Explicitly Make Objects Eligible for Garbage Collection

As all java people know how Memory Management & GarbageCollection work in java.
As of Summary: Garbage collector revolves around making sure that the heap has as much free space as possible. Objects are created in the heap. When JVM(Java Virtual Machine) senses that memory is running low, at that time it calls garbage collector to remove/delete the unused objects that are not needed any more from memory(heap).

Explicitly Makes Objects Eligible for Collection

(a). Nulling a Reference:

1. An object becomes eligible for garbage collection when there are no more reachable references to it. To remove a reference to an object is to set the reference variable that refers to the object to null.

Program code:

```
public class RemoveReference {
public static void main(String [] args) {
StringBuffer sb = new StringBuffer("Reference");
System.out.println(sb);
// Here the StringBuffer object is not eligible for collection
sb = null;
// Now the StringBuffer object is eligible for collection
}
}
```

(b). Reassigning a Reference Variable :

1. We can also decouple a reference variable from an object by setting the referencevariable to refer to another object.
2. The objects created in the method are eligible for garbage collection ,once the method has returned.
3. If an object is returned from the method, its reference might be assigned to a referencevariable in the method that called it; hence, it will not be eligible for collection

Program Code :

```
import java.util.Date;
class ReassignReference {
public static void main(String [] args) {
Date d = getDate(); // Reference return from the method

StringBuffer s1 = new StringBuffer("hello");
StringBuffer s2 = new StringBuffer("goodbye");
System.out.println(s1);
// At this point the StringBuffer "hello" is not eligible
s1 = s2; // Redirects s1 to refer to the "goodbye" object
// Now the StringBuffer "hello" is eligible for collection
}
public static Date getDate() {
Date d2 = new Date();
StringBuffer now = new StringBuffer(d2.toString());
// object eligible to garbage collector even we don't set the object as null explicitly.
System.out.println(now);
return d2; // Reference Return , not eligible for garbage collector
}
}
```

(c) .Isolating a Reference

1.There is another way in which objects can become eligible for garbage collection,even if they still have valid references! We call this scenario "**islands of isolation**."

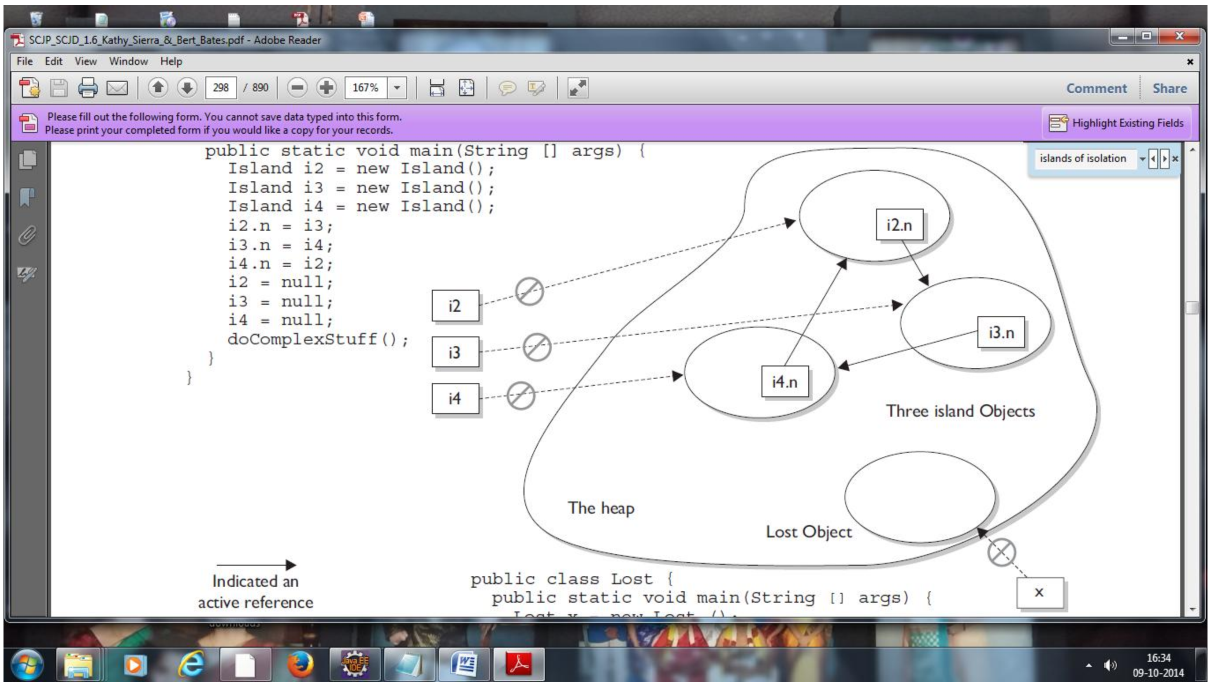
Program Code :

```
public class Island {
Island i;
public static void main(String [] args) {
Island i2 = new Island();
Island i3 = new Island();
Island i4 = new Island();

i2.i = i3; // i2 refers to i3
i3.i = i4; // i3 refers to i4
i4.i = i2; // i4 refers to i2

i2 = null;
i3 = null;
i4 = null;
// do complicated, memory intensive stuff
}
}
```

When the code reaches ***// do complicated***, the three Island objects (previously known as i2, i3, and i4) have instance variables so that they refer to each other, but their links to the outside world (i2, i3, and i4) have been null. These three objects are eligible for garbagecollection.


SCJP_SCJD_1.6_Kathy_Sierra_&_Bert_Bates.pdf - Adobe Reader
File Edit View Window Help
298 / 890
167%
Comment
Share
Highlight Existing Fields
islands of isolation
i2
i3
i4
i2.n
i3.n
i4.n
Three island Objects
The heap
Lost Object
x
Indicated an active reference
16:34
09-10-2014

Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

```
    public static void main(String [] args) {
      Island i2 = new Island();
      Island i3 = new Island();
      Island i4 = new Island();
      i2.n = i3;
      i3.n = i4;
      i4.n = i2;
      i2 = null;
      i3 = null;
      i4 = null;
      doComplexStuff();
    }
  }
```

```
public class Lost {
  public static void main(String [] args) {
```

(d) Forcing Garbage Collection

1. Garbage collection cannot be forced by a programmer.
2. However, Java provides some methods that allow you to request that the JVM perform garbage collection.
3. We can call garbage collector using Runtime java classes.

Ex: **Runtime.getRuntime().gc();**

Ex: **System.gc();**

```
class Test
{
Test t;
static int i=1;
protected void finalize()
{

System.out.println("Garbage collected from object" + i);
i++;
}

public static void main(String args[]){
Test t1=new Test();
Test t2=new Test();
Test t3=new Test();

//No Object Is Eligible for GC

t1.t=t2; //No Object Is Eligible for GC
t2.t=t3; //No Object Is Eligible for GC
t3.t=t1; //No Object Is Eligible for GC

t1=null;
//No Object Is Eligible for GC (Coz t2 still have a reference)

t2=null;
//No Object Is Eligible for GC (Coz t3 still hava a reference)

t3=null;
//All the 3 Object Is Eligible for GC (None of them have a reference)

System.gc();
}

}
```

output:

```
Garbage collected from object1
Garbage collected from object2
Garbage collected from object3
```

but sometimes you may not get the output becoz JVM does it automatically without calling the finalize() method. Here we are forcing JVM to call finalize() method which is not guaranteed always.