Appello di Programmazione 2
16 dicembre 2013

Esercizio 1.
Si progetti una classe `Articolo` con tre variabili di istanza: descrizione, prezzo unitario e quantità. Si progetti una classe `Magazzino` con due variabili d'istanza: un nome di tipo String e una lista di articoli di tipo array di `Articolo`.
Si scrivano i metodi della classe `Magazzino` per calcolare:
- dato un parametro intero n, il numero di articoli la cui quantità è minore o uguale a n (ritorna un intero);
- il valore totale degli articoli del magazzino (ritorna un double);
- se la quantità di tutti gli articoli è diversa da 0 (ritorna un boolean).

Infine si progetti una classe di test che costruisce un magazzino con tre articoli, invoca tutti i metodi, e ne stampa i risultati.

Esercizio 2.
Si considerino l'interfaccia `Archiviabile` e la classe `Biblioteca`:

```
public interface Archiviabile {
     int numeroOggettiInArchivio();
     int numeroPosizioniLibere();
}

public class Biblioteca {
          private String[] descrizioneOggetto;
}
```

dove l'array `descrizioneOggetto` contiene la descrizione del libro oppure `null` se la posizione è libera.

Si modifichi la classe `Biblioteca` affinché implementi l'interfaccia `Archiviabile`, dove il primo metodo calcola il numero totale di libri presenti in biblioteca, il secondo metodo le posizioni libere. Si progetti una classe `BibliotecaPerBambini` sottoclasse di `Biblioteca` con, in aggiunta, una variabile d'istanza `etaConsigliata` (di tipo array di interi) che implementi l'interfaccia `Archiviabile`.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

     private int n;

     public A(int n) {
          this.n=n;
     }

     public int getN() {
          return n;
     }

     public void setN(int n) {
          this.n=n;
     }

     public double calcola(int m) {
          n= n + m;
          return n;
     }

     public double calcola(double m) {
          return n + m + 5;
     }
}
```

```
public class B extends A {

     public B(int n) {
          super(n);
     }

     public double calcola(int m) {
          setN(getN() * m);
          return getN();
     }
}
```

```
public class Appello1 {

     public static void main(String[] args) {

          int i = 2;
          int k = 3;
          B b = new B(i);
          A a = b;
          System.out.println(a.calcola(k));
          System.out.println(b.calcola(b.calcola(k)));
     }
}
```

Appello di Programmazione 2
20 gennaio 2014

Esercizio 1.
Un multinsieme è una generalizzazione del concetto di insieme che permette elementi ripetuti. Ad esempio [[1, 2, 2, 2, 5, 6, 6]] è un multinsieme dove il numero 2 è ripetuto tre volte.

Si progetti una classe `Multinsieme` con una variabile d'istanza di tipo array di interi che conterrà gli elementi del multinsieme.
Si scrivano un costruttore ed i metodi della classe `Multinsieme` per:
- inserire un nuovo elemento (non è necessario mantenere l'ordinamento tra gli elementi);
- calcolare il numero di elementi contenuti nel multinsieme, tenendo conto delle ripetizioni (esempio: il multinsieme [[1, 2, 2, 2, 5, 6, 6]] contiene 7 elementi);
- eliminare una occorrenza di un elemento: il metodo deve ritornare `true` se è stato possibile eliminare l'elemento, `false` altrimenti (esempio: eliminando 2 da [[1, 2, 2, 2, 5, 6, 6]] otteniamo [[1, 2, 2, 5, 6, 6]] ed il metodo ritorna `true`);
- dato un numero "n", calcolare il numero di occorrenze di "n" (esempio: le occorrenze di 2 in [[1, 2, 2, 2, 5, 6, 6]] sono tre).

Infine si progetti una classe di test che costruisce un multinsieme, invoca tutti i metodi, e ne stampa i risultati.

Esercizio 2.
Data l'interfaccia:

```
public interface FormaGeometrica {

      int numeroLati();
      double calcolaPerimetro();
}
```

si progettino due classi Rettangolo (con due varibili di istanza) e Quadrato che implementino l'interfaccia `FormaGeometrica`, in cui Quadrato è una sottoclasse di Rettangolo.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

     private int n;

     public A() {
          n=0;
     }

     public int calcola(int a) {
          return n+a;
     }
}
```

```
public class B extends A {

     public B() {
          super();
     }

     public int calcola(int a) {
          return super.calcola(a)+1;
     }
}
```

```
public class Test {

     public static void main(String[] args) {
          B b = new B();
          A a = b;

          System.out.println(b.calcola(10));
          System.out.println(a.calcola(10));
     }

}
```

Appello di Programmazione 2
20 maggio 2014

Esercizio 1.
Un numero razionale (o frazione) è un numero che può essere espresso come frazione di due numeri interi. Ad esempio 3/4 e 7/100 sono numeri razionali.

Si progetti una classe `Razionale` con due variabili d'istanza intere per rappresentare il numeratore ed il denominatore del numero razionale. (Nota: non è necessario che i numeri razionali siano ridotti, ad esempio 1/2 e 2/4 sono ugualmente accettabili).

Si progettino un costruttore ed i seguenti metodi della classe `Razionale`:
- `int getNumeratore()` : restituisce il numeratore;
- `int getDenominatore()` : restituisce il denominatore;
- `boolean maggioreDi(int n)`: restituisce `true` se il numero razionale è maggiore di `n`, `false` altrimenti;
- `Razionale somma(Razionale r)`: restituisce un nuovo numero razionale ottenuto sommando il numero `r`.

Si progetti una classe di test che contenga i seguenti metodi statici:
- `Razionale[] serie(int n)`: restituisce una array di numeri razionali i cui elementi sono 1/n, 2/n, 3/n,... n/n;
- `Razionale sommatoria(Razionale[] a)`: calcola la somma di tutti i numeri razionali presenti nell'array `a`.

Infine si scriva il metodo main nel quale si calcoli la sommatoria 1/100 + 2/100 + 3/100 + 4/100+ … + 100/100 e si dica se è maggiore di 2 sfruttando i metodi statici sopra esposti.

Esercizio 2.
Si progettino una nuova classe `Razionale2,` sottoclasse di `Razionale,` ed una eccezione `EccezioneRazionale`.

Si progetti un costruttore della classe `Razionale2` affinché restituisca l'eccezione `EccezioneRazionale` se si cerca di costruire un numero razionale il cui denominatore è zero.

Si progetti un metodo per calcolare la divisione tra numeri razionali:
`Razionale2 divisione(Razionale2 r)`: effettua la divisione del numerro razionale ed `r`, e solleva l'eccezione `EccezioneRazionale` se `r` è zero.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {
```

```
	private int n=1;

	public int calcola(int a) {
		return n+a;
	}
}


public class B extends A {

	public B() {
		super();
	}

	public int calcola(int a) {
		return super.calcola(a)+1;
	}
}


public class Test {

	public static void main(String[] args) {
		A[] a = new A[2];
		a[0] = new A();
		a[1] = new B();

		System.out.println(a[0].calcola(10));
		System.out.println(a[1].calcola(10));
	}
 }
```

Appello di Programmazione 2
9 giugno 2014

Esercizio 1.
Si progetti una classe `InsiemeLimitato` che rappresenti un insieme di al più `n` elementi di tipo intero. La classe `InsiemeLimitato` utilizza un array di interi di lunghezza `n` per memorizzare gli elementi dell'insieme ed una variabile `numElementi` per tenere traccia del numero di elementi contenuti nell'insieme.

Si progetti un costruttore con un parametro `n` che indica il numero massimo di elementi che l'insieme può contenere.

Si progettino i seguenti metodi della classe `InsiemeLimitato`:
- `getNumElementi()`: restituisce il numero di elementi contenuti nell'insieme;
- `getSize()`: restituisce il numero massimo di elementi che l'insieme può contenere;
- `boolean contenuto(int e)`: restituisce `true` se l'elemento `e` è contenuto nell'insieme, `false` altrimenti;
- `boolean vuoto()`: restituisce `true` se l'insieme è vuoto, `false` altrimenti;
- `boolean pieno()`: restituisce `true` se l'insieme è pieno, `false` altrimenti;

Si progetti una classe di test che contenga i seguenti metodi statici:

- `boolean confronta(InsiemeLimitato i, InsiemeLimitato j)`: restituisce `true` se gli insiemi `i` e `j` hanno gli stessi elementi, `false` altrimenti;

- `InsiemeLimitato differenza(InsiemeLimitato i, InsiemeLimitato j)`: restituisce l'insieme differenza formato da tutti gli elementi che appartengono ad `i` e non appartengono a `j`.

Infine si scriva il metodo main nel quale si creano due insiemi e si chiamano tutti i metodi.

Esercizio 2.
Si progettino una eccezione `EccezioneInsiemeLimitato` e due metodi `inserisci` e `max` da aggiungere alla classe `InsiemeLimitato`:

- `void inserisci(int e)`: inserisce l'elemento `e` nell'insieme. Se l'insieme è pieno solleva l'eccezione `EccezioneInsiemeLimitato`. (Nota: ovviamente se un elemento appare già nell'insieme non deve essere inserito nuovamente).

- `int max()`: restituisce il massimo dell'insieme. Se l'insieme è vuoto solleva l'ecezione `EccezioneInsiemeLimitato`.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

      private int n;
```

```
	public A(int n) {
		this.n=n;
	}

	public void raddoppia() {
		n=n*2;
	}

	public int getN() {
		return n;
	}
}

public class B extends A{

	public B(int n) {
		super(n);
	}

	public void raddoppia() {
		super.raddoppia();
		super.raddoppia();
	}
}

public class Test {

	public static void main(String[] args) {
		A[] vettore = new A[3];
		vettore[0] = new A(1);
		vettore[1] = new B(1);
		vettore[2] = vettore[0];

		for(int i=0;i<vettore.length;i++) {
			A a = vettore[i];
			a.raddoppia();
			System.out.println(a.getN());
		}
	}
}
```

Appello di Programmazione 2
7 luglio 2014

Esercizio 1.
Si progettino le classi `Libro, LibroPerBambini` e `Biblioteca` per la gestione di una biblioteca.

La classe `Libro` ha 3 variabili di istanza `titolo, autore, inPrestito`. La variabile booleana `inPrestito` indica se il libro è attualmente in prestito. Si inseriscano nella classe `Libro` i segunti metodi:
- `String getDescrizione()`: restituisce una descrizione contenente il titolo e l'autore del libro;
- `boolean getInPrestito()`: restituisce `true` se il libro è in prestito, `false` altrimenti;
- `void setInPrestito(boolean stato)`: cambia lo stato del libro (in prestito oppure disponibile).

La classe `LibroPerBambini`, sottoclasse di `Libro`, deve contenere una variabile d'istanza `etaConsigliata` ed il metodo `getDescrizione()` deve restituire nella descrizione, oltre al titolo ed all'autore, anche l'età consigliata.

La classe `Biblioteca` gestisce la collezione di libri con una struttura dati di tipo `ArrayList`, che può contenere qualsiasi oggetto di tipo `Libro` e `LibroPerBambini`. La biblioteca implementa i seguenti metodi:
- `void addLibro(Libro l)`: aggiunge un nuovo libro alla biblioteca;
- `boolean inBiblioteca(Libro t)`: restituisce true se il libro è presente in biblioteca;
- `void prestato(Libro l)`: consente di dare in prestito un libro;
- `void restituito(Libro l)`: consente di restituire un libro dato in prestito;
- `int totInPrestito()`: conta il numero di libri in prestito;
- `boolean nessunPrestito()`: restituisce true se nella biblioteca non vi sono libri in prestito;
- `int perBambini(int n)`: conta quanti libri per bambini di età minore o uguale a n sono presenti in biblioteca.

Si scriva una classe di test che crea un libro, un libro per bambini, una biblioteca e chiama tutti i metodi.

Esercizio 2.
Si riscriva il metodo `void prestato(Libro l)` affinché sollevi l'eccezione EccezioneLibro nel caso in cui il prestito non sia possibile (il libro non è presente in biblioteca oppure è già in prestito).

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```java
public class A {
	private int n;

	public A(int n) {
		this.n=n;
	}

	public int getN() {
		return n;
	}

	public double calcola(int m) {
		return n+m;
	}

	public double calcola(double m) {;
		return n+m+1;
	}
}

public class B extends A {

	public B(int n) {
		super(n);
	}

	public double calcola(int m) {
		return getN() + m + 10;
	}
}

public class C extends B{

	public C(int n) {
		super(n);
	}

	public double calcola(int m) {
		return getN() + m + 100;
	}
}

public class Test {

	public static void main(String[] args) {
		C c = new C(2);
		B b = new B(5);
		A a = b;
		System.out.println(c.calcola(a.calcola(0)));
	}
}
```

Appello di Programmazione 2
11 settembre 2014

Esercizio 1.

Si progetti una classe `RegistroVendite` per registrare le vendite di un negozio (vettore di importi double).
Si progetti un costruttore in modo tale che un `RegistroVendite` possa registrare al massimo 10 vendite e che l'utente debba specificate quante vendite vuole inserire (max 10) e fornisca gli importi.
Si progettino inoltre i seguenti metodi per calcolare alcune statistiche sui dati contenuti nel vettore;

- `double totaleVendite()`: restituisce l'importo totale delle vendite
- `int venditeAlDiSopra(double valoreDiRiferimento)` restituisce il numero di vendite che hanno un valore superiore a valoreDiRiferimento, richiesto in input all'utente
- `double mediaVendite()`: restituisce il valore medio di tutte le vendite
- `double importoPiùAlto()`: restituisce il valore più alto fra tutte le vendite
- `RegistroVendite inserisci(double nuovoEl)`: inserisce una nuova vendita, se è possibile, altrimenti non fa niente.

Infine si progetti una classe di test che costruisce un `RegistroVendite`, invoca tutti i metodi, ed eventualmente ne stampa i risultati.

Esercizio 2.
Si progetti una nuova eccezione `EccezioneRegistroVendite`. Si modifichino il costruttore ed il metodo `inserisci` della classe `RegistroVendite` in modo da utilizzare l'eccezione. Il nuovo costruttore deve controllare che il numero di vendite da inserire sia al più 10 e che siano forniti gli importi per tutte e sole le vendite specificate, in caso contrario solleva l'eccezione `EccezioneRegistroVendite`. Il metodo `inserisci` effettua l'inserimento se possibile, altrimenti solleva l'eccezione `EccezioneRegistroVendite`.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

	private int n=1;

	public int calcola(int a) {
		return n+a;
	}
}


public class B extends A {

	public B() {
		super();
	}
```

```
	public int calcola(int a) {
		return super.calcola(a)+1;
	}
}


public class Test {

	public static void main(String[] args) {
		A[] a = new A[2];
		a[0] = new A();
		a[1] = new B();

		System.out.println(a[0].calcola(10));
		System.out.println(a[1].calcola(10));
	}
 }
```

Appello di Programmazione 2
23 gennaio 2015

Esercizio 1.
Si progettino le classi `Ristorante, Tavolo, Cameriere` e `Sommelier` per la gestione di un ristorante.

Le classi `Cameriere` e `Sommelier` implementano entrambe l'interfaccia `Personale` così definita:

```
public interface Personale {
     String getNome();
}
```

Ad ogni tavolo sono associati un cameriere ed un sommelier. La classe `Tavolo` ha 3 variabili di istanza: cameriere, sommelier e listaOrdini che tiene traccia degli ordini per il tavolo ed è implementata con un array di double di dimensione 100. La classe `Tavolo` ha inoltre i seguenti metodi:

- `void addOrdine(double costo)`: aggiunge un nuovo ordine alla listaOrdini del tavolo;
- `double calcolaTotaleTavolo()`: calcola il totale degli ordini presenti (o fatturato) nella listaOrdini del tavolo.

La classe `Ristorante` ha 3 variabili di istanza: `nome` (di tipo stringa), `listaTavoli` e `listaPersonale`, entrambe implementate con array list, ed i seguenti metodi:

- `calcolaTotale(Personale p)`: restituisce il totale fatturato dalla persona `p` calcolato sommando il totale del fatturato per tutti i tavoli a cui è associato;
- `double calcolaTotale()`: calcola il fatturato totale del ristorante, per tutti i tavoli;
- `Tavolo migliorTavolo()`: restituisce il tavolo con il più alto fatturato;
- `Personale migliore()`: restituisce il cameriere o sommelier con il più alto fatturato; il fatturato di un cameriere o di un sommelier è dato dalla somma dei fatturati dei tavoli a cui è associato;
- `int eccedenzaPersonale()`: calcola il numero di camerieri e sommelier che non hanno nessun tavolo associato;
- `int tavoliVuoti()`: calcola il numero di tavoli che non sono stati occupati (il cui fatturato è zero).

Si scriva una classe di test che crea un ristorante con due tavoli, un cameriere ed un sommelier. Si facciano due ordini per tavolo e si richiamino tutti i metodi del ristorante.

Esercizio 2.
Si progetti una nuova classe `SmartTavolo`, simile alle classe `Tavolo`, nella quale si tiene traccia non solo del costo delle cosumazioni effettuate, ma anche del loro nome. Ad esempio, ad un tavolo posso essere associate le seguenti consumazioni:
“pasta al pomodoro” 5.00 €
“sformato di verdure” 7.50 €
“tortino al cioccolato” 3.50 €
Si implementi un nuovo metodo che individui la consumazione più costosa ordinata nel tavolo.

Esercizio 3.

Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

   private int a=10;

   int calcola(int n) throws Exception {
      if (n==0) throw new Exception("ERRORE A");
      return n+a;
   }
}
```

```
public class B extends A {

   int calcola(int n) throws Exception {
      if (n==1)
         throw new Exception("ERRORE B");
      return 2*n+super.calcola(n+1);
   }
}
```

```
public class Test {

   public static void main(String[] args) {

      B b = new B();
      A a = b;
      try {
         System.out.println(a.calcola(0));
         System.out.println(a.calcola(1));
         System.out.println(b.calcola(0));
         System.out.println(b.calcola(1));
      }
      catch (Exception e) {
         System.out.println(e.getMessage());
      }
      System.out.print("END");
   }
}
```

Appello di Programmazione 2
11 giugno 2015

Esercizio 1.

Si realizzi una applicazione per gestire il rifornimento di un magazzino merci. Si progettino:

- una interfaccia Articolo con contenga i metodi:
  - String getCodice()
  - int getGiacenza()
  - int getMin()
  - double getPrezzoUnitario()
- una classe Prodotto che implementi l'interfaccia Articolo. Per ogni prodotto vogliamo memorizzare il codice (univoco), la giacenza (numero di pezzi in magazzino), la giacenza minima che deve essere sempre presente in magazzino ed il prezzo unitario.
- una classe Servizio che implementi l'interfaccia Articolo. Per i servizi, la giacenza ed il numero minimo di pezzi sono entrambi 0.
- una classe Magazzino con una struttura dati per memorizzare sia prodotti che servizi. La classe Magazzino deve implementare i seguenti metodi:
  - vuoto(): restituisce true se il magazzino è vuoto, false altrimenti;
  - valore(Articolo a): il valore di tutti i pezzi dell'articolo "a" presenti in magazzino (dato dal prezzo unitario per il numero di pezzi in giacenza);
  - valore(): il valore di tutti gli articoli in magazzino;
  - daAcquistare(): la lista degli articoli da acquistare (per cui la giacenza è inferiore al livello minimo)
  - costo(Articolo a): il costo da sostenere per acquistare l'articolo "a" affinché la giacenza raggiunga il minimo richiesto;
  - costo(): il costo da sostenere affinché tutti gli articoli raggiungano la giacenza minima.

Esercizio 2.

Si progetti una nuova eccezione MagazzinoException e si scriva il metodo controlloMagazzino() che restituisce true se per tutti gli articoli è presente la giacenza minima, altrimenti solleva l'eccezione MagazzinoException.

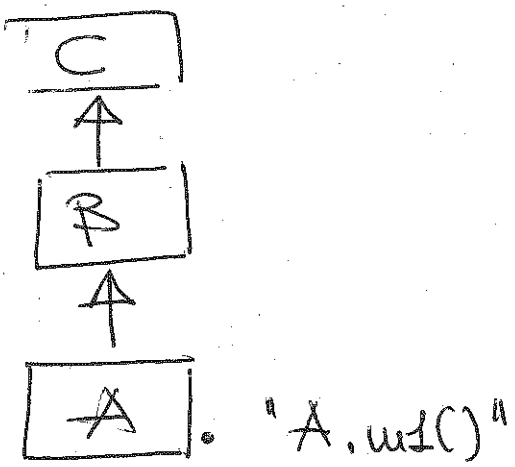
Esercizio 3.

Si dica cosa stampa il seguente programma motivando la risposta e indicando, per ogni chiamata ad un metodo, la lista delle firme candidate.

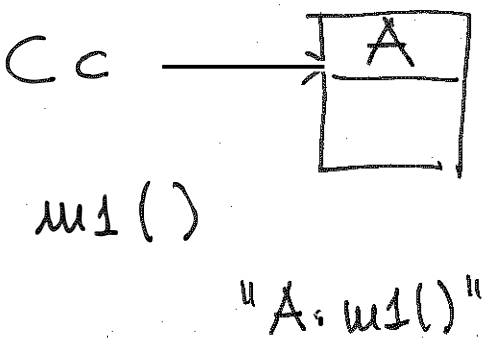
```java
public class A {
	public int m(A a, B b) {return 2; };
	public int m(B a, B b) {return 10; };
	public B   m(B c, A b) {return c; }
}

public class B extends A {
	public int m(A a, B b) {return 3; }
	public A   m(C c, B b) {return b; }
	public Object    m(C c, Object o) {return o; }
}

public class C extends B {
	public A   m(C c, A a) {return c; }
	public int m(B a, B b) {return 5; }
}
public class Test {
	public static void main(String[] args) {
		C gamma = new C();
		B beta = gamma;
		A alfa = gamma;
		System.out.println(alfa.m(beta, beta));
		System.out.println(gamma.m(beta.m(gamma, beta),beta));
	}
}
```

Quiz: Costrutti OO 1

1) Date le seguenti classi:



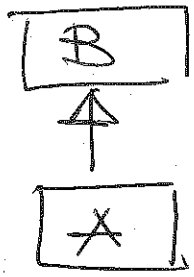

```
public class C {
public void m1() {
System.out.println("C.m1()");
}
}
public class B extends C{
public void m1() {
System.out.println("B.m1()");
}
}
public class A extends B{
public void m1() {
System.out.println("A.m1()");
}
}
public class P
{
        public static void test(C c){
           c.m1();
        }
}
public class MainClass{
public static void main(String args[]){
        C c=new A();
        P.test(c);
        }
}
```



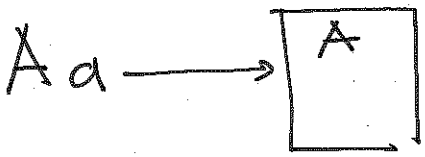
Qual è il risultato della compilazione ed esecuzione del programma ?

1. Errore di compilazione
2. Viene stampato "A.m1()"
3. Viene stampato "B.m1()"
4. Viene stampato "C.m1()"

2) Date le seguenti classi:




```
public class B
{
private int z;
        public B(int m){
           z=2*m;
        }
        public B(float m){
           z=3*(int) m;
        }
        public int getZ() {return z;}
}
public class A extends B
{
private int x;
private int y;
        public A(int i,int j){
           super((float)i/4);
           x=i;
           y=j;
        }
        public A(int i){
           this(i,10);
        }
        public void print(){
           System.out.println(x+y+getZ());
        }
}
public class MainClass{
public static void main(String args[]){
                A a=new A(5);
                a.print();
        }
}
```



Cosa viene stampato ?:

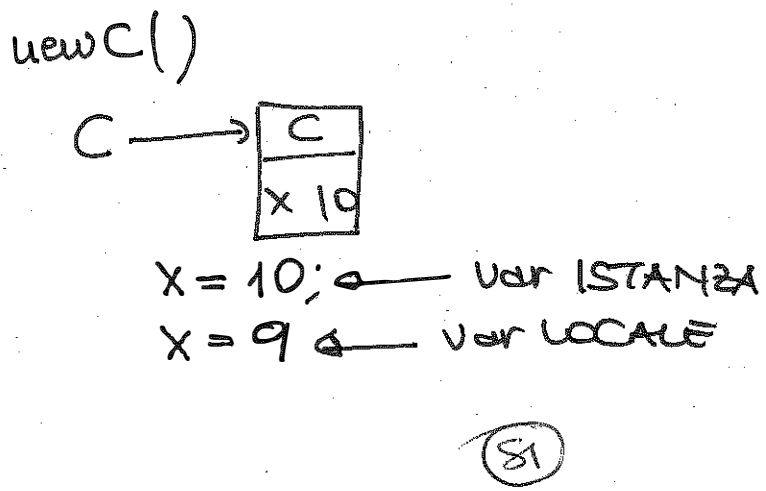
5. 17
6. 18
7. 15
8. 0

this (5,10)
super((float) 5/4)
z = 3
x = 5 y = 10
3+5+10 → 18

Quiz: Costrutti OO 1	Corso: "Programmazione orientata agli oggetti"	Docente: Andrea Bei

3) Date le seguenti classi:

```
public class C {
private int x;
public C(){
x=10;
}
public void m1(int h) {
System.out.print("x="+x+" ");
  {
  int x=3*h;
  System.out.println("x="+x);
 }
}
}
public class MainClass{
public static void main(String args[]){
      (new C()).m1(3);

          }
}
```

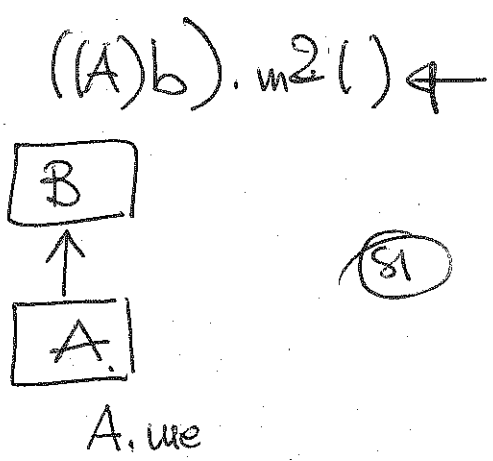


Qual è il risultato della compilazione ed esecuzione del programma ?

1. Errore di compilazione
2. stampa x=10 x=9
3. stampa x=10 x=10
4. stampa x=0 x=9

4)Date le seguenti classi:

```
public class B{
public void m1(){
System.out.println("B.m1()");
}
}
public class A extends B {
          public void m2(){
             System.out.println("A.m2()");
          }
}
public class MainClass{
public static void main(String args[]){
      B b=new A();
      b.m2();
          }
}
```



Qual è il risultato della compilazione ed esecuzione del programma ?

1. stampa: "A.m2()"

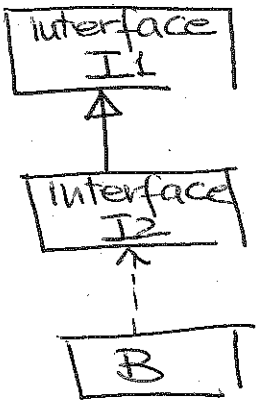
2. stampa: "B.m1()"
3. errore di compilazione
4. errore a tempo di esecuzione

5) Date le seguenti classi:

```
public abstract class B{
public abstract void m1();
public abstract void m2();
}
public class A extends B{
        public void m2(){
           System.out.println("A.m2()");
        }
}
public class MainClass{
public static void main(String args[]){
     B b=new A();
     b.m2();
}
}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. stampa: "A.m2()"
2. stampa: ""
3. errore a tempo di esecuzione
4. errore di compilazione



S1

Non implementa m1()
ERRORE a tempo di compilazione

6) Date le seguenti classi:

```
public interface Int1 {
        public void m1();
}
public interface Int2 extends Int1{
        public void m2();
}
public class B implements Int2{
        public void m2(){
        System.out.println("B.m2()");
        }
}
public class MainClass{
public static void main(String args[]){
     B b=new B();
     b.m2();
        }
}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. stampa: "B.m2()"
2. stampa: ""
3. Errore di compilazione
4. errore a tempo di esecuzione

7) Date le seguenti classi:

```
public class B
{
private int x;
public B(){
x=10;
}
public int getX(){
        return x;
        }
}
public class A extends B
{
        public void m2(){
         System.out.println("x= "+x);
        }
}
public class MainClass{
public static void main(String args[]){
    (new A()).m2();
        }
}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Errore di compilazione
2. stampa x=10
3. stampa x=0

8) Date le seguenti classi:

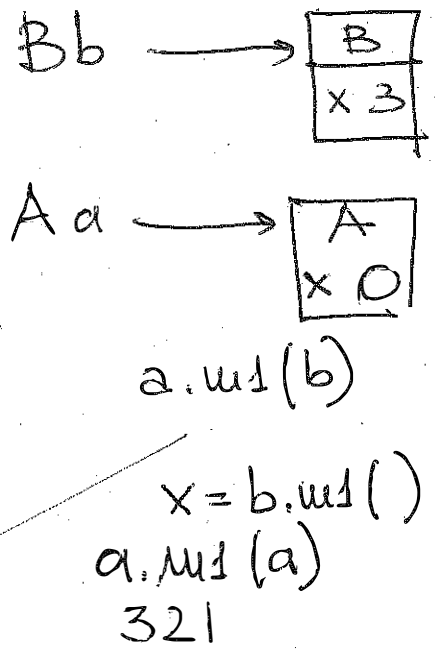
```
public class B
{
<??????> int x;
public B (){
x=10;
}
public int getX() {
        return x;
        }
}
public class A extends B
{
        public void m2() {
        x=0;
        }
}
public class C
{
        public void m2() {
        B b=new B() ;
        b.x=2;
        }
}
```

Come deve essere dichiarato l'attributo x di B affinchè la compilazione della classe A vada a buon fine e la compilazione della classe C dia errore ?

1. protected
2. public
3. private

Quiz: Costrutti OO 1	Corso: "Programmazione orientata agli oggetti"	Docente: Andrea Bei

9) Date le seguenti classi:




```
public class B {
int x;
public B(int i){
x=i;
}
public int m1(){
return x;
}
}
public class A
{
int x;
public void m1(B b) {
x=b.m1();
m1(this);
}
public void m1(A a) {
System.out.print(x--);
if (x>0) a.m1(this);
}
}
public class MainClass{
public static void main(String args[]){
        B b= new B(3);
                A a=new A();
                a.m1(b);
        }
}
```

Cosa viene stampato ?:

1. nulla
2. 3
3. 1
4. 321

10) Date le seguenti classi:

```
public class A{
private B b;
public A(B b){
this.b=b;
}
public void m1() {
b=new B();
}
public int getValue(){
return b.getValue();
}
}
public class B{
private int y ;
        public B(){
                y=10;
        }
        public int getValue(){
                return y;
        }
        public void setValue(int j){
                y=j;
        }
}
public class MainClass{
public static void main(String args[]){
        B b=new B();
        A a=new A(b);
        b.setValue(15);
        System.out.print(a.getValue());
        a.m1();
        System.out.print(a.getValue());
        }
}
```

Cosa viene stampato ?

1. nulla
2. 1010
3. 1015
4. 1510
5. 1515

11) Data la seguente classe

```
1: public class Q
2: {
3:    int maxElements;
4:
5:    void Q()
6:    {
7:       maxElements = 100;
8:       System.out.println(maxElements);
9:    }
10:
11:   Q(int i)
12:   {
13:      maxElements = i;
14:      System.out.println(maxElements);
15:   }
16:
17:   public static void main(String[] args)
18:   {
19:      Q a = new Q();
20:      Q b = new Q(999);
21:   }
22: }
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa 100 e 999.
2. Stampa 999 e 100.
3. Errore di compilazione a linea 3 (variabile maxElements non inizializzata).
4. Errore di compilazione a linea 19.

12) Date le seguenti classi:

```
1  class A
2  {
3     public int r=10;
4     void callme()
5     {
6        System.out.println("A");
7     }
8  }
9  class B extends A {
10
11    public void callme()
12    {
13       System.out.println("B");
14    }
15
16 }
17 class Q
18 {
19    public static void main(String args[])
20    {
21       B a = new B();
22       a.callme();
23       System.out.println(b.r);
24    }
25 }
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Errore di compilazione
2. Stampa B e 10
3. Stampa A e 10

Quiz: Costrutti OO 1	Corso: "Programmazione orientata agli oggetti"	Docente: Andrea Bei

13) Date le seguenti classi:

```
1  class Messaggio {
2  String text;
3  public Messaggio() { text ="Hello1"; }
4  }
5  class Super {
6   Messaggio msg;
7     public Super() { msg=new Messaggio(); }
8  }
9  class Ered extends Super
10 {
11    public static void main(String arg[])
12    {
13       Ered i=new Ered();
14       i.print();
15    }
16    public void print()
17    {
18       //Inserire il codice QUI !
19    }
20 }
```

Quale dei seguenti è il modo più semplice di stampare il valore della variabile text a linea 18 ?

1. System.out.println(msg.text);
2. System.out.println(super.msg.text);
3. System.out.println(Messaggio.text);
4. System.out.println(text);

14) Quali delle seguenti dichiarazioni di classe è corretta ?

```
1. public class Fred {
     public int x = 0;
     public Fred (int x) {
       this.x = x;
     }
   }

2. public class fred
     public int x = 0;
     public fred (int x) {
       this.x = x;
     }
   }

3. public class Fred extends MiaClasseBase, MiaAltraClasseBase{
     public int x = 0;
     public Fred (int xval) {
```

Non può ereditare da 2 classi

```
    x = xval;
  }
}
```

15) Date le seguenti classi:

```
1.  class Veicolo {
2.  public void guida() {
3.  System.out.println("Veicolo: guida");
4.  }
5.  }
6.  class Automobile extends Veicolo {
7.  public void guida() {
8.  System.out.println("Automobile: guida");
9.  }
10. }
11. public class Test {
12. public static void main (String args []) {
13. Veicolo v;
14. Automobile c;
15. v = new Veicolo();
16. c = new Automobile();
17. v.guida();
18. c.guida();
19. v = c;
20. v.guida();
21. }
22. }
```

Quali delle seguenti affermazioni è vera ?

1. Errore di compilazione su v= c;
2. Errore a tempo si esecuzione su v= c;
3. Stampa:

   Veicolo: guida
   Automobile: guida
   Automobile: guida

5. Stampa:

   Veicolo: guida
   Automobile: guida
   Veicolo: guida

16) Dove, in un costruttore, deve essere inserita l'istruzione super per chiamare il costruttore della superclasse ?

1. Ovunque
2. Deve essere la prima istruzione del costruttore

3. Deve essere l'ultima istruzione del costruttore
4. L'istruzione super non può essere inserita nel costruttore

17) Da quale istruzione nel codice seguente l'oggetto Impiegato("Roberto",48) può essere eliminato dal garbage collector ?

```
1. public class Test {
2. public static void main (String args []) {
3. Impiegato e = new Impiegato("Roberto", 48);
4. e.calcolaPaga();
5. System.out.println(e.stampaDettagli());
6. e = null;
7. e = new Impiegato("Federica", 36);
8. e.calcolaPaga();
9. System.out.println(e.stampaDettagli());
10. }
11. }
```

qui

1. Linea 10
2. Linea 11
3. Linea 6
4. Linea 8
5. Mai

18) Data la seguente classe:

```
public class A {int i1; void m1() {}}
```

Quali delle seguenti affermazioni è vera ?

1. La classe A eredita dalla classe Object. ✗
2. Il compilatore inserisce implicitamente un costruttore di default. ✗
3. Il costruttore di default accetta un parametro per ogni attributo di A. no
4. Il costruttore di default invoca il costruttore della superclasse ✗ (Object)

19) Date le seguenti classi:

```
class A {A(int i) {}}  // 1
class B extends A {}   // 2
```

B() { super(); }

Quali delle seguenti affermazioni sono vere ?

1. Il compilatore crea un costruttore di default per la classe A no
2. Il compilatore crea un costruttore di default per la classe B sì ma dà errore a tempo di COMPILAZIONE
3. Errore di compilazione a linea 1. no
4. Errore di compilazione a linea 2. sì

20) Quali delle seguenti affermazioni è vera ?

1. Il compilatore crea un costruttore di default solo se non esiste già un costruttore
2. Il costruttore di default ha zero argomenti.
3. Se la classe A ha una superclasse allora il costruttore di default di A invoca il costruttore a zero argomenti della superclasse.

21) Dato il seguente codice :

```
class Q {
 public static void main (String[] args) {
   private int x = 1;
   System.out.println(x);
}}
```

Quali delle seguenti affermazioni sono vere ?

1. Stampa: 1
2. Errore di esecuzione
3. Errore di compilazione
4. Nessuna

22) Data la seguente classe:

```
class Rosso {
 public int a;
 public static int b;
 public static void main (String[] in) {
   Rosso r1 = new Rosso(), r2 = new Rosso();
   r1.a++; r1.b++;
   System.out.print(r1.a+", "+r1.b+", "+r2.a+", "+r2.b);
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: 0, 0, 0, 0
2. Stampa: 0, 1, 1, 1
3. Stampa: 1, 1, 1, 0
4. Stampa: 1, 1, 0, 1
5. Stampa: 1, 1, 0, 0
6. Stampa: 1, 1, 1, 1
7. Compile-time error
8. Run-time error
9. None of the above

23) Quali delle seguenti affermazioni sono vere ?

1. Un metodo final non può essere sovrascritto
2. Tutti i metodi dichiarati in una classe final sono implicitamente final
3. Tutti i metodi dichiarati in una classe final devono essere esplicitamente dichiarati final altrimenti avviene un errore di compilazione

24) Data la seguente classe:

```
class Q {
  static int m1(int x) {return ++x ;}
  public static void main (String[] args) {
    int x = 1;
    int y = m1(x);
    System.out.println(x + "," + y);
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: 1,1
2. Stampa: 1,2
3. Stampa: 2,1
4. Stampa: 2,2
5. Errore di esecuzione
6. Errore di compilazione
7. Nessuna delle precedenti

25) Data la seguente classe:

```
class Q {
  private static int x=1;
  static void m1(int i) {x++; i++;}
  public static void main (String[] args) {
    int y=3; m1(y);
    System.out.println(x + "," + y);
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: 1,3
2. Stampa: 2,3
3. Stampa: 1,4
4. Stampa: 2,4
5. Errore di esecuzione
6. Errore di compilazione
7. Nessuna delle precedenti

26) Data la seguente classe:

```
class Q {
 private String name;
 public Q(String name) {this.name = name;}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
 public static void m1(Q r1, Q r2) {
  r1.setName("Uccello");
  r2 = r1;
 }
 public static void main (String[] args) {
  Q animale1 = new Q("Cane");
  Q animale2 = new Q("Gatto");
  m1(animale1,animale2);
  System.out.println(animale1.getName() + "," + animale2.getName());
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: Cane,Gatto
2. Stampa: Cane,Uccello
3. Stampa: Uccello,Gatto
4. Stampa: Uccello,Uccello
5. Errore di esecuzione
6. Errore di compilazione
7. Nessuna delle precedenti

27) Data la seguente classe:

```
class Q {
 private String name;
 public Q(String name) {this.name = name;}
 public void setName(String name) {this.name = name;}
 public String getName() {return name;}
 public static void m1(Q animale1, Q animale2) {
  animale1 = new Q("Pesce");
  animale2 = null;
 }
 public static void main (String[] args) {
  Q animale1 = new Q("Cane");
  Q animale2 = new Q("Gatto");
  m1(animale1,animale2);
  System.out.println(animale1.getName() + "," + animale2.getName());
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: Cane,Gatto
2. Stampa: Cane,Pesce
3. Stampa: Pesce,Gatto
4. Stampa: Pesce,Pesce
5. Errore di compilazione
6. Errore di esecuzione
7. Nessuna delle precedenti

28) Data la seguente classe:

```
class Q {
 static void m1(int[] i1, int[] i2) {
  int[] i3 = i1; i1 = i2; i2 = i3;
 }
 public static void main (String[] args) {
  int[] i1 = {1}, i2 = {3}; m1(i1, i2);
  System.out.print(i1[0] + "," + i2[0]);
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: 1,1
2. Stampa: 1,3
3. Stampa: 3,1
4. Stampa: 3,3
5. Errore di esecuzione
6. Errore di compilazione
7. Nessuna delle precedenti

29) Date le seguenti classi:

```
class A {
 void m1() {System.out.print("A.m1");}
}
class B extends A {
 void m1() {System.out.print("B.m1");}
 static void m1(String s) {System.out.print(s+",");}
}
class C {
 public static void main (String[] args) {
  B.m1("main");
  (new B()).m1();
}}
```

Qual è il risultato della compilazione ed esecuzione del programma ?

1. Stampa: main,B.m1
2. Errore di compilazione

3. Errore a Run-time
4. Nessuna delle precedenti

30) Quali delle seguenti affermazioni è vera ?

1. La relazione tra una classe e la superclasse è una relazione di composizione/aggregazione
2. La relazione tra una classe e la superclasse è una relazione di ereditarietà
3. La relazione tra una classe e un oggetto referenziato da un attributo della classe è una relazione di composizione/aggregazione
4. La relazione tra una classe e un oggetto referenziato da un attributo della classe è una relazione di ereditarietà

31) Date le seguenti classi:

```
class Zampa{}
abstract class Animale {
  public abstract void mangia() ;
  public abstract void dorme() ;
}
class Cane extends Animale {
  Zampa sinistraAnteriore;
  Zampa destraAnteriore;
  Zampa sinistraPosteriore;
  Zampa destraPosterire;
   Cane()
   {
   sinistraAnteriore= new Zampa();
   destraAnteriore=new Zampa();
   sinistraPosteriore= new Zampa();
   destraPosteriore=new Zampa();
   }
  public void mangia() {}
  public void dorme() {}
}
class Gatto extends Cane {
  public void disobbediente () {}
  public void siArrampicaSugliAlberi() {}
}
```

Quali delle seguenti affermazioni non è vera ?

1. Un Gatto eredita 4 istanze di Zampa dalla superclasse Cane
2. Un Gatto può mangiare e dormire
3. Un Gatto si arrampica sugli alberi
4. La relazione tra Cane e Animale è un esempio di uso approrpiato dell'ereditarietà
5. La relazione tra Gatto e Cane è un esempio di uso inappropriato dell'ereditarietà
6. Nessuna delle precedenti.

Soluzioni: Costrutti OO

Quesito	Soluzione	Commenti
1	2	Per il binding dinamico viene eseguita l'implementazione del metodo m1 della classe A
2	2	L'ordine di esecuzione è il seguente: • Invocazione del costruttore di A con un solo parametro (5) • Invocazione (tramite this) del costruttore di A con 2 parametri (5,10) • Invocazione del costruttore di B con il parametro di tipo float (5/4). Il cast a (int) rende 5/4 uguale a 1. Quindi z=3 • Esecuzione del metodo print 10+5+3
3	2	All'interno del metodo m1, per le regole sulla visibilità delle variabili, la prima istanza delle variabile x è l'attributo, la seconda è il valore della variabile dichiarata nel blocco interno al metodo m1 e definito dalle parentesi graffe.
4	3	L'errore di compilazione è dovuto al fatto che il tipo di dato B non definisce l'operazione m2. La variabile b punta però ad un oggetto il cui tipo di dato associato definisce l'operazione m2 (oggetto della classe A). Se si vuole invocare il metodo m2 si deve effettuare un cast e quindi sostituire all'istruzione b.m2();l'istruzione ((A) b).m2(); Quest'ultima istruzione non genera errori di compilazione.
5	4	La classe A deve implementare sia m1 che m2, altrimenti deve essere dichiarata astratta.
6	3	L'interfaccia Int2 ha i metodi m1 (ereditato) e m2. B dichiara di implementare Int2 quindi deve implementare sia m1 che m2 altrimenti si ha un errore di compilazione.
7	1	L'attributo x della classe A è di tipo private quindi non è visibile dalla classe B
8	1	Un attributo con modificatore di visibilità protected è visibile solo dalle sottoclassi.
9	4	Il metodo m1 della classe A è ricorsivo e chiama se stesso decrementando l'attributo x ad ogni chiamata fino a che x non diventa uguale a 0.
10	4	La sequenza delle operazioni è questa: • Viene creato un oggetto di tipo B (attributo y = 10) • L'oggetto di tipo B viene passato alla classe A • b.setValue(15) imposta y=15 • La prima stampa di a.getValue() restituisce 15 • a.m1() crea un altro oggetto di tipo B che assegna all'attributo b. Invocando di nuovo il costruttore di B ho nuovamente y=10 • La seconda stampa di a.getValue() restituisce 10
11	4	"void Q()" non è un costruttore perchè specifica il tipo ritornato (void) e nei costruttori questo non deve essere specificato. Dato che la classe Q ha già un costruttore "Q(int i)" non viene creato il costruttore di default. Quindi la classe non ha un costruttore a zero argomenti e quando viene invocato costruttore di questo tipo (riga 19) viene emesso un errore a tempo di compilazione.
12	1	La variabile b non è stata dichiarata

13	1	La variabile msg è visibile dalla classe Ered (friendly). Posso scrivere msg.text perché le classi sono tutte nello stesso package
14	1	La 2 non è corretta perché mancano le parentesi, la 3 non è corretta perché Fred eredita da due classi e in Java questo non è possibile,
15	3	• A riga 17 l'istruzione v.guida (); stampa: "Veicolo:guida" • A riga 18 l'istruzione c.guida(); stampa: "Automobile: guida" (nella classe Automobile viene fatto l'override del metodo guida) • A riga 19 l'istruzione v=c (possibile per il polimorfismo) fa si che il riferimento v punti allo stesso oggetto puntato dal riferimento c. Quindi v punta ad un oggetto di tipo Automobile. • A riga 20 l'istruzione v.guida(); provoca l'invocazione del metodo guida di Automobile() (binding dinamico) e quindi la stampa di "Automobile: guida"
16	2	Se non si inserisce come prima istruzione viene restituito un errore a tempo di compilazione.
17	3	A partire dalla linea 6 (*e=null*) il riferimento *e* non punta più all'oggetto considerato. Inoltre non esistendo, oltre ad *e*, nessun altro riferimento a tale oggetto questo può essere distrutto dal garbage collector che libera la memoria occupata.
18	1,2,4	• La classe A eredita implicitamente dalla classe Object. Questo vale in Java per tutte le classi che non ereditano esplicitamente da una classe. Infatti tutte le classi (anche le proprie) ereditano direttamente o indirettamente dalla classe Object. • Il compilatore inserisce un costruttore di default se la classe non ne definisce uno. • Il costruttore di default invoca il costruttore a zero argomenti della superclasse.
19	4	L'errore di compilazione a riga 2 deriva dal fatto che il compilatore inserisce un costruttore di default nella classe B. Tale costruttore invoca il costruttore a zero argomenti della superclasse. Dato che nella superclasse un costruttore di questo tipo non esiste, viene restituito un errore di compilazione a riga 2.
20	1,2,3	
21	3	Il modificatore di visibilità "private" può essere specificato per la dichiarazione degli attributi di una classe; non ha senso per le variabili locali di un metodo.
22	4	• r1.a vale 1 perché l'attributo a dell'oggetto r1 è stato incrementato dall'istruzione r1.a++ • r1.b vale 1 perché l'attributo statico b della classe Rosso è stato incrementato dall'istruzione r1.b++ • r2.a vale 0 perché l'attributo a dell'oggetto r1 non è stato ma inizializzato • r2.b vale 1 ed è lo stesso valore di r1.b dato che b è statico ed è un attributo della classe Rosso e non dei suoi oggetti. Quindi tutti gli oggetti della classe Rosso "vedono" lo stesso valore.
23	1,2	Una classe final non può essere ereditata e quindi i suoi metodi non possono essere sovrascritti ed in questo senso sono implicitamente final.
24	2	