Assignment-4
Garbage Collection

Q1)
```
public class Main{
    static class UnreachableObject{
        String name;
    }
    public UnreachableObject(String name){
        this.name = name;
    }
    public void display(){
        UnreachableObject obj = new UnreachableObject("Object");
        obj.show();
    }
    public void show(){
        UnreachableObject obj = new UnreachableObject("Object");
        System.out.println("Name: "+obj.name);
    }
    protected void finalize() throws Throwable{
        System.out.println("Object "+name+" is being garbage collected");
        super.finalize();
    }
}
public static void main(String[] args){
    UnreachableObject obj = new UnreachableObject("Main Object");
    obj.display();
    system.gc();
}
}
```

Output

Name: ~~Static~~ Object

Q2

```
class ReassigningReference {
    String name;
    public ReassigningReference (String name) {
        this.name = name;
    }
    protected void finalize () throws Throwable {
        System.out.println ("Object named '" + name + "' has been garbage
                             collected.");
        super.finalize ();
    }
}
public class Main {
    public static void main (String [] args) {
        ReassigningReference obj1 = new ReassigningReference ("Object 1");
        ReassigningReference obj2 = new ReassigningReference ("Object 2");
        obj1 = obj2;
        System.gc ();
        try {
            Thread.sleep (1000); // Pause for 1 sec
        } catch (InterruptedException e) {
            System.out.println (e);
        }
    }
}
```

Output:-

Object named 'Object 1' has been garbage collected.

Q3
```
public class NullifiedReference {
    String name;
    public NullifiedReference (String name) {
        this.name = name;
    }
    public static void main (String[] args) {
        NullifiedReference r = new NullifiedReference ("MyObject");
        reference = null;
        System.gc();
    }
    protected void finalize() {
        System.out.println ("Object '"+name+"' has been successfully
                             garbage collected.");
    }
}
```

Q4
```
class AnonymousObject {
    String name;
    public AnonymousObject (String name) {
        this.name = name;
    }
    public void display() {
        System.out.println ("Name: "+name);
    }
    protected void finalize() throws Throwable {
        SOP ("Object '"+name+"' has been garbage collected.");
        super.finalize();
    }
}
public class Main {
    public static void main (String[] args) {
        new AnonymousObject ("John").display();
        System.gc();
    }
}
```

Output
Name: John

Q3)
```
public class MyData{
    Private int value1;
    private double value2;
    public MyData (int value1, double value2){
        this.value1 = value1;
        this.value2 = value2;
    }
    public void setValue1 (int value1){
        this.value1 = value1;
    }
    public void setValue2 (double value2){
        this.value2 = value2;
    }
    public int getValue1(){
        return value1;
    }
    public double getValue2(){
        return value2;
    }
    public static void main (String [] args){
        MyData obj1 = new MyData(10, 3.14);
        MyData obj2 = new MyData(20, 6.28);
        obj1 = null;
        obj2 = null;
        Runtime rt = Runtime.getRuntime();
        long totalMemory = rt.totalMemory();
        System.out.println("Total memory allocated: " + totalMemory +
                           "bytes");
        rt.gc();
        long freeMemory = rt.freeMemory();
        long memoryOccupied = totalMemory - freeMemory;
        System.out.println("Memory occupied by objects: " + memoryOccupied +
                           "bytes");
    }
}
```

Name: Srikrushna Mishra Regd. Number: 2241013183

Output :-

Total memory allocated : 113573888 bytes

Memory occeepied by objects : 599704 bytes

Q6

```
import java.util.*;
public class MemoryIntensiveProgram {
    public static void main(String[] args) {
        List<Object> objectList = new ArrayList<>();
        long startTime = System.
        for(int i=0; i<1_000_000; i++) {
            objectList.add(new Object());
        }
        System.out.println("Timestamp: " + System.currentTimeMillis());
        System.out.println("Heap Size (MB): " + (Runtime.getRuntime().
                totalMemory() / (1024 * 1024)));
        System.out.println("Free Space (MB): " + (Runtime.getRuntime().
                freeMemory() / (1024 * 1024)));
    }
}
```

Output :-

Timestamp: 1712431629507

Heap Size (MB): 121

Free Space (MB): 90

Q7
```
import java.util.*;
class Student{
    String name;
    String address;
    String gender;
    String dob;
    public Student(String name, String address, String gender, String dob){
        this.name=name;
        this.address= address;
        this.gender=gender;
        this.dob= dob;
    }
    public String getName(){
      return name;
    }
    public String getAddress(){
      return address;
    }
    public String getGender(){
       return gender;
    }
    public String getDob(){
       return dob;
    }
    public void setName(String name){
        this.name=name;
    }
    public void setAddress(String address){
        this.address=address;
    }
    public void setGender(String gender){
        this.gender= gender;
    }
```

```
    public void setDob(String dob) {
        this.dob = dob;
    }
    protected void finalize() {
        System.out.println("Memory for student" + name + "successfully
                            reclaimed.");
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        List<Student> studentList = new ArrayList<>();
        while(true){
            SOP("\n1. Add student \n2. Search by Name \n3. Exit");
            int choice = sc.nextInt();
            sc.nextLine();
            switch(choice){
             case 1:
                SOP("Enter student name: ");
                String name = sc.nextLine();
                SOP("Enter address: ");
                String address = sc.nextLine();
                SOP("Enter gender: ");
                String gender = sc.nextline();
                SOP("Enter date of birth (dd/mm/yyyy): ");
                String dob = sc.nextline();
                Student s = new Student(name, address, gender, dob);
                studentList.add(s);
                break;
             case 2:
                SOP("Enter student name to search:");
                String search = sc.nextline();
```

```
        for (Student st : studentList){
            if (st.getName().equalsIgnoreCase(sName)){
                SOP("Student found:");
                SOP("Name: "+st.getName());
                SOP("Address: "+st.getAddress());
                SOP("Gender: "+st.getGender());
                SOP("DOB: "+st.getDob());
                break;
            }
        }
        break;
    case 3:
        SOP("Exiting Program");
        sc.close();
        return;
    default:
        SOP("Invalid choice. Please try again.");
    }
  }
}
}
```

Output:-

```
1. Add Student
2. Search by Name
3. Exit
1
Enter Student Name : Sam
Enter address : nayapali
Enter gender : male
Enter date of birth (dd/mm/yyyy) : 02-06-2004
```