Garbage Collection

Agenda

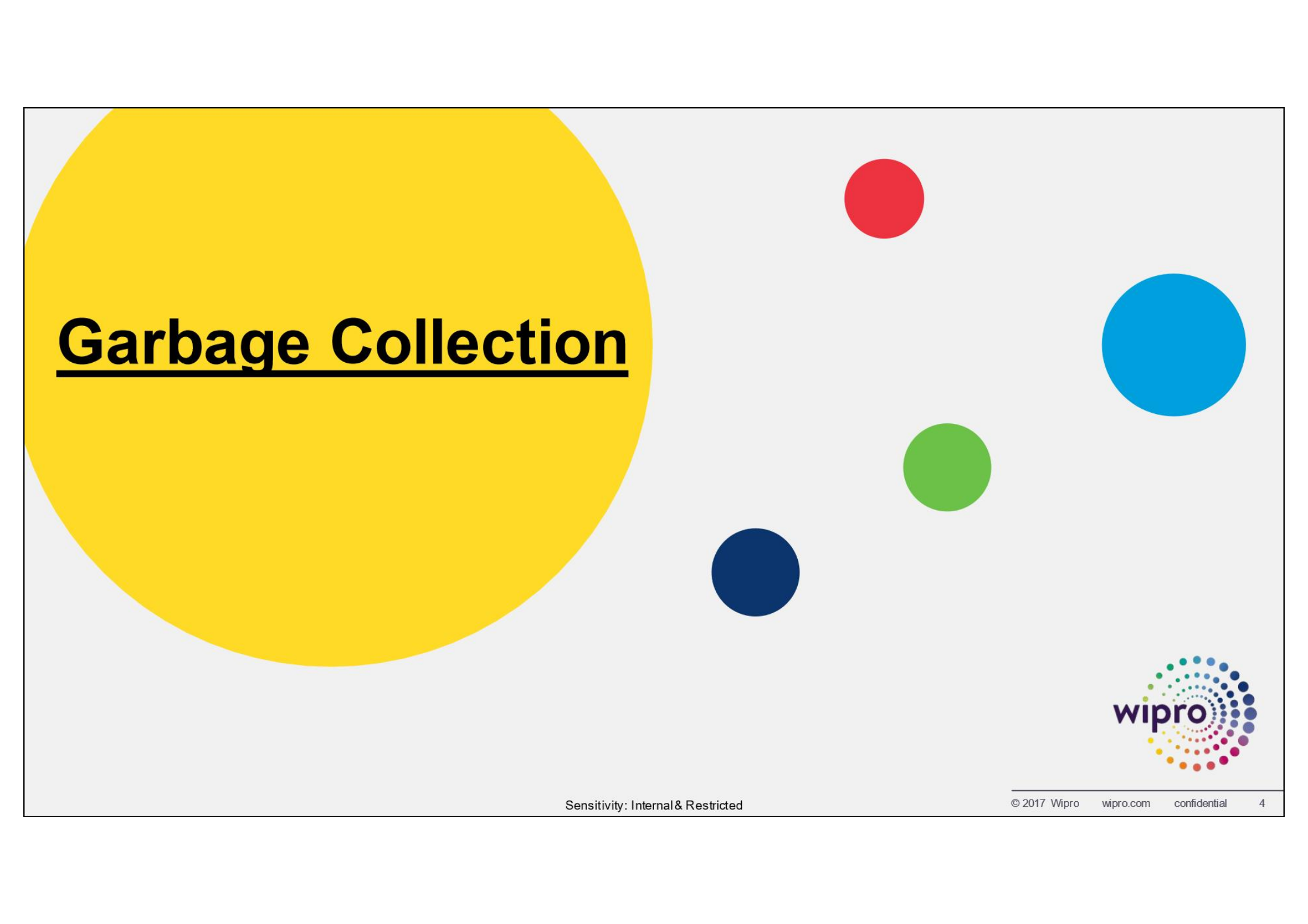
1 **Garbage Collection**

Objectives

At the end of this module, you will be able to:

- Get basic information about garbage collection
- Define finalize method

Garbage Collection



Garbage Collection- Introduction

You have created objects of classes in last several programs. What do you think will happen to the memory occupied by these object when the program finishes?

Once the program completes,

These objects become garbage…

Now what to do with these garbage???

We need to clean it up!!!

Java's Cleanup Mechanism – The Garbage Collector

- Java has its own Garbage Collector
- Consider the following:

Test test=new Test();

Here when the object is created, memory is allocated for this object.

test=null; => When you execute this, the reference is deleted but the memory occupied by the object is not released.

- We need to release the memory occupied by the 'test' object.

Java's Cleanup Mechanism – The Garbage Collector

- Objects on the heap must be deallocated or destroyed, and their memory released for later reallocation
- Java handles object deallocation automatically through **garbage collection**
- Objects which are occupying memory but not referenced will be reclaimed

Java's Cleanup Mechanism – The Garbage Collector

- The Garbage Collection is done automatically by JVM.
- If we need to manually do the garbage collection, we can use the following method:

```
Runtime rs = Runtime.getRuntime();
rs.gc();
```

The gc() method is used to manually run the garbage collection.

Example

```
import java.util.*;
class GarbageCollection {
public static void main(String s[]) throws Exception {
Runtime rs = Runtime.getRuntime();

System.out.println("Free memory in JVM before Garbage Collection = "+rs.freeMemory());

rs.gc();

System.out.println("Free memory in JVM after Garbage Collection = "+rs.freeMemory()); } }
```

Here the *rs.freeMemory()* method will give the free memory available in the system. Try this program and observe the results…

The finalize( ) Method

- Often, an object needs to perform some action when it is destroyed
- The action could pertain to:
  - releasing a file handle
  - reinitializing a variable, such as a counter
- Java's answer is a mechanism called finalization
- By using finalization, you can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector

The finalize( ) Method (Contd.).

- To add a finalizer to a class, you simply define the finalize( ) method
- The Java runtime calls that method whenever it is about to recycle an object of that class
- Inside the finalize( ) method, you will specify those actions that must be performed before an object is destroyed

Example Demonstrating finalize() method

```
public class CounterTest {
    public static int count;
        public CounterTest ( ) {
        count++;
    }
}


  public static void main(String args[ ])   {
            CounterTest ob1 = new CounterTest ( ) ;
        System.out.println("Number of objects :" +
                    CounterTest.count) ;
        CounterTest ob2 = new CounterTest ( );
        System.out.println("Number of objects :" +
        CounterTest.count) ;
```

Example Demonstrating finalize() method (Contd.).

```
Runtime r = Runtime.getRuntime( );
ob1 = null;
ob2 = null;
r.gc( );
}
protected void finalize( ) {
System.out.println("Program about to terminate");
        CounterTest.count --;
        System.out.println("Number of objects :"
  +                    CounterTest.count) ;
   }
}
```

Output:

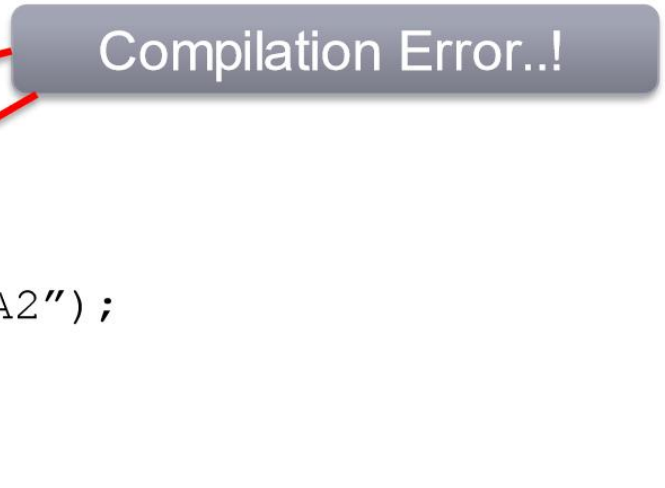
Number of objects :1
Number of objects :2
Program about to terminate
Number of objects :1
Program about to terminate
Number of objects :0

Point out the errors in the following code :

```
1. class A1 {
2.  final void m1() { }
3. }

4. final class A2 extends A1 {
5.  void m1(){
6.      System.out.println("Method m1 of A2");
7.  }
8. }

9. class A3 extends A2{
10. public static void main(String[] args) {
11.     System.out.println("Executed Successfully");
12. }
13.}
```

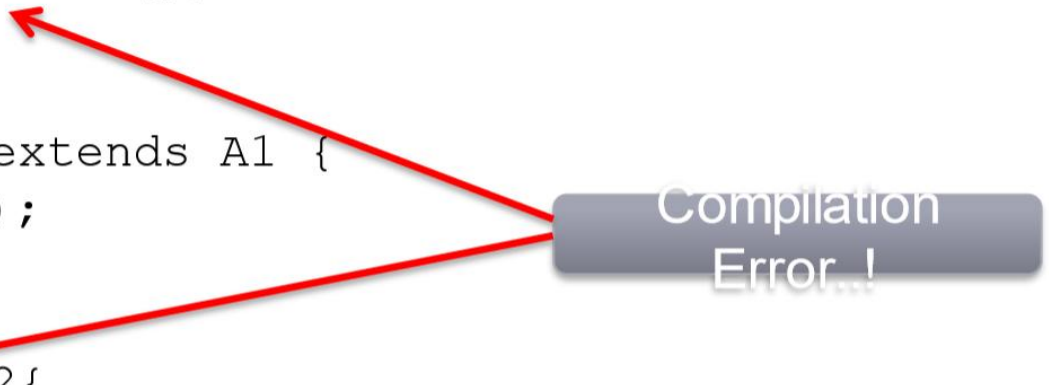


Point out the errors in the following code (Contd.).

```
1. abstract class A1 {
2.  abstract final void m1();
3. }

4. abstract class A2 extends A1 {
5.  abstract void m2();
6. }

7. class A3 extends A2{
8.  public static void main(String[] args) {
9.      System.out.println("Executed Successfully");
10. }
11.}
```



Quiz

In which of the below cases, an object is eligible for Garbage Collection?

a. Demo d;

b. Demo d= new Demo();

c. Demo d=new Demo(); d=null;

d. None of the above.

Summary

- In this session, you were able to:
  - Get basic information about garbage collection
  - Define finalize method

Thank You