

## Anteprima di test

parte teorica 19.01.17

**Data: Thu Jan 19 15:06:03 2017 Punteggi massimi: 73**

### 1. Dynamic Binding Java (7) (12 Punti)

---

**Date le seguenti dichiarazioni:**

```
1 class Persona{}
2 class Studente extends Persona{}
3
4 class Scuola{
5     void iscrivi(Persona l) {
6         System.out.println("S");
7     }
8 }
9
10class Liceo extends Scuola{
11     void iscrivi(Persona l) {
12         System.out.println("L");
13     }
14}
15
16class University extends Scuola{
17     void iscrivi(Studente l) {
18         System.out.println("U");
19     }
20}
21
22...
23Persona p = new Studente();
24Studente s = new Studente();
25Scuola ss = new Scuola();
26Scuola sl = new Liceo();
27Scuola su = new University();
```

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

**ss.iscrivi(p);S (1 Punto)**

**sl.iscrivi(p);L (1 Punto)**

**su.iscrivi(p);S (5 Punti)**

S  
L  
U

ss.iscrivi(s);S (1 Punto)  
sl.iscrivi(s);L (1 Punto)  
su.iscrivi(s);S (3 Punti)

S  
L  
S

## 2. overriding di equals (6) MQ (14 Punti)

Data la seguente classe e gli oggetti definiti come segue

```
1 public class Persona{
2     String nome;
3     String cognome;
4
5     public Persona(String n, String c) {
6         this.nome = n;
7         this.cognome = c;
8     }
9
10    public boolean equals(Persona a) {
11        return this.nome.equals(a.nome) && this.cognome == a.cognome;
12    }
13}
14...
15String n = new String("Angelo");
16String c = new String("Gargantini");
17
18Object o = new Persona ("Angelo","Gargantini");
19Persona p1 = new Persona ("Angelo","Gargantini");
20Persona p2 = new Persona ("Gargantini","Angelo");
21Persona p3 = new Persona ("angelo","Gargantini");
22Persona p4 = new Persona ("Angelo","Gargantini");
23Persona p5 = new Persona (n,c);
```

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));false (1 Punto)
System.out.println(p2.equals(p2));true (1 Punto)
```

```
System.out.println(p1.equals(p4));true (3 Punti)
```

```
System.out.println(p1.equals(p5));false (1 Punto)
```

FALSE  
FALSE  
FALSE  
FALSE  
TRUE  
TRUE  
FALSE

System.out.println(p1.equals(new Persona(n,c)));false (1 Punto)

FALSE

### 3. C++ virtual destructors ed ereditarietà (1) (7 Punti)

Date le seguenti classi:

```
1 struct A {
2     A() { cout << "A+"; }
3     virtual ~A() { cout << "A-"; }
4 };
5
6 struct B {
7     B() { cout << "B+"; }
8     ~B() { cout << "B-"; }
9 };
10
11 struct C {
12     C() { cout << "C+"; }
13     ~C() { cout << "C-"; }
14 };
15
16 struct X : A, B, protected C {
17     X() { cout << "X+"; }
18     ~X() { cout << "X-"; }
19 };
```

Scrivi l'output delle seguenti istruzioni (ERR se pensi che ci sia un errore):

A\* a = new A; delete a;A+A- (1 Punto)

A\* a = new X; delete a;A+B+C+X+X-C-B-A- (1 Punto)

B\* b = new B; delete b;B+B- (1 Punto)

B\* b = new X; delete b;A+B+C+X+B- (1 Punto)

C\* c = new C; delete c;C+C- (1 Punto)

C\* c = new X; delete c;ERR (1 Punto)

X\* x = new X; delete x;A+B+C+X+X-C-B-A- (1 Punto)

A+ A-  
A+ B+ C+ X+ X- C- B- A-  
B+ B-  
A+ B+ C+ X+ B-  
C+ C-  
ERR  
A+ B+ C+ X+ X- C- B- A-

### 4. Ridefinizione di metodi con classi (0) (3 Punti)

```
1 class Veicolo{
2     public Veicolo get(){return null;}
```

```

3 }
4 class Auto extends Veicolo {
5     public Veicolo get(){return null;}
6 }
7 class Bicicletta extends Veicolo {
8     private Veicolo get(){return null;}
9 }
10 class Autobus extends Veicolo {
11     public Autobus get(){return null;}
12 }

```



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

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

## 5. Passaggio di array in C (8) MQ (7 Punti)

Data la seguente funzione

```

1 void f(int a[], int n) {
2     printf("%d\n", n);
3     printf("%d\n", sizeof(a));
4     a = a + 1;
5     *a = *a + 1;
6 }
7
8 int main(void) {
9     int p[] = { 10, 20, 30 };
10    f(p, sizeof(p));
11    printf("%d\n", *p );
12    return 0;
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).

in f:

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

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

in main:

12  
4

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

3

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

**Date le seguenti classi**

```
1 class X {
2 private:
3     void pri() { cout << "X" << endl; }
4 public:
5     virtual void pub() { cout << "X" << endl; }
6 };
7
8 class Y : private X {
9 public:
10     void pri() { cout << "Y" << endl; }
11
12};
13
14class Z : private X {
15public:
16     virtual void pub() { cout << "Z" << endl; }
17};
18
19class V : public X {
20public:
21     void pri() { cout << "V" << endl; }
22};
23
24class W : public X {
25public:
26     virtual void pub() { cout << "W" << endl; }
27};
```

**Scrivi l'output delle seguenti coppie di istruzioni. Se pensi ci sia un errore scrivi ERR e ignora l'istruzione (solo quella che dà errore).**

**X x; x.pri();ERR (1 Punto)**  
**Y y; y.pri();Y (1 Punto)**  
**Z z; z.pri();ERR (1 Punto)**  
**V v; v.pri();V (1 Punto)**  
**W w; w.pri();ERR (1 Punto)**  
**x = y; x.pub();ERR (1 Punto)**  
**x = z; x.pub();ERR (1 Punto)**  
**x = v; x.pub();X (1 Punto)**

ERR  
Y  
ERR  
V  
ERR  
ERR  
ERR  
X

```

x = w; x.pub();X (1 Punto)
X* p = &x; p->pub();X (1 Punto)
p = &y; p->pub();ERR (1 Punto)
p = &z; p->pub();ERR (1 Punto)
p = &v; p->pub();X (1 Punto)
p = &w; p->pub();W (1 Punto)

```

Handwritten notes: X, X, R, R, W, X, R, R, W

## 7. Java varargs (2) MR (5 Punti)

Selezionare, fra i seguenti metodi, quello/i corretto/i (cioè che NON dà/danno errore in compilazione):

- ☒ static void prova1(int i, String... s) { /\*\*/ } (Selezionato = 1 Punto, Non selezionato = 0 Punti)
- ☐ static void prova2(String ...s, int i) { /\*\*/ } (Selezionato = 0 Punti, Non selezionato = 1 Punto)
- ☐ static void prova3(int ...i, String ... s) { /\*\*/ } (Selezionato = 0 Punti, Non selezionato = 1 Punto)
- ☐ static void prova4(int i,int ...j, String ...s) { /\*\*/ } (Selezionato = 0 Punti, Non selezionato = 1 Punto)
- ☒ static void prova5(String s, int ...i){ /\*\*/ } (Selezionato = 1 Punto, Non selezionato = 0 Punti)

## 8. Java generics - tipi (4) MR (11 Punti)

```

1class Shape { /* ... */ }
2class Circle extends Shape { /* ... */ }
3class Rectangle extends Shape { /* ... */ }

```

Dire, per ognuno dei seguenti pezzi di programma, se è corretto o se presenta errori. Nel caso di errore, indicare la PRIMA linea di codice che contiene l'errore:

// -- Codice 1:

```

class Node<T> { /* ... */ }
Node<Circle> nc = new Node<Circle>();
Node<Shape> ns = nc;    errore istruzione 3 (1 Punto)

```

// -- Codice 2:

```

class Node<?> { /* ... */ }
Node<Circle> nc = new Node<Circle>();

```

Handwritten notes: CB, CM, EF, WOND, SHAPE, CIR | RECT

Handwritten note: S

Handwritten note: 1

**Node<Circle> ns = nc;    errore istruzione 1 (2 Punti)**

**// -- Codice 3:**

**class Node<T> { /\* ... \*/ }**

**Node<Circle> nc = new Node<Shape>();**

**Node<Circle> ns = nc;    errore istruzione 2 (1 Punto)**

2

**// -- Codice 4:**

**class Node<T> { /\* ... \*/ }**

**Node<Shape> nc = new Node<Circle>();**

**Node<Shape> ns = nc;    errore istruzione 2 (1 Punto)**

2

**// -- Codice 5:**

**class Node<T> { /\* ... \*/ }**

**Node<?> nc = new Node<?>();**

**Node<Shape> ns = nc; errore istruzione 2 (1 Punto)**

2

**// -- Codice 6:**

**class Node<T> { /\* ... \*/ }**

**Node<Circle> nc = new Node<>();    corretto (1 Punto)**

2

**// -- Codice 7:**

**class Node<PIPP0> { /\* ... \*/ }**

**new Node<>(); corretto (1 Punto)**

2

**// -- Codice 8:**

**Shape s = null;**

2

Circle c = s; ~~corretto (1 Punto)~~ errore linea2

// -- Codice 9:

class Node<K> { /\* ... \*/ }

Node<Circle> nc = new Node<>();

Node<Circle> ns = nc; **corretto (1 Punto)**

2

// -- Codice 10:

class MyList extends ArrayList<Rectangle> { /\* ... \*/ }

MyList nc = new MyList();

ArrayList<Rectangle> ns = nc; **corretto (1 Punto)**

2

CLASSE\_B < C\_F\_1 > a = b dove

CLASSE\_B < C\_F\_1 > b = new CLASSE\_B < C\_F\_1 >

CLASSE\_B ? > B

CLASSE\_B < ? > b = new CLASSE\_B < ? > ()

CLASSE\_B < C\_F\_1 > b = new CLASSE\_B < C\_M >

null