Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 10 luglio 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Number`.
  Ogni oggetto della classe rappresenta un valore numerico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Number` vi è:

  `public abstract double doubleValue()`
  Restituisce un valore di tipo `double` uguale al numero rappresentato dall'oggetto che esegue il metodo (dopo avere effettuato le necessarie conversioni e approssimazioni; ad esempio se l'oggetto rappresenta il numero `1`, il metodo restituirà il valore `1` rappresentato come `double`).

- Classi `Integer`, `Long`, `Float`, `Double`.
  Gli oggetti rappresentano valori interi o in virgola mobile dei tipi primitivi `int`, `long`, `float` e `double`, rispettivamente. Queste classi (e non solo queste) sono definite estendendo direttamente `Number`.

  Tra i metodi forniti dalla classe `Double` vi sono:

  `public boolean isInfinite()`
  Restituisce `true` se l'oggetto che esegue il metodo rappresenta un valore infinito.

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

  La classe `Double` fornisce più costruttori. Tutti i costruttori prevedono almeno un argomento. Uno dei costruttori è:

  `public Double(double x)`
  Costruisce un oggetto che rappresenta il valore fornito tramite l'argomento.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Number[] ar`. Supponete di disporre di una porzione di codice alla fine della quale `ar` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Number` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor un valore di tipo `double` uguale alla *somma* degli oggetti contenuti nell'array che rappresentano un *valore positivo* (utilizzando il metodo opportuno tra quelli indicati sopra, per ciascun oggetto potete calcolare il corrispondente valore di tipo `double` e confrontare il risultato ottenuto con `0`).

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Double` contenuti nell'array `ar` che contengono un valore positivo e il numero di oggetti di tipo `Double` contenuti nell'array `ar` che contengono un valore negativo.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Double` contenuti nell'array `ar` che *non rappresentano* un valore infinito.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "gatto";
  private int w;

  public Strana() {
    w = 2 * s.length();
  }

  public Strana(String k) {
    this();
    w = w + k.length();
    s = k;
  }

  public int value() {
    return w;
  }

  public static int getStatico() {
    return s.length();
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana();
    System.out.println(x.value());            //2
    x = new Strana("ape");
    System.out.println(x.value());            //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

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

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

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Gamma` che estende `Strana`, una *classe astratta* `Omega` che estende `Number`, una classe concreta `Delta` che estende `Double` e implementa un'interfaccia `In`.

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

- ☐ Ogni istanza di `Strana` contiene il campo `s`
- ☐ Ogni istanza di `Strana` contiene il campo `w`
- ☐ `Omega` deve fornire l'implementazione del metodo `doubleValue`
- ☐ `Omega` può fornire l'implementazione del metodo `doubleValue`
- ☐ `Delta` è un sottotipo di `Object`
- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Delta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ La classe `Omega` possiede un costruttore
- ☐ Tutti i metodi di `Omega` devono essere astratti
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n; Object o; Gamma g; Strana s; Delta 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):

- ☐ `s = (Strana) o`
- ☐ `n = new Number()`
- ☐ `n = (Omega) n`
- ☐ `o = s`
- ☐ `g = (Gamma) i`
- ☐ `d = (Delta) o`
- ☐ `s = g`
- ☐ `g = s`
- ☐ `i = (Delta) s`
- ☐ `i = (Delta) n`

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

```
int x = 0;
try {
  for (String s: parole)
    x = s.charAt(0) == s.charAt(s.length() - 1) ? 0: 2 * x + 1;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 3;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 28;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 45;
} catch (NullPointerException e) {
  x = x + 16;
}
```

Ricordando che:

- il metodo `chatAt` della classe `String` restituisce il carattere che si trova nella posizione ricevuta tramite il parametro di tipo `int`, sollevando una `StringIndexOutOfBoundsException` se tale posizione non è presente nella stringa che esegue il metodo,
- `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`,
- "" indica la stringa vuota,

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

(a) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "`orso`", "`pollo`". ☐

(b) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`orso`", "`pollo`", "`cane`". ☐

(c) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`pollo`", "`cane`", "`orso`". ☐

(d) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "", "`orso`". ☐

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

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

f(4)	f(5)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 10 luglio 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Number`.
  Ogni oggetto della classe rappresenta un valore numerico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Number` vi è:

  `public abstract float floatValue()`
  Restituisce un valore di tipo `float` uguale al numero rappresentato dall'oggetto che esegue il metodo (dopo avere effettuato le necessarie conversioni e approssimazioni; ad esempio se l'oggetto rappresenta il numero `1`, il metodo restituirà il valore `1` rappresentato come `float`).

- Classi `Integer`, `Long`, `Float`, `Double`.
  Gli oggetti rappresentano valori interi o in virgola mobile dei tipi primitivi `int`, `long`, `float` e `double`, rispettivamente. Queste classi (e non solo queste) sono definite estendendo direttamente `Number`.

  Tra i metodi forniti dalla classe `Float` vi sono:

  `public boolean isInfinite()`
  Restituisce `true` se l'oggetto che esegue il metodo rappresenta un valore infinito.

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

  La classe `Float` fornisce più costruttori. Tutti i costruttori prevedono almeno un argomento. Uno dei costruttori è:

  `public Float(float x)`
  Costruisce un oggetto che rappresenta il valore fornito tramite l'argomento.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Number[] ar`. Supponete di disporre di una porzione di codice alla fine della quale `ar` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Number` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor un valore di tipo `float` uguale alla *somma* degli oggetti contenuti nell'array che rappresentano un *valore positivo* (utilizzando il metodo opportuno tra quelli indicati sopra, per ciascun oggetto potete calcolare il corrispondente valore di tipo `float` e confrontare il risultato ottenuto con `0`).

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Float` contenuti nell'array `ar` che contengono un valore positivo e il numero di oggetti di tipo `Float` contenuti nell'array `ar` che contengono un valore negativo.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Float` contenuti nell'array `ar` che *non rappresentano* un valore infinito.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "topo";
  private int w;

  public Strana() {
    w = 2 * s.length();
  }

  public Strana(String k) {
    this();
    w = w + k.length();
    s = k;
  }

  public int value() {
    return w;
  }

  public static int getStatico() {
    return s.length();
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana();
    System.out.println(x.value());            //2
    x = new Strana("formica");
    System.out.println(x.value());            //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

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

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

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Alfa` che estende `Strana`, una *classe astratta* `Omega` che estende `Number`, una classe concreta `Beta` che estende `Float` e implementa un'interfaccia `In`.

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

- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ La classe `Omega` possiede un costruttore
- ☐ Tutti i metodi di `Omega` devono essere astratti
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ `Omega` deve fornire l'implementazione del metodo `floatValue`
- ☐ `Omega` può fornire l'implementazione del metodo `floatValue`
- ☐ `Beta` è un sottotipo di `Object`
- ☐ Ogni istanza di `Strana` contiene il campo `s`
- ☐ Ogni istanza di `Strana` contiene il campo `w`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n; Object o; Alfa a; Strana s; Beta b; 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):

- ☐ `n = new Number()`
- ☐ `s = a`
- ☐ `a = s`
- ☐ `i = (Beta) s`
- ☐ `n = (Omega) n`
- ☐ `o = s`
- ☐ `a = (Alfa) i`
- ☐ `b = (Beta) o`
- ☐ `s = (Strana) o`
- ☐ `i = (Beta) n`

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

```
int x = 0;
try {
  for (String s: parole)
    x = s.charAt(0) == s.charAt(s.length() - 1) ? 0: 2 * x + 1;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 13;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 18;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 14;
} catch (NullPointerException e) {
  x = x + 33;
}
```

Ricordando che:

- il metodo `chatAt` della classe `String` restituisce il carattere che si trova nella posizione ricevuta tramite il parametro di tipo `int`, sollevando una `StringIndexOutOfBoundsException` se tale posizione non è presente nella stringa che esegue il metodo,
- `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`,
- "" indica la stringa vuota,

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

(a) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`pollo`", "`cane`", "`orso`". ☐

(b) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "", "`orso`". ☐

(c) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "`orso`", "`pollo`". ☐

(d) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`orso`", "`pollo`", "`cane`". ☐

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

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

f(4)	f(5)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 10 luglio 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Number`.
  Ogni oggetto della classe rappresenta un valore numerico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Number` vi è:

  `public abstract double doubleValue()`
  Restituisce un valore di tipo `double` uguale al numero rappresentato dall'oggetto che esegue il metodo (dopo avere effettuato le necessarie conversioni e approssimazioni; ad esempio se l'oggetto rappresenta il numero `1`, il metodo restituirà il valore `1` rappresentato come `double`).

- Classi `Integer`, `Long`, `Float`, `Double`.
  Gli oggetti rappresentano valori interi o in virgola mobile dei tipi primitivi `int`, `long`, `float` e `double`, rispettivamente. Queste classi (e non solo queste) sono definite estendendo direttamente `Number`.

  Tra i metodi forniti dalla classe `Double` vi sono:

  `public boolean isInfinite()`
  Restituisce `true` se l'oggetto che esegue il metodo rappresenta un valore infinito.

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

  La classe `Double` fornisce più costruttori. Tutti i costruttori prevedono almeno un argomento. Uno dei costruttori è:

  `public Double(double x)`
  Costruisce un oggetto che rappresenta il valore fornito tramite l'argomento.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Number[] el`. Supponete di disporre di una porzione di codice alla fine della quale `el` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Number` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor un valore di tipo `double` uguale alla *somma* degli oggetti contenuti nell'array che rappresentano un *valore positivo* (utilizzando il metodo opportuno tra quelli indicati sopra, per ciascun oggetto potete calcolare il corrispondente valore di tipo `double` e confrontare il risultato ottenuto con `0`).

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Double` contenuti nell'array `el` che contengono un valore positivo e il numero di oggetti di tipo `Double` contenuti nell'array `el` che contengono un valore negativo.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Double` contenuti nell'array `el` che *non rappresentano* un valore infinito.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "maiale";
  private int w;

  public Strana() {
    w = 2 * s.length();
  }

  public Strana(String k) {
    this();
    w = w + k.length();
    s = k;
  }

  public int value() {
    return w;
  }

  public static int getStatico() {
    return s.length();
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana();
    System.out.println(x.value());            //2
    x = new Strana("mucca");
    System.out.println(x.value());            //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

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

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

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Delta` che estende `Strana`, una *classe astratta* `Omega` che estende `Number`, una classe concreta `Alfa` che estende `Double` e implementa un'interfaccia `In`.

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

- ☐ Ogni istanza di `Strana` contiene il campo `w`
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ `Omega` deve fornire l'implementazione del metodo `doubleValue`
- ☐ Ogni istanza di `Strana` contiene il campo `s`
- ☐ Se il codice sorgente della classe `Delta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ La classe `Omega` possiede un costruttore
- ☐ Tutti i metodi di `Omega` devono essere astratti
- ☐ `Omega` può fornire l'implementazione del metodo `doubleValue`
- ☐ `Alfa` è un sottotipo di `Object`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n; Object o; Delta d; Strana s; Alfa a; 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):

- ☐ `i = (Alfa) s`
- ☐ `i = (Alfa) n`
- ☐ `d = (Delta) i`
- ☐ `a = (Alfa) o`
- ☐ `s = d`
- ☐ `s = (Strana) o`
- ☐ `n = new Number()`
- ☐ `n = (Omega) n`
- ☐ `o = s`
- ☐ `d = s`

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

```
int x = 0;
try {
  for (String s: parole)
    x = s.charAt(0) == s.charAt(s.length() - 1) ? 0: 2 * x + 1;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 7;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 17;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 27;
} catch (NullPointerException e) {
  x = x + 37;
}
```

Ricordando che:

- il metodo `chatAt` della classe `String` restituisce il carattere che si trova nella posizione ricevuta tramite il parametro di tipo `int`, sollevando una `StringIndexOutOfBoundsException` se tale posizione non è presente nella stringa che esegue il metodo,
- `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`,
- "" indica la stringa vuota,

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

(a) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`orso`", "`pollo`", "`cane`". ☐

(b) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "`orso`", "`pollo`". ☐

(c) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "", "`orso`". ☐

(d) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`pollo`", "`cane`", "`orso`". ☐

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

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

f(4)	f(5)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 10 luglio 2015

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe astratta `Number`.
  Ogni oggetto della classe rappresenta un valore numerico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Number` vi è:

  `public abstract float floatValue()`
  Restituisce un valore di tipo `float` uguale al numero rappresentato dall'oggetto che esegue il metodo (dopo avere effettuato le necessarie conversioni e approssimazioni; ad esempio se l'oggetto rappresenta il numero `1`, il metodo restituirà il valore `1` rappresentato come `float`).

- Classi `Integer`, `Long`, `Float`, `Double`.
  Gli oggetti rappresentano valori interi o in virgola mobile dei tipi primitivi `int`, `long`, `float` e `double`, rispettivamente. Queste classi (e non solo queste) sono definite estendendo direttamente `Number`.

  Tra i metodi forniti dalla classe `Float` vi sono:

  `public boolean isInfinite()`
  Restituisce `true` se l'oggetto che esegue il metodo rappresenta un valore infinito.

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

  La classe `Float` fornisce più costruttori. Tutti i costruttori prevedono almeno un argomento. Uno dei costruttori è:

  `public Float(float x)`
  Costruisce un oggetto che rappresenta il valore fornito tramite l'argomento.

Attenzione: nello svolgere gli esercizi fate riferimento a quanto indicato sopra e non all'implementazione delle classi, che è privata e dunque non accessibile al di fuori del codice di ciascuna delle classi stesse.

Negli esercizi 1, 2 e 3 considerate la dichiarazione di variabile `Number[] el`. Supponete di disporre di una porzione di codice alla fine della quale `el` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Number` (quindi nessuna delle posizioni contiene `null`).

1. Scrivete una porzione di codice per calcolare e visualizzare sul monitor un valore di tipo `float` uguale alla *somma* degli oggetti contenuti nell'array che rappresentano un *valore positivo* (utilizzando il metodo opportuno tra quelli indicati sopra, per ciascun oggetto potete calcolare il corrispondente valore di tipo `float` e confrontare il risultato ottenuto con `0`).

2. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Float` contenuti nell'array `el` che contengono un valore positivo e il numero di oggetti di tipo `Float` contenuti nell'array `el` che contengono un valore negativo.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor il numero di oggetti di tipo `Float` contenuti nell'array `el` che *non rappresentano* un valore infinito.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "oca";
  private int w;

  public Strana() {
    w = 2 * s.length();
  }

  public Strana(String k) {
    this();
    w = w + k.length();
    s = k;
  }

  public int value() {
    return w;
  }

  public static int getStatico() {
    return s.length();
  }
}
```

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana();
    System.out.println(x.value());            //2
    x = new Strana("coccodrillo");
    System.out.println(x.value());            //3
    System.out.println(Strana.getStatico());  //4
  }
}
```

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

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

5. Oltre alle classi utilizzate negli esercizi precedenti, considerate una classe concreta `Beta` che estende `Strana`, una *classe astratta* `Omega` che estende `Number`, una classe concreta `Gamma` che estende `Float` e implementa un'interfaccia `In`.

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

- ☐ Tutti i metodi di `Omega` devono essere astratti
- ☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
- ☐ Ogni istanza di `Strana` contiene il campo `s`
- ☐ Ogni istanza di `Strana` contiene il campo `w`
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ La classe `Omega` possiede un costruttore
- ☐ `Omega` deve fornire l'implementazione del metodo `floatValue`
- ☐ `Omega` può fornire l'implementazione del metodo `floatValue`
- ☐ `Gamma` è un sottotipo di `Object`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n; Object o; Beta b; Strana s; Gamma g; 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):

- ☐ `s = b`
- ☐ `b = (Beta) i`
- ☐ `g = (Gamma) o`
- ☐ `b = s`
- ☐ `n = new Number()`
- ☐ `n = (Omega) n`
- ☐ `o = s`
- ☐ `i = (Gamma) s`
- ☐ `i = (Gamma) n`
- ☐ `s = (Strana) o`

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

```
    int x = 0;
    try {
      for (String s: parole)
        x = s.charAt(0) == s.charAt(s.length() - 1) ? 0: 2 * x + 1;
      x = parole[parole.length / x].length();
    } catch (ArithmeticException e) {
      x = x + 11;
    } catch (ArrayIndexOutOfBoundsException e) {
      x = x + 35;
    } catch (StringIndexOutOfBoundsException e) {
      x = x + 25;
    } catch (NullPointerException e) {
      x = x + 21;
    }
    return x;
  }
```

Ricordando che:

- il metodo `chatAt` della classe `String` restituisce il carattere che si trova nella posizione ricevuta tramite il parametro di tipo `int`, sollevando una `StringIndexOutOfBoundsException` se tale posizione non è presente nella stringa che esegue il metodo,
- `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`,
- "" indica la stringa vuota,

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

(a) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "", "`orso`". [ ]

(b) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`pollo`", "`cane`", "`orso`". [ ]

(c) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`orso`", "`pollo`", "`cane`". [ ]

(d) L'array riferito da `parole` contiene (nell'ordine indicato) riferimenti a oggetti che rappresentano le stringhe "`cane`", "`orso`", "`pollo`". [ ]

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

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

f(4)	f(5)