Corso di laurea in Informatica AA 2005/06 Programmazione, M.A. Alberti

I Appello - 8 febbraio 2006

Cognome **Nome**

Matricola

1

Data la classe `Padre` e la sua sottoclasse `Figlio`:

```
public class Padre {
   protected static int m;
   private int n;
   Padre() {
      m++;
      n = 5;
   }
   protected int getn() {
      return n;
   }
   public int metodo_1(int i) {
      return n=i;
   }
   protected void metodo_2(int i) {
       n=i+2;
   }
   public static void metodo_3(int i)
   {
      stampa(i+3);
   }
   public static void main(…) {
      Padre p = new Padre();
      int x = 8;
      stampa(p.metodo_1(1));
      p.metodo_2(2); stampa(p.getn());
      Padre.metodo_3(x++);
      stampa(Padre.m);
   }
}
```

```
class Figlio extends Padre {
   static int n=20;
   int m;
   Figlio() {
   }
   public int getm() {
      return m;
   }
   public int metodo_1(int i) {
       return m += i;
   }
   public void metodo_2(double i) {
       m=(int)i*2;
   }
   public static void metodo_3(float i){
   }
   public static void metodo_4
                        (int i, int j) {
      stampa(n+i+j); n--
   }
}
```

Per ciascun metodo della sottoclasse `Figlio` dire se la definizione causa *errore* in compilazione e nel caso, specificare quale, o dire se si tratta di *sovrascrittura* o di *sovraccaricamento* o di *specializzazione*.

`metodo_1`:

`metodo_2`:

`metodo_3`:

`metodo_4`:

Definire il costruttore `Figlio()` in modo che il campo `n` sia decrementato di `1` ad ogni nuova istanza e il campo `m` sia posto a `1`

Nel contesto della classe `Figlio`, dopo aver eseguito: `Figlio f = new Figlio();` valutare:

`1. stampa(f.getn());`			
`2. Padre.main(null);`		`3. stampa(f.metodo_1(1));`	

		`4. stampa(f.getm());`	
`5. f.metodo_2(6.5);`		`10. Figlio.metodo_3(3);`	
`6. stampa(f.getm());`		`11. stampa(f.getn());`	
`7. Figlio.metodo_4(2,2);`		`12. Padre.main(null);`	
`8. stampa(Figlio.n);`			
`9. f.metodo_2(7);`			

Disegnate lo stato dello stack, dello heap e della memoria statica (per le sole parti rilevanti al fine di poter valutare il valore dei campi di classe e di istanza), durante l'esecuzione dell'istruzione 12., in particolare dopo aver eseguito l'ultima istruzione, immediatamente prima di concludere il metodo `main` della classe `Padre`.

stack	heap	statica

2

Considerate le classi `A` e `B`, estesnsione della classe `A`:

```
class A {
   public int codice;
   protected String tipo;

   A(int c, String t) {
      codice = c;
      tipo = t;
   }
}
```

```
class B extends A {
   private int codice;

   B(int c) {



   }
}
```

1. Definite il costruttore della classe `B`. Ricordate che la classe `B` è un'estensione della classe `A`, quindi nel definire il costruttore utilizzate opportunamente le seguenti indicazioni:
   1. ogni oggetto di classe `B` è caratterizzato da un numero di codice, archiviato nel campo `codice`
   2. il suo antenato ha per convenzione il campo `codice` il cui valore è dato dal valore – 1 del codice del figlio e il campo `tipo` posto al valore predefinito "`esteso`".
2. Osservate i due membri `codice` definiti nelle due classi. Il membro `codice` di classe `A` dichiarato `public` viene ereditato dalla classe `B`? SI NO
3. Si dice che il membro codice di classe `B` quello di classe `A`
4. Dire se l'istruzione seguente, che definisce un metodo di classe `B`, è corretta: SI NO

   ```
   int get_codice() { return super.codice + codice; }
   ```

5. Definita nella classe `A` un metodo `main` con le istruzioni seguenti, calcolate il valore che viene stampato:

public static void main (String[] s) { **A a = new A(21, "non esteso");** **B b = new B(222);** **System.out.println(a.codice);** **System.out.println(a.tipo);** **System.out.println(b.get_codice());** **System.out.println(b.tipo);** **}**	

3

Scrivere un programma per computare la famosa successione di Fibonacci `F(n)` data dai numeri `1,1,2,3,5,8,13,21, …` La sequenza è così definita:
`F(0) = 1, F(1) = 1, F(2) = F(1) + F(0), F(3) = F(2) + F(1),` …
Il programma, dato il parametro `n` dovrà riportare il valore del n-esimo numero della sequenza di Fibonacci.
Versione ricorsiva:

Versione iterativa:

Calcolare il valore di `F(5)`:

Dire il numero di chiamate ricorsive effettuate per calcolare `F(5)`:

4

Nel contesto delle seguenti dichiarazioni:

```
public class Parole {
     final char TAPPO = 0;          // TAPPO è null e indica fine stringa
     private String dato;
     Parole (String s){
          dato = new String (s) + TAPPO; }
     int lung () {




     }
     String rovescia (int i) {




     }
}
```

scrivere il metodo `rovescia`, che riporta la stringa rovesciata, in modo ricorsivo e il metodo `lung`, che calcola, in modo iterativo, la lunghezza della stringa non includendo il tappo.
Suggerimento per implementare il metodo `rovescia`: se il carattere corrispondente all'indice corrente `i` è il TAPPO, allora riporta la stringa vuota altrimenti riporta la stringa ottenuta richiamando opportunamente il metodo `rovescia` e concatenando il risultato con il carattere corrente. Il metodo `rovescia` verrà inizialmente chiamato passandogli 0 come parametro.

5

Data la frase "`Sempre caro mi fu quest'ermo colle`", scrivere un metodo `inpuInArray` che riceve la stringa come parametro e usa la classe `StringTokenizer` per selezionarne le singole parole e archiviarle in array di lunghezza appropriata da riportare all'ambiente chiamante.

Prototipo:

Codice

6

Data una stringa di caratteri **`partenza`** dire l'effetto causato dall'esecuzione del ciclo sulla stringa `arrivo`:

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

la stringa **`arrivo`** alla fine del ciclo:

E nel caso il ciclo fosse sostituito da un simile ciclo **`do-while`**?

```
do {
     copia.append(c);
     c = originale.charAt(++i);
} while (c != 'a');
```

la stringa **`arrivo`** alla fine del ciclo:

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

la stringa **`arrivo`** alla fine del ciclo **`do-while`**: