Test 1

```
1  public class Test {
2    public static void main(String[] args) {
3      A a = new A(); a.m();
4      B b = new C(); b.m1(1); b.m2(2);
5    }
6  }
7  interface I {
8    public int m1(int a);
9  }
10 interface J {
11   public int m2(int b);
12 }
13 interface K extends I,J {}
14 class A {
15   public String m() { return "this is A"; }
16 }
17 class B implements K {
18   int x = 500;
19   public int m1(int a) { return a * x; }
20   public int m2(int b) { return b + x; }
21 }
22 class C extends A,B {}
```

Test 2

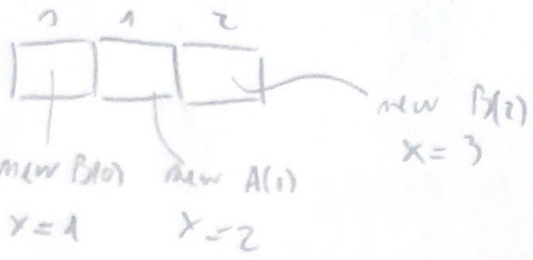
```
1  public class Test {
2    public static void main(String[] args) {
3      C obj = new C();
4      int a = C.m1();
5      int b = obj.m3();
6    }
7  }
8  class C {
9    static int x = 5;
10   int y = 3;
11   static public int m1() { return y + m2(); }
12   static private int m2() { return x; }
13   public int m3() { return m1(); }
14 }
```

Test 3

```
1  public class Test {
2    public static void main(String[] args) {
3      A a = new A();
4      System.out.println(a);
5    }
6  }
7  class A {
8    private B b;
9    public String toString() { return b.toString(); }
10 }
11 class B {
12   private int x;
13   public String toString() { return Integer.toString(x); }
14 }
```

Test 4

```
1  import java.util.*;
2  public class Test {
3    public static void main(String[] args) {
4      A[] a = new A[3];
5      for (int i = 0; i < a.length; i++) {
6        if (i%2 != 0) a[i]= new A(i);
7        else a[i]= new B(i);
8      }
9      ArrayList<A> l = new ArrayList<>(Arrays.asList(a));
10     for(A e: l) System.out.print(e.m(2) + "␣");
11   }
12 }
13 class A {
14   int x;
15   A(int x) { this.x = x + 1; }
16   public int m(int z) { return x + z; }
17 }
18 class B extends A {
19   B(int x) { super(x); }
20   public int m(int z) { return super.m(z) * 3; }
21 }
```

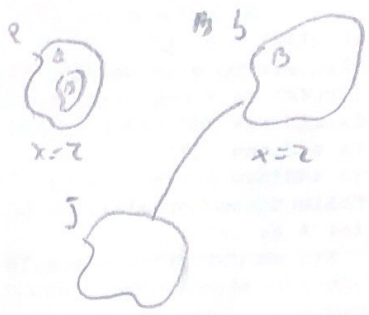


3 3

9 4 15

Test 5

```
1  public class Test {
2    public static void main(String[] args) {
3      A a = new B();
4      B b = new B();
5      J j = b;
6      if (a.equals(b))
7        System.out.println(j.m(2) + b.m("Java"));
8      else System.out.println(j.m(1));
9    }
10 }
11 interface I {
12   int m(int z);
13 }
14 interface J extends I {}
15
16 class A implements I {
17   int x = 2;
18   public int m(int z) { return x + z; }
19   public boolean equals(Object o) {
20     A obj = (A) o;
21     return x == obj.x;
22   }
23 }
24 class B extends A implements J {
25   public int m(String s) { return s.length(); }
26 }
```



4+4

Test 6

```
1  public class Test {
2    public static void main(String[] args) {
3      A obj = new B();
4      obj.m(new D());
5    }
6  }
7  class A {
8    void m(C c) { System.out.println("x"); }
9  }
10 class B extends A {
11   void m(C c) { System.out.println("y"); }
12   void m(D c) { System.out.println("z"); }
13 }
14 class C {}
15 class D extends C {}
```

y

Test 7

```
1  import java.util.*;
2
3  public class Test {
4    public static void main(String[] args) {
5      List<A> ls = new ArrayList<>();
6      ls.add(new A("Pordenone"));
7      ls.add(new A("Trento"));
8      ls.add(new A("Venezia"));
9      Collections.sort(ls);
10     for(A a: ls)
11       System.out.println(a.s.length());
12   }
13 }
14 class A implements Comparable<A> {
15   String s;
16   A(String s) { this.s = s; }
17   public int compareTo(A a) {
18     return (s.length() - a.s.length());
19   }
20 }
```

Pordenone
Venezia
Trento

9 7 6

Test 8

```
1  public class Test {
2    static String removeLast(String str) {
3      // ritorna la stringa ottenuta rimuovendo l'ultimo carattere dal parametro
4      return str.substring(0, str.length() - 1);
5    }
6    public static void main(String[] args) {
7      A obj1 = new A("Java");
8      A obj2 = new B("Python");
9      B obj3 = new B("Ruby");
10     I obj4 = new B("Scala");
11     System.out.println(obj1.m() + "␣" + obj2.m() + "␣" + obj3.m() + "␣" + obj4.m());
12   }
13 }
14 interface I {
15   public String m();
16 }
17 class A {
18   String s;
19   A(String s) { this.s = s; }
20   public String m() { return s; }
21 }
22 class B extends A implements I {
23   B(String s) { super(s); }
24   public String m() { return Test.removeLast(s); }
25 }
```

Jave Pytho Rub Scal

Test 9

A	Un tipo generico è una classe che contiene attributi di tipo `Object` nella propria definizione.	F
B	Nella gestione delle eccezioni in Java, un blocco `try` deve essere *sempre* seguito da almeno un blocco `catch`.	V
C	Siano dati i tipi riferimento `A` e `B`, e i tipi generici `C<A>` e `C<B>`. Se `B` è una sottoclasse di `A` allora `C<B>` è una sottoclasse di `C<A>`.	F
D	In Java, la parola chiave `final` può essere impiegata nella testata del metodo di una classe per impedirne la ridefinizione in una sottoclasse.	V
E	La coercizione di tipo (*cast*) presente in altri linguaggi quali il C è vietata in Java, in quanto potenziale fonte di errori. Pertanto, un'istruzione `A a = (B) b;` dà *sempre* un errore *in compilazione*, indipendentemente dalla definizione dei tipi `A` e `B`.	F
F	Quando una classe `C` implementa un'interfaccia `I`, deve fornire (o ereditare) l'implementazione di tutti i metodi di `I`, oppure dichiarare `C` come `abstract`; in caso contrario, viene generato un errore in compilazione.	V
G	Il tipo *statico* di un oggetto `p` è quello ad esso associato all'atto della dichiarazione, es., `Persona p = ...`. Il tipo *dinamico* di `p`, potenzialmente diverso da quello statico, è invece quello associato a tale oggetto in un certo punto dell'esecuzione del programma.	V
H	In Java non esiste ereditarietà multipla fra interfacce: un tipo interfaccia può estendere da un solo altro tipo interfaccia.	F