Corso di laurea in Comunicazione digitale — Programmazione I

AA 2009/10 — Prof. M.A. Alberti

IV Appello - 7 luglio 2010

Cognome **Nome**

Matricola

1 (6 punti)
Date le classi:

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

  SuperCls (int i) {
    n = i;
    x = i+3;
  }

  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){ …
  }
}
```

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

Dire se le definizioni dei membri sottoindicati della classe `SottoCls` costituiscono *sovraccaricamento*, *sovrascrittura* o *errore*:

`metodoA`

`metodoB`

`metodoC`

`metodoD`

Dire se e' corretta la definizione del costruttore:

`SottoCls (int i)`

Corretti gli eventuali errori, specificandoli quali siano

Nel metodo main della superclasse, dopo aver eseguito

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

dire quali siano i valori di ritorno o l'output delle seguenti espressioni o istruzioni:

```
p.n
p.x
f.n
f.metodoA (f.n, x)
p.x
p.metodoC (p.n);
f.metodoC (f.n);
p.x (attenzione!!)
f.metodA (f.n, x)
```

È corretta l'istruzione:

```
int z = p.metodoB(15);
```

È corretta l'istruzione:

```
int z = f.metodoB(15);
```

Spiegare:

---

2 (5 punti)

Scrivere la classe `Sacchetti.java` che genera 100 numeri pseudo-casusali compresi tra 1 e 12 (estremi inclusi) e che conta le occorrenze dei numeri generati tra 1 e 4, quelli tra 5 e 8 e gli altri e scriva un rapporto finale. I contatori si chiamano `sacco_1`, `sacco_2` e `sacco_3`.

```
import java.util.Random;

public class Sacchetti{
   public static void main (String[] args){




}
```

3 (4 punti)

Data la stringa di caratteri `partenza` specificare l'effetto causato dall'esecuzione del ciclo sulla stringa `arrivo`. Non preoccupatevi del tipo `StringBuffer` se non lo conoscete, per questo esercizio equivale al tipo `String` con la differenza che posso invocare il metodo `append` che fa esattamente quello che vi aspettate che faccia: appende il carattere specificato in input alla stringa su cui è invocato.

```
String partenza = "i segni";
StringBuffer arrivo = new StringBuffer();
int i = 0;
char c = partenza.charAt(i);
while (c != 'i') {
    arrivo.append(c);
    c = partenza.charAt(++i);
}
```

Variabile `arrivo` alla fine del ciclo while:

Se, mantenendo le inizializzazioni, il ciclo fosse:

```
do {
    arrivo.append(c);
    c = partenza.charAt(++i);
} while (c != 'i');
```

la variabile `arrivo` alla fine del ciclo sarebbe:

Ripetere l'esercizio per la stringa `String partenza = "disegni";`
Variabile `arrivo` alla fine del ciclo while:

Variabile `arrivo` alla fine del ciclo do-while:

4 (4 punti)

Assumendo le dichiarazione seguenti:

```
final int MAX = 10;
final int MIN = 5;
int i, s=0, valore;
```

Dire qual'è il valore di `i, s` e `valore` dopo aver eseguito l'istruzione:
`for (i=2; valore > MIN && valore <= MAX; valore++, i++, s+=++i);`
Calcolate tre cicli `for` per diversi valori della variabile `valore`, considerandoli indipendenti l'uno dall'altro.

valore	i	s	valore
3			
8			
12			

Se i cicli fossero eseguiti sequenzialmente otterremmo valori diversi? SI NO
Per quale variabile?

5 (5 punti)

Scrivere una funzione **ricorsiva** per il calcolo del numero delle cifre in un numero naturale. Si ricorda che se il numero dato è compreso tra 0 e 9 allora è composta da 1 cifra. Nel caso di numeri più lunghi si potrà procedere con successive divisioni per 10 .

```
int numero_cifre(int n) {



}
```

---

6 (4 punti)

Disegnate la traccia dell'andamento dello stack delle chiamate dei metodi durante l'esecuzione del metodo **`numero_cifre`** dell'esercizio precedente quando viene richiamato **`numero_cifre(2010).`** Si ricordi che, nel record d'attivazione del metodo i campi `ind` e `val` indicano rispettivamente l'indirizzo e il valore di rientro relativi ad un metodo chiamato dal metodo corrente. Inoltre si noti che nella figura è tracciata solo la parte di stack relativa alla prima chiamata del metodo `numeroCifre` dall'inidirizzo `K` da parte di qualche altro metodo che trascuriamo. La freccia indica la direzione di crescita dello stack

	n 2010					
val ?	**val**					
ind K	**ind**					

Record 1 Record 2 →

---

7 (4 punti)

Utilizzando la classe `StringTokenizer` del pacchetto `java.util` scrivete un ciclo `while` per testare se ci sono ancora token e stamparli. Inizializzate l'oggetto `st` di classe `StringTokenizer` con la stringa `"questo e' un esame"`

Dire qual'è il tipo riportato da `nextToken()`: