1(A) LINEAR SEARCH

AIM:

To write a java program using linear search

ALGORITHMS:

1. Start the program
2. Declare the necessary variable for linear search program
3. After that Find() methods are used to return the position of the element stored location.
4. Print the output of the program
5. Stop the program

PROGRAM:

```
class LinearSearch

{

public static void main(String args[])

{

int a[]={1,5,-2,8,7,11,40,32};

System.out.println(“the element is at location “+Find(a,7));

System.out.println(“the element is at location “+Find(a,11));

}

public static  int Find(int a[], int key)

{

int i;

for(i=0;i<a.length;i++)

{

if(a[i] = = key)

return i+1;

}

return -1;

}

}
```

Output:

```
  The element is at location  5

  The element is at location  6
```

RESULT:

Thus the java program for linear search program has been implemented and executed successfully.

1(b) BINARY SEARCH

AIM:

To write a java program using Binary search

ALGORITHMS:

1. Start the program
2. Declare the necessary variable for binary search program
3. Find() methods are used to return the position of the elements on stored elements
4. Print the output of the program
5. Stop the program

PROGRAM:

```
class BinarySearch

{

public static void main(String args[])

{

int[] a= {10,20,30,40,50,60};

int key=40;

Find(a,0,5,key);

}

public static void Find(int[] a, int low, int high, int key)

{

int mid;

if(low>high)

{

System.out.println("Error! The element is not present in the list");

return;

}

mid=(low+high)/2;

if(key= = a[mid])

System.out.println("the element is present at loaction" +(mid+1));

else if(key<a[mid])

Find(a,low,mid-1,key);

else if(key>a[mid])

Find (a,mid+1,high,key);

}

}
```

OUTPUT:

```
The element is present at loaction 4
```

RESULT:

Thus the java program for Binary search program has been implemented and executed successfully.

1(c) SELECTION SORT

AIM:

To write a java program for Selection sort using quadratic sorting algorithms

ALGORITHMS:

1. Start the program
2. Initialize minimum value(**min_idx**) to location 0.
3. Traverse the array to find the minimum element in the array.
4. While traversing if any element smaller than **min_idx** is found then swap both the values.
5. Then, increment **min_idx** to point to the next element.
6. Repeat until the array is sorted.
7. Stop the program

PROGRAM:

```
import java.util.*;
class Selectionsort
{
void sort(int arr[])
{
int n=arr.length;
        for (int i = 0; i < n-1; i++)
        {
            int = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }
    public static void main(String args[])
    {
        SelectionSort ob = new SelectionSort();
        int arr[] = {64,25,12,22,11};
        ob.sort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
```

OUTPUT:

```
Sorted Array: 11, 12,22,25,64
```

RESULT:

Thus the java program for selection sort program using quadratic sorting algorithms has been implemented and executed successfully.

1(D) INSERTION SORT

AIM:

To write a java program for Selection sort using quadratic sorting algorithms.

ALGORITHMS:

1. Start the program to sort an array of size N in ascending order:
2. Iterate from arr[1] to arr[N] over the array.
3. Compare the current element (key) to its predecessor.
4. If the key element is smaller than its predecessor, compare it to the elements before.
5. Move the greater elements one position up to make space for the swapped element.
6. Print the output of insertion sort
7. Stop the program

PROGRAM:

```
class InsertionSort
 {
     void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i)
    {
            int key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key)
          {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

  static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }
   public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6 };
        InsertionSort ob = new InsertionSort();
        ob.sort(arr);
        printArray(arr);
    }
}
```

OUTPUT:

```
5          6          11          12          13
```

RESULT:

Thus the java program for insertion sort program using quadratic sorting algorithms has been implemented and executed successfully.

.

2(A) STACK OPERATIONS

AIM:

To develop a java program for stack data structure using class and object

ALGORITHMS:

1. Start the program
2. Declare the necessary variables of stack program
3. push inserts an item at the top of the stack (i.e., above its current top element).
4. pop removes the object at the top of the stack and returns that object from the function. The stack size will be decremented by one.
5. isEmpty tests if the stack is empty or not.
6. isFull tests if the stack is full or not.
7. peek returns the object at the top of the stack without removing it from the stack or modifying the stack in any way.
8. Print size returns the total number of elements present in the stack.
9. Stop the program

PROGRAM:

```
class Stack
{
   private int arr[];
   private int top;
   private int capacity;
        Stack(int size)
   {
       arr = new int[size];
       capacity = size;
       top = -1;
   }
 public void push(int x)
   {
       if (isFull())
       {
           System.out.println("Overflow\nProgram Terminated\n");
           System.exit(-1);
       }
        System.out.println("Inserting " + x);
       arr[++top] = x;
   }
   public int pop()
   {
               if (isEmpty())
       {
           System.out.println("Underflow\nProgram Terminated");
           System.exit(-1);
       }
        System.out.println("Removing " + peek());
       return arr[top--];
   }
```

```
  public int peek()
    {
        if (!isEmpty())
{
            return arr[top];
        }
        else {
            System.exit(-1);
        }
        return -1;
    }
        public int size()
{
        return top + 1;
    }

        public boolean isEmpty()
{
        return top == -1;               // or return size() == 0;
    }
         public boolean isFull()
 {
        return top == capacity - 1;     // or return size() == capacity;
    }
}

 class Main
{
    public static void main (String[] args)
    {
        Stack stack = new Stack(3);
         stack.push(1);      // inserting 1 in the stack
        stack.push(2);      // inserting 2 in the stack
         stack.pop();        // removing the top element (2)
        stack.pop();        // removing the top element (1)
         stack.push(3);      // inserting 3 in the stack
         System.out.println("The top element is " + stack.peek());
        System.out.println("The stack size is " + stack.size());
         stack.pop();        // removing the top element (3)
        if (stack.isEmpty())
    {
            System.out.println("The stack is empty");
        }
        else
    {
            System.out.println("The stack is not empty");
        }
      }
    }
```

OUTPUT:

```
Inserting 1
Inserting 2
Removing 2
Removing 1
Inserting 3
The top element is 3
The stack size is 1
Removing 3
The stack is empty
```

RESULT:

Thus the java program for stack data structure using class and object has been implemented and executed successfully.

2(B) QUEUE OPERATIONS

AIM:

To develop a java program for Queue data structure using class and object

ALGORITHMS:

Queue operations work as follows:

1. Two pointers FRONT and REAR
2. FRONT track the first element of the queue
3. REAR track the last element of the queue
4. initially, set value of FRONT and REAR to -1

Enqueue Operation:

1. check if the queue is full
2. for the first element, set the value of FRONT to 0
3. increase the REAR index by 1
4. add the new element in the position pointed to by REAR

Dequeue Operation:

1. check if the queue is empty
2. return the value pointed by FRONT
3. increase the FRONT index by 1
4. for the last element, reset the values of FRONT and REAR to -1

PROGRAM:

```
public class Queue
{
  int SIZE = 5;
  int items[] = new int[SIZE];
  int front, rear;
  Queue()
 {
    front = -1;
    rear = -1;
  }
  boolean isFull()
 {
    if (front == 0 && rear == SIZE - 1)
{
      return true;
    }
    return false;
  }
  boolean isEmpty()
 {
    if (front == -1)
      return true;
    else
      return false;
  }
  void enQueue(int element)
{
    if (isFull())
{
      System.out.println("Queue is full");
    }
 Else
 {
```

```
          if (front == -1)
            front = 0;
          rear++;
          items[rear] = element;
          System.out.println("Inserted " + element);
        }
      }

      int deQueue()
     {
        int element;
        if (isEmpty())
     {
          System.out.println("Queue is empty");
          return (-1);
        }
    else
     {
          element = items[front];
          if (front >= rear)
    {
            front = -1;
            rear = -1;
          }
          else
     {
            front++;
          }
          System.out.println("Deleted -> " + element);
          return (element);
        }
      }
      void display()
    {
        int i;
        if (isEmpty())
```

```
 {
      System.out.println("Empty Queue");
    }
else
{
      System.out.println("\nFront index-> " + front);
      System.out.println("Items -> ");
      for (i = front; i <= rear; i++)
        System.out.print(items[i] + "  ");
      System.out.println("\nRear index-> " + rear);
    }
  }
  public static void main(String[] args)
{
    Queue q = new Queue();
        q.deQueue();
    q.enQueue(1);
    q.enQueue(2);
    q.enQueue(3);
    q.enQueue(4);
    q.enQueue(5);
    q.enQueue(6);
    q.display();
    q.deQueue();
    q.display();
  }
}
```

OUTPUT:

```
2       3       4       5       6
```

RESULT:

Thus the java program for queue data structure using class and object has been implemented and executed successfully.

3. PAYSLIP GENERATION USING INHERITANCE

AIM:

To develop a java application to generate pay slip for different category of employee using the concepts of inheritance.

ALGORITHMS:

1. Start the program
2. Create the class employee with name, empid,address, mailid, mobile no as data members
3. Inherit the classes programmer, asstprofessor, associateprofessor and professor from employee class
4. Add basic pay(bp) as data member of all the ibherited classes
5. Calculate DA as 97% bp,HRA, 10% of BP, PF as 12% BP staff club fund as 0.1% of BP.
6. Calculate gross salary and net salary
7. Generate payslip for all categories of employee
8. Stop the program

PROGRAM:

Salary.java

```
import java.util.Scanner;
class Employee
{
int empid;
long mobile;
String name, address, mailid;
Scanner get = new Scanner(System.in);
void getdata()
{
System.out.println("Enter Name of the Employee");
name = get.nextLine();
System.out.println("Enter Mail id");
mailid = get.nextLine();
System.out.println("Enter Address of the Employee:");
address = get.nextLine();
System.out.println("Enter employee id ");
empid = get.nextInt();
System.out.println("Enter Mobile Number");
mobile = get.nextLong();
}
void display()
{
System.out.println("Employee Name: "+name);
 System.out.println("Employee id : "+empid);
System.out.println("Mail id : "+mailid);
System.out.println("Address: "+address);
System.out.println("Mobile Number: "+mobile);
}
}
class Programmer extends Employee
{
double salary,bp,da,hra,pf,club,net,gross;
void getprogrammer()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateprog()
{
da=(0.97*bp);
 hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************************");
 System.out.println("PAY SLIP FOR PROGRAMMER");
System.out.println("************************************************");
System.out.println("Basic Pay: Rs. "+bp);
System.out.println("DA: Rs. "+da);
System.out.println("HRA: Rs. "+hra);
System.out.println("PF: Rs. "+pf);
System.out.println("CLUB: Rs. "+club);
System.out.println("GROSS PAY: Rs. "+gross);
System.out.println("NET PAY: Rs. "+net);
}
```

```
}
class Asstprofessor extends Employee
{
double salary,bp,da,hra,pf,club,net,gross;
void getasst()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateasst()
{
da=(0.97*bp);
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************");
System.out.println("PAY SLIP FOR ASSISTANT PROFESSOR");
System.out.println("***********************************");
System.out.println("Basic Pay: Rs. "+bp);
System.out.println("DA: Rs. "+da);
System.out.println("HRA: Rs. "+hra);
System.out.println("PF: Rs. "+pf);
 System.out.println("CLUB: Rs. "+club);
System.out.println("GROSS PAY: Rs. "+gross);
System.out.println("NET PAY: Rs. "+net);
}
}
```

```
class Associateprofessor extends Employee
{
double salary,bp,da,hra,pf,club,net,gross;
void getassociate()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateassociate()
{
da=(0.97*bp);
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
 gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************");
System.out.println("PAY SLIP FOR ASSOCIATE PROFESSOR");
System.out.println("************************************");
 System.out.println("Basic Pay: Rs. "+bp);
System.out.println("DA: Rs. "+da);
System.out.println("HRA: Rs. "+hra);
System.out.println("PF: Rs. "+pf);
 System.out.println("CLUB: Rs. "+club);
System.out.println("GROSS PAY: Rs. "+gross);
System.out.println("NET PAY: Rs. "+net);
}
}
```

```
class Professor extends Employee

{

double salary,bp,da,hra,pf,club,net,gross;

void getprofessor()

{

System.out.println("Enter basic pay");

bp = get.nextDouble();

}

void calculateprofessor()

{

da=(0.97*bp);

hra=(0.10*bp);

pf=(0.12*bp);

club=(0.1*bp);

gross=(bp+da+hra);

net=(gross-pf-club);

System.out.println("************************");

System.out.println("PAY SLIP FOR PROFESSOR");
System.out.println("************************");

System.out.println("Basic Pay: Rs. "+bp);

System.out.println("DA: Rs. "+da);

System.out.println("HRA: Rs. "+hra);

System.out.println("PF: Rs. "+pf);

 System.out.println("CLUB: Rs. "+club);

System.out.println("GROSS PAY: Rs. "+gross);

System.out.println("NET PAY: Rs. "+net);

}

}
```

```
class Salary

{

public static void main(String args[])

{

int choice,cont;

do

{

System.out.println("PAYROLL");

System.out.println(" 1.PROGRAMMER \t 2.ASSISTANT PROFESSOR \t 3.ASSOCIATE
PROFESSOR \t 4.PROFESSOR ");

Scanner c = new Scanner(System.in);

System.out.print("Enter Your Choice:");

choice=c.nextInt();

switch(choice)

{

case 1:

{

Programmer p=new programmer();

p.getdata();

p.getprogrammer();

p.display();

p.calculateprog();

break;

}

case 2:

{

Asstprofessor asst = new Asstprofessor();

asst.getdata();

asst.getasst();

asst.display();
asst.claculateasst();

break;

}

case 3:

{

Associateprofessor asso = new Associateprofessor();

asso.getdata();

asso.getassociate();

asso.display();

asso.claculateassociate();

break;

}

case 4:

{

professor prof = new professor();

prof.getdata();

prof.getassociate();

prof.display();

prof.claculateassociate();

break;

}

}

System.out.println(“please enter 0 to quit and 1 to continue”);

cont=s.nextInt();

}

while(con= = 1);

}

}
```

OUTPUT:

RESULT:

Thus the java application to generate pay slip for different category of employee has been implemented using inheritance and the program was executed successfully.

4.ABSTRACT CLASS IMPLEMENTATION

AIM:

To write a java program to create an abstract class

ALGORITHMS:

1. Start the program
2. Declare the necessary variable for class shapes
3. Declare the abstarct printarea() method in abstarct shape class
4. Classes are created as rectangle, triangle and circle classes that classes inherits from shape using extends keyword
5. Define the definition part of printarea in rectangel class, triangle class and circle class
6. Calculate of rectangle area, triangle area and circle area using the printarea() method
7. Print the outputs
8. Stop the program

PROGRAM:

```
 import java.util.*;
abstract class shape
{
            int a,b;
            abstract public void printarea();
}
class rectangle extends shape
{
           public int area_rect;
           public void printarea()
           {
                        Scanner s=new Scanner(System.in);
                        System.out.println("Enter the length and breadth of rectangle");
                        a=s.nextInt();
                        b=s.nextInt();
                        area_rect=a*b;
                        System.out.println("Length of rectangle: "+a +"breadth of rectangle: "+b);
                        System.out.println("The area of rectangle is:"+area_rect);
           }
 }
class triangle extends shape
{
           double  area_tri; public
           void printarea()
           {
                      Scanner s=new Scanner(System.in);
                      System.out.println("Enter the base and height of triangle:");
                      a=s.nextInt();
                      b=s.nextInt();
                      System.out.println("Base of triangle: "+a +"height of triangle: "+b);
                      area_tri=(0.5*a*b);
                      System.out.println("The area of triangle is:"+area_tri);
           }
}
class circle extends shape
{
        double area_circle;
        public void printarea()
        {
                Scanner s=new Scanner(System.in);
                System.out.println("Enter the radius of circle:");
                a=s.nextInt();
                area_circle=(3.14*a*a);
                System.out.println("Radius of circle:"+a);
                System.out.println("The area of circle is:"+area_circle);
        }
}

public class Shapeclass

{
         public static void main(String[] args)
        {
                rectangle r=new rectangle();
                r.printarea();
                triangle t=new triangle();
                t.printarea();
                circle r1=new circle();
                r1.printarea();
        }
}
```

OUTPUT:

```
Enter the length and breadth of rectangle
10
20
Length of rectangle:10
breadth of rectangle:  20
The area of rectangle is: 200

Enter the base and height of triangle:
10
20
Base of triangle: 10
height of triangle:  20
The area of triangle is: 100

Enter the radius of circle: 15
Radius of circle: 15
The area of circle is: 706.5
```

RESULT:

Thus the java program for abstract class has been implemented and executed successfully.

5.INTERFACE CONCEPTS

AIM:

To write a java program using interface concepts

ALGORITHMS:

1. Start the program
2. Defining the interface
3. Declare the necessary variable for class shapes
4. Declare the abstarct printarea() method in abstarct shape class
5. Classes are created as rectangle, triangle and circle classes that classes inherits from shape using implements interface.
6. Define the definition part of printarea in rectangel class, triangle class and circle class
7. Calculate of rectangle area, triangle area and circle area using the printarea() method
8. Print the outputs
9. Stop the program

PROGRAM:

```
import java.util.*;
interface myinterface
{
public void printarea();
}
abstract class shapes
{
double a,b;
abstract void printarea();
}
class rectangle extends shapes implements myinterface
{
public  void printarea()
{
System.out.println("calculating area of rectangle");
Scanner input=new Scanner(System.in);
System.out.println("enter length:");
a=input.nextDouble();
System.out.println("enter breadth:");
b=input.nextDouble();
double area=a*b;
System.out.println(" area od rectangle:"+area);
}
}
class triangle extends shapes implements myinterface
{
System.out.println("calculating area of triangle");
Scanner input=new Scanner(System.in);
System.out.println("enter height:");
a=input.nextDouble();
System.out.println("enter breadth:");
b=input.nextDouble();
double area=0.5*a*b;
System.out.println(" area od triangle:"+area);
}
}
class circle extends shapes implements myinterface
{
System.out.println("calculating area of circle");
Scanner input=new Scanner(System.in);
System.out.println("enter radius:");
a=input.nextDouble();
double area=3.14*a*a;
System.out.println(" area od circle:"+area);
}
}
class abstractclassdemo
{
public static void main(String args[])
{
shapes obj;
obj=new rectangle();
obj.printarea();
obj=new triangle();
obj.printarea();
obj=newcircle();
obj.printarea();
}
}
```

OUTPUT:

```
Calculating area of rectangle
Enter length: 10
Enter breadth: 20
Area of rectangle : 200.0

Calculating area of triangle
Enter height: 10
Enter breadth: 5
Area of triangle : 25.0

Calculating area of circle
Enter radius: 10
Area of rectangle : 314.0
```

RESULT:

Thus the java program for interface concept has been implemented and executed successfully.

6(a) EXCEPTION HANDLING

AIM:

To develop a java program using exception handling

ALGORITHMS:

1. Start the program
2. Declare the necessary variables for exception handling program
3. Define the try – catch block for handling the exception
4. Try block is used to try block of source code that is to be monitored for the exception.
5. Catch block is used to handles the specific type of exception along with the try block.
6. Print the exception information using printStackTrace() method
7. Stop the program

PROGRAM:

```
import  java.io.*;
class  GFG
 {
    public  static  void  main (String[] args)
{
      int  a=5;
      int  b=0;
        try
      {
          System.out.println(a/b);
        }
      catch(ArithmeticException e)
     {
        e.printStackTrace();
      }
    }
}
```

OUTPUT:

```
java.lang.ArithmeticException: Division by zero at GFG.main(File.java:10)
```

RESULT:

Thus the java program for exception handling has been implemented and executed successfully.

6(b) USER DEFINED EXCEPTION

AIM:

To develop a java program using User defined Exception

ALGORITHMS:

1. Start the program
2. Declare the necessary variables for user defined exception program
3. Initialize the values using Constructor method.
4. Define the try – catch block for handling the exception
5. Try block is used to try block of source code that is to be monitored for the exception.
6. Catch block is used to handles the specific type of exception along with the try block.
7. Print the output
8. Stop the program

PROGRAM :

```
class MyException extends Exception

{

            String str1;
            MyException(String str2)

             {

                        str1=str2;

              }

             public String toString()

            {

                        return ("MyException Occurred: "+str1) ;

             }

}

class example

{

             public static void main(String args[])

            {

                        try

                        {           System.out.println("Starting of try block");

                                    throw new MyException("This is My error Message");

                        }

                        catch(MyException exp)

                        {

                                    System.out.println("Catch Block") ;
                                    System.out.println(exp) ;

                        }

            }

}
```

OUTPUT:

```
Starting of try block
Catch Block
Myexception Occurred: this is my error Message
```

RESULT:

Thus the java program for user defined exception handling has been implemented and executed successfully.

7. MULTITHREADING IMPLEMENTATION

AIM:

To develop a java program using Multithreading implementation

ALGORITHMS:

1. Start the program
2. Create a class even which implements first thread that compute the square of the number
3. Run() method implements the code to be excuted when thread gets executed.
4. Create a class odd which implements seconf thread that computes the cube of the number.
5. Create the third thread that generates random number. If the random number is even it display the square of the number. If the random number is odd, it display the cube of given number.
6. The multithreading is performed and the task switched between multiple thread.
7. The sleep() method makes the thread to suspend for the specified time.
8. Stop the program

PROGRAM:

```
import java.util.*;
class even implements Runnable

{
        public int x;
        public even(int x)
        {
                this.x = x;
        }
        public void run()
        {
              System.out.println("New Thread "+ x +" is EVEN and Square of " + x + " is: " + x * x);
        }
}
class odd implements Runnable

{
        public int x;
        public odd(int x)
        {
                this.x = x;
        }
        public void run()
        {
              System.out.println("New Thread "+ x +" is ODD and Cube of " + x + " is: " + x * x * x);
        }
}
class A extends Thread

{
        public void run()
        {
                int num = 0;
                Random r = new Random();
                try
                {
                        for (int i = 0; i < 5; i++)
                        {
                                num = r.nextInt(100);
                                System.out.println("Main Thread and Generated Number is " + num);
                                if (num % 2 == 0)
                                {
                                        Thread t1 = new Thread(new even(num));
                                        t1.start();
                                }
                                else
                                {
                                        Thread t2 = new Thread(new odd(num));

                                         t2.start();
                                 }
```

```
                    Thread.sleep(1000);
                    System.out.println("------------------------------- ");
                }
            }
            catch (Exception ex)
            {
                System.out.println(ex.getMessage());
            }
        }
    }

    public class multithreadprog

    {
        public static void main(String[] args)
        {
            A a = new A();
            a.start();
        }
    }
```

OUTPUT :

```
Main thread and Generated Number is 37
New Thread 37 is ODD and Cube of 37 is : 50653
Main thread and Generated Number is  4
New Thread 4 is EVEN and Square of 4 is : 16
Main thread and Generated Number is 69
New Thread 69 is ODD and Cube of 69 is : 328509
Main thread and Generated Number is 32
New Thread 32 is EVEN and Square of 32 is : 1024
Main thread and Generated Number is 26
New Thread 26 is EVEN and Square of 26 is : 676
```

RESULT:

Thus the java program for multithreaded has been implemented and executed successfully.

8(A) FILE OPERATION

AIM:

To develop a java program that copies the content of one file to another file by removing unnecessary spaces between words.

ALGORITHMS:

1. Start the program
2. Declare the necessary classes for file operation.
3. Read() method is used to read the byte of data
4. Write () method is used to write the byte of data
5. Close () method is used to close the stream.
6. Get the source file name and destination file name using Scanner and File Classes
7. CopyContent() method called to copy the contents from one file to another file
8. Stop the program

PROGRAM:

```
import java.io.*;
import java.util.*;
  public class CopyFromFileaToFileb
{
    public static void copyContent(File a, File b) throws Exception
    {
        FileInputStream in = new FileInputStream(a);
        FileOutputStream out = new FileOutputStream(b);
          try
      {
              int n;
                   while ((n = in.read()) != -1)
           {
               out.write(n);
            }
        }
        finally
      {
            if (in != null)
         {
                 in.close();
            }
            if (out != null)
           {
               out.close();
            }
        }
        System.out.println("File Copied");
    }
```

```
      public static void main(String[] args) throws Exception
    {
       Scanner sc = new Scanner(System.in);
            System.out.println( "Enter the source filename from where you have to read/copy :");
       String a = sc.nextLine();
         File x = new File(a);
            System.out.println( "Enter the destination filename where you have to write/paste :");
       String b = sc.nextLine();
            File y = new File(b);
         copyContent(x, y);
    }
}
```

OUTPUT:

```
Enter the source filename from where you have to read/copy :
sourcefile.txt
Enter the destination filename where you have to write/paste :
destinationfile.txt
File Copied
```

RESULT:

Thus the java program copies the content of one file to another file has been implemented and executed successfully.

8(B) COUNT THE WORDS IN A FILE

AIM:

To develop a java program for count the words in a file using File Operations.

ALGORITHMS:

1. Start the program
2. Define String line
3. Set count =0
4. Use File Reader to open file in read mode
5. READ line from file
6. REPEAT STEP 7 to STEP 8 UNTIL reach the end of file
7. SPLIT lines into words and STORE in array string words[].
8. count = count + words.length
9. print the Count of Words in a file.
10. Stop the progam

PROGRAM:

```
import java.io.BufferedReader;
import java.io.FileReader;
  public class CountWordFile
{
    public static void main(String[] args) throws Exception
 {
        String line;
        int count = 0;
        FileReader file = new FileReader("data.txt ");
        BufferedReader br = new BufferedReader(file);
        while((line = br.readLine()) != null)
{
            String words[] = line.split("");
             count = count + words.length;
          }
          System.out.println("Number of words present in given file: " + count);
        br.close();
    }
}
```

OUTPUT:

```
Number of words present in given file: 60
```

RESULT:

Thus the java program to count the words in a file has been implemented and executed successfully.

9 GENERIC CLASSES

AIM:

To develop a java program using generic class.

ALGORITHMS:

1. Start the program
2. Define generic class
3. Constructor method will be ivoked from main
4. Define generic method for PUSH and POP operations inside the generic class
5. Declaring integer and character values to be pushed onto the stack.
6. Ist is an instance for integer stack and cst is an instance for character stack
7. The constructor stack(size) will be invoked
8. Pushing onto integer stack using push()
9. Pushing onto character stack using push()
10. Popping from the character using pop()
11. Popping from the integer using pop()
12. print the outputs
13. Stop the program

PROGRAM 1:

```
mport java.util.*;
public class stack<T>
{
public arraylist<T> obj;
public stack(int size)
{
obj=new arraylist<T>(size);
}
public void push(T item)
{
obj.add(item);
}
public T pop()
{
if(obj.isEmpty())
{
System.out.println(“\n Stack is empty”);
return null;
}
return obj.remove(obj.size()-1);
}
}
```

PROGRAM 2:

```
 import java.io.*;
 import java.util.*
public class stackgeneric
 {
 public static void main(String args[])
 {
 int[] ia={1,2,3,4,5};
 char[] ca={‘A’,’B’,’C’,’D’,’E’};
 stack<Integer>  ist=new stack<Integer>(5);
 stack<Character>  ist=new stack<Character>(5);
 System.out.println(“\n pushing the elements in integer stack”);
 for(int i=0;i<5;i++)
 ist.push(ia[i]);
 System.out.println(“\n pushing the elements in character stack”);
for(int i=0;i<5;i++)
ist.push(ca[i]);
 System.out.println(“\n popping the elements from Character stack”);
for(int i=0;i<2;i++)
 System.out.println(“\n %c” ,cst.pop());
 System.out.println(“\n popping the elements from integer stack”);
 for(int i=0;i<5;i++)
 System.out.println(“\n %d”, ist.pop());
 System.out.println(“\n  performing one more pop integer stack”);
 System.out.println(“\n %d”, ist.pop());
 }
 }
```

OUTPUT:

```
Pushing the elements in integer stack

Pushing the elements in character stack

Popping two elements from character stack

E

D

Popping all the elements from integer stack

5

4

3

2

1

Performing one more pop for integer stack

Stack is empty

Null
```

RESULT:

Thus the java program to generic class has been implemented and executed successfully.

10.(a) JavaFX Control

AIM:

To write a javaFX application for creating a login form

ALGORITHMS:

1. Start the program
2. Declare the necessary packages for the progam
3. Label control display simple text on the screen
4. Button control is used to control specific function of the applications
5. GridPane places its nodes into grid of row and columns. Nodes may span multiple rows or columns. It is most flexible built-in layout pane.
6. Textfield is used for getting the input from user, the textfield is provided
7. addRow() is used to add the contents in row
8. setTitle() is used to set the title name on the screen
9. print the output
10. Stop the program

PROGRAM:

```
package myjavafxapplication;
import java.io.FileInputString;
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.event.ActionEvent;
importjavafx.event.EventHandler;
import javafx.scene.Scene;
import javxfx.scene.control.Button;
import javxfx.scene.control.Label;
import javxfx.scene.control..PasswordField;
import javxfx.scene.control..TextField;
import javxfx.scene.control.gridPane;
import javafx.stage.Stage;
public class myjavafxapplication extends Application
{
@override
public void start(Stage primaryStage) throws Exception
{
Label L1=new Label("Enter your name");
TextField tf1=new TextField();
Label L1=new Label("Enter your password");
TextField tf2=new TextField();
Button btn=new Button("Submit");
GridPane root=new GridPane();
root.setVgap(10);
root.addRow(0,L1,ft1);

root.addRow(0,L2,ft2);

root.addRow(2, btn);

Scene s=new Scene(root,300,200);

primaryStage.setTitle("text Field demo");

primaryStage.setScene(s);

primaryStage.show();

}
public staic void main(String[] args)

{

launch(args);

}

}
```

OUTPUT:

RESULT:

Thus the javaFX application for creating a login form has been implemented and executed successfully.

10.(b) LAYOUTS

AIM:

To write a javaFX application for creating a HBox Layout

ALGORITHMS:

1. Start the program
2. Declare the necessary packages for the progam
3. HBox layout arranges the children in the form of horizontal rows.
4. Button control is used to control specific function of the applications
5. getChildren() is used to returns the nodes in HBox.
6. addAll() is used to add the button on the screen
7. setTitle() is used to set the title name on the screen
8. print the output
9. Stop the program

PROGRAM:

```
package myjavafxapplication;

import javafx.application.Application;

import javafx.scene.Scene;

import javxfx.scene.control.Button;

import javxfx.scene.layout.Hbox;

import javafx.stage.Stage;

public class myjavafxapplication extends Application

{

@override

public void start(Stage primaryStage)

{

Hbox root=new Hbox();

Button B1=new Button(“One”);

Button B2=new Button(“Two”);

Button B3=new Button(“Three”);

Button B4=new Button(“Four”);

Button B5=new Button(“Five”);

Scene s=new Scene(root,200,100);

root.getChildren().addAll(B1,B2,B3,B4,B5);

primaryStage.setScene(s);

primaryStage.setTitle(“Hbox demo”);

primaryStage.show();

}
public staic void main(String[] args)

{

launch(args);

}

}
```

OUTPUT:

RESULT:

Thus the javaFX application for creating a HBox layout has been implemented and executed successfully.

10.(b) MENUS

AIM:

To develop a javaFX application for creating menus.

ALGORITHMS:

1. Start the program
2. Create a Menu bartand it can be instantiated using new keyword
   MenuBar m=new MenuBar();
3. Now create the menu. The name of the menu is passed as an arguments to the Menu Class
   Menu f=new Menu("File");
4. Create the menuitem. The name of the menu is passed as an arguments to Menuitem class
   MenuItem mi=new MenuItem("new");
5. Add menu items to the menu using add or addAll method
6. Add menu to menubar using add or addAll method
7. print the output
8. Stop the program

PROGRAM:

```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javxfx.scene.control.Menu;
import javxfx.scene.control.MenuBar;
import javxfx.scene.control.MenuItem;
import javxfx.scene.layout.BorderPane;
import javafx.stage.Stage;
public class main extends Application
{
@override
public void start(Stage stage)
{
MenuBar mb=new MenuBar();
Menu filename=new Menu("File");
Menu editmenu=new Menu("Edit");
Menu aboutmenu=new Menu("About");
MenuItem newitem=new MenuItem("New");
MenuItem openfileitem=new MenuItem("Open File");
MenuItem saveitem=new MenuItem("Save");
MenuItem exititem=new MenuItem("Exit");
MenuItem cutitem=new MenuItem("Cut");
MenuItem copyitem=new MenuItem("Copy");
MenuItem pasteitem=new MenuItem("Paste");
fileMenu.getItems().addAll(newitem,openfileitem,saveitem,exititem);
editMenu.getItems().addAll(cutitem,copyitem,pasteitem);
menuBar.getMenus().addAll(fileMenu,editMenu,aboutMenu);

BorderPane r=new BorderPane();

r.setTop(menuBar);

Scene s=new Scene(r,400,300);

stage.setTitle(“JavaFX Menu Demo”);

stage.setScene(s);

stage.show();

}
public staic void main(String[] args)

{

Application.launch(args);

}

}
```

OUTPUT:

RESULT:

Thus the javaFX application for creating a Menu has been implemented and executed successfully.