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	Riportare la sequenza di V e F →	

TEST 9	Riportare la sequenza di V e F →	

TEST 10	Riportare la sequenza di V e F →	

A

Test 1

Il seguente codice viene compilato abilitando le assertions.

```
00  class DemoAssert {
01    int x=5;
02    public void stampaInteroPositivo(int i) {
03      assert i >= 0 : stampaErroreAssert();
04      System.out.println(i);
05    }
06    public int stampaErroreAssert()  {
07      System.out.println("XX");
08      return -1;
09    }
10    public static void main(String args[]) {
11      int x=-10;
12      DemoAssert test = new DemoAssert();
13      test.stampaInteroPositivo(x);
14    }
15  }
```

Comment [mr1]: PER RISOLVERE QUESTO ESERCIZIO OCCORRE SAPERE COS'E' E COME SI COMPORTA LA "assert", DISCUSSA NELLA LEZIONE 4

Test 2

Il seguente codice viene compilato abilitando le assertions.

```
00  class DemoAssert {
01    int x=5;
02    public void stampaInteroPositivo(int i) {
03      assert i >= 0 : stampaErroreAssert();
04      System.out.println(i);
05    }
06    public int stampaErroreAssert()  {
07      System.out.println("XX");
08      return -1;
09    }
10    public static void main(String args[]) {
11      x=10;
12      DemoAssert test = new DemoAssert();
13      test.stampaInteroPositivo(x);
14    }
15  }
```

Comment [mr2]: PER RISOLVERE L?ESERCIZIO E' NECESSARIO AVER CAPITO COS'E' UN CONTESTO STATICO, DISCUSSO NELLA LEZIONE 13

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   }
```

Comment [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.

A

Test 4

```
00   #include <iostream.h>
01   void rimescola(int& k, int m, int* n) {
02     k = m; m = *n; *n = k; n[-1] = k;
03   }
04   int main(){
05     int vet[] = {5,4,3,2,1};
06     rimescola(vet[0],vet[2],vet+4);
07     for (int i=0;i<5;i++) cout<<vet[i];
08     return 0;
09   }
```

Comment [mr4]: La soluzione di questo esercizio dovrebbe essere nota fin da Programmazione 1. In particolare, la relazione tra arrays e puntatori è stata ampiamente discussa nella lezione 2

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=null;
08     try {
09       z = Class.forName("uno.B").newInstance();
10     } catch (Exception ex) { ex.printStackTrace();}
11     if (z instanceof uno.A) ((A) z).f(1);
12     if (z instanceof uno.B) ((B) z).f(2);
13   }
14 }
15 class B extends A{
16   void f(int k) {
17     System.out.print(k);
18   }
19 }
```

Comment [mr5]: Per risolvere questo esercizio è necessario evere chiaro il comportamento della intanceof (lezione 11) e il comportamento del binding dinamico (lezione 7, 8 ecc.)

Test 6

```
01 public class A {
02   public A() {System.out.print("1");}
03   public void finalize(){System.out.print("3");}
04   public static void main(String Args[]){
05      A x;
06      A z=new A();
07      A y=z;
08      z=null;
09      System.gc();
10      System.out.print("5");
11      y=null;
12      System.gc();
13   }
14 }
```

Comment [mr6]: Questo esercizio richiede di aver compreso cosa sono iriferimenti ad oggetti e come funzionano (concetto discusso ed esemplificato innumerevoli volte in aula).

A

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  }
```

Comment [mr7]: Per risolvere questo esercizio (forse il più difficile del compito) è necessario avere chiaro il meccanismo della visibilità delle variabili (scope) ripreso molte volte a lezione. In particolare, usando la "regola delle graffe" e guardando dove si trova la variabile x referenziata nel costruttore di B risulta chiaro qual'e' il suo valore.

Test 8– Riportare sul foglio delle risposte una sequenza ordinata indicando V per le affermazioni vere e F per quelle false (Es. VVF)

Container eredita da Component
Component è figlia di Container
JComponent è figlia di Component

Comment [mr8]: La gerarchia della tre principali classi di grafica è stata ampiamente discussa nella lezione 12

Comment [mr9]: VFF

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

Java non usa lo stack ma solo la heap
Java usa lo stack solo per le variabili statiche
Java non usa la heap ma solo lo stack

Comment [mr10]: La comprensione del meccanismo di heap e stack (discussi nella prima parte del corso) è fondamentale per avere un buon modello mentale di quanto accade in un programma.

Comment [mr11]: FFF

Test 10 – scrivere nel campo per l'output del test la sequenza risultante indicando V per le affermazioni vere e F per quelle false

Quando si scrive la clone si deve anche scrivere anche la equals
Quando si scrive la equals si deve anche scrivere anche la clone
Per le classi che non hanno una struttura profonda non serve scrivere la equals

Comment [mr12]: Clone ed equals sono state oggetto della lezione 11

Comment [mr13]: FFF