



## Temi Teoria

Informatica III A + B (12 crediti) (Università degli Studi di Bergamo)

## Domanda 1

Date le seguenti dichiarazioni:

```

1 class Computer {
2     void setCPU(int l) {
3         System.out.println("C");
4     }
5 }
6 class Notebook extends Computer {
7     void setCPU(int l) {
8         System.out.println("N");
9     }
10 }
11 class Tablet extends Computer {
12     void setCPU(short l) {
13         System.out.println("T");
14     }
15 }
16 ...
17 Object oc = new Computer();
18 Computer cc = new Computer();
19 Computer cn = new Notebook();
20 Computer ct = new Tablet();
21 short myfreq = 30;

```

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

oc.setCPU(myfreq)   
 cc.setCPU(myfreq)   
 cn.setCPU(myfreq)   
 ct.setCPU(myfreq)

## Domanda 2

La soluzione migliore è:

Data la seguente classe e gli oggetti definiti come segue

```

1 public class C {
2     String name;
3
4     public C(String s) {
5         name = s;
6     }
7
8     public boolean equals(C c) {
9         return this.name == c.name;
10    }
11 }
12
13 String p1 = new String("pippo");
14 String p2 = new String("pippo");
15 Object o = new C(p1);
16 C c1 = new C(p1);
17 C c2 = new C(p2);

```

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

System.out.println(p1.equals(o));   
 System.out.println(o.equals(o));   
 System.out.println(o.equals(new String("Pippo")));   
 System.out.println(o.equals(c1));   
 System.out.println(c1.equals(c1));   
 System.out.println(c1.equals(c2));

## Domanda 3

La soluzione migliore è:

Date le seguenti classi:

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

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

X\* x = new X; delete x; X+X-

X\* x = new S; delete x; X+Y+Z+S+X-

Y\* y = new Y; delete y; Y+Y-

Y\* y = new S; delete y; X+Y+Z+S+S-Z-Y-X-

Z\* z = new Z; delete z; Z+Z-

Z\* z = new S; delete z; ERR

S\* s = new S; delete s; X+Y+Z+S+S-Z-Y-X-

S puoi chiamare il costruttore di Z, lo eredita in modo private MA al suo interno lo puoi richiamare!

Ne basta uno della superclasse che sia virtual per distruggere tutti gli oggetti

errore perché con estensione private nascondo al mondo che S sia un sottotipo di Z

## Domanda 4

La soluzione migliore è:

Data la seguente classe:

```
class A {
```

```
protected Object foo(int x){return new Integer(x);}
```

```
}
```

Cosa accade se in una sottoclasse B introduco un metodo m fatto nei seguenti modi? Scegli "errore" se ho un errore in compilazione "overload" se faccio overload o "override" se sto facendo override.

```
protected Object foo(int x){return null;} override
```

```
Object foo(int x){return null;} errore
```

```
private Object foo(int x){return null;} errore errore perchè la visibilità può solo aumentare
```

```
protected Object foo(float x){return null;} overload
```

```
protected String foo(int x){return null;} override string è sottotipo di object
```

## Domanda 5

2) non mettendo niente cambia la visibilità e diventa visibile solo agli elementi dello stesso package (protected invece permette di farlo vedere, oltre a tutti i componenti dello stesso package, anche a sottoclassi al di fuori del package). Quindi visibilità diminuisce

La soluzione migliore è:

```
1 ArrayList<Integer> list1 = new ArrayList<Integer>();
2 ArrayList<Integer> list2 = new ArrayList<Integer>();
3
4 compare(list1, list2);
```

List va bene perché è interfaccia/supertipo di ArrayList

Selezionare le definizioni corrette (cioè che non danno errore) del metodo "compare" fra le seguenti:

- ☒ void compare(List<? extends Integer> list1, List<?> list2) { }
- ☒ void compare(ArrayList<?> list1, ArrayList<?> list2) { }
- ☐ void compare(List<? implements Comparable<Integer>> list1, List<Integer> list2) { }
- ☐ void compare(LinkedList<Integer> list1, LinkedList<Integer> list2) { }
- ☒ void compare(ArrayList<? super Integer> list1, ArrayList<? super Integer> list2) { }
- ☒ void compare(List<?>... list) { }
- ☒ void compare(List<? extends Comparable<?>> list1, ArrayList<Integer> list2) { }
- ☒ void compare(ArrayList<Integer> list1, ArrayList<Integer> list2) { }
- ☒ void compare(ArrayList<? extends Number> list1, ArrayList<? extends Number> list2) { }
- ☐ void compare(List<String> list1, List<String> list2) { }
- ☐ void compare(ArrayList<Object> list1, ArrayList<Object> list2) { }
- ☐ void compare(List<> list1, List<> list2) { }

## Domanda 6

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

## Domanda 7

ESC + compressor x pu

7  
 6m  
 7 mil  
 7  
 7 RBZ  
 Erken  
 Ermen  
 7  
 7  
 7  
 Ermen  
 Ermen  
 7  
 7 RBZ

Z a1; a1.m();	Z
ZPRI1 a2; a2.m();	ERR
ZPRI2 a3; a3.m();	ZPRI2
ZPUB1 a4; a4.m();	Z
ZPUB2 a5; a5.m();	ZPUB2
a1 = a2; a1.m();	ERR
a1 = a3; a1.m();	ERR
a1 = a4; a1.m();	Z
a1 = a5; a1.m();	Z
Z* p = &a1; p -> m();	Z
p = &a2; p -> m();	ERR
p = &a3; p -> m();	ERR
p = &a4; p -> m();	Z
p = &a5; p -> m();	ZPUB2

Data la seguente funzione

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

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

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

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

```

1 ArrayList<Integer> li = new ArrayList<Integer>();
2
3 li.add(1); li.add(2); li.add(3);
4
5 stampaLista(li);

```

Dire, per ognuna delle seguenti implementazioni di "stampaLista" (NOTA: cambia solo il parametro in input), cosa stampano in output (se stampano correttamente la lista oppure se danno errore in compilazione):

```

void stampaLista(List<Integer> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3
void stampaLista(ArrayList<Object> list) { for (Object elem : list) System.out.print(elem + " "); } errore
void stampaLista(List<?> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3
void stampaLista(List<? extends Number> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3
void stampaLista(List<? super Number> list) { for (Object elem : list) System.out.print(elem + " "); } errore
void stampaLista(List<Number> list) { for (Object elem : list) System.out.print(elem + " "); } errore
void stampaLista(List<? extends Comparable<?>> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3
void stampaLista(List<? implements Serializable> list) { for (Object elem : list) System.out.print(elem + " "); } errore

```

mi aspetto List<Number>. Non ce covarianza.  
List<Integer> non è sottotipo di List<Number>

## Risultati del test 1 per Luiza Jully Bentivoglio

**Nome: Luiza Jully Bentivoglio**

**Data del test: 2017-02-09 10:38:38, Data di stampa: 2017-03-07 15:52:40**

**Hai Superato il test.**

### Risultati del test

Ordine	Titolo	Punteggi massimi	Punteggio raggiunto	Percentuale risolta
1	Java generics - wildcard (1) MR	8	8	100.00 %
2	Java varargs (2) MR	5	5	100.00 %
3	C++ virtual destructors ed ereditarietà (2)	7	7	100.00 %
4	Ridefinizione di un metodo Java (2) MR	9	3	33.33 %
5	Passaggio di array in C (3)	7	7	100.00 %
6	C++ virtual functions ed ereditarietà (8) - FG	14	8	57.14 %
7	overriding di equals (1)	5	5	100.00 %
8	Java generics - tipi (2) MR	1	1	100.00 %
9	Passaggio per riferimento in c++ (6) MQ	6	2	33.33 %
10	Dynamic Binding Java (4)	15	9	60.00 %
<b>Totale</b>		<b>77</b>	<b>55</b>	<b>71.43 %</b>

### Risultati delle domande in dettaglio

#### 1. Java generics - wildcard (1) MR

```
1ArrayList<Integer> li = new ArrayList<Integer>();  
2  
3li.add(1); li.add(2); li.add(3);  
4  
5stampaLista(li);
```

**Dire, per ognuna delle seguenti implementazioni di "stampaLista" (NOTA: cambia solo il parametro in input), cosa stampano**

in output (se stampano correttamente la lista oppure se danno errore in compilazione):

**void stampaLista(List<Integer> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3** ✓

**void stampaLista(ArrayList<Object> list) { for (Object elem : list) System.out.print(elem + " "); } errore** ✓

**void stampaLista(List<?> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3** ✓

**void stampaLista(List<? extends Number> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3** ✓

**void stampaLista(List<? super Number> list) { for (Object elem : list) System.out.print(elem + " "); } errore** ✓

**void stampaLista(List<Number> list) { for (Object elem : list) System.out.print(elem + " "); } errore** ✓

**void stampaLista(List<? extends Comparable<?>> list) { for (Object elem : list) System.out.print(elem + " "); } 1 2 3** ✓

**void stampaLista(List<? implements Serializable> list) { for (Object elem : list) System.out.print(elem + " "); } errore** ✓

## 2. Java varargs (2) MR

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

- ☒ **static void prova1(int i, String... s) { /\*\*/ }** ✓
- ☐ **static void prova2(String ...s, int i) { /\*\*/ }** ✓
- ☐ **static void prova3(int ...i, String ... s) { /\*\*/ }** ✓
- ☐ **static void prova4(int i,int ...j, String ...s) { /\*\*/ }** ✓
- ☒ **static void prova5(String s, int ...i){ /\*\*/ }** ✓

## 3. C++ virtual destructors ed ereditarietà (2)

Date le seguenti classi:

```

1 struct X {
2     X() { cout << "X+"; }
3     ~X() { cout << "X-"; }
4 };
5
6 struct Y {
7     Y() { cout << "Y+"; }
8     virtual ~Y() { cout << "Y-"; }
9 };
10

```



```

11struct Z {
12    Z() { cout << "Z+"; }
13    ~Z() { cout << "Z-"; }
14};
15
16struct S : X, Y, private Z {
17    S() { cout << "S+"; }
18    ~S() { cout << "S-"; }
19};

```

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

**X\* x = new X; delete x;X+X-** ✓

**X\* x = new S; delete x;X+Y+Z+S+X-** ✓

**Y\* y = new Y; delete y;Y+Y-** ✓

**Y\* y = new S; delete y;X+Y+Z+S+S-Z-Y-X-** ✓

**Z\* z = new Z; delete z;Z+Z-** ✓

**Z\* z = new S; delete z;ERR** ✓

**S\* s = new S; delete s;X+Y+Z+S+S-Z-Y-X-** ✓

#### 4. Ridefinizione di un metodo Java (2) MR

**Data la seguente classe:**

```
class A {
```

```
protected Object foo(int x){return new Integer(x);}

```

```
}

```

**Cosa accade se in una sottoclasse B introduco un metodo m fatto nei seguenti modi? Scegli "errore" se ho un errore in compilazione "overload" se faccio overload o "override" se sto facendo override.**

**errore** corrisponde a `protected String foo(int x){return null;}` ✗

**override** corrisponde a `protected Object foo(int x){return null;}` ✓

**overload** corrisponde a `protected Object foo(float x){return null;}` ✓

## 5. Passaggio di array in C (3) 0

Data la seguente funzione

```

1 void so(char a[10]){
2     printf("%d\n", sizeof(a));
3     a = a +1;
4 }
5
6
7 int main(void) {
8     char p[5] = "ciao";
9     printf("%d\n", sizeof(p)/sizeof(char));
10    so(p);
11    printf("%c\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.

nel main:

`printf("%d\n", sizeof(p)/sizeof(char));`5✓

in so:

`printf("%d\n", sizeof(a));`4✓ il puntatore vale comunque 4 anche se punta a char (=1)

nel main di nuovo

`printf("%d\n", *p);`c✓

## 6. C++ virtual functions ed ereditarietà (8) - FG 0

Date le seguenti classi

```

1 struct A {
2     virtual void m() { cout << "A" << endl; }
3 };
4
5 class B : A {
6 };
7
8 class C : A {
9     virtual void m() { cout << "C" << endl; }
10};
```

se non scrivo in modo esplicito come derivo:  
 se ho una classe-> eredito in modo private  
 se ho una struct-> eredito in modo public  
 per capire se class o struct devi guardare la classe derivata (B e C), non la classe madre

```

11
12struct D : A {
13};
14
15struct E : A {
16    virtual void m() { cout << "E" << endl; }
17};

```

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

A a; a.m();A ✓

B b; b.m();A ✗ ERR→B eredita in modo private da A dato che non ce scritto nulla e B è una classe

C c; c.m();A ✗ C

D d; d.m();A ✓

E e; e.m();E ✓

a = b; a.m();A ✗ ERR

a = c; a.m();A ✗ ERR

a = d; a.m();A ✓

a = e; a.m();A ✓

A\* p = &a; p->m();A ✓

p = &b; p->m();A ✗ entrambi ERR

p = &c; p->m();A ✗

p = &d; p->m();A ✓

p = &e; p->m();E ✓

## 7. overriding di equals (1)

**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 ✓
o.equals(o) true ✓
o.equals(a1) false ✓
o.equals(a2) false ✓
a1.equals(a2) true ✓
```

## 8. Java generics - tipi (2) MR

---

```
1class A {
2    public static void main(String[] args) {
3        List<String> la = new ArrayList<>();
4        System.out.println(la.size());
5    }
6}
```

**Il codice mostrato**

[0] ☐ dà errore in esecuzione

[0] ☐ dà errore in compilazione

[1] ☒ è corretto e stampa "0" ✓

[0] ☐ è corretto e stampa "1"

## 9. Passaggio per riferimento in c++ (6) MQ

---

**Data la seguente funzione**

```
1void foo(int& x, int y){
2    x = x + y;
3    y = x--;
4}
```

**Qual'è l'output prodotto dalle seguenti istruzioni? Se pensi conteng un errore, scrivi ERR**

```
int main() {
```

```

int a = 15;
int b = 25;
int& h = b;
foo(a,b);
cout << a << endl;40 ✗
cout << b << endl;25 ✓
cout << h << endl;25 ✓
foo(h,b);
cout << a << endl;40 ✗
cout << b << endl;50 ✗
cout << h << endl;50 ✗
return 0;
}

```

39  
25  
25

y=x-- -> esegue x=x+1 e poi lo assegna a y. La modifica su X rimane  
y=x-1 -> così non modifico la X

## 10. Dynamic Binding Java (4)

Date le seguenti dichiarazioni:

```

1 class Mobile{
2     void size(double x){
3         System.out.println("M");
4     }
5 }
6
7 class Armadio extends Mobile{
8     void size(int x){
9         System.out.println("A");
10    }
11}
12
13class Cassapanca extends Armadio{
14    void size(double x){
15        System.out.println("C");
16    }
17}
18
19...
20Object om = new Mobile();
21Mobile ma = new Armadio();
22Mobile mc = new Cassapanca();
23int i = 9;
24long l = 10;
25double d = 20;

```

long e double sono compatibili  
double può contenere un long

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

**om.size(i);errore** ✓

**ma.size(i);M** ✓

**ma.size(l);errore** ✗ M

**ma.size(d);M** ✓

**mc.size(i);M** ✗ C

**mc.size(l);errore** ✗ C

**mc.size(d);C** ✓

## 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 sequenti 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)**

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

## 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)
```

potrebbe essere true perchè JVM fa l'interning accorgendosi che è lo stesso oggetto allora avranno lo stesso riferimento della stringa cognome per i due oggetti

la differenza con p5 è che sto in modo esplicito dichiarando new String() -> quindi avro un nuovo e diverso riferimento. Per questo l'ultima è false mentre la penultima è true



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

### 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)**

### 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;} errore perche riduco la visibilità
9 }
10 class Autobus extends Veicolo {
11     public Autobus get(){return null;}
12 } Autobus sottotipo di veicolo quindi ok

```



**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:**

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

## **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)**

```
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)
```

## 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>();
```

**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)**

**// -- Codice 4:**

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

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

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

**// -- Codice 5:**

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

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

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

**// -- Codice 6:**

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

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

**// -- Codice 7:**

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

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

**// -- Codice 8:**

**Shape s = null;**

**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)**

// -- Codice 10:

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

**MyList nc = new MyList();**

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

## Anteprima di test

parte teorica 4 Febbraio 15

**Data: Thu Feb 19 11:31:51 2015 Punteggi massimi: 42**

### 1. Dynamic Binding Java (8 Punti)

**Date le seguenti dichiarazioni:**

```
1 class Computer {
2     void setCPU(int l) {
3         System.out.println("C");
4     }
5 }
6 class NoteBook extends Computer {
7     void setCPU(int l) {
8         System.out.println("N");
9     }
10}
11class Tablet extends Computer {
12    void setCPU(short l) {
13        System.out.println("T");
14    }
15}
16...
17Object oc = new Computer();
18Computer cc = new Computer();
19Computer cn = new NoteBook();
20Computer ct = new Tablet();
21short myfreq = 30;
```

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

**oc.setCPU(myfreq) errore (2 Punti)**

**cc.setCPU(myfreq) C (1 Punto)**

**cn.setCPU(myfreq) N (1 Punto)**

**ct.setCPU(myfreq) C (4 Punti)**

### 2. Passaggio di array in C (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)**

**in f:**

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

**nel main di nuovo**

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

### 3. C++ virtual functions ed ereditarietà (3) (14 Punti)

**Date le seguenti classi**

```
1 class A{
2 private:
3     void pri(){cout << "A" << endl;}
4 public:
5     virtual void pub(){ cout << "A" << endl;}
6 };
```

```

7
8 class APRI1: private A{
9 public:
10     void pri(){cout << "APRI1" << endl;}
11
12};
13
14class APRI2: private A{
15public:
16     virtual void pub(){ cout << "APRI2" << endl;}
17};
18
19class APUB1: public A{
20public:
21     void pri(){cout << "APUB1" << endl;}
22};
23
24class APUB2: public A{
25public:
26     virtual void pub(){ cout << "APUB2" << endl;}
27};

```

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

**A a1; a1.pri();errore (1 Punto)**  
**APRI1 a2; a2.pri();APRI1 (1 Punto)**  
**APRI2 a3; a3.pri();errore (1 Punto)**  
**APUB1 a4; a4.pri();APUB1 (1 Punto)**  
**APUB2 a5; a5.pri();errore (1 Punto)**  
**a1 = a2; a1.pub();errore (1 Punto)**  
**a1 = a3; a1.pub();errore (1 Punto)**  
**a1 = a4; a1.pub();A (1 Punto)**  
**a1 = a5; a1.pub();A (1 Punto)**  
**A\* p = &a1; p -> pub();A (1 Punto)**  
**p = &a2;p -> pub();errore (1 Punto)**  
**p = &a3;p -> pub();errore (1 Punto)**  
**p = &a4;p -> pub();A (1 Punto)**  
**p = &a5;p -> pub();APUB2 (1 Punto)**

#### 4. Overriding/Overloading (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
10class Phone extends Elaboratore {
11     SmallValue getVal() {
12         return new SmallValue();
13     }
14}

```

**Quali di queste affermazioni sono giuste?**

- |                                                                                            |                                                           |
|--------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| <input checked="" type="checkbox"/> Phone fa overriding del metodo getVal di Elaboratore   | <b>(Selezionato = 1 Punto, Non selezionato = 0 Punti)</b> |
| <input checked="" type="checkbox"/> Phone è una sottoclasse di Elaboratore                 | <b>(Selezionato = 1 Punto, Non selezionato = 0 Punti)</b> |
| <input type="checkbox"/> Phone contiene un errore: non può definire getVal in questo modo! | <b>(Selezionato = 0 Punti, Non selezionato = 1 Punto)</b> |
| <input type="checkbox"/> Phone fa overloading del metodo getVal di Elaboratore             | <b>(Selezionato = 0 Punti, Non selezionato = 1 Punto)</b> |

#### 5. 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)
```

### 6. overriding di equals (4 Punti)

---

Data la seguente classe e gli oggetti definiti come segue

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

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

```
System.out.println(o.equals(a1));false (1 Punto)
System.out.println(a1.equals(o));false (1 Punto)
System.out.println(a1.equals(a1));true (1 Punto)
System.out.println(a1.equals(a2));true (1 Punto)
```

## Anteprima di test

parte teorica 20 Gennaio 2015

**Data: Mon Jan 26 12:44:27 2015 Punteggi massimi: 38**

### 1. Overriding/Overloading (2) (3 Punti)

---

**Dato il seguente codice**

```
1class Elaboratore{
2  void setQuantity(int q){}
3}
4
5class Phone extends Elaboratore{
6  void setQuantity(long l){}
7}
```

**Quali di queste sono giuste**

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

### 2. overriding di equals (8 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 == a.name;
10  }
11}
12String p1 = new String("pipipo");
13String p2 = new String("pipipo");
14Object o = new A(p1);
15A a1 = new A(p1);
16A a2 = new A(p2);
```

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

```
System.out.println(p1.equals(o));false (1 Punto)
System.out.println(o.equals(o));true (1 Punto)
System.out.println(o.equals(new Integer(5)));false (1 Punto)
System.out.println(o.equals(a1));false (1 Punto)
System.out.println(a1.equals(a1));true (1 Punto)
System.out.println(a1.equals(a2));false (3 Punti)
```

### 3. Passaggio per riferimento in c++ (9 Punti)

---

**Data la seguente funzione**

```
1void copy(int& x, int& y) {
```

```

2      x = y;
3      x--;
4      y++;
5}

```

**Qual'è l'output prodotto dalle seguenti istruzioni? Se pensi conteng un errore, scrivi errore**

```

int main() {
int a = 5;
int b = 6;
int& h = a;

copy(a,b);
cout << a << endl;5 (1 Punto)
cout << b << endl;7 (1 Punto)
cout << h << endl;5 (1 Punto)
copy(h,b);
cout << a << endl;6 (1 Punto)
cout << b << endl;8 (1 Punto)

cout << h << endl;6 (1 Punto)

b=7;

cout << a << endl;6 (1 Punto)
cout << b << endl;7 (1 Punto)

cout << h << endl;6 (1 Punto)

return 0;
}

```

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

---

**Date le segenti dichiarazioni:**

```

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

# Anteprima di test

## parte teorica

Data: Fri Jul 4 13:22:51 2014 Punteggi massimi: 53

### 1. Dynamic Binding Java (8 Punti)

Date le seguenti dichiarazioni:

```
1 class Computer {
2     void setCPU(int l) {
3         System.out.println("C");
4     }
5 }
6 class NoteBook extends Computer {
7     void setCPU(int l) {
8         System.out.println("N");
9     }
10}
11class Tablet extends Computer {
12    void setCPU(short l) {
13        System.out.println("T");
14    }
15}
16...
17Object oc = new Computer();
18Computer cc = new Computer();
19Computer cn = new NoteBook();
20Computer ct = new Tablet();
21short myfreq = 30;
```

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

oc.setCPU(myfreq) errore (2 Punti)

cc.setCPU(myfreq) C (1 Punto)

cn.setCPU(myfreq) N (1 Punto)

ct.setCPU(myfreq) C (4 Punti)

### 2. overriding di equals (8 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 == a.name;
10    }
11}
12String p1 = new String("pippo");
13String p2 = new String("pippo");
14Object o = new A(p1);
15A a1 = new A(p1);
```

```
16A a2 = new A(p2);
```

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

```
System.out.println(p1.equals(o));false (1 Punto)
System.out.println(o.equals(o));true (1 Punto)
System.out.println(o.equals(new Integer(5)));false (1 Punto)
System.out.println(o.equals(a1));false (1 Punto)
System.out.println(a1.equals(a1));true (1 Punto)
System.out.println(a1.equals(a2));false (3 Punti)
```

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

Date le seguenti classi

```
1 class A{
2 private:
3     void pri(){cout << "A" << endl;}
4 public:
5     virtual void pub(){ cout << "A" << endl;}
6 };
7
8 class APRI1: private A{
9 public:
10     void pri(){cout << "APRI1" << endl;}
11
12};
13
14class APRI2: private A{
15public:
16     virtual void pub(){ cout << "APRI2" << endl;}
17};
18
19class APUB1: public A{
20public:
21     void pri(){cout << "APUB1" << endl;}
22};
23
24class APUB2: public A{
25public:
26     virtual void pub(){ cout << "APUB2" << endl;}
27};
```

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

```
A a1; a1.pri();errore (1 Punto)
APRI1 a2; a2.pri();APRI1 (1 Punto)
APRI2 a3; a3.pri();errore (1 Punto)
APUB1 a4; a4.pri();APUB1 (1 Punto)
APUB2 a5; a5.pri();errore (1 Punto)
a1 = a2; a1.pub();errore (1 Punto)
a1 = a3; a1.pub();errore (1 Punto)
a1 = a4; a1.pub();A (1 Punto)
a1 = a5; a1.pub();A (1 Punto)
A* p = &a1; p -> pub();A (1 Punto)
p = &a2;p -> pub();errore (1 Punto)
```

p = &a3;p -> pub();errore (1 Punto)  
p = &a4;p -> pub();A (1 Punto)  
p = &a5;p -> pub();APUB2 (1 Punto)

#### 4. Passaggio per riferimento in c++ (9 Punti)

Data la seguente funzione

```
1 void copy(int& x, int& y) {  
2     x = y;  
3     x--;  
4     y++;  
5 }
```

Qual'è l'output prodotto dalle seguenti istruzioni? Se pensi conteng un errore, scrivi errore

```
int main() {  
int a = 5;  
int b = 6;  
int& h = a;  
  
copy(a,b);  
cout << a << endl;5 (1 Punto)  
cout << b << endl;7 (1 Punto)  
cout << h << endl;5 (1 Punto)  
copy(h,b);  
cout << a << endl;6 (1 Punto)  
cout << b << endl;8 (1 Punto)  
  
cout << h << endl;6 (1 Punto)  
  
b=7;  
  
cout << a << endl;6 (1 Punto)  
cout << b << endl;7 (1 Punto)  
  
cout << h << endl;6 (1 Punto)  
  
return 0;  
}
```

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

Dato il seguente codice

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

Quali di queste sono giuste

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

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

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

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

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

#### 6. Overriding/Overloading (2) (3 Punti)

Dato il seguente codice

```
1 class Elaboratore{
2     void setQuantity(int q){}
3 }
4
5 class Phone extends Elaboratore{
6     void setQuantity(long l){}
7 }
```

Quali di queste sono giuste

- ☐ Phone contiene un errore: non può definire setQuantity in questo modo!
- ☒ Phone fa overloading del metodo setQuantity di Elaboratore
- ☐ Phone fa overriding del metodo setQuantity di Elaboratore

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

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

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

#### 7. Passaggio di array in C (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)



in f:

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

nel main di nuovo

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

## 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)
```

### 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)**

### **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 ridefiniti 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)**

**in f:**

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

**nel main di nuovo**

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

## 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)
    //
```

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



## 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;
```

quindi int<long<double

**Qual è l'input prodotto dalle sequenti 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)**

**pa.setAge(ageL);errore (1 Punto)** errore perchè long è piu grande di int e non posso scendere da una superclasse

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

**aa.setAge(ageL);A (2 Punti)**

se sono in una sottoclasse posso permettermi di non adattarmi e di salire alla classe superiore per trovare una firma più vicina possibile. Se parto dalla superclasse invece dmi devo adattare

## 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)
```

## 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 redefiniti 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{
10 public:
11     virtual void m(){ cout << "ZPRI2" << endl;}
12 };
13
14 class ZPUB1: public Z{
15 };
16
17 class ZPUB2: public Z{
18 public:
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)**

```
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)
```

## 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)  
    }
```

Errori perche C è una sottoclasse! Non puoi mandargli fratelli o Madri