Cognome.....

Nome.....

Matricola.....

Programmazione

Compitino del 26 gennaio 2011

TEMPO DISPONIBILE: 1 ora e 30 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Number` (esercizi 1, 2, 3, 4, 5): ogni oggetto della classe rappresenta un numero. La classe possiede un costruttore privo di argomenti. Nelle librerie standard alcune classi involucro (ad esempio `Integer`, `Long` e `Double`) sono definite estendendo `Number`.

  Tra i metodi forniti da `Number` vi è:

  - `public abstract double doubleValue()`
    Restituisce il numero rappresentato dall'oggetto che esegue il metodo, come valore del tipo primitivo `double`.

- Classe `ElencoNumeri` (esercizi 1, 2, 3): ogni oggetto della classe rappresenta un elenco di oggetti `Number`. Tra i metodi forniti dalla classe vi sono:

  - `public double somma()`
    Restituisce il somma di tutti i numeri presenti nell'elenco che esegue il metodo. I numeri vengono sommati considerando il loro valore nel tipo `double`.
    Ad esempio, se l'elenco contiene due oggetti di tipo `Integer` che rappresentano gli interi `2` e `3`, un oggetto di tipo `Long` che rappresenta l'intero `8` e due oggetti di tipo `Double` che rappresentano i valori `2.4` e `5.1`, il metodo `somma` restituisce il valore `20.5`.
  - `public int quanti()`
    Restituisce il numero totale di numeri presenti nell'elenco che esegue il metodo (nel caso dell'esempio precedente il metodo `quanti` restituisce `5`).
  - `public double media()`
    Restituisce la media di tutti i numeri presenti nell'elenco che esegue il metodo (nel caso dell'esempio precedente il metodo `media` restituisce `4.1`). Se nell'elenco non ci sono numeri, il metodo solleva un'eccezione non controllata di tipo `ArithmeticException`.

1. Scrivete l'implementazione del metodo `media`, *senza conoscere* l'implementazione della classe `ElencoNumeri`, ma servendovi esclusivamente dei metodi che essa fornisce. Si ricordi che la classe `ArithmeticException` fornisce un costruttore che riceve come argomento una stringa (un messaggio d'errore).

La classe `ElencoNumeri` è implementata mediante un unico campo

```
private Number[] numeri
```

che si riferisce ad un array contenente i numeri presenti nell'elenco (in alcune posizioni l'array può contenere `null`).

2. Scrivete l'implementazione del metodo `somma`.

3. La classe `Double` fornisce il metodo:

- `public boolean isNaN()`
  Restituisce `true` se e solo se l'oggetto che esegue il metodo rappresenta il valore `NaN` (Not-a-Number).

Scrivete l'implementazione del seguente metodo della classe `ElencoNumeri`:

- `public int quantiDoubleNaN()`
  Restituisce il numero di oggetti presenti nell'elenco che sono di tipo `Double` e che rappresentano il valore `NaN`.

4. Oltre alle classi indicate in precedenza, considerate due classi concrete `Beta` e `Gamma` e un'interfaccia `In`, tali che:
- `Gamma` estende direttamente `Number` e implementa `In`,
- `Beta` estende direttamente `Gamma`.

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

☐ Almeno un costruttore di `Gamma` richiama un costruttore di `Number`
☐ Ogni metodo di `In` è astratto
☐ `Double` è un supertipo di `Beta`
☐ `Gamma` deve fornire l'implementazione del metodo `doubleValue`
☐ `Beta` è un sottotipo di `In`
☐ `Gamma` possiede un costruttore
☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
☐ `Beta` deve fornire l'implementazione del metodo `doubleValue`
☐ Almeno un costruttore di `Gamma` richiama un costruttore di `In`
☐ `Beta` è una sottoclasse di `In`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n; Gamma x; Beta y; Double d; 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):

☐ `y = (Beta) n`
☐ `d = i`
☐ `n = (Gamma) i`
☐ `x = y`
☐ `y = x`

☐ `n = y`
☐ `n = new Number(...)`
☐ `n = new Gamma(...)`
☐ `n = i`
☐ `y = new Gamma(...)`

5. Considerate le seguenti classi:

```
public class Alfa extends Number {

  private int x = 8, y = 14;
  private static int z = 2;

  public Alfa(String s) {
    x = s.length();
    z = z + 1;
  }

  public Alfa(int i) {
    this("");
    y = i;
  }

  public int t() {
    return x + y;
  }

  public static int getStatico() {
    return z;
  }

  ...altri metodi...
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Alfa.getStatico());  //1
    Alfa a = new Alfa(5);
    System.out.println(a.t());              //2
    a = new Alfa("grillo");
    System.out.println(a.t());              //3
    System.out.println(Alfa.getStatico());  //4
  }
}
```

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

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

6. Considerate la seguente dichiarazione di variabile

```
String[] parole;
```

e il seguente frammento di codice:

```
int x = 0;
try {
  x = parole[x].length() / parole[x + 1].length();
} catch (ArithmeticException e) {
  x = x + 6;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 7;
} catch (NullPointerException e) {
  x = x + 10;
}
```

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`,
- `""` rappresenta la stringa vuota,

indicate il valore della variabile `x` dopo l'esecuzione, nei seguenti casi:

a. l'array riferito da `parole` contiene, nell'ordine, i riferimenti alle stringhe `"formica"`, `"ape"`, `""`.

valore di `x`

b. l'array riferito da `parole` contiene, nell'ordine, i riferimenti alle stringhe `""`, `"formica"`, `"ape"`.

valore di `x`

c. `parole` contiene `null`.

valore di `x`

d. l'array riferito da `parole` contiene, nell'ordine, i riferimenti alle stringhe `"formica"`, `""`, `"ape"`.

valore di `x`

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

```
... int f(int x, int y) {
  if (x > y)
    return x;
  else
    return 2 * f(x - 1, y / 2) + 1;
}
```

`f(3, 3)`

`f(4, 6)`