Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 23 febbraio 2016

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe `Object`.
  Ogni istanza di questa classe rappresenta un oggetto generico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Object` vi è:

  ```
  public boolean equals(Object o)
  ```

- Classe `String`.
  Ogni istanza di questa classe rappresenta una stringa di caratteri. La classe implementa l'interfaccia `Comparable` e pertanto fornisce il seguente metodo:

  `public int compareTo(String altra)`
  Confronta la stringa che esegue il metodo con quella fornita tramite l'argomento. Restituisce un valore negativo quando, in ordine lessicografico, la stringa che esegue il metodo precede quella fornita tramite l'argomento, un valore positivo quando la stringa che esegue il metodo segue quella fornita tramite l'argomento, 0 quando le due stringhe sono uguali.

  Tra gli altri metodi forniti dalla classe vi è:

  `public int length()`
  Restituisce la lunghezza della stringa che esegue il metodo.

  Inoltre, il metodo `equals` di `Object` è ridefinito allo scopo di controllare l'uguaglianza tra stringhe.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Object[] dati` e `String s`. Supponete di disporre di una porzione di codice alla fine della quale `dati` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Object` (quindi nessuna delle posizioni contiene `null`) e `s` si riferisca a un oggetto di tipo `String` costruito in precedenza.

1. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di stringhe contenute nell'array `dati` che siano uguali alla stringa riferita dalla variabile `s`.

2. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di oggetti contenuti nell'array `dati` che *non siano istanze* della classe `String`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media delle lunghezze delle stringhe contenute nell'array `dati` che, in ordine lessicografico, precedono la stringa riferita dalla variabile `s`. Nel caso l'array non contenga alcuna stringa che soddisfi tale condizione, al posto della media deve essere visualizzato un opportuno messaggio.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "gatto";
  private int k = 400;
  private int w;

  public Strana() {}

  public Strana(String t) {
    w = k / 2;
    k = k + t.length();
    s = k + w + s;
  }

  public int intValue() {
    return k + w;
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.length());  //1
    Strana s = new Strana("cicala");
    System.out.println(s.intValue());     //2
    s = new Strana();
    System.out.println(s.intValue());     //3
    System.out.println(Strana.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 astratta `Alfa` che estende `Strana`, una classe concreta `Beta` che estende `Alfa` e una classe concreta `Gamma` che estende `Object`. Considerate inoltre un'interfaccia `In` implementata solo dalla classe `Beta`.

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

- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Beta` deve fornire l'implementazione di tutti i metodi di `Alfa`
- ☐ `Gamma` può contenere un metodo astratto
- ☐ `Beta` deve fornire l'implementazione dei metodi di `In`
- ☐ La classe `Alfa` possiede un costruttore
- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 3 campi (oltre a quelli ereditati dalla superclasse)
- ☐ I costruttori di `Strana` richiamano il costruttore senza argomenti della superclasse
- ☐ Non esistono superclassi di `Strana`

b. Considerate le seguenti dichiarazioni di variabile:

```
Object o, Strana s, Alfa a, Beta b, Gamma g, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `o = i`
- ☐ `i = (In) o`
- ☐ `o = new Object()`
- ☐ `a = b`
- ☐ `s = (Alfa) i`
- ☐ `g = b`
- ☐ `i = b`
- ☐ `s = (Beta) i`
- ☐ `s = g`
- ☐ `g = s`

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

```
int x = 4;
try {
  for (Object o: oggetti)
    x = o instanceof String ? ((String)o).length() : x / 2;
  String s = (String) oggetti[x / x];
  x = s.length();
} catch (ArithmeticException e) {
  x = 31 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 81 + x;
} catch (NullPointerException e) {
  x = 91 + x;
} catch (ClassCastException e) {
  x = 51 + x;
}
```

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`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,
- `""` 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 `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, un riferimento a `""`, `null`.

(b) `oggetti` contiene `null`.

(c) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, e i riferimenti a `""` e a `"null"`.

(d) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a `"gatto"`, il riferimento a un oggetto costruito mediante `new Object()`, `null`.

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 + 2;
  else
    return 3 * f(x / 2);
}
```

f(4)	f(10)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 23 febbraio 2016

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe `Object`.
  Ogni istanza di questa classe rappresenta un oggetto generico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Object` vi è:

  ```
  public boolean equals(Object o)
  ```

- Classe `String`.
  Ogni istanza di questa classe rappresenta una stringa di caratteri. La classe implementa l'interfaccia `Comparable` e pertanto fornisce il seguente metodo:

  `public int compareTo(String altra)`
  Confronta la stringa che esegue il metodo con quella fornita tramite l'argomento. Restituisce un valore negativo quando, in ordine lessicografico, la stringa che esegue il metodo precede quella fornita tramite l'argomento, un valore positivo quando la stringa che esegue il metodo segue quella fornita tramite l'argomento, 0 quando le due stringhe sono uguali.

  Tra gli altri metodi forniti dalla classe vi è:

  `public int length()`
  Restituisce la lunghezza della stringa che esegue il metodo.

  Inoltre, il metodo `equals` di `Object` è ridefinito allo scopo di controllare l'uguaglianza tra stringhe.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Object[] elem` e `String s`. Supponete di disporre di una porzione di codice alla fine della quale `elem` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Object` (quindi nessuna delle posizioni contiene `null`) e `s` si riferisca a un oggetto di tipo `String` costruito in precedenza.

1. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di stringhe contenute nell'array `elem` che siano uguali alla stringa riferita dalla variabile `s`.

2. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di oggetti contenuti nell'array `elem` che *non siano istanze* della classe `String`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media delle lunghezze delle stringhe contenute nell'array `elem` che, in ordine lessicografico, precedono la stringa riferita dalla variabile `s`. Nel caso l'array non contenga alcuna stringa che soddisfi tale condizione, al posto della media deve essere visualizzato un opportuno messaggio.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "elefante";
  private int k = 200;
  private int w;

  public Strana() {}

  public Strana(String t) {
    w = k / 2;
    k = k + t.length();
    s = k + w + s;
  }

  public int intValue() {
    return k + w;
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.length());  //1
    Strana s = new Strana("topo");
    System.out.println(s.intValue());     //2
    s = new Strana();
    System.out.println(s.intValue());     //3
    System.out.println(Strana.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 astratta `Gamma` che estende `Strana`, una classe concreta `Delta` che estende `Gamma` e una classe concreta `Alfa` che estende `Object`. Considerate inoltre un'interfaccia `In` implementata solo dalla classe `Delta`.

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

- ☐ Se il codice sorgente della classe `Gamma` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 3 campi (oltre a quelli ereditati dalla superclasse)
- ☐ I costruttori di `Strana` richiamano il costruttore senza argomenti della superclasse
- ☐ Non esistono superclassi di `Strana`
- ☐ `Alfa` può contenere un metodo astratto
- ☐ `Delta` deve fornire l'implementazione dei metodi di `In`
- ☐ La classe `Gamma` possiede un costruttore
- ☐ Se il codice sorgente della classe `Alfa` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Delta` deve fornire l'implementazione di tutti i metodi di `Gamma`

b. Considerate le seguenti dichiarazioni di variabile:

```
Object o, Strana s, Gamma g, Delta d, Alfa a, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `i = (In) o`
- ☐ `i = d`
- ☐ `s = (Delta) i`
- ☐ `s = a`
- ☐ `o = new Object()`
- ☐ `g = d`
- ☐ `s = (Gamma) i`
- ☐ `a = d`
- ☐ `o = i`
- ☐ `a = s`

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

```
int x = 4;
try {
  for (Object o: oggetti)
    x = o instanceof String ? ((String)o).length() : x / 2;
  String s = (String) oggetti[x / x];
  x = s.length();
} catch (ArithmeticException e) {
  x = 91 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 81 + x;
} catch (NullPointerException e) {
  x = 51 + x;
} catch (ClassCastException e) {
  x = 31 + x;
}
```

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`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,
- `""` 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 `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, e i riferimenti a `""` e a `"null"`.

(b) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a `"gatto"`, il riferimento a un oggetto costruito mediante `new Object()`, `null`.

(c) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, un riferimento a `""`, `null`.

(d) `oggetti` contiene `null`.

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 + 3;
  else
    return 2 * f(x / 2);
}
```

f(4)	f(10)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 23 febbraio 2016

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe `Object`.
  Ogni istanza di questa classe rappresenta un oggetto generico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Object` vi è:

  ```
  public boolean equals(Object o)
  ```

- Classe `String`.
  Ogni istanza di questa classe rappresenta una stringa di caratteri. La classe implementa l'interfaccia `Comparable` e pertanto fornisce il seguente metodo:

  `public int compareTo(String altra)`
  Confronta la stringa che esegue il metodo con quella fornita tramite l'argomento. Restituisce un valore negativo quando, in ordine lessicografico, la stringa che esegue il metodo precede quella fornita tramite l'argomento, un valore positivo quando la stringa che esegue il metodo segue quella fornita tramite l'argomento, 0 quando le due stringhe sono uguali.

  Tra gli altri metodi forniti dalla classe vi è:

  `public int length()`
  Restituisce la lunghezza della stringa che esegue il metodo.

  Inoltre, il metodo `equals` di `Object` è ridefinito allo scopo di controllare l'uguaglianza tra stringhe.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Object[] vett` e `String s`. Supponete di disporre di una porzione di codice alla fine della quale `vett` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Object` (quindi nessuna delle posizioni contiene `null`) e `s` si riferisca a un oggetto di tipo `String` costruito in precedenza.

1. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di stringhe contenute nell'array `vett` che siano uguali alla stringa riferita dalla variabile `s`.

2. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di oggetti contenuti nell'array `vett` che *non siano istanze* della classe `String`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media delle lunghezze delle stringhe contenute nell'array `vett` che, in ordine lessicografico, seguono la stringa riferita dalla variabile `s`. Nel caso l'array non contenga alcuna stringa che soddisfi tale condizione, al posto della media deve essere visualizzato un opportuno messaggio.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "formica";
  private int k = 800;
  private int w;

  public Strana() {}

  public Strana(String t) {
    w = k / 2;
    k = k + t.length();
    s = k + w + s;
  }

  public int intValue() {
    return k + w;
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.length());  //1
    Strana s = new Strana("gatto");
    System.out.println(s.intValue());     //2
    s = new Strana();
    System.out.println(s.intValue());     //3
    System.out.println(Strana.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 astratta `Beta` che estende `Strana`, una classe concreta `Alfa` che estende `Beta` e una classe concreta `Delta` che estende `Object`. Considerate inoltre un'interfaccia `In` implementata solo dalla classe `Alfa`.

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

- ☐ `Alfa` deve fornire l'implementazione di tutti i metodi di `Beta`
- ☐ Non esistono superclassi di `Strana`
- ☐ `Delta` può contenere un metodo astratto
- ☐ Se il codice sorgente della classe `Delta` 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
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 3 campi (oltre a quelli ereditati dalla superclasse)
- ☐ I costruttori di `Strana` richiamano il costruttore senza argomenti della superclasse
- ☐ `Alfa` deve fornire l'implementazione dei metodi di `In`
- ☐ La classe `Beta` possiede un costruttore

b. Considerate le seguenti dichiarazioni di variabile:

```
Object o, Strana s, Beta b, Alfa a, Delta d, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `s = d`
- ☐ `d = s`
- ☐ `s = (Beta) i`
- ☐ `d = a`
- ☐ `i = a`
- ☐ `o = i`
- ☐ `i = (In) o`
- ☐ `o = new Object()`
- ☐ `b = a`
- ☐ `s = (Alfa) i`

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

```
int x = 4;
try {
  for (Object o: oggetti)
    x = o instanceof String ? ((String)o).length() : x / 2;
  String s = (String) oggetti[x / x];
  x = s.length();
} catch (ArithmeticException e) {
  x = 81 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 51 + x;
} catch (NullPointerException e) {
  x = 31 + x;
} catch (ClassCastException e) {
  x = 91 + x;
}
```

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`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,
- `""` indica la stringa vuota,

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

(a) `oggetti` contiene `null`. ☐

(b) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, un riferimento a `""`, `null`. ☐

(c) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a `"gatto"`, il riferimento a un oggetto costruito mediante `new Object()`, `null`. ☐

(d) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, e i riferimenti a `""` e a `"null"`. ☐

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 + 1;
  else
    return 3 * f(x / 2);
}
```

f(4)	f(10)

Cognome.....

Nome.....

Matricola.....

Programmazione

Prova scritta del 23 febbraio 2016

TEMPO DISPONIBILE: 1 ora e 40 minuti

Negli esercizi proposti si utilizzano le seguenti classi:

- Classe `Object`.
  Ogni istanza di questa classe rappresenta un oggetto generico. La classe possiede un costruttore privo di argomenti. Tra i metodi della classe `Object` vi è:

  ```
  public boolean equals(Object o)
  ```

- Classe `String`.
  Ogni istanza di questa classe rappresenta una stringa di caratteri. La classe implementa l'interfaccia `Comparable` e pertanto fornisce il seguente metodo:

  `public int compareTo(String altra)`
  Confronta la stringa che esegue il metodo con quella fornita tramite l'argomento. Restituisce un valore negativo quando, in ordine lessicografico, la stringa che esegue il metodo precede quella fornita tramite l'argomento, un valore positivo quando la stringa che esegue il metodo segue quella fornita tramite l'argomento, 0 quando le due stringhe sono uguali.

  Tra gli altri metodi forniti dalla classe vi è:

  `public int length()`
  Restituisce la lunghezza della stringa che esegue il metodo.

  Inoltre, il metodo `equals` di `Object` è ridefinito allo scopo di controllare l'uguaglianza tra stringhe.

Negli esercizi 1, 2 e 3 considerate le dichiarazioni di variabile `Object[] memo` e `String s`. Supponete di disporre di una porzione di codice alla fine della quale `memo` si riferisca a un array in cui ogni posizione contiene il riferimento a un oggetto di tipo `Object` (quindi nessuna delle posizioni contiene `null`) e `s` si riferisca a un oggetto di tipo `String` costruito in precedenza.

1. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di stringhe contenute nell'array `memo` che siano uguali alla stringa riferita dalla variabile `s`.

2. Scrivete una porzione di codice per contare e visualizzare sul monitor il numero di oggetti contenuti nell'array `memo` che *non siano istanze* della classe `String`.

3. Scrivete una porzione di codice per calcolare e visualizzare sul monitor la media delle lunghezze delle stringhe contenute nell'array `memo` che, in ordine lessicografico, seguono la stringa riferita dalla variabile `s`. Nel caso l'array non contenga alcuna stringa che soddisfi tale condizione, al posto della media deve essere visualizzato un opportuno messaggio.

4. Considerate le seguenti classi:

```
public class Strana {
  private static String s = "cicala";
  private int k = 1000;
  private int w;

  public Strana() {}

  public Strana(String t) {
    w = k / 2;
    k = k + t.length();
    s = k + w + s;
  }

  public int intValue() {
    return k + w;
  }

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

```
class Prova {
  public static void main(String[] args) {
    System.out.println(Strana.length());  //1
    Strana s = new Strana("ape");
    System.out.println(s.intValue());     //2
    s = new Strana();
    System.out.println(s.intValue());     //3
    System.out.println(Strana.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 astratta `Delta` che estende `Strana`, una classe concreta `Gamma` che estende `Delta` e una classe concreta `Beta` che estende `Object`. Considerate inoltre un'interfaccia `In` implementata solo dalla classe `Gamma`.

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

- ☐ I costruttori di `Strana` richiamano il costruttore senza argomenti della superclasse
- ☐ Non esistono superclassi di `Strana`
- ☐ Se il codice sorgente della classe `Beta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ `Gamma` deve fornire l'implementazione di tutti i metodi di `Delta`
- ☐ Se il codice sorgente della classe `Delta` non contiene un costruttore allora il compilatore segnala un errore
- ☐ Ogni istanza di `Strana` contiene esattamente 2 campi (oltre a quelli ereditati dalla superclasse)
- ☐ Ogni istanza di `Strana` contiene esattamente 3 campi (oltre a quelli ereditati dalla superclasse)
- ☐ `Beta` può contenere un metodo astratto
- ☐ `Gamma` deve fornire l'implementazione dei metodi di `In`
- ☐ La classe `Delta` possiede un costruttore

b. Considerate le seguenti dichiarazioni di variabile:

```
Object o, Strana s, Delta d, Gamma g, Beta b, In i;
```

Nel riquadro accanto a ciascun assegnamento scrivete SI se l'assegnamento è compilato correttamente, NO se non è compilato correttamente:

- ☐ `i = g`
- ☐ `s = (Delta) i`
- ☐ `b = g`
- ☐ `s = (Gamma) i`
- ☐ `i = (In) o`
- ☐ `o = new Object()`
- ☐ `d = g`
- ☐ `s = b`
- ☐ `b = s`
- ☐ `o = i`

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

```
int x = 4;
try {
  for (Object o: oggetti)
    x = o instanceof String ? ((String)o).length() : x / 2;
  String s = (String) oggetti[x / x];
  x = s.length();
} catch (ArithmeticException e) {
  x = 51 + x;
} catch (ArrayIndexOutOfBoundsException e) {
  x = 31 + x;
} catch (NullPointerException e) {
  x = 91 + x;
} catch (ClassCastException e) {
  x = 81 + x;
}
```

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`,
- `ClassCastException` viene sollevata quando si tenti di forzare un oggetto a una sottoclasse di cui l'oggetto non è istanza,
- `""` 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 `oggetti` contiene (nell'ordine indicato) il riferimento a `"gatto"`, il riferimento a un oggetto costruito mediante `new Object()`, `null`.

(b) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, e i riferimenti a `""` e a `"null"`.

(c) `oggetti` contiene `null`.

(d) l'array riferito da `oggetti` contiene (nell'ordine indicato) il riferimento a un oggetto costruito mediante `new Object()`, un riferimento a `""`, `null`.

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 + 2;
  else
    return 2 * f(x / 2);
}
```

f(4)	f(10)