

Anteprima di test

parte teorica aprile 16

Data: Mon Jan 9 16:13:47 2017 Punteggi massimi: 34

1. overriding di equals (1) (5 Punti)

Data la seguente classe e gli oggetti definiti come segue

```
1 public class A {
2     String name;
3
4     public A(String s) {
5         name = s;
6     }
7
8     public boolean equals(A a) {
9         return this.name.equals(a.name);
10    }
11}
12
13String pippo = "pippo";
14Object o = new A(pippo);
15A a1 = new A("pippo");
16A a2 = new A(pippo);
```

Quanto valgono (metti errore se pensi ci sia un errore)?

```
pippo.equals(o) false (1 Punto)
o.equals(o) true (1 Punto)
o.equals(a1) false (1 Punto)
o.equals(a2) false (1 Punto)
a1.equals(a2) true (1 Punto)
```

FALSE
TRUE
FALSE
FALSE
TRUE

2. Dynamic Binding Java (1) (4 Punti)

Date le seguenti dichiarazioni:

```
1 class Elaboratore {
2     void setCPU(int l) {
3         System.out.println("E");
4     }
5 }
```

```

5 }
6
7 class Phone extends Elaboratore {
8     void setCPU(int l) {
9         System.out.println("P");
10    }
11}
12
13class Computer extends Elaboratore {
14    void setCPU(short l) {
15        System.out.println("C");
16    }
17}
18
19...
20Object oe = new Elaboratore ();
21Elaboratore ee = new Elaboratore ();
22Elaboratore ep = new Phone ();
23Elaboratore ec = new Computer ();
24short myfreq = 30;

```

Quale è l' input prodotto dalle sequenti istruzioni (errore se pensi ci sia un errore)?

oe.setCPU(myfreq) errore (1 Punto)

ee.setCPU(myfreq) E (1 Punto)

ep.setCPU(myfreq) P (1 Punto)

ec.setCPU(myfreq) E (1 Punto)

ERRORE
illegibile

3. passaggio parametri (1 Punto)

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

la variabile x viene passata per

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

4. Return result address (1 Punto)

Che cos'è il return result-address?

- ☒ [] un campo contenente l'indirizzo dove salvare il risultato della funzione

(1 Punto)

- ☒ un campo contenente l'indirizzo della funzione chiamata "return" **(-1 Punti)**
- ☐ un campo contenente l'indirizzo della prima istruzione da eseguire quando la funzione termina **(-1 Punti)**
- ☐ un campo contenente l'istruzione return della funzione **(-1 Punti)**

5. Overriding/Overloading (0) (4 Punti)

Dato il seguente codice

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

Quali di queste affermazioni sono giuste?

- ☒ Phone fa overriding del metodo getVal di Elaboratore **(Selezionato = 1 Punto, Non selezionato = 0 Punti)**
- ☐ Phone contiene un errore: non può definire getVal in questo modo! **(Selezionato = 0 Punti, Non selezionato = 1 Punto)**
- ☒ Phone è una sottoclasse di Elaboratore **(Selezionato = 1 Punto, Non selezionato = 0 Punti)**
- ☐ Phone fa overloading del metodo getVal di Elaboratore **(Selezionato = 0 Punti, Non selezionato = 1 Punto)**

6. Ridefinizione di metodi con classi (1) (3 Punti)

```
1 class Veicolo{
2     public Auto m(){return null;}
3 }
4 class Auto extends Veicolo {
5     public Auto m(){return null;}
6 }
7 class Bicicletta extends Veicolo {
8     private Auto m(){return null;}
9 }
```

```
9 }
10 class Autobus extends Veicolo {
11     public Veicolo m(){return null;}
12 }
```



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

- ☒ il metodo m di Autobus (Selezionato = 1 Punto, Non selezionato = -1 Punto)
- ☐ il metodo m di Auto (Selezionato = -1 Punto, Non selezionato = 1 Punto)
- ☒ il metodo m di Bicicletta (Selezionato = 1 Punto, Non selezionato = -1 Punto)

7. Passaggio di array in C (5) (7 Punti)

Data la seguente funzione

```
1 void f(int a[]){
2     printf("%d\n", sizeof(a));
3     a = a +1;
4 }
5
6
7 int main(void) {
8     int p[] = {10,20,30};
9     printf("%d\n", sizeof(p));
10    f(p);
11    printf("%d\n", *p);
12    return EXIT_SUCCESS;
13 }
```

Qual'è l'output prodotto dalle seguenti istruzioni (in ordine di esecuzione)? Se pensi contenga un errore, scrivi errore

Assumi che un puntatore vale 4 byte come anche un intero (32 bit).

nel main:

printf("%d\n", sizeof(p)); 12 (1 Punto)

12

in f:

printf("%d\n", sizeof(a)) 4 (4 Punti)

4

nel main di nuovo

printf("%d\n", *p); 10 (2 Punti)

10

8. C++ virtual functions ed ereditarietà - calls (9 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_veicolo(veicolo v) {
24     cout << v.pub();
25     cout << v.pri();
26     cout << v.vpub() << endl;
27}
28void f_camper(camper c) {
29     cout << c.pub();
30     cout << c.pri();
31     cout << c.vpub() << endl;
32}
33void f_automobile(automobile a) {
34     cout << a.pub();
35     cout << a.pri();
36     cout << a.vpub() << endl;
37}
38void f_p_veicolo(veicolo* v) {
39     cout << v->pub();
40     cout << v->pri();
41     cout << v->vpub() << endl;
42}
43void f_p_camper(camper* c) {
44     cout << c->pub();
45     cout << c->pri();
```

```

46      cout << c->vpub() << endl;
47}
48void f_p_automobile(automobile* a) {
49      cout << a->pub();
50      cout << a->pri();
51      cout << a->vpub() << endl;
52}
53void f_r_veicolo(veicolo& v) {
54      cout << v.pub();
55      cout << v.pri();
56      cout << v.vpub() << endl;
57}
58void f_r_camper(camper& c) {
59      cout << c.pub();
60      cout << c.pri();
61      cout << c.vpub() << endl;
62}
63void f_r_automobile(automobile& a) {
64      cout << a.pub();
65      cout << a.pri();
66      cout << a.vpub() << endl;
67}

```

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_veicolo(v);23 (1 Punto)
    f_veicolo(c);ERR (1 Punto)
    f_veicolo(a);23 (1 Punto)

    //
    f_p_veicolo(&v);23 (1 Punto)
    f_p_veicolo(&c);ERR (1 Punto)
    f_p_veicolo(&a);27 (1 Punto)
    //

```

23
ERR
23

23
ERR
27

```
f_r_veicolo(v);23 (1 Punto)
f_r_veicolo(c);ERR (1 Punto)
f_r_veicolo(a);27 (1 Punto)
}
```

23
ERR
27