```
Garbage Collection:To destroy useless objects
======================
1.Introduction
2.The ways to make an object elgible for GC
3.The Methods for requesting JVM to run GC
4.Finalization
=========================================

1.Nullyfying the reference variable

class Test
{}{
    public static void main(String[] args)
    {}{
        Student s1=new Student();
        Student s2=new Student();
        s1=null;
        //at this stage one object is elgible for GC
        s2=null;
        //at this stage 2 objects is elgible for GC
    }
}
class Student
{}{
}
--------------

//2.Reassigning the reference variables
class Test
{}{
    public static void main(String[] args)
    {}{
        Student s1=new Student();
        Student s2=new Student();
        //no

        s1=new Student();
        s2=s1;

    }
}
class Student
{}{
}
---------------------------

//3.objects created inside a method
class Test
{}{
```

```
50      public static void main(String[] args)
51      {}{
52          m1();
53      }
54
55      public static void m1()
56      {}{
57          Student s1=new Student();
58          Student s2=new Student();
59
60      }
61  }
62  class Student
63  {}{
64  }
65  -----------------
66  IV.Island of Isolation
67
68  class Test
69  {}{
70      Test i;
71      public static void main(String[] args)
72      {}{
73          Test t1=new Test();
74          Test t2=new Test();
75          Test t3=new Test();
76          t1.i=t2;
77          t2.i=t3;
78          t3.i=t1;
79          //at this stage no object is elgible for GC
80          t1=null;//at this stage no object is elgible for GC
81          t2=null;//at this stage no object is elgible for GC
82          t3=null;//at this stage All Three objects are elgible for GC
83
84      }
85  }
86
87
88  The Methods for requesting JVM to run GC
89  ========================================
90   2 ways
91
92   1.By using System class
93
94      System.gc();
95   2.By using Rutime class
96     Runtime r=Runtime.getRuntime();
97
98
```

```
 99 import java.util.*;
100 class MyTestRuntime
101 {}{
102 public static void main(String[] args)
103     {}{
104     Runtime r=Runtime.getRuntime();
105     System.out.println("Total Memory:="+r.totalMemory());
106     System.out.println("Free Memory:="+r.freeMemory());
107
108     for(int i=0;i<100000;i++)
109         {}{
110         Date d=new Date();
111         d=null;
112         }
113
114         System.out.println("Free Memory After:="+r.freeMemory());
115         r.gc();
116         System.out.println("After Gc Free Memory:="+r.freeMemory());
117
118
119     }
120
121 }
122 output:
123
124 D:\Yellaswamy_ClassNotes\GarbageCollectorEx>javac MyTestRuntime.java
125
126 D:\Yellaswamy_ClassNotes\GarbageCollectorEx>java MyTestRuntime
127 Total Memory:=62390272
128 Free Memory:=61089816
129 Free Memory After:=58593264
130 After Gc Free Memory:=61773704
131
132 D:\Yellaswamy_ClassNotes\GarbageCollectorEx>
133
134 Finalization:
135 ==========================
136 1.Just before destroying an object Garbage Collector calls finalize() t
137 2.once finalize() completes automatically GC destroys that object.
138 3.finalize() present in object class with the following prototype
139     protected void finalize() throws Throwable
140 4.Based on our requirement we can override finalize() method in our cla
141
142
143 case 1:
144 ==========
145 class Test
146 {}{
147
```

```
148 public static void main(String[] args)
149     {}{
150     //String s=new String("Ashok");
151     Test s=new Test();
152     s=null;
153     System.gc();
154     System.out.println("end of main method");
155 
156     }
157 
158     public void finalize()
159     {}{
160         System.out.println("finalize() method called");
161     }
162 }
163 
164 case 2:
165 ==========
166 class Test
167 {}{
168 
169 public static void main(String[] args)
170     {}{
171     Test t=new Test();
172     t.finalize();
173     t.finalize();
174     t=null;
175     System.gc();
176     //t=null;
177     System.out.println("end of main method");
178 
179     }
180 
181     public void finalize()
182     {}{
183         System.out.println("finalize() method called");
184     }
185 }
186 
187 case 3:
188 ===========
189 class Test
190 {}{
191 
192 public static void main(String[] args)
193     {}{
194     Test t=new Test();
195     //t.finalize();
196 
```

```
197     t=null;
198     System.gc();
199     //t=null;
200     System.out.println("end of main method");
201
202     }
203
204     public void finalize()
205     {}{
206         System.out.println("finalize() method called");
207         System.out.println(10/0);
208     }
209 }
210
211
```