Experiment No. __9__

Date of performance: 09/10/22

Date of Submission: 09/10/22

SAP Id: 500091584

Roll No.: R2142210822

Name of the Student: Ujjwal Kumar Gupta

---

1. **Title:** **Threads and Collections**

2. **Objective:** **Threads and Collections**

3. **List of lab activities:**

1) Write a program to implement the concept of threading by extending Thread Class and Runnableinterface.2) Write a program for generating
2) 2 threads, one for printing even numbers and the other for printingodd numbers.
3) 3) Write a program to launch 10 threads. Each thread increments a counter variable. Run the programwith synchronization.
4) 4) Write a Java program to create five threads with different priorities. Send two threads of the highestpriority to sleep state. Check the aliveness of the threads and mark which thread is long lasting

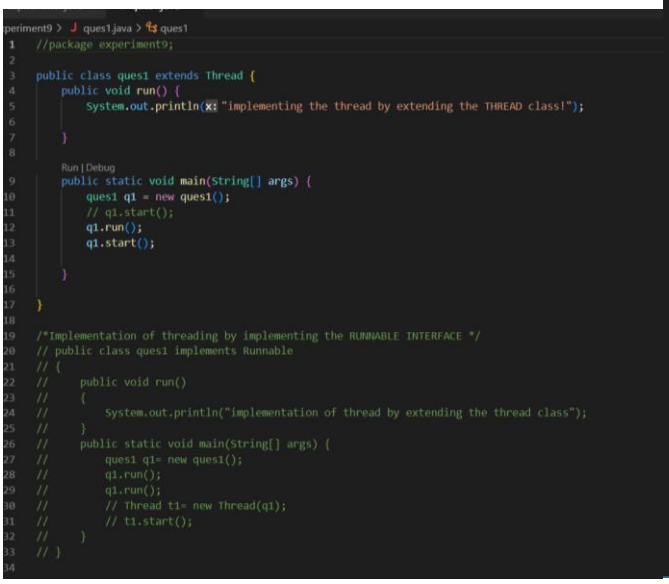
TITLE: Collections

1) Write a program for the following:Read all elements from ArrayList by using Iterator.Create duplicate object of an ArrayList instance.Reverse ArrayList content.

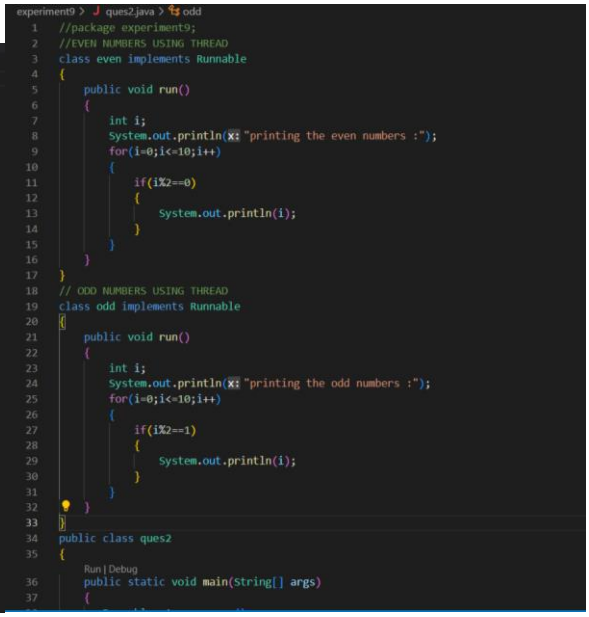
2) Write a program for the following HashMapfind whether specified key exists or not.find whether specified value exists or notget all keys from the given HashMapget all key-value pair as Entry objects

3) Write a program for the following HashSetcopy another collection object to HashSet object.delete all entries at one call from HashSetsearch user defined objects from HashSet

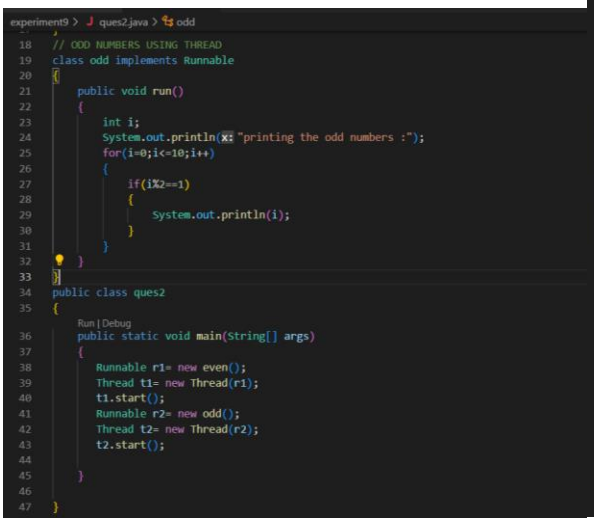
4. **Algorithm/Flowchart and Code followed by Output screenshot (2samples for each program):**



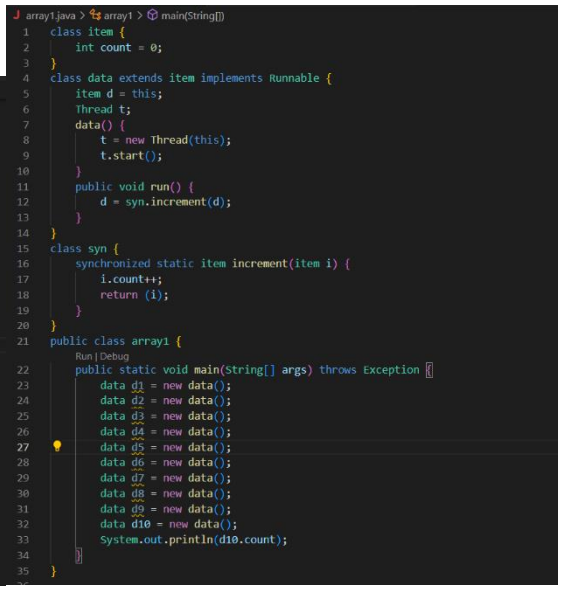
```
//package experiment9;

public class ques1 extends Thread {
    public void run() {
        System.out.println("implementing the thread by extending the THREAD class!");

    }

    Run | Debug
    public static void main(String[] args) {
        ques1 q1 = new ques1();
        // q1.start();
        q1.run();
        q1.start();

    }

}

/*Implementation of threading by implementing the RUNNABLE INTERFACE */
// public class ques1 implements Runnable
// {
//     public void run()
//     {
//         System.out.println("implementation of thread by extending the thread class");
//     }
//     public static void main(String[] args) {
//         ques1 q1= new ques1();
//         q1.run();
//         q1.run();
//         // Thread t1= new Thread(q1);
//         // t1.start();
//     }
// }
```



```
//package experiment9;
//EVEN NUMBERS USING THREAD
class even implements Runnable
{
    public void run()
    {
        int i;
        System.out.println("printing the even numbers :");
        for(i=0;i<=10;i++)
        {
            if(i%2==0)
            {
                System.out.println(i);
            }
        }
    }
}
// ODD NUMBERS USING THREAD
class odd implements Runnable
{
    public void run()
    {
        int i;
        System.out.println("printing the odd numbers :");
        for(i=0;i<=10;i++)
        {
            if(i%2==1)
            {
                System.out.println(i);
            }
        }
    }
}
public class ques2
{
    Run | Debug
    public static void main(String[] args)
    {
```



```
// ODD NUMBERS USING THREAD
class odd implements Runnable
{
    public void run()
    {
        int i;
        System.out.println("printing the odd numbers :");
        for(i=0;i<=10;i++)
        {
            if(i%2==1)
            {
                System.out.println(i);
            }
        }
    }
}
public class ques2
{
    Run | Debug
    public static void main(String[] args)
    {
       Runnable r1= new even();
       Thread t1= new Thread(r1);
       t1.start();
       Runnable r2= new odd();
       Thread t2= new Thread(r2);
       t2.start();

    }

}
```



```
class item {
    int count = 0;
}
class data extends item implements Runnable {
    item d = this;
    Thread t;
    data() {
        t = new Thread(this);
        t.start();
    }
    public void run() {
        d = syn.increment(d);
    }
}
class syn {
    synchronized static item increment(item i) {
        i.count++;
        return (i);
    }
}
public class array1 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        data d1 = new data();
        data d2 = new data();
        data d3 = new data();
        data d4 = new data();
        data d5 = new data();
        data d6 = new data();
        data d7 = new data();
        data d8 = new data();
        data d9 = new data();
        data d10 = new data();
        System.out.println(d10.count);
    }
}
```

```java
riority.java > priority > main(String[])
public class priority extends Thread {
    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        Thread t1= new Thread();
        Thread t2= new Thread();
        Thread t3= new Thread();
        Thread t4= new Thread();
        Thread t5= new Thread();
        t1.setPriority(newPriority: 6);
        t2.setPriority(newPriority: 1);
        t3.setPriority(newPriority: 9);
        t4.setPriority(newPriority: 10);
        t5.setPriority(newPriority: 4);
        t1.sleep(millis: 200);
        if(t1.isAlive())
        {
            System.out.println(x: "t1 is alive");
        }
        else
        System.out.println(x: "t1 not alive");
        t2.start();
        if(t2.isAlive())
        System.out.println(x: "t2 alive");
        else
        System.out.println(x: "t2 not alive");

        t3.sleep(millis: 1500);
        if(t3.isAlive())
        System.out.println(x: "t3 alive");
        else
        System.out.println(x: "t3 not alive");

        t4.start();
        if(t4.isAlive())
        System.out.println(x: "t4 is alive");
        else
        System.out.println(x: "t4 not alive");
         t5.start();
```

```java
priority.java > priority > main(String[])
        else
        System.out.println(x: "t2 not alive");

        t3.sleep(millis: 1500);
        if(t3.isAlive())
        System.out.println(x: "t3 alive");
        else
        System.out.println(x: "t3 not alive");

        t4.start();
        if(t4.isAlive())
        System.out.println(x: "t4 is alive");
        else
        System.out.println(x: "t4 not alive");
         t5.start();
         if(t5.isAlive())
         System.out.println(x: "t5 is alive");
         else
         System.out.println(x: "t5 not alive");
    }

}
```

```java
array.java > array > main(String[])
import java.util.*;
 public class array
 {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add(e: "Mango");//Adding object in arraylist
            list.add(e: "Apple");
            list.add(e: "Banana");
            list.add(e: "Grapes");
            //Printing the arraylist object
            Iterator itr=list.iterator();//getting the Iterator
            while(itr.hasNext())
            {//check if iterator has the elements
             System.out.println(itr.next());//printing the element and move to next
            }
            System.out.println(x: "\nafter cloning the list");
            ArrayList list2= new ArrayList();
            list2= (ArrayList)list.clone();
            Iterator itr2=list2.iterator();//getting the Iterator
            while(itr2.hasNext())
            {//check if iterator has the elements
             System.out.println(itr2.next());//printing the element and move to next
            }
            //System.out.println(list2);
            System.out.println(x: " \naftr reversing the array");
            Collections.reverse(list);
            for(String str: list)
            {
                System.out.println(str);
            }

    }
}
```

5. Brief notes about all the concepts related to the lab experiment