Corso di laurea in Informatica AA 2006/07 Programmazione, M.A. Alberti

V Appello 2006/07 - 19 settembre 2007

Cognome **Nome**

Matricola

1

Definire il prototipo di un metodo `filtra` che riceve in input un array di interi e riporta all'ambiente chiamante un array di interi:

Definire ora il corpo del metodo, con le seguenti specifiche: l'array di interi passato come parametro di input , di lunghezza assegnata `MAX`, dovrà essere riportato all'ambiente chiamante opportunamente modificato. In particolare, i dati nelle posizioni dispari saranno mantenuti inalterati gli altri saranno posti a `0`.

2

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.

3
Dichiarate la struttura dati `matr` matrice di interi di dimensione `M` x `N`, dove `M` e `N` sono costanti intere date:

4
Definite (prototipo e corpo) il metodo `moltiplica` che riceve in input due matrici quadrate e ne calcola il prodotto da riportare all'ambiente

---

5
La sequenza di Fibonacci è la seguente, in cui ogni elemento è costituito dalla somma dei due che lo precedono e i primi due elementi sono dati (`0` e `1`):
`0 1 1 2 3 5 8 13 21 34 …`
Un metodo che la calcoli deve quindi riportare il valore della sequenza corrispondente al parametro passato in input, ad esempio:
`fibonacci(0)` deve riportare `0`
`fibonacci(3)` deve riportare `2`
Scrivere un metodo ricorsivo che implementa la funzione di Fibonacci

```
int fibonacci (int n) {

}
```

6

Si esegua il metodo `fibonacci(5)`. Si dica

a. quante chiamate ricorsive vengono effettuate:

b. quante chiamate al metodo `fibonacci(1)` vengono effettuate:

7

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:

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`

   `metodoB`

   `metodoC`

   `metodoD`

c. Dire se e' corretta la definizione del costruttore

   `SottoCls (int i)`

8

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