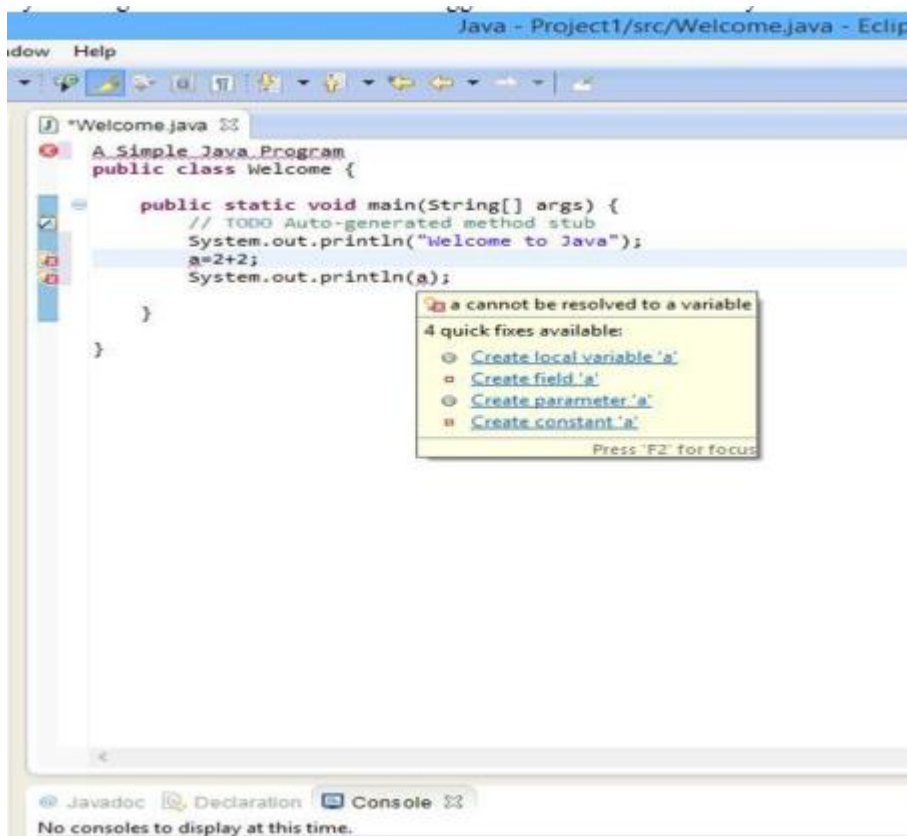
## Step5: Enter Java Code.

Here new class Welcome.java is created. It appears with the method stub **“public static void main(String[] args)”** along with some automatically generated comments. A method will contain a sequence of instructions to be executed by the program. A comment is a statement that is ignored by the compiler. Comments are used by programmers to document their code. Edit this file and insert the code for Java Program.



### Step6: Watch out for errors in code.

Any errors will be underlined in red, and icon with an "X" will show up on the left. Fix errors. By mousing over an error icon, can see a suggestion box that lists the ways can fix the error.

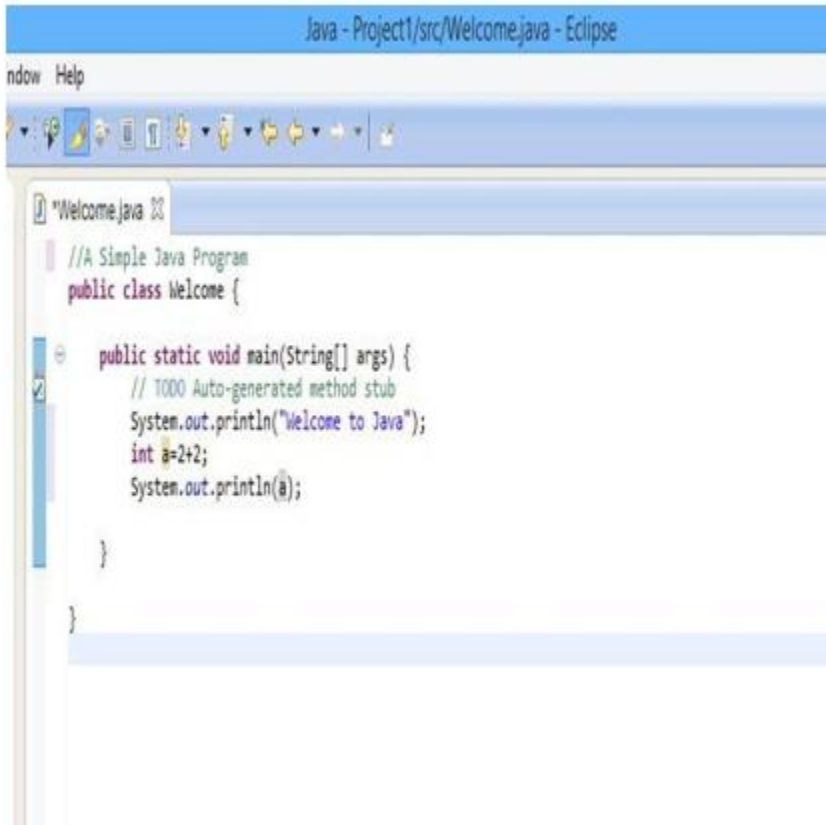


**Step7: Ensure that entire program is free of errors.**

There are three types of errors must beware of: syntax errors, run-time errors and logic errors. The compiler will alert syntax errors. Examples of syntax errors are misspelled variable names or missing semi-colons. Until remove all syntax errors from code program will not compile. The compiler will not catch run-time errors or logic errors.

# OOP With JAVA

---



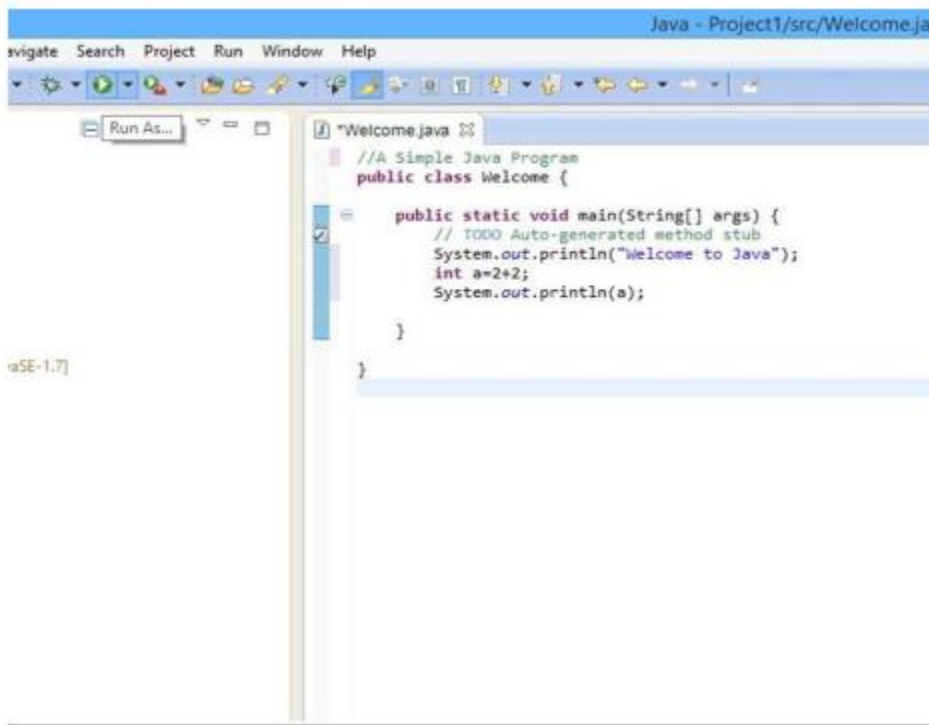
## **Step8: Compile Java Program.**

Now the program is free for errors, click the triangular icon to run program. Another way to run

## OOP With JAVA

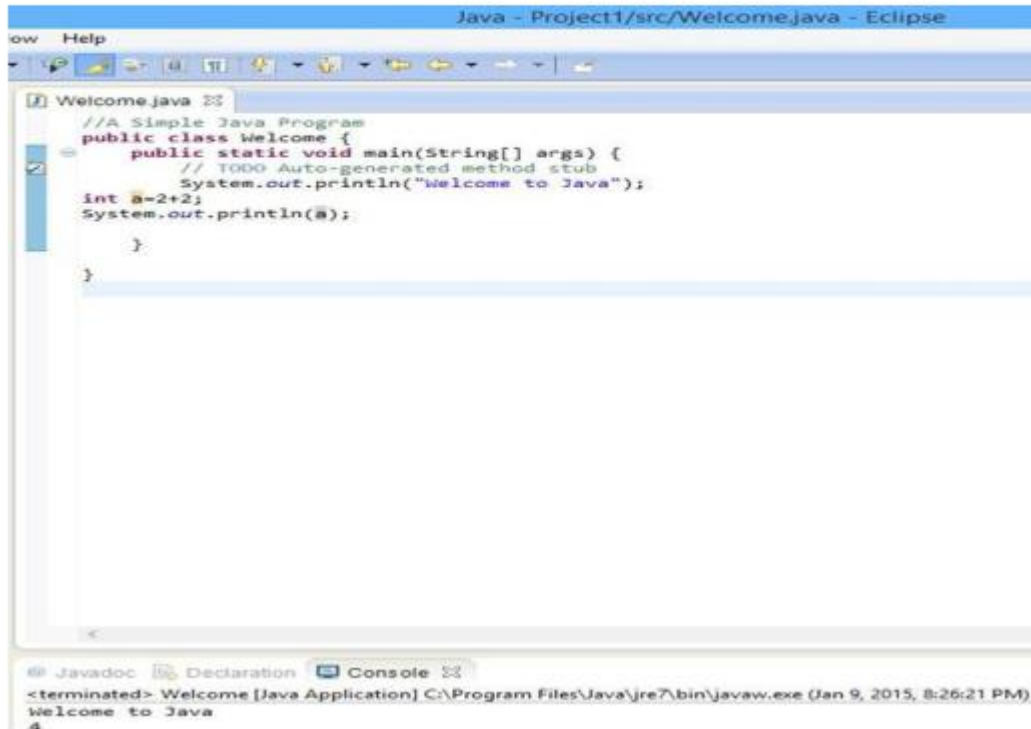
---

program is to select **“Run”** from the main menu and then select **“Run”** again from the drop-down menu. The shortcut is Ctrl+F11.



### Step9: Verify the output is what you expected.

When program runs, the output will be displayed on console at the bottom of the screen.



The screenshot shows the Eclipse IDE interface. The main editor window displays a Java file named 'Welcome.java' with the following code:

```
//A Simple Java Program
public class Welcome {
    public static void main(String[] args) {
        // TODO: Auto-generated method stub
        System.out.println("Welcome to Java");
        int a=2+2;
        System.out.println(a);
    }
}
```

Below the editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> Welcome [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jan 9, 2015, 8:26:21 PM)
Welcome to Java
4
```

### Step10: Fix any run-time or logic errors.

If the output is different from what you expected, then there might have been an error even though the program compiled. For example, if the output was zero instead of **four**, then there was a mistake in the program's calculation

### PROGRAM – 2

#### create a test project, add a test class, and run it

**Aim:** Use Eclipse or Net bean platform and acquaint with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

**Program:**

```
class PrimeNumbers
{
public static void main(String[] args)
{
try
{
System.out.println("***** PRIME NUMBERS *****");
Scanner objScanner = new Scanner(System.in);
System.out.print("\n Enter n Value:");
long n = objScanner.nextInt();
for (long i = 2; i <= n; i++)
{
boolean isprime = isNumPrime(i);
if (isprime)
{
System.out.print(i + " ");
} } }
catch (Exception e)
{
e.printStackTrace();
} }
public static boolean isNumPrime(long number)
{
boolean result = true;
for (long i = 2; i <= number / 2; i++)
{
```

```
if ((number % i) != 0)
{
    result = true

}
else
{
    result = false;
    break;
} }
return result;
} }
```

### VIVA QUESTIONS:

#### 1. What do you mean by Platform Independence?

Platform Independence you can run and compile program in one platform and can execute in any other platform.

#### 2. What is JIT Compiler?

Just-In-Time(JIT) compiler is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time

#### 3. What all memory areas are allocated by JVM?

Heap, Stack, Program Counter Register and Native Method Stack

#### 4. What is the base class of all classes?

java.lang.Object

#### 5. What are two different ways to call garbage collector?

System.gc() OR Runtime.getRuntime().gc().

#### 6. Use of finalize() method in java?

finalize () method is used to free the allocated resource.

#### 7. List two java IDEs?

1. Eclipse, 2.Net beans and 3.IntelliJ

#### 8. What are java buzzwords?

Java buzzwords explain the important features of java. They are Simple,Secured, Portable, architecture neutral, high performance, dynamic, robust,interpreted etc.

#### 9. Is byte code is similar to .obj file in C?



Yes, both are machine understandable codes No, .obj file directly understood by machine, byte code requires JVM.

### 10. What are length and length() in Java?

Both gives number of char/elements, length is variable defined in Array class,length() is method defined in String class.

## PROGRAM -2

### Design of Simple Calculator

**Aim:** Write a java program that works as a simple calculator. Use a GridLayout to arrangeButtons

for digits and for the + - \* % operations. Add a text field to display the result. Handle any possible exceptions like divide by zero.

**THEORY:** GridLayout is one of the Layout managers.A layout manager automatically arranges your

controls with in a window by using some type of algorithm.Grid Layout lays out component in a two

dimensional grid. When you instantiate a GridLayout,you define the number of rows and columns

#### Program:

```
import java.awt.*; import
java.awt.event.*; import
java.applet.*;
/*<applet code="Cal" width=300 height=300></applet>*/
public class Cal extends Applet implements ActionListener
{
String msg=" "; int
v1,v2,result; TextField t1;
Button b[]=new Button[10];
Button add,sub,mul,div,clear,mod,EQ; char OP;
public void init()
{
Color k=new Color(120,89,90); setBackground(k);
```

```
t1=new TextField(10);
GridLayoutgl=new GridLayout(4,5); setLayout(gl);
for(int i=0;i<10;i++)
{
b[i]=new Button(""+i);
}
add=new Button("add"); sub=new Button("sub"); mul=new Button("mul"); div=new
Button("div");
mod=new Button("mod"); clear=new Button("clear"); EQ=new Button("EQ");
t1.addActionListener(this);
add(t1);
for(int i=0;i<10;i++)
{
add(b[i]);}
add(add);
add(sub);

add(mul);
add(div);
add(mod);
add(clear);
add(EQ);
for(int i=0;i<10;i++)
{
b[i].addActionListener(this);
}
add.addActionListener(this);
sub.addActionListener(this);
mul.addActionListener(this);
div.addActionListener(this);
mod.addActionListener(this);
clear.addActionListener(this);
EQ.addActionListener(this);
}
public void actionPerformed(ActionEventae)
{
String str=ae.getActionCommand(); char ch=str.charAt(0);
if ( Character.isDigit(ch))
```

```
t1.setText(t1.getText()+str); else
if(str.equals("add"))
{
v1=Integer.parseInt(t1.getText());
OP='+';
t1.setText("");
}
else if(str.equals("sub"))
{
v1=Integer.parseInt(t1.getText()); OP='-';
t1.setText("");
} else if(str.equals("mul"))
{
v1=Integer.parseInt(t1.getText());
OP='*';
t1.setText("");
}
else if(str.equals("div"))
{
v1=Integer.parseInt(t1.getText());
OP='/';
t1.setText("");
}

else if(str.equals("mod"))
{
v1=Integer.parseInt(t1.getText());
OP='%';
t1.setText("");
}
if(str.equals("EQ"))
{
v2=Integer.parseInt(t1.getText());
if(OP=='+')
result=v1+v2; else if(OP=='-')
result=v1-v2; else if(OP=='*')
result=v1*v2; else if(OP=='/')
result=v1/v2; else if(OP=='%')
```

```
result=v1%v2;
t1.setText(""+result);
}
if(str.equals("clear"))
{
t1.setText("");
} } }
```

### VIVA QUESTIONS:

#### 1. What is object cloning?

The object cloning is used to create the exact copy of an object.

#### 2. When parseInt() method can be used?

This method is used to get the primitive data type of a certain String.

#### 3. java.util.regex consists of which classes?

java.util.regex consists of three classes - Pattern class, Matcher class and PatternSyntaxException class.

#### 4. Which package is used for pattern matching with regular expressions?

java.util.regex package is used for this purpose.

#### 5. Define immutable object?

An immutable object can't be changed once it is created.

#### 6. Explain Set Interface?

It is a collection of element which cannot contain duplicate elements. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

#### 7. What is function overloading?

If a class has multiple functions by same name but different parameters, it is known as Method Overloading.

#### 8. What is function overriding?

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding.

#### 9. What is the difference between yielding and sleeping?

When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state.

#### 10. What are Wrapper classes?

These are classes that allow primitive types to be accessed as objects. Example: Integer, Character, Double, Boolean etc.

## PROGRAM -3

### **Creates a User Interface to perform Integer Divisions**

#### **Aim:**

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

**THEORY:** The AWT supports a rich assortment of graphics methods. All graphics are drawn relative to a window. This can be the main window of an applet, a child window of an applet, or a standalone application window. The origin of each window is at the top-left corner and is 0,0 coordinates are specified in pixels. All output to a window takes place through a graphics context.

#### **Program:**

```
import java.awt.*;                                import
java.awt.event.*;                                import
java.applet.*;
/*<applet      code="Div"width=230      height=250></applet>*/      public
class      Div      extends      Applet      implements      ActionListener
{
String                                          msg;
TextField                                          num1,num2,res;Label      11,12,13;
Button                                          div;
public                                          void      init()
{
l1=new      Label("Number      1");      l2=new
Label("Number      2");      l3=new
Label("result");      num1=new
TextField(10);      num2=new
TextField(10);      res=new
TextField(10);      div=new
Button("DIV");
div.addActionListener(this);      add(l1);
add(num1);
add(l2);
add(num2);
```

```
add(13);
add(res);
add(div);
}
public void actionPerformed(ActionEvent ae)
{
    String arg=ae.getActionCommand();
    if(arg.equals("DIV"))
    {
        String s1=num1.getText();
        String s2=num2.getText();
        int num1=Integer.parseInt(s1); int
        num2=Integer.parseInt(s2);
        if(num2==0)
        {
            try
            {
                System.out.println(" ");
            }
            catch(Exception e)
            {
                System.out.println("ArithmeticException"+e);
            }
            msg="Arithmetic";
            repaint();
        }
        else if((num1<0)||(num2<0))
        {
            try
            {
                System.out.println("");
            }
            catch(Exception e)
            {
                System.out.println("NumberFormatException"+e);
            }
            msg="NumberFormatException";
        }
    }
}
```

```
repaint();
}
else
{
int num3=num1/num2;
res.setText(String.valueOf(num3));
} } }
public void paint(Graphics g)
{
g.drawString(msg,30,70);
} }
```

### VIVA QUESTIONS:

#### 1. What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors. It is mainly used to handle checked exceptions.

#### 2. What is difference between Checked Exception and Unchecked Exception?

##### i). Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

##### ii). Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException etc. Unchecked exceptions are not checked at compile-time.

#### 3. What is the base class for Error and Exception?

Throwable.

#### 4. What is finally block?

finally block is a block that is always executed

#### 5. Can finally block be used without catch?

Yes, by try block. finally must be followed by either try or catch.

#### 6. Is there any case when finally will not be executed?

finally block will not be executed if program exits (either by calling System.exit() or by causing a fatal error that causes the process to abort)

#### 7. What is exception propagation?

Forwarding the exception object to the invoking method is known as exception propagation.

#### 8. What is nested class?

A class which is declared inside another class is known as nested class. There are 4 types of

nested class member inner class, local inner class, anonymous inner class and static nested class.

### 9. What is nested interface?

Any interface i.e. declared inside the interface or class, is known as nested interface. It is static by default.

### 10. Can an Interface have a class?

Yes, they are static implicitly.

## PROGRAM -5:

### Multithreaded Program

**Aim:** Write a Java program that implements a multithreaded program has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd the third thread will print the value of cube of the number.

**THEORY:** The java run-time system depends on the threads for many things, and all the class

libraries are designed with multithreading in mind. In fact ,java uses threads to enable the entire environment to be asynchronous. This helps reduce inefficiency by preventing the waste of CPU cycles. The benefits of java's multithreading is that the main loop/polling mechanism is eliminated.

one thread can pause without stopping other parts of your program. when a thread blocks in a java

program, only the single thread that is blocked pauses. All other threads continue to run.

### Program:

```
import java.io.*; import
java.util.*;
class First extends Thread
{
public void run()
{
for(;;)
{
int r;
Random d = new Random(); roll =
d.nextInt(200) + 1;
```



```
System.out.println(r);
Thread t2=new Second(r);
Thread t3=new Third(r); try
{
Thread.sleep(1000);
if(roll%2==0)
t2.start(); else
t3.start();
}
catch(InterruptedException e){ }
} } }
class Second extends Thread

{
int r1; Second(int r)
{
r1=r;
}
public void run()
{
System.out.println("The square of number"+r1+"is:"+r1*r1);
} }
class Third extends Thread
{
int r1; Third(int r)
{
r1=r;
}
public void run()
{
System.out.println("The Cube of the Number"+r1+"is: "+r1*r1*r1);
} }
class Mthread
{
public static void main(String[] args)
{
Thread t1=new First(); System.out.println("press
Ctrl+c to stop....."); t1.start();
```

## } } VIVA QUESTIONS

### 1) What is multithreading?

Multithreading is a process of executing multiple threads simultaneously. Its main advantage is:

- Threads share the same address space.
- Thread is lightweight.
- Cost of communication between process is low.

### 2) What is thread?

A thread is a lightweight subprocess. It is a separate path of execution. It is called separate path of execution because each thread runs in a separate stack frame.

### 3) What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

### 4) What does join() method?

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

### 5) Is it possible to start a thread twice?

No, there is no possibility to start a thread twice. If we do, it throws an exception.

### 6) Can we call the run() method instead of start()?

yes, but it will not work as a thread rather it will work as a normal object so there will not be context-switching between the threads.

### 7) What about the daemon threads?

The daemon threads are basically the low priority threads that provide the background support to the user threads. It provides services to the user threads.

### 8) Can we make the user thread as daemon thread if thread is started?

No, if you do so, it will throw Illegal Thread State Exception

### 9) What is shutdown hook?

The shutdown hook is basically a thread i.e. invoked implicitly before JVM shuts down. So we can use it to perform clean up resource.

### 10) When should we interrupt a thread?

We should interrupt a thread if we want to break out the sleep or wait state of a thread.

## PROGRAM -6 :

**Aim:** Write a Java program for the following:

- i) Create a doubly linked list of elements.
- ii) Delete a given element from the above list.
- iii) Display the contents of the list after deletion

**THEORY:** A doubly-linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.

- i) Create a doubly linked list of elements

**Program:**

```
import java.util.NoSuchElementException;
public class DoublyLinkedListImpl<E> {
    private Node head;
    private Node tail;
    private int size;
    public DoublyLinkedListImpl() {
        size = 0;
    }
    /**
     * this class keeps track of each element information
     * @author java2novice
     *
     */
}
```

```
private class Node {
    E element;
    Node next;
    Node prev;
    public Node(E element, Node next, Node prev) {
        this.element = element;
        this.next = next;
        this.prev = prev;
    }
}
/**
 * returns the size of the linked list
 * @return
 */
public int size() { return size; }
/**

 * return whether the list is empty or not
 * @return
 */
public boolean isEmpty() { return size == 0; }
/**
 * adds element at the starting of the linked list
 * @param element
 */
public void addFirst(E element) {
    Node tmp = new Node(element, head, null);
    if(head != null ) {head.prev = tmp;}
    head = tmp;
    if(tail == null) { tail = tmp;}
    size++;
    System.out.println("adding: "+element);
}
/**
 * adds element at the end of the linked list
 * @param element
 */
public void addLast(E element) {
    Node tmp = new Node(element, null, tail);
    if(tail != null) {tail.next = tmp;}
    tail = tmp;
    if(head == null) { head = tmp;}
    size++;
    System.out.println("adding: "+element);
}
```

```
/**
 * this method walks forward through the linked list
 */
public void iterateForward(){
    System.out.println("iterating forward..");
    Node tmp = head;
    while(tmp != null){
        System.out.println(tmp.element);
        tmp = tmp.next;
    }
}

/**
 * this method walks backward through the linked list
 */
public void iterateBackward(){
    System.out.println("iterating backword..");
    Node tmp = tail;
    while(tmp != null){
        System.out.println(tmp.element);
        tmp = tmp.prev;
    }
}

/**
 * this method removes element from the start of the linked list
 * @return
 */
public E removeFirst() {
    if (size == 0) throw new NoSuchElementException();
    Node tmp = head;
    head = head.next;
    head.prev = null;
    size--;
    System.out.println("deleted: "+tmp.element);
    return tmp.element;
}

/**
 * this method removes element from the end of the linked list
 * @return
 */
public E removeLast() {
    if (size == 0) throw new NoSuchElementException();
    Node tmp = tail;
    tail = tail.prev;
    tail.next = null;
```

```
size--;  
System.out.println("deleted: "+tmp.element);  
return tmp.element;  
}  
public static void main(String a[]){  
DoublyLinkedListImpl<Integer> dll = new DoublyLinkedListImpl<Integer>();  
dll.addFirst(10);  
dll.addFirst(34);  
dll.addLast(56);  
dll.addLast(364);  
dll.iterateForward();  
dll.removeFirst();  
dll.removeLast();  
dll.iterateBackward();  
}  
}
```

ii) Delete a given element from the above list.

**Program:**

```
public class Delete
{
    public static void main(String[] args)
    {
        int n, x, flag = 1, loc = 0;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter no. of elements you want in array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter all the elements:");
        for (int i = 0; i < n; i++)
        {
            a[i] = s.nextInt();
        }
        System.out.print("Enter the element you want to delete:");
        x = s.nextInt();
        for (int i = 0; i < n; i++)
        {
            if(a[i] == x)
            {
                flag = 1;
                loc = i;
                break;
            }
            else
            {
                flag = 0;
            }
        }
        if(flag == 1)
        {
            for(int i = loc+1; i < n; i++)
            {
                a[i-1] = a[i];
            }
            System.out.print("After Deleting:");
            for (int i = 0; i < n-2; i++)
            {
                System.out.print(a[i]+" ");
            }
        }
        System.out.print(a[n-2]);
    }
    else
    {
        System.out.println("Element not found");
    }
}
```

```
    }
    System.out.print(a[n-2]);
}
else
{
    System.out.println("Element not found");
}
}
```

iii) Display the contents of the list after deletion.

**Program:**

```
import java.util.Scanner;
public class JavaProgram
{
```

```
public static void main(String args[])
{
    int size, i, del, count=0;
    int arr[] = new int[50];
    Scanner scan = new Scanner(System.in);
    System.out.print("Enter Array Size : ");
    size = scan.nextInt();
    System.out.print("Enter Array Elements : ");
    for(i=0; i<size; i++)
    {
        arr[i] = scan.nextInt();
    }
    System.out.print("Enter Element to be Delete : ");
    del = scan.nextInt();
    for(i=0; i<size; i++)
    {
        if(arr[i] == del)
        {
            for(int j=i; j<(size-1); j++)
            {
                arr[j] = arr[j+1];
            }
            count++;
            break;
        }
    }
    if(count==0)
    {
        System.out.print("Element Not Found..!!");
    }
    else
    {
        System.out.print("Element Deleted Successfully..!!");
        System.out.print("\nNow the New Array is :\n");
        for(i=0; i<(size-1); i++)
        {
            System.out.print(arr[i]+ " ");
        }
    }
}
```



```
}  
}  
}  
}
```

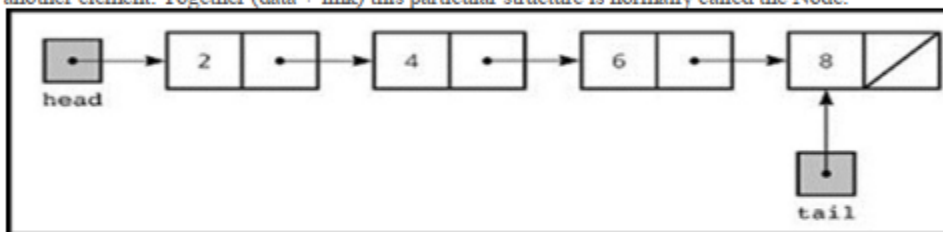
## **VIVA QUESTIONS:**

### **1. What is a Linked list?**

Linked list is an ordered set of data elements, each containing a link to its successor (and typically its predecessor).

### **2. Can you represent a Linked list graphically?**

The fundamental data structure for the linked record involves 3 segments: the data itself and also the link to another element. Together (data + link) this particular structure is normally called the Node.



### **3. How many pointers are required to implement a simple Linked list?**

You can find generally 3 pointers engaged:

- A head pointer, pointing to the start of the record.
- A tail pointer, pointing on the last node of the list. The key property in the last node is that its subsequent pointer points to nothing at all (NULL).
- A pointer in every node, pointing to the next node element.

### **4. How many types of Linked lists are there?**

Singly linked list, [doubly linked list](#), multiply linked list, Circular Linked list.

### **5. How to delete a node from linked list?**

- The following are the steps to delete node from the list at the specified position.  
Set the head to point to the node that head is pointing to.
- Traverse to the desired position or till the list ends; whichever comes first
- You have to point the previous node to the next node.

### **6. How to reverse a singly linked list?**

- First, set a pointer (\*current) to point to the first node i.e. current=head.
- Move ahead until current!=null (till the end)
- set another pointer (\*next) to point to the next node i.e. next=current->next
- store reference of \*next in a temporary variable (\*result) i.e. current->next=result
- swap the result value with current i.e. result=current
- And now swap the current value with next. i.e. current=next
- return result and repeat from step 2
- A linked list can also be reversed using recursion which eliminates the use of a temporary variable.

### **7. Compare Linked lists and Dynamic Arrays**

A dynamic array is a data structure that allocates all elements contiguously in memory, and keeps a count of the

present number of elements. If the area reserved for the dynamic array is exceeded, it's reallocated and

traced, a  
costly operation.

Linked lists have many benefits over dynamic arrays. Insertion or deletion of an element at a specific point of a

list, is a constant-time operation, whereas insertion in a dynamic array at random locations would require moving half the elements on the average, and all the elements in the worst case.

Whereas one can delete an element from an array in constant time by somehow marking its slot as vacant, this

causes fragmentation that impedes the performance of iteration.

### 8. What is a Circular Linked list?

In the last node of a singly linear list, the link field often contains a null reference. A less common convention is

to make the last node to point to the first node of the list; in this case the list is said to be „circular“ or „circularly linked“

### PROGRAM -8 :

#### Abstract Class

**Aim:** Write a java program to create an abstract class named shape that contains two integers and an empty method named printArea() Provide three classes named Rectangle,, Triangle and Circle such that each one of the classes extends the class shape. Each one of the class contains only the method printArea() that print the area of the given shape.

**THEORY:** To create an abstract class that shows the hiding of elements in a class. At the same time

inheritance property is used to extend the class shape into different geometrical figures. This represents the reusability of code for a programmer.

#### Program:

// Abstract class that contains abstract method.

abstract class Shape

{

int h=10,w=5;

abstract void printArea();

}

// Classes that illustrates the abstract method.

class Rectangle extends Shape

{

void printArea()

{

```
float area=0.5*h*w;
System.out.println("The Area of rectangle is"+area);
} }
class Triangle extends Shape
{
void printArea()
{
float area=h*w;
System.out.println("The Area of Triangle is"+area);
} }
class Circle extends Shape
{
void printArea()

{
System.out.println("Circle area is not possibles ");
} }
// Class that create objects and call the method.
class ShapeDemo
{
public static void main(String args[])
{
Rectangle obj1 = new Rectangle (); Triangle obj2 = new
Triangle (); Circle obj3 = new Circle ();
obj1. printArea (); obj2. printArea ();
obj3. printArea ();
} }
```

### **VIVA QUESTIONS:**

#### **1. What is abstraction?**

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

Abstraction lets you focus on what the object does instead of how it does it.

#### **2. What is the difference between abstraction and encapsulation?**

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

#### **3. What is abstract class?**

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

**4. Can there be any abstract method without abstract class?**

No, if there is any abstract method in a class, that class must be abstract.

**5. Can you use abstract and final both with a method?**

No, because abstract method needs to be overridden whereas you can't override final method.

**6. Is it possible to instantiate the abstract class?**

No, abstract class can never be instantiated.

**7. What is interface?**

Interface is a blueprint of a class that have static constants and abstract methods. It can be used to achieve fully abstraction and multiple inheritance.

**8. Can you declare an interface method static?**

No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.

**9. Can an Interface be final?**

No, because its implementation is provided by another class.

**10. What is marker interface?**

An interface that have no data member and method is known as a marker interface. For example Serializable, Cloneable etc.