C

Test 1 `java B`

```
00 public class B {
01     static int x = 2;
02     B() { x++; }
03     B(int y) {x=x+y;}
04     public void finalize(){x--;}
05     public static void main(String[] args) {
06         A a2 = new A();
07         System.out.print(a2);
08         A a1 = new A(2);
09         System.out.print(a1);
10     }
11     public String toString(){return ""+x;}
12 }
13 class A extends B {
14     int x=4;
15     A() { x++; }
16     A(int x) {super(x); x++;}
17     public String toString(){return super.toString()+x;}
18 }
```

Test 2 `java uno.C`

```
01 package uno;
02 public class C {
03   void f(int k) {
04     System.out.print(k*3);
05   }
06   public static void main (String args[]){
07     Object z = new B();
08     if (z instanceof uno.C) ((C) z).f(3);
09     if (z instanceof uno.B) ((B) z).f(6);
10   }
11   public static void main (String var){
12     Object z = new B();
13     if (z instanceof uno.C) ((C) z).f(3);
14   }}
15 class B extends C{
16   void f(int k) {
17     System.out.print(k);
18   }}
```

Test 3 `java Test3`

```
00 public class Test3 {
01   public static void main(String[] args) {
02     A a = new A();
03     a.m1(5);
04     System.out.println("y="+a.y);
05 } }
06 class A {
07   int y = 4;
08   void m1(int y) { m2(y++); }
09   void m2(int y) { --y; }
10 }
```

C

Test 4

```
01  #include <cstdlib>
02  #include <iostream>
03  using namespace std;
04  void f(int k){ k++; }
05  void g(int *k){ (*k)++; }
06  void h(int *k){ --k;}
07  int main() {
08      int k[]={3,5,8,13};
09      f(k[2]);
10      g(&k[3]);
11      h(k);
12      for (int i=0; i<4; i++) { cout << k[i]; }
13      return 0;
14  }
```

Test 5 **java Sette**

```
01  public class Sette {
02      Sette(){
03          List<String> a = new ArrayList<String>();
04          List<String> b = new HashSet<String>();
05          for (int k=0;k<10;k++) {
06              String s="A"+(k%3);
07              a.add(s);
08              b.add(s);
09          }
10          System.out.println(a.size()+" "+b.size());
11      }
12      public static void main(String[] a) { new Sette(); }
13      public static void main(String a) {new Sette(); new Sette();}
14  }
```

Test 6 **java Sette**

```
01  public class Sette {
02      Sette(){
03          Collection<String> a = new ArrayList<String>();
04          Collection<String> b = new HashSet<String>();
05          for (int k=0;k<10;k++) {
06              String s="A"+(k%5);
07              a.add(s);
08              b.add(s);
09          }
10          System.out.println(a.size()+" "+b.size());
11      }
12      public static void main(String[] a) {
13           new Sette();
14  }}
```

C

Test 7 java B1

01	public class A { // this class is in the A.java file
02	public int x=0;
03	A(int x) {x=6;}
04	}
05	public class B1 { // this class is in the B1.java file
06	public static void main(String a[]){
07	A a1=new A();
08	A a2=new A(9);
09	System.out.println(a1.x+" "+a2.x);
10	}
11	}

Test 8 java D

01	public class D {
02	static int x=1;
03	public static void main(String[] args) {
04	D a5=new D(); a5.f();
05	a5=new D(); a5.f();
06	System.gc(); System.runFinalization();
07	}
08	void f() {Pippo a=new Pippo2();
09	}
10	public void finalize() { System.out.print("Z"); }
11	class Pippo {
12	int k;
13	Pippo() {k=++x;}
14	public void finalize() { System.out.print(k); }
15	}
16	class Pippo2 extends Pippo {
17	Pippo2() {k=x++;}
18	}}

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	Nei metodi statici non è possibile leggere e scrivere le variabili di istanza
9.2	Si consideri un attributo x dichiarato come protected nella classe C del package P1; x non è visibile da una classe D appartenente a un package P2, a meno che D non erediti da C.
9.3	Si consideri un attributo x dichiarato come protected nella classe C del package P1; x è visibile da un altra classe D appartenente allo stesso package P1 solo se D eredita da C.
9.4	In Java esiste ereditarietà singola: quindi, un tipo interfaccia può ereditare da un solo tipo interfaccia.
9.5	Sia I un'interfaccia, e C una classe che la implementa. L'assegnamento C c = new I(); non causa errori in compilazione, ma può generare errori a runtime.
9.6	Un metodo generico è un qualsiasi metodo che contiene parametri di tipo Object nella propria definizione.
9.7	Il costruttore di una classe chiama automaticamente il costruttore della superclasse con gli stessi parametri; se in questa non è disponibile un costruttore con la stessa firma, viene chiamato il costruttore senza parametri.
9.8	Le variabili dichiarate static non sono modificabili dai metodi privati