Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 17 giugno 2013

TEMPO DISPONIBILE: 1 ora e 30 minuti

1. Siano `x` e `y` due variabili di tipo `int` alle quali sono assegnati dei valori iniziali. Scrivete, per i casi indicati, i valori delle due variabili dopo l'esecuzione dei seguenti frammenti di codice:

(a)
```
x = x + (x = y) + x;
y = x + y;
```

(b)
```
y = x + (x = y);
x = x + y;
```

(a) assegnamenti iniziali: `x = 2` `y = 4` valori finali?		(a) assegnamenti iniziali: `x = 5` `y = 7` valori finali?	
`x`	`y`	`x`	`y`

(b) assegnamenti iniziali: `x = 2` `y = 4` valori finali?		(b) assegnamenti iniziali: `x = 5` `y = 7` valori finali?	
`x`	`y`	`x`	`y`

2. Data una variabile `x` di tipo `double`, considerate la seguente condizione:

"Il valore contenuto in `x` è inferiore a `15` ma non a `10`"

(a) Esprimete in linguaggio Java la condizione precedente *senza utilizzare* l'operatore di negazione.

(b) Esprimete in linguaggio Java la *negazione* della condizione precedente *senza utilizzare* l'operatore di negazione.

3. Considerate il seguente metodo ricorsivo. Scrivete il risultato restituito dalle chiamate indicate nei due riquadri:

```
... int f(int x) {
  if (x == x * x)
    return 1;
  else
    return x * f(x / 2) + 3;
}
```

`f(7)`	`f(0)`

4. Considerate la dichiarazione di variabile `String[] nomi` e il seguente frammento di codice:

```
int x = 1;
try {
  x = nomi[x - 1].length() / nomi[x + 1].length();
} catch (ArithmeticException e) {
  x = x + 9;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 11;
} catch (NullPointerException e) {
  x = x + 8;
}
```

Ricordando che:

- `ArithmeticException` viene sollevata in caso di anomalie nel calcolo di operazioni aritmetiche,
- `ArrayIndexOutOfBoundsException` viene sollevata quando si tenti di accedere a una posizione inesistente in un array,
- `NullPointerException` viene sollevata quando si tenti di accedere a un oggetto tramite un riferimento `null`,

indicate nel riquadro corrispondente, in ciascuno dei seguenti casi, il valore della variabile `x` dopo l'esecuzione:

(a) l'array riferito da `nomi` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe `""`, `"formica"`, `"ape"`.

(b) l'array riferito da `nomi` contiene come unico elemento il riferimento a un oggetto che rappresenta la stringa `"ape"`.

(c) l'array riferito da `nomi` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe `"formica"`, `"ape"`, `""`.

(d) l'array riferito da `nomi` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe `"formica"`, `""`, `"ape"`.

Negli esercizi seguenti si supponga di disporre di una classe concreta di nome `Alfa`, che possiede un *unico costruttore*. Il costruttore riceve come argomento un valore di tipo `int`. Tra i metodi di `Alfa` vi è `public int length()`, che restituisce il numero di cifre del valore `int` specificato al momento della costruzione dell'oggetto. Ad esempio, il metodo `length()` di un oggetto costruito invocando `new Alfa(123)` restituisce `3`.

5. Considerate le seguenti classi:

```
public class Beta extends Alfa {
  private static int x = 5;
  private String y;

  public Beta(String s, int u) {
    super(u);
    y = s;
    x = x + u;
  }

  public int length() {
    return super.length() + y.length();
  }

  public static int getStatico() {
    return x;
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Beta.getStatico());     //1
    Alfa a = new Beta("coccodrillo", 300);
    System.out.println(a.length());            //2
    a = new Alfa(a.length());
    System.out.println(a.length());            //3
    System.out.println(Beta.getStatico());     //4
  }
}
```

Scrivete in ogni riquadro l'output prodotto dall'istruzione di stampa seguita dal commento indicato:

//1	//2	//3	//4

6. Oltre alle classi precedenti, considerate due classi concrete `Sigma` e `Omega`, un'interfaccia `In`, tali che:

- `Omega` estende `Beta`,
- `Sigma` estende `Alfa` e implementa `In`.

a. Nel riquadro che precede ciascuna affermazione, scrivete V se l'affermazione è vera, F se è falsa:

☐ `Sigma` deve fornire l'implementazione dei metodi di `In`

☐ Se il codice sorgente della classe `Omega` non contiene un costruttore allora il compilatore segnala un errore

☐ Ogni istanza di `Sigma` è anche un'istanza di `Alfa`

☐ Ogni istanza di `Sigma` è anche un'istanza di `Beta`

☐ `Sigma` è una *sottoclasse* di `In`

☐ `In` è un *supertipo* di `Sigma`

☐ Ogni istanza di `Beta` possiede un solo campo (oltre a quelli ereditati dalla superclasse)

☐ Ogni istanza di `Beta` possiede due campi (oltre a quelli ereditati dalla superclasse)

☐ `Sigma` *deve ridefinire* il metodo `length`

☐ `Sigma` *può ridefinire* il metodo `length`

b. Considerate le seguenti dichiarazioni di variabile:

```
Alfa a; Beta b; Omega o; Sigma s; In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente (supponete che al posto di `...` vi siano gli argomenti opportuni):

☐ `o = new Sigma(...)`

☐ `s = new Sigma(...)`

☐ `s = i`

☐ `s = new Omega(...)`

☐ `a = (Sigma) i`

☐ `i = (Sigma) a`

☐ `i = s`

☐ `s = a`

☐ `a = o`

☐ `b = (Beta) a`

7. Trattate *in modo sintetico* uno solo tra i seguenti argomenti. (Esempio)

(a) Spiegate il concetto di *supertipo* e *sottotipo* facendo riferimento esplicitamente sia alle classi, sia alle interfacce.

(b) I significati della parola riservata `this`.