Corso di laurea in Comunicazione digitale — Fondamenti di architettura e programmazione

AA 2008/09 — Prof. M.A. Alberti

V Appello - 15 settembre 2009

Cognome **Nome**

Matricola

1 (12 punti)

Date le classi:

```
public class SuperCls
{
  private int n;
  protected static int x;

  SuperCls(int i) {
    n = i;
    x = i+2;
  }

  public int metodoA(int i) {
    return i;}

  public void metodoB(int i) {
    System.out.println (i);
  }

  public void metodoC(int i){
    System.out.println (i*2);
    x = i;
  }

  public static void metodD(){ …
  }
}
```

```
public class SottoCls extends SuperCls
{
  public int n;
  private int x;

  SottoCls(int i) {
    n = i;
    x = i*2;
  }

  public int metodoA(int i, int y){
    return i*y;}

  public int metodoB(int i) {
    return (i+100);
  }

  public void metodoC(int i){
    System.out.println (i);
  }


 public void metodD(int i){ …
  }
}
```

a. Dire se ci sono errori che impediscono la compilazione in `SuperCls`, eventualmente corregerli:
Non ci sono errori

b. Dire se le definizioni dei membri sottoindicati della classe `SottoCls` costituiscono sovraccaricamento, sovrascrittura o errore (in questo caso specificare quando l'errore viene rilevato e da chi):

`metodoA` **sovraccaricamento**

`metodoB` **sovrascrittura**

`metodoC` **sovrascrittura**

`metodoD` non crea alcun problema ed e' semplicemente un metodo diverso con firma diversa, il primo metodo e' statico e quindi di classe e il secondo d'istanza. Pur essendo possibile

definire questi due metodi con lo stesso nome non lo si raccomanda per ragioni di leggibilita' del codice.

c. Dire se e' corretta la definizione del costruttore `SottoCls (int i)` **No. Manca l'invocazione al costruttore della superclasse `super (i)` ad esempio;**

2

Corretti gli eventuali errori, che potevano impedire la compilazione delle classi dell'esercizio precedente, dopo aver eseguito le due istruzione seguenti:

```
SuperCls p = new SuperCls (3);
SottoCls f = new SottoCls (5);
```

dire quali saranno i valori di ritorno o l'output di:

1. `p.n` **3**
2. `p.x` **5**
3. `f.n` **5**
4. `f.metodoA (f.n, x)` **35**
5. `p.x` **(attenzione!!)** **7**
6. `p.metodoC (p.n);` **6**
7. `f.metodoC (f.n);` **5**
8. `p.x` **3**
9. `f.metodoA (f.n, x)` **15**

---

3 (4 punti)

Scrivere l'espressione booleana che controlla che la variabile numero dichiarata di tipo `int` appartenga all'intervallo `[-5, 5)` - estremo inferiore incluso e superiore escluso - oppure sia uguale a `99`.

```
(( -5 <= n < 5 ) || n == 99 )
```

---

4 (6 punti)

Definite un ciclo `while` che incrementa di `1` ad ogni passo una variabile intera, che chiamiamo `num`, inizialmente posta a un valore intero pseudo-casuale, compreso tra `0` e `MAX`, e si fermi quando la variabile raggiunge un multiplo di `4`.
Nel codice scrivere anche l'inizializzazione della variabile di controllo del ciclo. Alla fine del ciclo un contatore opportunamente inizializzato e gestito ad ogni iterazione darà indicazione di quanti passi del ciclo sono stati effettuati.

```
final int MAX = 100;
int num = (int)(Math.random()*MAX);

    while (num%4!=0) num++;
```

---

5 (8 punti)

Definire una classe `Tabelline` che inizializza una struttura dati a matrice di dimensione `10 x 10` con le tabelline dei primi 10 numeri da 1 a 10 (cioè righe e colonne con lo stesso indice contengono la tabellina del numero rappresentato dall'indice). Ovvero inizializzi la matrice nel seguente modo:

```
1     2    3    4    5    6    7    8    9   10
2     4    6    8   10   12   14   16   18   20
3     6    9   12   15   18   21   24   27   30
4     8   12   16   20   24   28   32   36   40
5    10   15   20   25   30   35   40   45   50
6    12   18   24   30   36   42   48   54   60
7    14   21   28   35   42   49   56   63   70
8    16   24   32   40   48   56   64   72   80
9    18   27   36   45   54   63   72   81   90
10   20   30   40   50   60   70   80   90  100
```

```
public class Tabelline {

   public static int[][] calcola(){
      int[][] matrice = new int[10][];
      for (int i = 0; i < matrice.length; i++) {
         matrice[i] = new int[10];
            for (int j = 0; j < matrice[i].length; j++) {
               matrice[i][j] = (i+1) * (j+1);
            }
       }
       return matrice;
   }

….eventuali altri metodi ….
}
```

---

6 (2 punti)

Stabilite qual'è l'output del seguente codice Java, supponendo che sia stata dichiarata la variabile `int i`:

`for (i = 1; i < 12; i++) {`
`  if (i%4 == 0) break;`
`  System.out.println (i);`
`}`
`System.out.println (i);`
	`1`
	`2`
	`3`
	`4  //valore di i fuori dal loop`

`for (i = 1; i < 12; i++) {`
`  if (i%4 == 0) continue;`
`  System.out.println (i);`
`}`
`System.out.println (i);`
	`1`
	`2`
	`3`
	`5`
	`6`
	`7`
	`9`
	`10`
	`11`
	`12  //valore di i fuori dal loop`

`for (i = 1; i < 12; i++) {`
`  if (i%4 == 0) System.exit(0);`
`  System.out.println (i);`
`}`
`System.out.println (i);`
	`1`
	`2`
	`3`