

Risorse » Ingegneria » Dip.IIMM » Informatica » Informatica 3 - parte A » test AA 2015/16 » test info3 21.11.16

## Anteprima di test

test info3 21.11.16

**Data: Mon Jan 9 16:12:07 2017 Punteggi massimi: 61**

### 1. Dynamic Binding Java (2) (15 Punti)

---

**Date le seguenti dichiarazioni:**

```
1 class Persona {
2     void setAge(int l) {
3         System.out.println("P");
4     }
5 }
6
7 class Studente extends Persona {
8     void setAge(int l) {
9         System.out.println("S");
10    }
11}
12
13class Anziano extends Persona {
14    void setAge(long l) {
15        System.out.println("A");
16    }
17}
18
19...
20Object op = new Persona();
21Persona pp = new Persona();
22Persona ps = new Studente();
23Persona pa = new Anziano();
24Anziano aa = new Anziano();
25int age = 30;
26long ageL = 200;
```

**Qual è l'input prodotto dalle seguenti istruzioni (errore se pensi ci sia un errore)?**

**op.setAge(age);**errore (2 Punti)

**pp.setAge(age);**P (1 Punto)

**ps.setAge(age);**S (1 Punto)

**pa.setAge(age);**P (4 Punti)

ERRORE  
P  
S  
P

pa.setAge(ageL);errore (1 Punto)  
 aa.setAge(age);P (4 Punti)  
 aa.setAge(ageL);A (2 Punti)

P  
A

## 2. overriding di equals (4) (9 Punti)

Data la seguente classe e gli oggetti definiti come segue

```
1 public class Complex {
2     int re;
3     int im;
4
5     public Complex (int re, int im) {
6         this.re = re;
7         this.im = im;
8     }
9
10    public boolean equals(Complex a) {
11        return this.re == a.re && this.im == a.im;
12    }
13}
14...
15Object o = new Complex (1,2);
16Complex p1 = new Complex (1,2);
17Complex p2 = new Complex (2,1);
18Complex p3 = new Complex (2,1);
```

Quale è l'output prodotto dalle seguenti istruzioni (metti errore se pensi ci sia un errore)?

```
System.out.println(o.equals(p1));false (2 Punti)
System.out.println(p1.equals(o));false (3 Punti)
System.out.println(o.equals(p2));false (2 Punti)
System.out.println(p2.equals(p3));true (1 Punto)
System.out.println(p2.equals(p2));true (1 Punto)
```

FALSE  
FALSE  
FALSE  
TRUE  
TRUE

## 3. passaggio parametri (1 Punto)

```
int foo(int x) { ... }
```

la variabile x viene passata per

- ☒ valore (1 Punto)  
☐ riferimento (0 Punti)

#### 4. Overriding/Overloading (1) (4 Punti)

Dato il seguente codice

```
1 class Value {}
2 class SmallValue extends Value {}
3
4 class Computer {
5     Value getVal() {
6         return new Value();
7     }
8 }
9
10 class Notebook extends Computer {
11     SmallValue getVal() {
12         return new SmallValue();
13     }
14 }
```

Quali di queste sono giuste

- ☒ Notebook fa overriding del metodo getVal di Computer
- ☐ Notebook fa overloading del metodo getVal di Computer
- ☒ Notebook è una sottoclasse di Computer
- ☐ Notebook contiene un errore: non può definire getVal in questo modo!

*(Selezionato = 1 Punto, Non selezionato = 0 Punti)*

*(Selezionato = 0 Punti, Non selezionato = 1 Punto)*

*(Selezionato = 1 Punto, Non selezionato = 0 Punti)*

*(Selezionato = 0 Punti, Non selezionato = 1 Punto)*

#### 5. Ridefinizione di metodi con classi (3) MQ (3 Punti)

```
1 class Sportivo{
2     public Calciatore m(){return null;}
3 }
4 class Calciatore extends Sportivo {
5     public Calciatore m(){return null;}
6 }
7 class Pallavolista extends Sportivo {
8     private Calciatore m(){return null;}
9 }
10 class Golfista extends Sportivo {
11     public Sportivo m(){return null;}
12 }
```



Quali di questi metodi sono ridefiniti in modo sbagliato (errore in compilazione)?

- ☒ il metodo m di Pallavolista (*Selezionato = 1 Punto, Non selezionato = -1 Punto*)  
☒ il metodo m di Golfista (*Selezionato = 1 Punto, Non selezionato = -1 Punto*)  
☐ il metodo m di Calciatore (*Selezionato = -1 Punto, Non selezionato = 1 Punto*)

## 6. C++ virtual functions ed ereditarietà (2) (14 Punti)

Date le seguenti classi

```
1 class Z{
2 public:
3     virtual void m(){ cout << "Z" << endl;}
4 };
5
6 class ZPRI1: private Z{
7 };
8
9 class ZPRI2: private Z{
10public:
11     virtual void m(){ cout << "ZPRI2" << endl;}
12};
13
14class ZPUB1: public Z{
15};
16
17class ZPUB2: public Z{
18public:
19     virtual void m(){ cout << "ZPUB2" << endl;}
20};
```

Scrivi l'output delle seguenti coppie di istruzioni. ERR se pensi ci sia un errore.

Z a1; a1.m();Z (1 Punto)  
 ZPRI1 a2; a2.m();ERR (1 Punto)  
 ZPRI2 a3; a3.m();ZPRI2 (1 Punto)  
 ZPUB1 a4; a4.m();Z (1 Punto)  
 ZPUB2 a5; a5.m();ZPUB2 (1 Punto)  
 a1 = a2; a1.m();ERR (1 Punto)  
 a1 = a3; a1.m();ERR (1 Punto)  
 a1 = a4; a1.m();Z (1 Punto)

Z  
 ERR  
 ZPRI Δ  
 Z  
 ZPUBZ  
 ERR  
 ERR  
 Z

```

a1 = a5; a1.m();Z (1 Punto)
Z* p = &a1; p -> m();Z (1 Punto)
p = &a2;p -> m();ERR (1 Punto)
p = &a3;p -> m();ERR (1 Punto)
p = &a4;p -> m();Z (1 Punto)
p = &a5;p -> m();ZPUB2 (1 Punto)

```

2  
2  
ERR  
ERR  
2  
2PUB2

## 7. C++ virtual functions ed ereditarietà - calls (2) (15 Punti)

Date le seguenti classi e le funzioni definite sotto.

```

1 #include <iostream>
2 using namespace std;
3 class veicolo {
4 private:
5     int pri() {         return 1;         }
6 public:
7     int pub() {         return 2;         }
8
9     virtual int vpub() {         return 3;         }
10};
11
12class camper: private veicolo {
13public:
14     int vpub() {         return 5;         }
15};
16
17class automobile: public veicolo {
18private:
19     int pri() {         return 6;         }
20public:
21     int vpub() {         return 7;         }
22};
23void f_camper(camper c) {
24     cout << c.pub();
25     cout << c.pri();
26     cout << c.vpub() << endl;
27}
28void f_p_camper(camper* c) {
29     cout << c->pub();
30     cout << c->pri();
31     cout << c->vpub() << endl;
32}
33void f_r_camper(camper& c) {
34     cout << c.pub();
35     cout << c.pri();
36     cout << c.vpub() << endl;

```

37}

Scrivi l'output delle seguenti istruzioni. Se una funzione chiamata f\_\* contiene un errore, ignora solo la riga della f\_\* che contiene l'errore. Se una delle seguenti istruzioni è sbagliata (anche se f\_\* chiamata fosse corretta), scrivi ERR.

```
int main() {  
    veicolo v;  
    camper c;  
    automobile a;  
  
    f_camper(v);ERR (1 Punto)  
    f_camper(c);5 (3 Punti)  
    f_camper(a);ERR (1 Punto)  
  
    //  
    f_p_camper(&v);ERR (1 Punto)  
    f_p_camper(&c);5 (3 Punti)  
    f_p_camper(&a);ERR (1 Punto)  
  
    //  
    f_r_camper(v);ERR (3 Punti)  
    f_r_camper(c);5 (1 Punto)  
    f_r_camper(a);ERR (1 Punto)  
}
```

ERR  
5  
ERR  
  
ERR  
5  
ERR  
  
ERR  
5  
ERR