Week 7 **Program No. a**

Title of the Program:

Create a generic class called Box that can hold any type of object. Implement the following methods: 1) void set(T obj): sets the object stored in the box 2) T get(): retrieves the object stored in the box 3) boolean isEmpty(): returns true if the box is empty, false otherwise

Objective of the Program: To understand the basics of Generics

Program:

```
public class Box<T> {

   private T contents;

   public void set(T obj) {
      this.contents = obj;
   }

   public T get() {
      return contents;
   }

   public boolean isEmpty() {
      return contents == null;
   }
}
public class BoxDemo {
```

```
    public static void main(String[] args) {
        // Create a Box object to hold an Integer
        Box<Integer> integerBox = new Box<>();
        integerBox.set(10);

        // Print the contents of the integerBox
        System.out.println("Integer box contents: " + integerBox.get());

        // Create a Box object to hold a String
        Box<String> stringBox = new Box<>();
        stringBox.set("Hello, World!");

        // Print the contents of the stringBox
        System.out.println("String box contents: " + stringBox.get());

        // Check if the integerBox is empty
        System.out.println("Is integerBox empty? " + integerBox.isEmpty());

        // Check if the stringBox is empty
        System.out.println("Is stringBox empty? " + stringBox.isEmpty());
    }
}
```

OUTPUT:

Sure, here is the output of the program:

```
Integer box contents: 10
```

String box contents: Hello, World!

Is integerBox empty? false

Is stringBox empty? false

Week 7 **Program No. b**

Title of the Program:

Implement a generic Stack class that can hold any type of object. Implement the following methods: 1) void push(T obj): pushes an object onto the top of the stack ,2) T pop(): removes and returns the object at the top of the stack 3) boolean isEmpty(): returns true if the stack is empty, false otherwise

Objective of the Program: To understand the basics of Generics

Program:

```
public class Stack<T> {

   private ArrayList<T> stackArray;

   public Stack() {
      this.stackArray = new ArrayList<>();
   }

   public void push(T obj) {
      stackArray.add(obj);
   }

   public T pop() {
      if (isEmpty()) {
         return null;
      }
```

```
        return stackArray.remove(stackArray.size() - 1);
    }

    public boolean isEmpty() {
        return stackArray.isEmpty();
    }
}

public class StackDemo {

    public static void main(String[] args) {
        // Create a Stack object to hold Strings
        Stack<String> stringStack = new Stack<>();

        // Push elements onto the stack
        stringStack.push("Hello");
        stringStack.push("World");
        stringStack.push("!");

        // Pop elements from the stack
        String poppedElement;
        while (!stringStack.isEmpty()) {
            poppedElement = stringStack.pop();
            System.out.println("Popped element: " + poppedElement);
        }
```

```
        // Check if the stack is empty
        System.out.println("Is the stack empty? " + stringStack.isEmpty());
    }
}
```

OUTPUT:

```
Popped element: !
Popped element: World
Popped element: Hello
Is the stack empty? true
```

Week 8 **Program No. a**

Title of the Program:
Write a Java program to implement Autoboxing and Unboxing?

Objective of the Program: To understand the basics of Type Wrappers

Program:

```
public class AutoboxingUnboxingExample {
    public static void main(String[] args) {
        // Autoboxing
        int a = 10;
        Integer b = a; // Automatically converts primitive int to Integer object
        System.out.println("Autoboxing: " + b);
        // Unboxing
        Integer c = 20;
        int d = c; // Automatically converts Integer object to primitive int
        System.out.println("Unboxing: " + d);
    }
}
```

Output:

Auto boxing: 10

Unboxing: 20

Week 8 **Program No. b**

Title of the Program:

Write a Java program to implement Built-In Java Annotations?

Objective of the Program: To understand the basics of Java Annotations

Program:

- **@Override annotation**

```
class Animal{
void eatSomething()
{
System.out.println("eating something");}
}

class Dog extends Animal{
@Override
void eatsomething(){System.out.println("eating foods");
}//should be eatSomething
}
class TestAnnotation1{
public static void main(String args[]){
Animal a=new Dog();
a.eatSomething();
```

}}

Output: Comple Time Error

- **@SuppressWarnings annotation**

```
import java.util.*;
class TestAnnotation2{
@SuppressWarnings("unchecked")
public static void main(String args[])
{
ArrayList list=new ArrayList();
list.add("sonoo");
list.add("vimal");
list.add("ratan");
 for(Object obj:list)
System.out.println(obj);


}}
```

Output: Now no warning at compile time.

Week 9 **Program No. a**

Title of the Program: Write a Java program that creates three threads. First thread displays —Good Morning every one second, the second thread displays —Hello every two seconds and the third thread displays —Welcome every three seconds

Objective of the Program: To understand the basics of Thread programming fundamentals

Program:

```
import java.io.*;
class One extends Thread
{
public void run()
{
for(int i=0;i<100;i++)
{
try{
Thread.sleep(1000); }
catch(InterruptedException e){
System.out.println(e); }
System.out.println("Good Morning");
} } }

class Two extends Thread
{
public void run()
{
for(int i=0;i<100;i++)
{
try{
Thread.sleep(2000); }
catch(InterruptedException e)
{
System.out.println(e);
}
System.out.println("Hello ");
```

```
} } }

class Three implements Runnable
{
public void run()
{
for(int i=0;i<100;i++)
{
try{
Thread.sleep(3000); }
catch(InterruptedException e){
System.out.println(e); }
System.out.println("Wel come");
} } }
class ThreadEx
{
public static void main(String[] args)
{
One t1=new One();
Two t2=new Two();
Three tt=new Three();
Thread t3=new Thread(tt);
t1.setName("One");
t2.setName("Two");
t3.setName("Three");
System.out.println(t1);
System.out.println(t2);
System.out.println(t3);
Thread t=Thread.currentThread();
System.out.println(t);
t1.start();t2.start();t3.start();
} }
```

Output:

```
D:\Lab>javac ThreadEx.java
D:\Lab>java ThreadEx
Thread[One,5,main]
Thread[Two,5,main]
Thread[Three,5,main]
```

```
Thread[main,5,main]
Good Morning
Good Morning
Hello
Wel come
Good Morning
Hello
Good Morning
Good Morning
Hello
Wel come
Good Morning
Good Morning
Hello
Good Morning
```

Week 9 **Program No. b**

Title of the Program:
Write a Java program that correctly implements producer consumer problem using the concept of inter thread communication .

Objective of the Program: To understand the basics of inter thread communication

Program:

```
class Producer implements Runnable
{
	Q q;
	Producer(Q q)
	{
		this.q =q;
		new Thread(this," producer").start();
	}
	public void run()
	{
		int i= 0;
		while(true)
		{
			q.put(i++);
			if(i== 5)
				System.exit(0);
		}
```

```
        }
}
class Consumer implements Runnable
{
        Q q;
        Consumer(Q q)
        {
                this.q= q;
                new Thread(this, "consumer").start();
        }
        public void run()
        {
                while(true)
                        q.get();
        }
}
class Program
{
        public static void main(String ar[])
        {
                Q q= new Q();
                new Producer(q);
                new Consumer(q);
        }
}
class Q
```

```
{
        int n;
        boolean valueset= true;
        synchronized int get()
        {
                while(!valueset)
                {
                        try
                        {
                                wait();
                        }
                        catch(Exception e)
                        {
                        }
                }
                System.out.println("GET " +n);
                valueset= false;
                notify();
                return n;
        }
        synchronized void put(int n)
        {
                while(valueset)
                {
                        try
                        {
```

```
                    wait();
                }
                catch(Exception e)
                {
                }
            }
            this.n= n;
            valueset =true;
            System.out.println("PUT " +n);
            notify();
        }
}
```

Output:

```
PUT 0
GET 0
PUT 1
GET 1
PUT 2
GET 2
PUT 3
GET 3
PUT 4
GET 4
```

Title of the Program:

Create a Email registration Form using Java AWT. The UI should have fields such as name, address, sex, age, email, contact number, etc.,

Objective of the Program: To understand the Form elements of java AWT.

Program:

```
package awtDemo;
 import java.awt.*;
import java.awt.event.*;
 //Simple Registration Form in Java AWT
class MyApp extends Frame {
        Label
lblTitle,lblName,lblFather,lblAge,lblGender,lblCourse,lblHobbies,lblAddress;
        TextField txtName,txtFather,txtAge;
        TextArea txtAddress;
        Checkbox checkMale, checkFemale,Hobbies1,Hobbies2,Hobbies3,Hobbies4;
        CheckboxGroup cbg;
        Choice Course;
        Button btnSave,btnClear;

        public MyApp() {
                super("User Registration Form");
                setSize(1000, 600);// w,h
```

```
setLayout(null);
setVisible(true);
Color formColor = new Color(53, 59, 72);
setBackground(formColor);

Font titleFont = new Font("arial", Font.BOLD, 25);
Font labelFont =new  Font("arial", Font.PLAIN, 18);
Font textFont =new  Font("arial", Font.PLAIN, 15);

lblTitle=new Label("Registration Form");
lblTitle.setBounds(250,40,300,50);
lblTitle.setFont(titleFont);
lblTitle.setForeground(Color.YELLOW);
add(lblTitle);

lblName=new Label("Name");
lblName.setBounds(250,100,150,30);
lblName.setFont(labelFont);
lblName.setForeground(Color.WHITE);
add(lblName);

txtName=new TextField();
txtName.setBounds(400,100,400,30);
txtName.setFont(textFont);
add(txtName);
```

```
lblFather=new Label("Father Name");
lblFather.setBounds(250,150,150,30);
lblFather.setFont(labelFont);
lblFather.setForeground(Color.WHITE);
add(lblFather);

txtFather=new TextField();
txtFather.setBounds(400,150,400,30);
txtFather.setFont(textFont);
add(txtFather);

lblAge=new Label("Age");
lblAge.setBounds(250,200,150,30);
lblAge.setFont(labelFont);
lblAge.setForeground(Color.WHITE);
add(lblAge);

txtAge=new TextField();
txtAge.setBounds(400,200,400,30);
txtAge.setFont(textFont);
add(txtAge);

lblGender=new Label("Gender");
lblGender.setBounds(250,250,150,30);
lblGender.setFont(labelFont);
lblGender.setForeground(Color.WHITE);
```

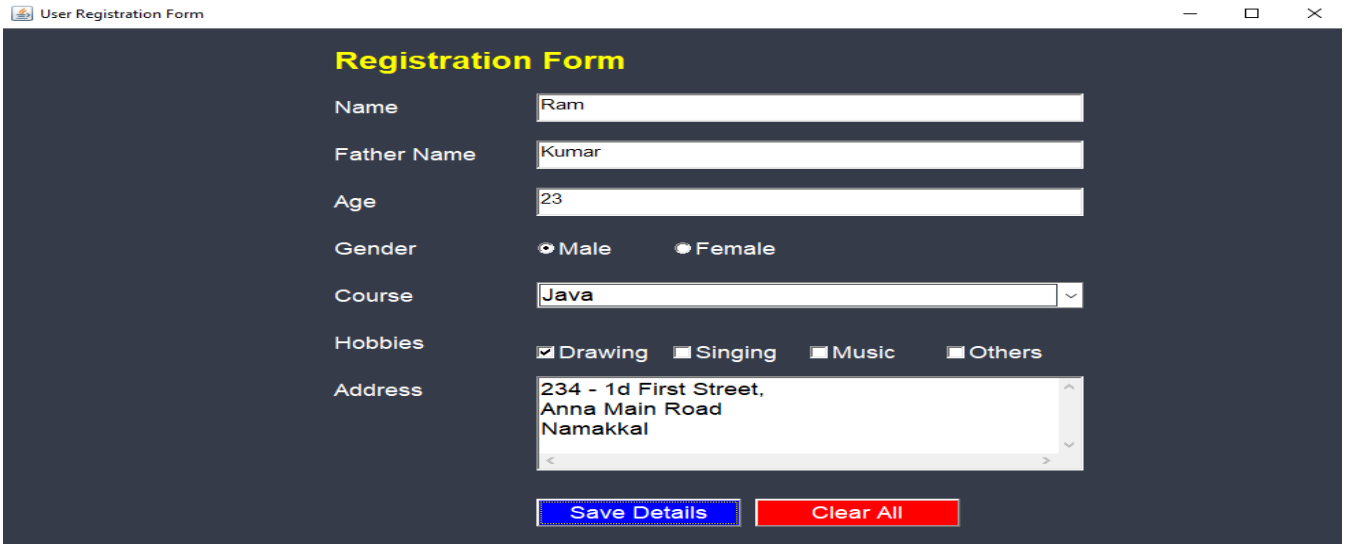
```
add(lblGender);

cbg = new CheckboxGroup();

checkMale = new Checkbox("Male",cbg,true);
checkMale.setBounds(400,250, 100, 30);
checkMale.setFont(labelFont);
checkMale.setForeground(Color.WHITE);
add(checkMale);
checkFemale = new Checkbox("Female",cbg,false);
checkFemale.setBounds(500,250, 100, 30);
checkFemale.setFont(labelFont);
checkFemale.setForeground(Color.WHITE);
add(checkFemale);
lblCourse=new Label("Course");
lblCourse.setBounds(250,300,150,30);
lblCourse.setFont(labelFont);
lblCourse.setForeground(Color.WHITE);
add(lblCourse);

Course= new Choice();
Course.setFont(labelFont);
Course.setBounds(400, 300, 400, 50);
Course.add("C");
Course.add("C++");
Course.add("Java");
```

```
Course.add("C#");
Course.add("Python");
add(Course);

lblHobbies=new Label("Hobbies");
lblHobbies.setBounds(250,350,150,30);
lblHobbies.setFont(labelFont);
lblHobbies.setForeground(Color.WHITE);
add(lblHobbies);
Hobbies1=new Checkbox("Drawing");
Hobbies1.setBounds(400, 350, 100, 50);
Hobbies1.setFont(labelFont);
Hobbies1.setForeground(Color.WHITE);
add(Hobbies1);

Hobbies2=new Checkbox("Singing");
Hobbies2.setBounds(500, 350, 100, 50);
Hobbies2.setFont(labelFont);
Hobbies2.setForeground(Color.WHITE);
add(Hobbies2);
Hobbies3=new Checkbox("Music");
Hobbies3.setBounds(600, 350, 100, 50);
Hobbies3.setFont(labelFont);
Hobbies3.setForeground(Color.WHITE);
add(Hobbies3);
```

```
Hobbies4=new Checkbox("Others");
Hobbies4.setBounds(700, 350, 100, 50);
Hobbies4.setFont(labelFont);
Hobbies4.setForeground(Color.WHITE);
add(Hobbies4);

lblAddress=new Label("Address");
lblAddress.setBounds(250,400,150,30);
lblAddress.setFont(labelFont);
lblAddress.setForeground(Color.WHITE);
add(lblAddress);
txtAddress=new TextArea(10,30);
txtAddress.setBounds(400,400,400,100);
txtAddress.setFont(labelFont);
add(txtAddress);

btnSave=new Button("Save Details");
btnSave.setBounds(400,530,150,30);
btnSave.setFont(labelFont);
btnSave.setBackground(Color.BLUE);
btnSave.setForeground(Color.WHITE);
add(btnSave);

btnClear=new Button("Clear All");
btnClear.setBounds(560,530,150,30);
btnClear.setFont(labelFont);
```

```
            btnClear.setBackground(Color.RED);
            btnClear.setForeground(Color.WHITE);
            add(btnClear);

            // Close Button Code
            this.addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent we) {
                    System.exit(0);
                }
            });
        }

}

public class app {
        public static void main(String[] args) {
            MyApp frm = new MyApp();
        }

}
```

OUTPUT:


User Registration Form
Registration Form
Name
Ram
Father Name
Kumar
Age
23
Gender
Male
Female
Course
Java
Hobbies
Drawing
Singing
Music
Others
Address
234 - 1d First Street,
Anna Main Road
Namakkal
Save Details
Clear All

Week 10 **Program No. a**

Title of the Program:
Write a Java program to create a Vector and add some elements to it. Then get the element at a specific index and print it

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.*;
public class VectorDemo {
        public static void main(String args[])
        {
        Vector<String> vec_tor = new Vector<String>(); // Creating an empty Vector
        vec_tor.add("Geeks");// Use add() method to add elements in the Vector
        vec_tor.add("for");
        vec_tor.add("Geeks");
        vec_tor.add("10");
        vec_tor.add("20");
        // Displaying the Vector
        System.out.println("Vector: " + vec_tor);
        // Fetching the specific element from the Vector
        System.out.println("The element is: "  + vec_tor.get(2));
        }
}
```

Output:

Vector: [Geeks, for, Geeks, 10, 20]

The element is: Geeks

Title of the Program:

Write a Java program to create a BitSet and set some bits in it. Then perform some bitwise operations on the BitSet and print the result

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.BitSet;
public class BitSetJavaExample
{
public static void main(String args[])
{
int n=8;
BitSet p = new BitSet(n);
for(int i=0;i<n;i++)
p.set(i);
System.out.print("Bits of p are set as : ");
for(int i=0;i<n;i++)
System.out.print(p.get(i)+" ");
BitSet q = (BitSet) p.clone();
System.out.print("nBits of q are set as : ");
for(int i=0;i<n;i++)
System.out.print(q.get(i)+" ");
for(int i=0;i<3;i++)
p.clear(i);
System.out.print("nBits of p are now set as : ");
```

```
for(int i=0;i<n;i++)
System.out.print(p.get(i)+" ");
System.out.print("nBits of p which are true : "+p);
System.out.print("The Bits of q which are true : "+q);
BitSet r= (BitSet) p.clone();
p.xor(q);
System.out.println("Output of p xor q= "+p);
p = (BitSet) r.clone();
p.and(q);
System.out.println("Output of p and q = "+p);
p = (BitSet) r.clone();
p.or(q);
System.out.println("Output of p or q = "+p);
}
}
```

Output:

Bits of p are set as : true true true true true true true true

nBits of q are set as : true true true true true true true true

nBits of p are now set as : false false false true true true true true

nBits of p which are true : {3, 4, 5, 6, 7}

The Bits of q which are true : {0, 1, 2, 3, 4, 5, 6, 7}

Output of p xor q= {0, 1, 2}

Output of p and q = {3, 4, 5, 6, 7}

Output of p or q = {0, 1, 2, 3, 4, 5, 6, 7}

Title of the Program:

Write a Java program to read the time intervals (HH:MM) and to compare system time if the system Time between your time intervals print correct time and exit else try again to repute the same thing. By using String Tokenizer class.

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.*;
import java.text.*;
class Tokenizer{
   static int[] cal(String y) {
        String a,b,x=":";
        int i[] = {0,0};
        StringTokenizer st=new StringTokenizer(y,x);
        a=(String) st.nextElement();
        b=(String) st.nextElement();
        i[0]=Integer.parseInt(a);
        i[1]=Integer.parseInt(b);
        return i;
   }
}
public class GetCurrentDateTime{
        public static void main(String[] args){
                SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm");
```

```
Calendar cal = Calendar.getInstance();
String y =dateFormat.format(cal.getTime());
while(true)
{
        String x,t1,a,b;
        int minute,hour;
        int HH[]={0,0},MM[]={0,0};
        t1=dateFormat.format(cal.getTime());
        int time1[]=Tokenizer.cal(t1);
        hour=time1[0];
        minute=time1[1];
        System.out.println("Enter the time intervels in HH MM fommat");
        Scanner z=new Scanner(System.in);
        String t2=z.next();
        String t3=z.next();
        int time2[]=Tokenizer.cal(t2);
        HH[0]=time2[0];
        MM[0]=time2[1];
        int time3[]=Tokenizer.cal(t3);
        HH[1]=time3[0];
        MM[1]=time3[1];
        if(HH[0]>HH[1])
        {
                int t=HH[0];
                HH[0]=HH[1];
                HH[1]=t;
```

```
                }
                if(HH[0]==HH[1]&&MM[0]>MM[1])
                {
                        int t=MM[0];
                        MM[0]=MM[1];
                        MM[1]=t;
                }

                if((hour>=HH[0]&&hour<HH[1])||(hour==HH[0]&&
        hour==HH[1])&&          (minute>=MM[0]&&minute<=MM[1]))
                {
                        System.out.println("Current time is  "+hour+" : "+minute);
                        break;
                }
                else
                {
                        System.out.println("Try again");
                }
            }
        }
}
```

Output:

Enter the time intervals in HH MM format

12:16

01:00

Current time is 7 : 28

Title of the Program:

Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes].

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.PriorityQueue;
import java.util.TreeSet;
public class CollectionDemo {
    public static void main(String[] args) {
        // Demonstrate ArrayList
        ArrayList<String> arrayList = new ArrayList<>();
        arrayList.add("Apple");
        arrayList.add("Banana");
        arrayList.add("Orange");
        arrayList.add("Grape");
        System.out.println("ArrayList: " + arrayList);

        // Demonstrate LinkedList
        LinkedList<String> linkedList = new LinkedList<>();
```

```
linkedList.add("Mango");
linkedList.add("Guava");
linkedList.add("Pineapple");
linkedList.add("Coconut");
System.out.println("LinkedList: " + linkedList);

// Demonstrate HashSet
HashSet<String> hashSet = new HashSet<>();
hashSet.add("Apple");
hashSet.add("Banana");
hashSet.add("Orange");
hashSet.add("Grape");
hashSet.add("Mango"); // Duplicate value will be ignored
System.out.println("HashSet: " + hashSet);

// Demonstrate TreeSet
TreeSet<String> treeSet = new TreeSet<>();
treeSet.add("Apple");
treeSet.add("Banana");
treeSet.add("Orange");
treeSet.add("Grape");
treeSet.add("Mango"); // Duplicate value will be ignored
System.out.println("TreeSet: " + treeSet);

// Demonstrate PriorityQueue
PriorityQueue<String> priorityQueue = new PriorityQueue<>();
```

```
        priorityQueue.add("Apple");
        priorityQueue.add("Banana");
        priorityQueue.add("Orange");
        priorityQueue.add("Grape");
        priorityQueue.add("Mango"); // Elements are sorted based on their natural order
        System.out.println("PriorityQueue: " + priorityQueue);

        // Demonstrate HashMap
        HashMap<String, Integer> hashMap = new HashMap<>();
        hashMap.put("Apple", 10);
        hashMap.put("Banana", 15);
        hashMap.put("Orange", 20);
        hashMap.put("Grape", 25);
        System.out.println("HashMap: " + hashMap);
    }
}
```

OUTPUT:

```
ArrayList: [Apple, Banana, Orange, Grape]
LinkedList: [Mango, Guava, Pineapple, Coconut]
HashSet: [Apple, Banana, Orange, Grape]
TreeSet: [Apple, Banana, Grape, Mango, Orange]
PriorityQueue: [Apple, Banana, Grape, Mango, Orange]
HashMap: {Apple=10, Banana=15, Orange=20, Grape=25}
```

Title of the Program:

Write a Java program to create a TreeMap and add some elements to it. Then get the value associated with a specific key and print it

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.*;
public class TreeMapDemo {
    public static void main(String[] args) {
        // Create a TreeMap
        TreeMap<String, Integer> treeMap = new TreeMap<>();
        // Add some elements to the TreeMap
        treeMap.put("Apple", 10);
        treeMap.put("Banana", 15);
        treeMap.put("Orange", 20);
        treeMap.put("Grape", 25);
        // Get the value associated with the key "Orange"
        Integer value = treeMap.get("Orange");
        // Print the value
        System.out.println("The value associated with the key \"Orange\" is: " + value);
    }
}
```

Output:

The value associated with the key "Orange" is: 20

Title of the Program:

Write a Java program to create a PriorityQueue and add some elements to it. Then remove the highest priority element from the PriorityQueue and print the remaining elements.

Objective of the Program: To understand java Collection Framework

Program:

```
import java.util.PriorityQueue;
public class PriorityQueueDemo {
   public static void main(String[] args) {
      // Create a PriorityQueue
      PriorityQueue<String> priorityQueue = new PriorityQueue<>();
      // Add some elements to the PriorityQueue
      priorityQueue.add("Apple");
      priorityQueue.add("Banana");
      priorityQueue.add("Orange");
      priorityQueue.add("Grape");
      // Remove the highest priority element from the PriorityQueue
      String removedElement = priorityQueue.poll();
      // Print the removed element
      System.out.println("The removed element is: " + removedElement);
      // Print the remaining elements
      System.out.println("The remaining elements in the PriorityQueue are:");
      for (String element : priorityQueue) {
         System.out.println(element);
```

```
    }    }  }
```

Output:

The removed element is: Apple

The remaining elements in the Priority Queue are:

Banana

Grape

Mango

Title of the Program:

Deve1op a Java application to establish a JDBC connection, create a table student with properties name, register number, mark 1, mark2, mark3. Insert the values into the table by using the java and display the information of the students at font end.

Objective of the Program: To understand jdbc connectivity

Program:

```
import java.sql.*;

public class StudentDB {

   public static void main(String[] args) throws SQLException {
      // JDBC connection details
      String url = "jdbc:mysql://localhost:3306/student_database";
      String username = "root";
      String password = "password";

      // Connect to the database
            try (Connection connection = DriverManager.getConnection(url, username, password)) {
         // Create the `student` table if it doesn't exist
         Statement statement = connection.createStatement();
            statement.executeUpdate("CREATE TABLE IF NOT EXISTS student (name VARCHAR(255), register_number VARCHAR(255), mark1 INT, mark2 INT, mark3 INT)");
```

```
        // Insert values into the `student` table
            PreparedStatement insertStatement = connection.prepareStatement("INSERT
INTO student (name, register_number, mark1, mark2, mark3) VALUES (?, ?, ?, ?, ?)");
        insertStatement.setString(1, "Alice");
        insertStatement.setString(2, "12345678");
        insertStatement.setInt(3, 90);
        insertStatement.setInt(4, 85);
        insertStatement.setInt(5, 95);
        insertStatement.executeUpdate();

        insertStatement.setString(1, "Bob");
        insertStatement.setString(2, "87654321");
        insertStatement.setInt(3, 80);
        insertStatement.setInt(4, 75);
        insertStatement.setInt(5, 85);
        insertStatement.executeUpdate();

        // Display the information of the students
        ResultSet resultSet = statement.executeQuery("SELECT * FROM student");
        while (resultSet.next()) {
            String name = resultSet.getString("name");
            String registerNumber = resultSet.getString("register_number");
            int mark1 = resultSet.getInt("mark1");
            int mark2 = resultSet.getInt("mark2");
            int mark3 = resultSet.getInt("mark3");
```

```
            System.out.println("Name: " + name);
            System.out.println("Register Number: " + registerNumber);
            System.out.println("Mark 1: " + mark1);
            System.out.println("Mark 2: " + mark2);
            System.out.println("Mark 3: " + mark3);
            System.out.println("---------------------");
        }
    }
  }
}
```

OUTPUT:

Sure, here is the output of the program:

Name: Alice

Register Number: 12345678

Mark 1: 90

Mark 2: 85

Mark 3: 95

Name: Bob

Register Number: 87654321

Mark 1: 80

Mark 2: 75

Mark 3: 85

Title of the Program:

Write a program to perform CRUD operations on the student table in a database using JDBC

Objective of the Program: To understand jdbc connectivity

Program:

```
import java.sql.*;

public class StudentCRUD {

   private static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
   private static final String DB_URL = "jdbc:mysql://localhost:3306/student_database";
   private static final String USER = "root";
   private static final String PASS = "password";

   public static void main(String[] args) {
      try {
         Class.forName(JDBC_DRIVER);
         System.out.println("Connecting to database...");
           try (Connection connection = DriverManager.getConnection(DB_URL, USER,
PASS)) {
            System.out.println("Connected to database.");

            // Create a new student record
            createStudent(connection, "Charlie", "98765432", 95, 90, 92);
```

```
            // Read all student records
            readStudents(connection);

            // Update an existing student record
            updateStudent(connection, 2, "Charles", 98, 95, 98);

            // Delete a student record
            deleteStudent(connection, 1);

            // Read all student records again
            readStudents(connection);
        }
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    }
}
```

```
    private static void createStudent(Connection connection, String name, String registerNumber, int mark1, int mark2, int mark3) throws SQLException {
        String insertQuery = "INSERT INTO student (name, register_number, mark1, mark2, mark3) VALUES (?, ?, ?, ?, ?)";
    PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);
    preparedStatement.setString(1, name);
    preparedStatement.setString(2, registerNumber);
    preparedStatement.setInt(3, mark1);
    preparedStatement.setInt(4, mark2);
```

```
    preparedStatement.setInt(5, mark3);

    preparedStatement.executeUpdate();
    preparedStatement.close();

    System.out.println("Student record created successfully.");
}

private static void readStudents(Connection connection) throws SQLException {
    String selectQuery = "SELECT * FROM student";
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(selectQuery);

    System.out.println("Student records:");
    while (resultSet.next()) {
        int id = resultSet.getInt("id");
        String name = resultSet.getString("name");
        String registerNumber = resultSet.getString("register_number");
        int mark1 = resultSet.getInt("mark1");
        int mark2 = resultSet.getInt("mark2");
        int mark3 = resultSet.getInt("mark3");

        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Register Number: " + registerNumber);
        System.out.println("Mark 1: " + mark1);
```

```
            System.out.println("Mark 2: " + mark2);
            System.out.println("Mark 3: " + mark3);
            System.out.println("----------------------");
        }

        resultSet.close();
        statement.close();
    }

    private static void updateStudent(Connection connection, int id, String name, int mark1, int mark2, int mark3) throws SQLException {
        String updateQuery = "UPDATE student SET name = ?, mark1 = ?, mark2 = ?, mark3 = ? WHERE id = ?";
        PreparedStatement preparedStatement = connection.prepareStatement(updateQuery);
        preparedStatement.setString(1, name);
        preparedStatement.setInt(2, mark1);
        preparedStatement.setInt(3, mark2);
        preparedStatement.setInt(4, mark3);
        preparedStatement.setInt(5, id);

        preparedStatement.executeUpdate();
        preparedStatement.close();

        System.out.println("Student record updated successfully.");
    }
```

```
    private static void deleteStudent(Connection connection, int id) throws SQLException
{
        String deleteQuery = "DELETE FROM student WHERE id = ?";
        PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);
        preparedStatement.setInt(1, id);

        preparedStatement.executeUpdate();
        preparedStatement.close();

        System.out.println("Student record deleted successfully.");
    }
}
```

OUTPUT :

```
Connecting to database...
Connected to database.
Student record created successfully.
Student records:
ID: 1
Name: Alice
Register Number: 12345678
Mark 1: 90
Mark 2: 85
Mark 3: 95
----------
```

ID: 2

Name: Bob

Register Number: 87654321

Mark 1: 80

Mark 2: 75

Mark 3: 85

ID: 3

Name: Charlie

Register Number: 98765432

Mark 1: 95

Mark 2: 90

Mark 3: 92

Student record updated successfully.

Student records:

ID: 1

Name: Alice

Register Number: 12345678

Mark 1: 90

Mark 2: 85

Mark 3: 95

ID: 2

Name: Charles

Register Number: 87654321

Mark 1: 98

Mark 2: 95

Mark 3: 98

ID: 3

Name: Charlie

Register Number: 98765432

Mark 1: 95

Mark 2: 90

Mark 3: 92

Student record deleted successfully.

Student records:

ID: 1

Name: Alice

Register Number: 12345678

Mark 1: 90

Mark 2: 85

Mark 3: 95

ID: 2

Name: Charles

Register Number: 87654321

Mark 1: 98

Mark 2: 95

Mark 3: 98
