Test 1

```
00 public class E4 {
01  int x=4;
02     E4(int x) {
03        f(x); f();
04        System.out.println(x);
05     }
06     void f() { x++; System.out.print(x);}
07     void f(int x) { this.x++; x--;System.out.print(x);}
08     public static void main(String arg[]) {
09         int x=2;
10         new E4(2);
11 }}
```

Test 2

```
01 public class E3 {
02     static int counter=4;
03     private int value=2;
04     E3(){value=++counter;}
05     public String toString(){
06         return this.getClass().getName()+value+" ";}
07     public void finalize(){System.out.print("F"+value);}
08 }
09 class G extends E3{
10     public static void main(String d[]){
11         LinkedList<E3> x=new LinkedList<E3>();
12         E3 a1=new G();
13         G a2=new G();
14         E3 a3=new E3();
15         x.add(a1);x.add(a3);
16         a1=null; a2=null; a3=null;
17         Iterator<E3> it=x.iterator();
18         while (it.hasNext()){System.out.print(it.next());}
19         System.gc();System.runFinalization();
20 }}
```

Test 3

```
01 public class E2 {
02     static HashSet hs=new HashSet();
03     public int hashCode() {return 0;}
04     public boolean equals(Object x) {
           return (x.getClass().equals(this.getClass()));}
05     public static void main(String s[]){
06         f(new E2()); f(new E2()); f(new A2());
07         f(new A2()); f(new A3()); f(new A3());
08     }
09     static void f(E2 x) {int v=0;
           if (hs.add(x)) v=1; System.out.print(v);
       }
10 }
11 class A2 extends E2 {
12       public boolean equals(Object x) {return x instanceof A2;}
13 }
14 class A3 extends A2 {}
```

Test 4

```
01  class A1 {int x=7; }
02  class B1 extends A1 {int k=1;}
03  public class E1 {
04      public static void main(String[] args) {
05          A1 a1=new B1();
06          A1 a2=(A1)(new B1());
07          B1 b1=new A1();
08          System.out.println(a1.x+a2.x+b1.x);
09      }
10  }
```

Test 5

```
01  #include <cstdlib>
02  #include <iostream>
03  using namespace std;
04  void f(char x[2],int index,char value){
05      x[index]=value;
06  }
07  int main(int argc, char** argv) {
08      char a[]="ABCDEFGHIL";
09      strcpy(&a[3],"000");
10      f(&a[2],4,'$');
11      f(&a[2],6,0);
12      cout<<a;
13      return 0;
14  }
```

Test 6

```
00  public class C {
01      int s=5;
02      void f() {System.out.print("A"+(++s));}
03      public static void main(String[]a){
04          C y=new C();
05          C x=new C() {
06              void f() {System.out.print("B"+(--s));}
07          };
08          x.f();
09          y.f();
10      }
11  }
```

Test 7

```
00  public class D {
01      static int x=2;
02      public static void main(String[] args) {
03          D a5=new D();       a5.f();
04          a5=new D();         a5.f();
05          System.gc();        System.runFinalization();
06      }
07      void f() {Pippo a=new Pippo2();
08      }
09      public void finalize() { System.out.print("X"); }
10      class Pippo {
11          int k;
12          Pippo() {k=++x;}
13          public void finalize() { System.out.print(k); }
14      }
15      class Pippo2 extends Pippo {
16              Pippo2() {k=x++;}
17      }
18  }
```

Test 8

```
00  public class E6 {
01      static Collection l1 = new LinkedList();
02      int x=6;
03      E6(){}
04      E6(int x){
05          l1.add(this);
06          l1.add(new E5A());
07      }
08      public static void main(String z[]) {
09          new E6(3);
10          Iterator iter = l1.iterator();
11          while (iter.hasNext()) {
12              ((E6)(iter.next())).f();
13      }}
14      public void f() { System.out.print(x); }
15  }
16  class E5A extends E6 {
17      public void f() {
18          x++; super.f(); System.out.print(2);
19  }}
```

Test 9 – scrivere nel campo per l'output del test la sequenza risultante indicando V per le affermazioni vere e F per quelle false

9.1	Se una classe è astratta è permesso usarla per effettuare ereditarietà multipla
9.2	E' corretto scrivere Integer k=3;
9.3	L'esistenza di un metodo f(int x) e di uno f(String s) nella stessa classe è un esempio di overloading
9.4	L'esistenza in una classe di un metodo f(int x), e in una sua sottoclasse di un metodo f(float x) è un esempio di overriding
9.5	Se a.hashCode==b.hashCode, a.equals(b) deve essere vero
9.6	Se a.hashCode!=b.hashCode, a.equals(b) deve essere falso
9.7	Il metodo finalize() chiama automaticamente il corrispondente metodo della superclasse
9.8	Il costruttore chiama automaticamente il costruttore della superclasse con gli stessi parametri. Se nella superclasse non è disponibile un costruttore con parametri dello stesso tipo, viene chiamato il costruttore vuoto.