A

NOME, COGNOME	
NUMERO DI MATRICOLA	

Istruzioni: leggere il codice dei test sui fogli allegati.
Indicare la risposta sul presente foglio, cerchiando la voce A, B o C. Se si prevede un errore indicare la riga e riportare la motivazione nel campo libero. Se si prevede una corretta esecuzione del codice riportarne l'output nel campo libero.

TEST 1	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

TEST 2	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	il codice esegue correttamente, e l'output è →	

TEST 3	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	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	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	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	A	compile error alla riga _____ perchè →	
	B	runtime error alla riga _____ perchè →	
	C	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 →								

Risposte errate al punto 9 sottraggono punti

A

A

Test 1

```
00 import java.util.*;
01 class A {
02   private static A a;
03   private static int instancecount=0;
04   private A()  {instancecount++;}
05   static A getInstance() {if (a==null) a=new A(); return a;}
06   void printCount(){System.out.println(instancecount);}
07 }
08 public class Prova {
09   public static void main(String args[]) {
10     Collection s = new LinkedList();
11     for (int i = 1; i < 4; i++) s.add(A.getInstance());
12     Iterator i = s.iterator();
13     while (i.hasNext()) ( (A) i.next()).printCount();
14   }
15 }
```

Commentato [mr1]: 1 1 1

Test 2

```
00 import java.util.*;
01 class A {
02   private static A a;
03   private static int instancecount=0;
04   private A()  {instancecount++;}
05   static A getInstance() {if (a==null) a=new A(); return a;}
06   void printCount(){System.out.println(instancecount);}
07 }
08 public class Prova {
09   public static void main(String args[]) {
10     Collection s = new HashSet();
11     for (int i = 1; i < 4; i++) s.add(new A());
12     Iterator i = s.iterator();
13     while (i.hasNext()) ( (A) i.next()).printCount();
14   }
15 }
```

Commentato [mr2]: Errore di compilazione alla riga 11 – costruttore privato

A

Test 3

```
A00    package esame; // NOTA :QUESTA CLASSE E' NEL FILE A.java
A01    public class A {
A02      int x=1;
A03      public static void main(String string[]) {
A04         (new abcd.B()).f();
A05      }
A06    }
B01    package abcd; // NOTA :QUESTA CLASSE E' NEL FILE B.java
B02    public class B extends esame.A{
B03      public void f(){
B04        System.out.println(++x);
B05      }
B06   }
```

Commentato [mr3]: La soluzione di questo esercizio si basa sul concetto di scope delle variabili. La variabile x non ha qualificatori public, private o protected quindi la sua vilibilità è di package.

ERRORE DI COMPILAZIONE ALLA RIGA B04 – x ha visibilità di package

Test 4

```
01  #include <iostream.h>
02  void f(char *x, int * y) {
03    (*y)++;
04    x[*y]++;  }
05  void g(char x[], int y) {
06    y--;
07    x[y]- -;  }
08  int main(){
09    char x[2];
10    int y;
11    x[0]='B';  x[1]='C';   y=0;
12    f(x,&y);
13    g(x,y);
14    cout<<x[0]<<" "<<x[1]<<" "<<y;
15    return 0; }
```

Commentato [mr4]:
OUTPUT: A D 1

Test 5

```
01  package uno;
02  public class A {
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.A) ((A) z).f(1);
09      if (z instanceof uno.B) ((B) z).f(2);
10    }
11  }
12  class B extends A{
13    void f(int k) {
14      System.out.print(k);
15    }
16  }
```

Commentato [mr5]:
Output: 12

A

Test 6

```
01 class A {
02   A(int x) {System.out.print("X");}
03   A() {System.out.print("Y");}
04   public void finalize() {System.out.print("Z");}
05 }
06 class B extends A {
07   B(int x) {System.out.print("A");}
08   B() {System.out.print("B");}
09   public void finalize() {System.out.print("C");}
10 }
11 public class Prova {
12   public static void main(String args[]) {
13     A a=new B(3);
14     a = null;
15     System.gc();
16     System.runFinalization();
17   }
18 }
```

Commentato [mr6]: YAC

Test 7

```
01 package uno;
02 public class A {
03   int x=10;
04   A(){int x=12; new B();}
05   public static void main(String args[]){
06     int x=11;
07     new A();
08   }
09   class B{
10     B() {System.out.println(x);}
11   }
12 }
```

Commentato [mr7]:
Output: 10

Test 8

```
01 class A {
02   int x = 3;
03   class B {
04       int x = 4;
05       B(int x) { System.out.print(x); }
06   }
07   A(int x) { new B(x); System.out.print(x); }
08   A(int y) { new B(y); System.out.print(y-4); }
09   A() { new B(x); System.out.print(x); }
10   public static void main(String s[]) {
11     A a=new A(3);  }
12 }
```

Commentato [mr8]: Errore in compilazione alla riga 8 – duplicate method

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	Poichè Java usa sempre dynamic binding, esso usa sempre la heap e mai lo stack.
9.2	Il garbage collector di Java sospende l'esecuzione del programma finchè non ha finito di liberare la memoria.
9.3	Un costruttore non può mai essere protected.
9.4	Di default l'operatore = = e il metodo equals fanno la stessa cosa.
9.5	Un oggetto ed un suo clone sono identici.
9.6	Se A è padre di B la scrittura B a=new A(); genera errore a compile time
9.7	Se A è padre di B la scrittura A a=(A)(new B()); genera errore a runtime
9.8	Il main può accedere a qualunque variabile di istanza della classe in cui è contenuto.

Commentato [mr9]:
1-F
2-F
3-F
4-T
5-F
6-T
7-F
8-F