Cognome .....

Nome .....

Matricola .....

Programmazione

Compitino del 17 gennaio 2007

NOTA: Negli esercizi 1, 2, 3, 4 fate riferimento alla classe astratta `Figura`, con le sottoclassi concrete `Rettangolo`, `Quadrato` e `Cerchio`. Si ricordi che:

- `Rettangolo` e `Cerchio` estendono `Figura`, `Quadrato` estende `Rettangolo`;
- in `Figura` vi sono due metodi astratti `public double getArea()` e `public double getPerimetro()` che restituiscono, rispettivamente, l'area e il perimetro della figura che esegue il metodo;
- in `Rettangolo` sono definiti i metodi `public double getBase()` e `public double getAltezza()` che restituiscono, rispettivamente, la base e l'altezza del rettangolo che li esegue.
- in `Quadrato` è definito un metodo `public double getLato()` il cui compito è restituire il lato del quadrato che lo esegue;
- in `Cerchio` è definito un metodo `public double getRaggio()` il cui compito è restituire il raggio del cerchio che lo esegue.

1. Considerate la dichiarazione di variabile: `Figura[] figure`. Supponete di disporre di una porzione di codice, alla fine della quale il riferimento `figure` si riferisce a un array che contiene, in ogni posizione, il riferimento a un oggetto di tipo `Figura`. Scrivete le istruzioni per visualizzare la media delle aree di tutte le figure presenti nell'array (supponete di disporre di un riferimento `out` di tipo `ConsoleOutputManager` a un oggetto che rappresenta il monitor).

2. Nell'ipotesi che l'array riferito da `figure` non sia vuoto, scrivete delle istruzioni, che dovranno essere collocate dopo quelle dell'esercizio precedente, che permettano di visualizzare:

- il numero di rettangoli presenti nell'array, che non siano dei quadrati;
- il numero di rettangoli presenti nell'array, che non siano dei quadrati e che abbiano area superiore alla media calcolata dal codice scritto per l'esercizio precedente.

3. Considerate la dichiarazione di variabile: `Figura[] figure`. Supponete di disporre di una porzione di codice, alla fine della quale il riferimento `figure` si riferisce a un array che contiene, in ogni posizione, il riferimento a un oggetto di tipo `Figura`. Scrivete le istruzioni per ottenere, in una variabile `sommaRaggi` (da dichiarare), la somma dei raggi dei cerchi.

4. Considerate le seguenti classi:

```
public class Alfa extends Rettangolo {

  private double k = 10;
  private static int cont = 7;

  public Alfa(double y) {
    super(y, y);
    k = k + y;
    cont++;
  }

  public Alfa(int x) {
    this((double) x);
    k = x;
  }

  public double getArea() {
    return super.getArea() + k;
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Alfa.getStatico());//1
    Alfa a = new Alfa(2.0);
    Rettangolo r = a;
    a = new Alfa(2);
    System.out.println(r.getArea());      //2
    System.out.println(a.getArea());      //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

5. Considerate la seguente dichiarazione di variabile

```
String[] parole;
```

e il seguente frammento di codice:

```
int x = 0;
try {
  for (String s : parole)
    x = x + s.length();
  x = parole[x / parole.length - 1].length();
} catch (ArithmeticException e) {
  x = x + 6;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 8;
} catch (NullPointerException e) {
  x = x + 14;
}
```

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

a. l'array riferito da `parole` contiene, nell'ordine, i riferimenti alle stringhe `"gatto"`, `"cane"`, `"topo"`.

valore di `x`

b. l'array riferito da `parole` è vuoto.

valore di `x`

c. `parole` contiene `null`.

valore di `x`

d. l'array riferito da `parole` contiene, nell'ordine, i riferimenti alle stringhe `"il"`, `"mio"`, `"cane"`.

valore di `x`

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

```
... int f(int x) {
  if (x % 2 == 0)
    return f(x + 1) + f(x - 1);
  else if (x > 4)
    return 2 * f(x - 2);
  else
    return x;
}
```

`f(2)`

`f(4)`

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`.

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)`