Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 15 giugno 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 il numero 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 un valore di tipo `double` uguale alla media dei valori dei *soli oggetti* di tipo `Double` contenuti nell'array `ar` (nel caso non vi siano oggetti di tipo `Double` fornite un messaggio appropriato).

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 il valore `NaN` (Not-a-Number).

4. Considerate le seguenti classi:

```
public class Strana {
  private String s;
  private static int y = 4;

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

  public Strana(int k) {
    this((new Integer(k)).toString());
  }

  public int length() {
    return s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana("elefante");
    System.out.println(x.length());           //2
    System.out.println(Strana.getStatico());  //3
    x = new Strana(444);
    System.out.println(x.length());           //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` e implementa un'interfaccia `In`, una classe concreta `Omega` che estende `Number`, una classe concreta `Delta` che estende `Double`.

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

- ☐ Se il codice sorgente della classe `Omega` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Delta` deve fornire l'implementazione del metodo `isNaN`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
- ☐ `Delta` è usa sottoclasse di `Gamma`
- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Delta` deve fornire l'implementazione del metodo `doubleValue`
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Gamma` deve fornire l'implementazione del metodo `length`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n, Object o, Gamma g, Long l, 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):

- ☐ `n = new Omega(...)`
- ☐ `o = n`
- ☐ `l = (Long) n`
- ☐ `n = new Number()`
- ☐ `g = (Gamma) i`
- ☐ `o = i`
- ☐ `l = (Long) i`
- ☐ `g = d`
- ☐ `d = g`
- ☐ `d = (Delta) 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(x) == 'c' ? 2 * x + 1 : 0;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 22;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 15;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 27;
} catch (NullPointerException e) {
  x = x + 44;
}
```

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`,

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`", "`oca`", "`pollo`". ☐

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

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

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

7. Date una variabile `a` di tipo `double`, si considerino le seguenti condizioni:

il valore contenuto in `a` è maggiore di 12

il valore contenuto in `a` è al massimo 15

(a) Esprimete in linguaggio Java la *congiunzione* delle due condizioni.

(b) Esprimete in linguaggio Java la *negazione* dell'espressione ottenuta al punto (a), *senza utilizzare* l'operatore di negazione.

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 15 giugno 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 il numero 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 un valore di tipo `float` uguale alla media dei valori dei *soli oggetti* di tipo `Float` contenuti nell'array `ar` (nel caso non vi siano oggetti di tipo `Float` fornite un messaggio appropriato).

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 il valore `NaN` (Not-a-Number).

4. Considerate le seguenti classi:

```
public class Strana {
  private String s;
  private static int y = 7;

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

  public Strana(int k) {
    this((new Integer(k)).toString());
  }

  public int length() {
    return s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana("gatto");
    System.out.println(x.length());           //2
    System.out.println(Strana.getStatico());  //3
    x = new Strana(1024);
    System.out.println(x.length());           //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` e implementa un'interfaccia `In`, una classe concreta `Omega` che estende `Number`, una classe concreta `Beta` che estende `Float`.

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
- ☐ `Beta` deve fornire l'implementazione del metodo `floatValue`
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Alfa` deve fornire l'implementazione del metodo `length`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ `Beta` è usa sottoclasse di `Alfa`
- ☐ Se il codice sorgente della classe `Omega` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Beta` deve fornire l'implementazione del metodo `isNaN`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n, Object o, Alfa a, Long l, 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):

- ☐ `o = n`
- ☐ `l = (Long) i`
- ☐ `a = b`
- ☐ `b = a`
- ☐ `l = (Long) n`
- ☐ `n = new Number()`
- ☐ `a = (Alfa) i`
- ☐ `o = i`
- ☐ `n = new Omega(...)`
- ☐ `b = (Beta) 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(x) == 'c' ? 2 * x + 1 : 0;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 12;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 19;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 28;
} catch (NullPointerException e) {
  x = x + 36;
}
```

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`,

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`", "`oca`". ☐

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

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

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

7. Date una variabile `b` di tipo `float`, si considerino le seguenti condizioni:

il valore contenuto in `b` non supera 3

il valore contenuto in `b` è più di 33

(a) Esprimete in linguaggio Java la *disgiunzione* delle due condizioni.

(b) Esprimete in linguaggio Java la *negazione* dell'espressione ottenuta al punto (a), *senza utilizzare* l'operatore di negazione.

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 15 giugno 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 il numero 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 un valore di tipo `double` uguale alla media dei valori dei *soli oggetti* di tipo `Double` contenuti nell'array `el` (nel caso non vi siano oggetti di tipo `Double` fornite un messaggio appropriato).

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 il valore `NaN` (Not-a-Number).

4. Considerate le seguenti classi:

```
public class Strana {
  private String s;
  private static int y = 3;

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

  public Strana(int k) {
    this((new Integer(k)).toString());
  }

  public int length() {
    return s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana("topo");
    System.out.println(x.length());           //2
    System.out.println(Strana.getStatico());  //3
    x = new Strana(12321);
    System.out.println(x.length());           //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` e implementa un'interfaccia `In`, una classe concreta `Omega` che estende `Number`, una classe concreta `Alfa` che estende `Double`.

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

- ☐ `Alfa` deve fornire l'implementazione del metodo `isNaN`
- ☐ `Delta` deve fornire l'implementazione del metodo `length`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ Se il codice sorgente della classe `Omega` 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
- ☐ `Alfa` deve fornire l'implementazione del metodo `doubleValue`
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`
- ☐ `Alfa` è usa sottoclasse di `Delta`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n, Object o, Delta d, Long l, 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):

- ☐ `a = d`
- ☐ `a = (Alfa) o`
- ☐ `d = (Delta) i`
- ☐ `o = i`
- ☐ `l = (Long) i`
- ☐ `n = new Omega(...)`
- ☐ `o = n`
- ☐ `l = (Long) n`
- ☐ `n = new Number()`
- ☐ `d = a`

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(x) == 'c' ? 2 * x + 1 : 0;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 40;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 35;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 18;
} catch (NullPointerException e) {
  x = x + 25;
}
```

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`,

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`", "`oca`", "`oca`". ☐

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

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

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

7. Date una variabile `c` di tipo `double`, si considerino le seguenti condizioni:

il valore contenuto in `c` è almeno 13

il valore contenuto in `c` è minore di 127

(a) Esprimete in linguaggio Java la *congiunzione* delle due condizioni.

(b) Esprimete in linguaggio Java la *negazione* dell'espressione ottenuta al punto (a), *senza utilizzare* l'operatore di negazione.

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 15 giugno 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 il numero 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 un valore di tipo `float` uguale alla media dei valori dei *soli oggetti* di tipo `Float` contenuti nell'array `el` (nel caso non vi siano oggetti di tipo `Float` fornite un messaggio appropriato).

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 il valore `NaN` (Not-a-Number).

4. Considerate le seguenti classi:

```
public class Strana {
  private String s;
  private static int y = 8;

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

  public Strana(int k) {
    this((new Integer(k)).toString());
  }

  public int length() {
    return s.length();
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.getStatico());  //1
    Strana x = new Strana("ape");
    System.out.println(x.length());           //2
    System.out.println(Strana.getStatico());  //3
    x = new Strana(99);
    System.out.println(x.length());           //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` e implementa un'interfaccia `In`, una classe concreta `Omega` che estende `Number`, una classe concreta `Gamma` che estende `Float`.

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

- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Beta` deve fornire l'implementazione del metodo `length`
- ☐ Se il codice sorgente della classe `Omega` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Gamma` deve fornire l'implementazione del metodo `isNaN`
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Gamma` deve fornire l'implementazione del metodo `floatValue`
- ☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
- ☐ Ogni istanza di `Strana` contiene esattamente un campo (oltre a quelli ereditati dalla superclasse)
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ `Gamma` è usa sottoclasse di `Beta`

b. Considerate le seguenti dichiarazioni di variabile:

```
Number n, Object o, Beta b, Long l, 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):

- ☐ `l = (Long) i`
- ☐ `b = (Beta) i`
- ☐ `o = i`
- ☐ `b = g`
- ☐ `o = n`
- ☐ `l = (Long) n`
- ☐ `n = new Number()`
- ☐ `g = b`
- ☐ `g = (Gamma) o`
- ☐ `n = new Omega(...)`

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(x) == 'c' ? 2 * x + 1 : 0;
  x = parole[parole.length / x].length();
} catch (ArithmeticException e) {
  x = x + 33;
} catch (ArrayIndexOutOfBoundsException e) {
  x = x + 27;
} catch (StringIndexOutOfBoundsException e) {
  x = x + 11;
} catch (NullPointerException e) {
  x = x + 19;
}
```

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`,

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 "`oca`", "`pollo`", "`cane`". ☐

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

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

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

7. Date una variabile `d` di tipo `float`, si considerino le seguenti condizioni:

il valore contenuto in `d` è minore di 16

il valore contenuto in `d` è almeno 25

(a) Esprimete in linguaggio Java la *disgiunzione* delle due condizioni.

(b) Esprimete in linguaggio Java la *negazione* dell'espressione ottenuta al punto (a), *senza utilizzare* l'operatore di negazione.