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

Compitino III Appello - 15 giugno 2006

Cognome **Nome**

Matricola

1

Data la classe `Vecchia` e la sua sottoclasse `Nuova`:

```
public class Vecchia {
   private int n;
   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(…) {
      Vecchia v = new Vecchia();
      stampa(v.metodo_1(1));
      v.metodo_2(2); stampa(v.getn());
     Vecchia.metodo_3(9);
   }
}
```

```
class Nuova extends Vecchia {
   static int n=20;
   int m;
   Nuova() {
      n--;
      m=1;
   }
   public int getm() {
      return m;
   }
   public int metodo_1(int i) {
       return m=super.metodo_1(i)+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 `Nuova` 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:`

Nel contesto della classe `Nuova`, dopo aver eseguito: `Nuova nv = new Nuova();` valutare le istruzioni:

`Vecchia.main(null);`		`stampa(Nuova.n);`	
		`nv.metodo_2(7.5);`	
		`stampa(nv.getm());`	
`stampa(nv.metodo_1(1));`		`nv.metodo_2(7);`	
`stampa(nv.getn());`		`Nuova.metodo_3(3);`	
`Nuova.metodo_4(2,2);`		`stampa(nv.getn());`	

2

Date le classi `Padre` e `Figlio`:

```
class Padre {
   int n;
   public Padre (int i) {
       n = i;
   }
}
class Figlio extends Padre {
   Figlio() {
      super();
   }
}
```

```
1.  public class Test {
2.     public void metodo() {
3.        Padre p = new Padre(5);
4.        Figlio f = new Figlio(10);
5.     }
6.     public static void main (String[] a) {
          Test t = new Test();
          t.metodo();
7.     }
8.  }
```

La classe `Test` viene compilata correttamente? Indicare in caso contrario la riga che causa errore:
Modificare le classi `Padre` o `Figlio` per evitare l'eventuale errore, lasciando inalterata la classe `Test`.

3

Scrivere un programma per computare la funzione di Ackermann `A(m, n)` definita come segue:

```
            | n+1 se m = 0
A(m, n)  =  | A(m-1, 1) se m>0 e n=0
            | A(m-1, A(m, n-1)) se m>0 e n >0
```

.

Calcolare `A(1, 2)` :

Dire il numero di chiamate ricorsive effettuate:

4

Date le due versioni seguenti per il calcolo della potenza:

```
static int pot (int k, int n) {
   if (n == 0)
      return 1;
   else return k * pot(k, n-1);

}
```

```
static int pot (int k, int n) {
   if (n == 0)
      return 1;
   else {
      int t = pot (k, n/2);
      if ((n % 2) == 0)
         return t * t;
      else return k * t * t;
   }
}
```

Produrre una tabella che mostra quante moltiplicazioni e quante chiamate al metodo pot vengono effettuate in ciascuna versione nel calcolare:

`pot( 3, n ) con n= 0, 1, 2, 4, 8, 16, 32`

metodo/n	0	1	2	4	8	16	32	64
metodo A moltiplicazioni								
metodo A chiamate								
metodo B moltiplicazioni								
metodo b chiamate								

.

5

Valutare l'espressione aritmetica 2 * (1 + 2) * (3 + 5)
Disegnare l'andamento dello stack nella griglia proposta (ogni colonna rappresenta una fotografia dello stack ad un passo della computazione, il fondo dello stack sia in basso) durante la valutazione dell'espressione, dopo averla trascritta in forma postfissa:

6

Dato il seguente spezzone di codice:

```
public class Arte {
   private int anno;
   int periodo() {
      return anno;
   }
}
```

```
public class Disegno extends Arte {
   String autore;

}
```

Dire se le seguenti istruzioni sono lecite o no:

a. `Disegno d = new Arte();` SI NO

b. `Disegno d = new Disegno(); int x = d.periodo();` SI NO

c. `Arte y; int p = y.periodo();` SI NO

d. `Arte s =  new Disegno(); s.autore="leonardo";` SI NO

In caso di risposta NO, correggete, se possible, le istruzioni per renderle lecite:

a.

b.

c.

d.

7

In una implementazione dinamica della struttura dati stack, definita come segue, dare la definizione della funzione `top()` che accede al valore del nodo in cima allo stack e lo riporta all'ambiente senza modificare la struttura e di un'eccezione non controllata `EmptyStackException()`.

```
public class Stack  {
   private NodoStack cima;
   private class NodoStack  {
      Object dato;
      NodoStack pros;
   }

   public Stack()  {
      cima = null;
   }
```

```
public void push(Object o)  {
      NodoStack t = new NodoStack();
      t.dato = o;
      t.pros = cima;
      cima = t;
}
```

Si scriva una istruzione `try-catch` in cui si cattura l'eventuale eccezione lanciata dal metodo `top()` nel contesto di un programma in cui sia stata introdotta la variabile `pila`, onde poter recuperare e inserire almeno un nodo. Gli oggetti da inserire nella pila siano di classe `Integer`.

```
Stack pila = new Stack();
```

8

Avendo definito `class Settimana implements java.util.Iterator { };` dire se sono lecite le seguenti istruzioni:

a. `Settimana s = new Settimana();`

   `while(s.hasNext()) <computa(s.next()>` SI NO

b. `Iterator i = new Settimana();` SI NO

c. Qual'è il concetto della programmazione a oggetti per cui diversi oggetti che hanno una interfaccia comune rispondono in modo diverso quando un metodo di quell'interfaccia è invocato:

   ……..

d. Casting è appropriato quando si riceve un riferimento a una classe antenato e si sappia che l'oggetto è una particolare sotto-classe e si voglia usare l'oggetto nella sua funzionalità completa. SI NO