WEEK -7

7.a 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

```
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) {
   Box<Integer> integerBox = new Box<>();
integerBox.set(10);
      System.out.println("Integer box contents: " + integerBox.get());
 Box<String> stringBox = new Box<>();
 stringBox.set("Hello, World!");
      System.out.println("String box contents: " + stringBox.get());
      System.out.println("Is integerBox empty? " + integerBox.isEmpty());
      System.out.println("Is stringBox empty? " + stringBox.isEmpty());
  }}
```

Output:

```
C:\Users\VNR\Desktop\24071A6714>javac BoxDemo.java
C:\Users\VNR\Desktop\24071A6714>java BoxDemo
Integer box contents:10
String box
contents:Hello,world!
Is integerBox empty.false
Is stringBox empty.false
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

7.b 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) {
      Stack<String> stringStack = new Stack<>();
stringStack.push("Hello");
stringStack.push("World");
stringStack.push("!");
String poppedElement;
```

```
while (!stringStack.isEmpty()) {
poppedElement = stringStack.pop();
System.out.println("Popped element: " + poppedElement);
}
System.out.println("Is the stack empty? " + stringStack.isEmpty());
}
}
```

Output:

```
C:\Users\VNR\Desktop\24071A6714>javac StackDemo.java
C:\Users\VNR\Desktop\24071A6714>java StackDemo

Popped element:!

Popped element:World

Popped element:Hello

Is the stack empty.true
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:_____

Experiment No:_____ **Date:**_____

Week-8

8.a 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) {
int a = 10;
Integer b = a;
System.out.println("Autoboxing: " + b);
Integer c = 20;
int d = c;
System.out.println("Unboxing: " + d);
}
}
```

Output:

```
C:\Users\VNR\Desktop24071A6714>javac AutoboxingUnboxingExample.java

C:\Users\VNR\Desktop\24071A6714>java AutoBoxingUnboxingExample

Autoboxing:10

 Unboxing:20
```

8.b 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");
}
}
class TestAnnotation1{
public static void main(String args[]){
Animal a=new Dog();
a.eatSomething();
}}
@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:

```
C:\Users\VNR\Desktop\24071A6714>javac  TestAnnotation1.java
C:\Users\VNR\Desktop\24071A6714>java TestAnnotation
Compile    Time Error
Now no  warning at compile time
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

Week-9

9.aTitle 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:

```
C:\Users\VNR\Desktop\24071A6714>javac ThreadEx.java

C:\Users\VNR\Desktop\24071A6714>java ThreadEx.java

Thread(One,5,main)

Thread(Two,5,main)

Thread(Three,5,main)

Good Morning

Good Morning

Hello

Wel come

Good Morning

Hello
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

9.b Write a Java program that correctly implements producer consumer problem using the concept of inte thread communication.

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

```
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();
 }
```

```
C:\Users\VNR\Desktop\24071A6714>javac Program.java
C:\Users\VNR\Desktop\24071A6714>java Program
PUT 0
GET 0
PUT 1
GET 1
 PUT 2
 GET 2
 PUT 3
 GET 3
PUT 4
GET 4
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

Week-10

10.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

```
import java.util.*;
public class VectorDemo {
public static void main(String args[]) {
Vector<String> vec_tor = new Vector<String>();
vec_tor.add("Geeks");
vec_tor.add("for");
vec_tor.add("Geeks");
vec_tor.add("10");
vec_tor.add("20");
System.out.println("Vector: " + vec_tor);
System.out.println("The element is: " + vec_tor.get(2));
}
}
```

Output:

```
C:\Users\VNR\Desktop\24071A6714>javac VectorDemo.java
C:\Users\VNR\Desktop\24071A6714>java VectorDemo
Vector: [Geeks, for, Geeks, 10, 20]
The element is: Geeks
```

VNR VJIET

Name of the Laboratory: _____

Name of the Experiment: _____

Experiment No: _____ **Date:** _____

10.b 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:

```
C:\Users\VNR\Desktop\24071A6714>javac BitSetJavaExample.java
C:\Users\VNR\Desktop\24071A6714>java BitSetJavaExample
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}
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

10.c 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

```
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:

```
C:\Users\VNR\Desktop\24071A6714>javac  GetCurrentDateTime.java
C:\Users\VNR\Desktop\24071A6714>java GetCurrentDateTime
 Enter the time intervals in HH MM format
 12:16
 01:00
 Current time is  7 : 28
```

WEEK 11

11. aWrite 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) {
        ArrayList<String> arrayList = new ArrayList<>();
arrayList.add("Apple");
arrayList.add("Banana");
arrayList.add("Orange");
arrayList.add("Grape");
        System.out.println("ArrayList: " + arrayList);
LinkedList<String> linkedList = new LinkedList<>();
        linkedList.add("Mango");
linkedList.add("Guava");
linkedList.add("Pineapple");
linkedList.add("Coconut");
        System.out.println("LinkedList: " + linkedList);
HashSet<String> hashSet = new HashSet<>();
hashSet.add("Apple");
 hashSet.add("Banana");
hashSet.add("Orange");
hashSet.add("Grape");
        hashSet.add("Mango");
System.out.println("HashSet: " + hashSet);
TreeSet<String> treeSet = new TreeSet<>();
treeSet.add("Apple");
```

```
 treeSet.add("Banana");
treeSet.add("Orange");
treeSet.add("Grape");
treeSet.add("Mango");
System.out.println("TreeSet: " + treeSet);
      PriorityQueue<String> priorityQueue = new PriorityQueue<>();
    priorityQueue.add("Apple");
    priorityQueue.add("Banana");
     priorityQueue.add("Orange");
     priorityQueue.add("Grape");
     priorityQueue.add("Mango");
     System.out.println("PriorityQueue: " + priorityQueue);
       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);
              }
}
```

```
C:\Users\VNR\Desktop\24071A6714>javac CollectionDemo.java
C:\Users\VNR\Desktop\24071A6714>java CollectionDemo
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)
```

11.b 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) {

  TreeMap<String, Integer> treeMap = new TreeMap<>();
       treeMap.put("Apple", 10);
treeMap.put("Banana", 15);
 treeMap.put("Orange", 20);
 treeMap.put("Grape", 25);

       Integer value = treeMap.get("Orange");
       System.out.println("The value associated with the key \"Orange\" is: " + value);
    }
}
```

Output:

```
C:\Users\VNR\Desktop\24071A6714 >javac TreeMapDemo.java
C:\Users\VNR\Desktop\24071A6714 >java TreeMapDemo
The value associated with the key "Orange" is: 20
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:_____

Experiment No:_____ **Date:**_____

11.c 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

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

Output:

```
C:\Users\VNR\Desktop\24071A6714>javac  PriorityQueueDemo.java

C:\Users\VNR\Desktop\24071A6714>java PriorityQueueDemo
The removed element is: Apple
The remaining elements in the Priority Queue are:
Banana
Grape
Mango
```

VNR VJIET

Name of the Laboratory:

Name of the Experiment:

Experiment No: **Date:**

WEEK 12

12.a 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

```
import java.sql.*;
public class StudentDB {
public static void main(String[] args) throws SQLException {
String url = "jdbc:mysql://localhost:3306/student_database";
        String username = "root";
String password = "password";
  try (Connection connection = DriverManager.getConnection(url, username,
    password)) {
   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)");
   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(5, 85);
insertStatement.executeUpdate();
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:

```
C:\Users\VNR\Desktop\24071A6714>javac  StudentDB.java

C:\Users\VNR\Desktop\24071A6714>java StudentDB
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
-----------------------
```

12. bWrite 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.");
createStudent(connection, "Charlie", "98765432", 95, 90, 92);
readStudents(connection);
updateStudent(connection, 2, "Charles", 98, 95, 98);
deleteStudent(connection, 1);
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.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 ps = connection.prepareStatement(updateQuery);
ps.setString(1, name);
ps.setInt(2, mark1);
ps.setInt(3, mark2);
ps.setInt(4, mark3);
ps.setInt(5, id);
ps.executeUpdate();
ps.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:

```
C:\Users\VNR\Desktop\24071A6714>javac  StudentCRUD.java

C:\Users\VNR\Desktop\24071A6714>java StudentCRUD
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
```