A

Istruzioni

Staccare questo foglio dal resto del blocchetto, ed utilizzarlo per segnare le risposte (sul retro di questa pagina). Segnare SUBITO nome, cognome e numero di matricola negli appositi spazi. Al termine della prova si dovrà consegnare SOLO questo foglio.

Il codice Java delle domande non mostra gli *import*, che si assumono essere correttamente presenti.

Vi sono tre tipi di risposte:

A) il codice esegue correttamente. Si indichi l'output

TEST x	il codice esegue correttamente, e l'output è →	

B) il codice potrebbe generare errori. Si indichi in quale riga l'errore avviene, e se si tratta di un compile error o di un runtime error. Se non vi sono errori, si indichi l'output. Va cerchiata l'opzione scelta (A, B o C)

TEST x	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

C) Domande vero/falso. Per questo tipo di domande, le risposte errate SOTTRAGGONO punti. Nel dubbio si consiglia di lasciare in bianco. Si riporti V o F nelle 8 caselle.

TEST x	Riportare la sequenza di V e F								

A

NOME, COGNOME	
NUMERO DI MATRICOLA	

TEST 1	il codice esegue correttamente, e l'output è →	

TEST 2	il codice esegue correttamente, e l'output è →	

TEST 3	il codice esegue correttamente, e l'output è →	

TEST 4	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

TEST 5	il codice esegue correttamente, e l'output è →	

TEST 6	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

TEST 7	il codice esegue correttamente, e l'output è →	

TEST 8	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

TEST 9	Riportare la sequenza di V e F									

A

Test 1

```
01 public class E3 {
02     static int counter=2;
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 2

```
00 public class E4 {
01   int x=3;
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 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 {}
```

A

Test 4

```
01  class A1 {int x=5; }
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          B1 b1=(B1)(new A1());
07          A1 a2=(A1)(new B1());
08          System.out.println(a1.x+a2.x+b1.x);
09      }
10  }
```

Test 5

```
01  using namespace std;
02  #include <cstdlib>
03  #include <iostream>
04  void f(int x[4],int value){ x[0]+=value; }
05  void g(int x,int value){ x+=value; }
06  int main(int argc, char** argv) {
07      int z=5;
08      int a[]={1,2,3,4,5,6,7,8,9};
09      g(z,1);
10      f(&z,2);
11      f(a+2,3);
12      g(a[2],4);
13      cout<<z<<a[2];
14      return 0;
15  }
```

Test 6

```
00  public class C {
01      int s=7;
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      public static void main(String a){
12          C y=new C();
13      }
14  }
```

A

Test 7

```
00  public class D {
01      static int x=1;
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 ll = new LinkedList();
02      int x=3;
03      E6(int x){
04          ll.add(this);
05          ll.add(new E6A());
06      }
07      public static void main(String z[]) {
08          new E6(3);
09          Iterator iter = ll.iterator();
10          while (iter.hasNext()) {
11              ((E6)(iter.next())).f();
12      }}
13      public void f() { System.out.print(x); }
14      public static void main(String z) {
15          new E6A();
16          System.out.print(ll.size());
17  }}
18  class E6A extends E6 {
19      E6A(){x++;}
20      public void f() {
21          x++; super.f(); System.out.print(2);
22  }}
```

A

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	**Ereditarietà multipla è permessa con le interfacce e le classi astratte.**
9.2	**E' corretto scrivere Integer k=3;**
9.3	**Nel main si può leggere una qualunque variabile di istanza della classe in cui è contenuto.**
9.4	**Se a.hashCode!=b.hashCode, a.equals(b) deve essere falso**
9.5	**L'esistenza di un metodo f(int x) in una classe e di uno f(String s) in una sua sottoclasse è un esempio di overloading**
9.6	**In una classe ci può essere un solo metodo main**
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.**