ASSIGNMENT – 4

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

Registration Number: 2341013237

Section: 23412G1

Branch: CSE

Q1. Write a Java program to demonstrate garbage collection using an UnreachableObject class. It should include a constructor to initialize an object with a given name, a show() method creating an instance and invoking display(), and the display() method creating another instance. The main() method should call show() and explicitly invoke the garbage collector. The program must override the finalize() method to print the object's name upon successful garbage collection, illustrating how unreachable objects are collected.

Solution:

```java
class UnreachableObject {
    String name;
    public UnreachableObject(String name) {
        this.name = name;
    }
    public void show() {
        UnreachableObject obj1 = new UnreachableObject(name:"Object1");
        display();
    }
    public void display() {
        UnreachableObject obj2 = new UnreachableObject(name:"Object2");
    }
    protected void finalize() throws Throwable {
        System.out.println(name + " is being collected as garbage.");
    }
}
public class Q1 {
    Run | Debug
    public static void main(String[] args) {
        UnreachableObject obj = new UnreachableObject(name:"Main Object");
        obj.show();
        obj = null;
        System.gc();
        try {
            Thread.sleep(millis:1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(x:"End of main method.");
    }
}
```

Output:

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\18-03-2025 ASSIGNMENT-4 (Chap-15)\" && javac Q1.java && java Q1
Q1.java:13: warning: [removal] finalize() in Object has been deprecated and marked for removal
    protected void finalize() throws Throwable {
                   ^
1 warning
Main Object is being collected as garbage.
Object2 is being collected as garbage.
Object1 is being collected as garbage.
End of main method.

[Done] exited with code=0 in 2.058 seconds
```

Q2. Develop a Java program to demonstrate reference reassignment and garbage collection using the ReassigningReference class. The class should have a constructor to initialize an object with a given name. In the main() method, create two instances of ReassigningReference, reassign one reference to another, and then explicitly invoke the garbage collector. Override the finalize() method to print the object's name upon successful garbage collection.

Solution:

```java
class ReassigningReference {
    String name;
    public ReassigningReference(String nm) {
        this.name = nm;
    }

    protected void finalize() {
        System.out.println("Garbage collected From: " + name);
    }
}
public class Q2 {
    Run | Debug
    public static void main(String[] args) {
        ReassigningReference obj1 = new ReassigningReference(nm:"Object 1");
        ReassigningReference obj2 = new ReassigningReference(nm:"Object 2");
        // Reassigning obj1 to obj2, making the first Object1 eligible for garbage collection
        obj1 = obj2;

        // Explicitly invoking garbage collector
        System.gc();
        try {
            Thread.sleep(millis:500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println(x:"End of main method.");
    }
}
```

Output:

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\18-03-2025 ASSIGNMENT-4 (Chap-15)\" && javac Q2.java && java Q2
Q2.java:7: warning: [removal] finalize() in Object has been deprecated and marked for removal
    protected void finalize() {
                   ^
1 warning
Garbage collected From: Object 1
End of main method.

[Done] exited with code=0 in 1.545 seconds
```

Q3. Write a Java program to demonstrate nullification of references and garbage collection using the NullifiedReference class. The class should have a constructor to initialize an object with a given name. In the main() method, create an instance of NullifiedReference, assign it, then nullify the reference, and explicitly invoke the garbage collector. Override the finalize() method to print the object's name upon successful garbage collection.

Solution:

```java
class NullifiedReference {
    String name;
    public NullifiedReference(String nm)    {           name = nm;                          }
    public void finalize()                  {   System.out.println("Object's Name: "+name);     }
}
public class Q3 {
    Run | Debug
    public static void main(String[] args) {
        NullifiedReference o1 = new NullifiedReference(nm:"Object 1(3)");
        o1 = null;
        System.gc();

        try {
            Thread.sleep(millis:200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(x:"End of Main Program(3).");
    }
}
```

Output:

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\18-03-2025 ASSIGNMENT-4 (Chap-15)\" &&
javac Q3.java && java Q3
Q3.java:4: warning: [removal] finalize() in Object has been deprecated and marked for removal
    public void finalize()                 {   System.out.println("Object's Name: "+name);     }
                ^
1 warning
Object's Name: Object 1(3)
End of Main Program(3).

[Done] exited with code=0 in 1.496 seconds
```

Q4. Create a Java program to demonstrate anonymous objects and garbage collection using the AnonymousObject class. The class should have a constructor to initialize an object with a name. In the main() method, create an anonymous object and explicitly invoke the garbage collector. Override the finalize() method to print the object's name upon successful garbage collection

Solution:

```
Q1.java 2   Q2.java 1   Q3.java 1   Q4.java   Q5.java
Q4.java
class AnonymousObject {
    String name;
    public AnonymousObject (String nm) {
        name=nm;
    }
    public void finalize() {
        System.out.println("Object's Name: "+name);
    }
}
public class Q4 {
    Run | Debug
    public static void main(String[] args) {
        new AnonymousObject(nm:"Object 1(4)");
        System.gc();

        try {
            Thread.sleep(millis:200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(x:"End of Main Program(4).");
    }
}
```

Output:

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS          Filter          Code

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\18-03-2025 ASSIGNMENT-4 (Chap-15)\" &&
javac Q4.java && java Q4
Q4.java:6: warning: [removal] finalize() in Object has been deprecated and marked for removal
    public void finalize() {
                ^
1 warning
Object's Name: Object 1(4)
End of Main Program(4).

[Done] exited with code=0 in 1.266 seconds
```

Q5. Develop a Java class with private integer and double data members, along with methods for initialization, setting, and updating these members. Create two objects of this class and invoke the necessary methods to modify the data. Use the Runtime class to calculate the total allocated memory and memory occupied by the objects. Apply any technique to make the objects unreachable, making them eligible for garbage collection. Finally, recheck the utilized and total memory using the Runtime class.

Solution:

```
J Q1.java 2   J Q2.java 1   J Q3.java 1   J Q4.java   J Q5.java ×   J MemoryIntensiveApp.java
J Q5.java > Q5 > main(String[])
class DataObject {
    private int intValue;       private double doubleValue;

    public DataObject(int iv, double dv) {
        intValue = iv;              doubleValue = dv;
    }

    public void updateValues(int intValue, double doubleValue) {
        this.intValue = intValue;
        this.doubleValue = doubleValue;
    }

    public void printValues() {
        System.out.println("Integer Value: " + intValue + " & Double Value: " + doubleValue);
    }

    protected void finalize() {
        System.out.println(x:"Object's Garbage has been collected");
    }
}
public class Q5 {
    Run | Debug
    public static void main(String[] args) {
        Runtime runtime = Runtime.getRuntime();

        System.out.println(x:"Initial memory usage:");
        long initialMemory = runtime.totalMemory() - runtime.freeMemory();
        System.out.println("Total Memory: " + runtime.totalMemory() + " bytes");
        System.out.println("Used Memory: " + initialMemory + " bytes\n");

        DataObject obj1 = new DataObject(iv:10, dv:10.5);
        DataObject obj2 = new DataObject(iv:20, dv:20.5);

        obj1.printValues();
        obj2.printValues();
```

```java
public class Q5 {
    public static void main(String[] args) {
        DataObject obj1 = new DataObject(iv:10, dv:10.5);
        DataObject obj2 = new DataObject(iv:20, dv:20.5);

        obj1.printValues();
        obj2.printValues();

        obj1.updateValues(intValue:25, doubleValue:25.5);
        System.out.println();
        obj2.updateValues(intValue:30, doubleValue:30.5);

        obj1.printValues();
        obj2.printValues();

        obj1 = null;
        obj2 = null;

        System.gc();

        try {
            Thread.sleep(millis:200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println(x:"\nFinal Memory usage after making objects unreachable and requesting garbage collection:");

        long finalMemory = runtime.totalMemory() - runtime.freeMemory();
        System.out.println("Total Memory: " + runtime.totalMemory() + " bytes");
        System.out.println("Used Memory: " + finalMemory + " bytes");

        System.out.println("Memory freed: " + (initialMemory - finalMemory) + " bytes");
    }
}
```

Q6. Write a memory-intensive Java program that creates a large number of objects and test it using the G1 garbage collector. Print timestamps along with the total heap size and free memory. Use the following commands to retrieve heap memory details:

- **Total heap memory: Runtime.getRuntime().totalMemory();**
- **Free heap memory: Runtime.getRuntime().freeMemory();**

Solution:

```java
import java.util.*;
class Q6_MemoryIntensiveApp {
    private int[] largeArray;
    public Q6_MemoryIntensiveApp(int size) {
        this.largeArray = new int[size];
    }
    public static void printMemoryUsage(Runtime runtime) {
        System.out.println("Total heap memory: " + runtime.totalMemory());
        System.out.println("Free heap memory: " + runtime.freeMemory());
        System.out.println("Used memory: " + (runtime.totalMemory() - runtime.freeMemory()));
        System.out.println("-".repeat(count:20));
    }
    Run | Debug
    public static void main(String[] args) {
        // Get Runtime instance
        Runtime runtime = Runtime.getRuntime();
        // List to hold objects and increase memory usage (Without <> operator)
        List<Q6_MemoryIntensiveApp> objectList = new ArrayList<Q6_MemoryIntensiveApp>();
        System.out.println(new Date() + " Starting memory-intensive task...");
        printMemoryUsage(runtime);
        // Create a large number of objects to fill up heap (Without underscore)
        for (int i = 0; i < 50000; i++) { // Adjust count based on system capacity
            objectList.add(new Q6_MemoryIntensiveApp(size:10000)); // Remove underscore
            if (i % 10000 == 0) { // Remove underscore
                System.out.println(new Date() + " Created" + i + " objects");
                printMemoryUsage(runtime);
            }
        }
        System.out.println(new Date() + " - Finished object creation. Now clearing memory...");
        printMemoryUsage(runtime);
        // Clear the list to make objects eligible for garbage collection
        objectList.clear();
        System.gc();
        // Explicitly request garbage collection
        // Pause to allow garbage collection
        try {
            Thread.sleep(millis:2000);
        } catch (InterruptedException e) {
            e.printStackTrace();        }
        System.out.println(new Date() + " - After garbage collection: ");
        printMemoryUsage(runtime);
    }
}
```

Output:

```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\18-03-2025 ASSIGNMENT-4 (Chap-15)\" && javac Q6_MemoryIntensiveApp.java && java Q6_MemoryIntensiveApp
Tue Mar 18 21:14:43 IST 2025 Starting memory-intensive task...
Total heap memory: 264241152
Free heap memory: 260297904
Used memory: 3943248
--------------------
Tue Mar 18 21:14:43 IST 2025 Created0 objects
Total heap memory: 264241152
Free heap memory: 260046848
Used memory: 4194304
--------------------
Tue Mar 18 21:14:44 IST 2025 Created10000 objects
Total heap memory: 792723456
Free heap memory: 387229136
Used memory: 405494320
--------------------
Tue Mar 18 21:14:44 IST 2025 Created20000 objects
Total heap memory: 2162163712
Free heap memory: 1353031344
Used memory: 809132368
--------------------
Tue Mar 18 21:14:44 IST 2025 Created30000 objects
Total heap memory: 2984247296
Free heap memory: 1771333136
Used memory: 1212914160
--------------------
Tue Mar 18 21:14:44 IST 2025 Created40000 objects
Total heap memory: 2984247296
Free heap memory: 1368639936
Used memory: 1615607360
--------------------
Tue Mar 18 21:14:45 IST 2025 - Finished object creation. Now clearing memory...
Total heap memory: 2984247296
Free heap memory: 964896096
Used memory: 2019351200
--------------------
Tue Mar 18 21:14:47 IST 2025 - After garbage collection:
Total heap memory: 50331648
Free heap memory: 46429096
Used memory: 3902552
--------------------

[Done] exited with code=0 in 4.637 seconds
```

Q7. Create a Java program for university student enrollment using a Student class to manage course details and student information. Implement efficient garbage collection for memory management and use the Runtime class to monitor memory usage. Override the finalize() method to print a message upon successful garbage collection.

Solution:

```java
import java.util.*;
class Student {
    private int studentId;    private String name;    private List<String> courses;

    public Student(int studentId, String name) {
        this.studentId = studentId;
        this.name = name;
        this.courses = new ArrayList<>();
        System.out.println("Student " + name + " (ID: " + studentId + ") enrolled.");
    }
    // Enroll in a course
    public void enrollCourse(String course) {
        courses.add(course);
        System.out.println(name + " enrolled in: " + course);
    }

    // Withdraw from all courses (make eligible for garbage collection)
    public void withdraw() {
        courses.clear();
        System.out.println(name + " has withdrawn from all courses.");
    }

    protected void finalize() throws Throwable {
        System.out.println("Student " + name + " (ID: " + studentId + ") is being garbage collected.");
    }
}
```

```java
public class Q7 {
    Run | Debug
    public static void main(String[] args) {
        Runtime runtime = Runtime.getRuntime();

        // Memory before student enrollment
        long totalBefore = runtime.totalMemory();
        long freeBefore = runtime.freeMemory();
        System.out.println("\nMemory Before Enrollment - Total Heap: " + totalBefore + ", Free Heap: " + freeBefore);

        // Creating students
        Student student1 = new Student(studentId:69, name:"Arpit");
        Student student2 = new Student(studentId:78, name:"Ruchi");

        // Enrolling students in courses
        student1.enrollCourse(course:"Computer Science & Engineering");
        student2.enrollCourse(course:"Civil Engineering");

        // Memory after enrollment
        long totalAfterEnroll = runtime.totalMemory();
        long freeAfterEnroll = runtime.freeMemory();
        System.out.println("\nMemory After Enrollment - Total Heap: " + totalAfterEnroll + "\t Free Heap: " + freeAfterEnroll+"\n");

        // Withdraw students and make them eligible for GC
        student1.withdraw();
        student2.withdraw();

        student1 = null;
        student2 = null;

        // Request garbage collection
        System.out.println(x:"\nRequesting Garbage Collection...");
        System.gc();
```

```java
        // Request garbage collection
        System.out.println(x:"\nRequesting Garbage Collection...");
        System.gc();

        try {
            Thread.sleep(millis:200);
        } catch (InterruptedException e) {
            e.printStackTrace();        }

        // Memory after garbage collection
        long totalAfterGC = runtime.totalMemory();
        long freeAfterGC = runtime.freeMemory();
        System.out.println("\nMemory After Garbage Collection - Total Heap: " + totalAfterGC + "\t Free Heap: " + freeAfterGC);

        System.out.println(x:"\nEnd of Enrollment System.");
    }
}
```