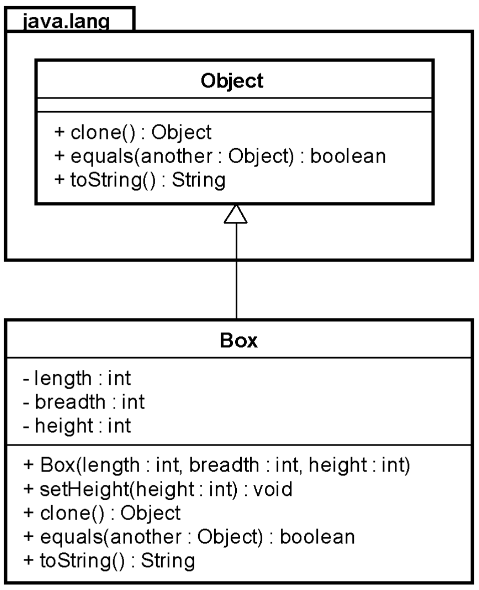
Inheritance 2

Resource: http://blue.smu.edu.sg/inheritance2-resource.zip

1. **[Method overriding]** A Box class represents a cuboid with 3 attributes (height, breadth & length).


java.lang
Object
+ clone() : Object
+ equals(another : Object) : boolean
+ toString() : String
Box
- length : int
- breadth : int
- height : int
+ Box(length : int, breadth : int, height : int)
+ setHeight(height : int) : void
+ clone() : Object
+ equals(another : Object) : boolean
+ toString() : String

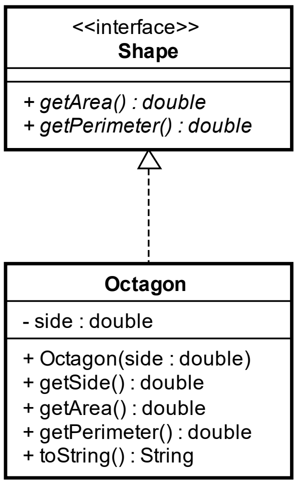
a. Implement the Box class overriding the equals(), toString() and clone() methods.
   - `equals()` takes in an `Object` and returns `true` if the input is another `Box` object that has the same dimensions.
   - `toString()` returns the dimensions as a `String`: "height: <height>, breadth: <breadth>, length: <length>"
   - `clone()` returns an entirely new `Box` object with the same dimensions.

   - Use `BoxTest` to check that you have written Box correctly. The output expected is as follows:

```
testing equals:
b1 and b2 are equal :true
```

```
b2 and b1 are equal :true

testing toString:
b1 :height: 1, breadth: 2, length: 3
b2 :height: 1, breadth: 2, length: 3

testing clone:
b1 :height: 1, breadth: 2, length: 3
b3 :height: 1, breadth: 2, length: 3

after changing the height of b1:
b1 :height: 5, breadth: 2, length: 3
b3 :height: 1, breadth: 2, length: 3
```

2. **[Interface]** Given the following class diagram:



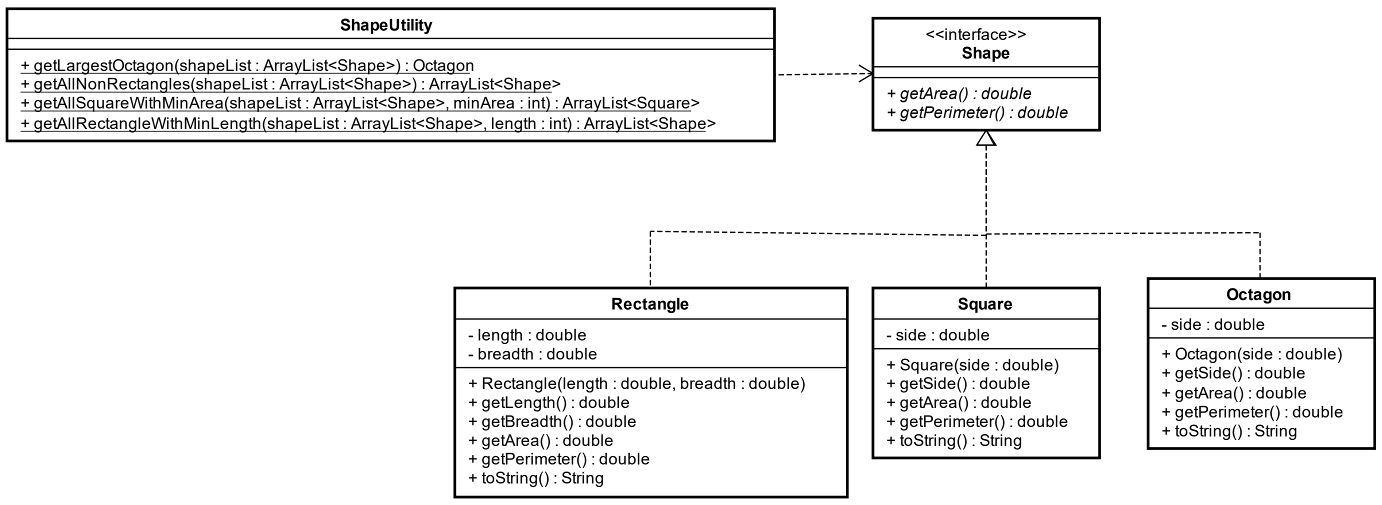
Note: The arrow should be dotted.

a. Implement the Shape interface.
b. Implement the Octagon class given the following formula:

Area of an octagon = $side^2 \times \left(2 + 2(\sqrt{2})\right)$
Perimeter of an octagon = $8 \times side$

An octagon whose side is 12 will have an area and perimeter of 695.29 and 96 respectively.

3. **[instanceof & casting]** Given the following class diagram.


ShapeUtility
+ getLargestOctagon(shapeList : ArrayList<Shape>) : Octagon
+ getAllNonRectangles(shapeList : ArrayList<Shape>) : ArrayList<Shape>
+ getAllSquareWithMinArea(shapeList : ArrayList<Shape>, minArea : int) : ArrayList<Square>
+ getAllRectangleWithMinLength(shapeList : ArrayList<Shape>, length : int) : ArrayList<Shape>
<<interface>>
Shape
+ getArea() : double
+ getPerimeter() : double
Rectangle
- length : double
- breadth : double
+ Rectangle(length : double, breadth : double)
+ getLength() : double
+ getBreadth() : double
+ getArea() : double
+ getPerimeter() : double
+ toString() : String
Square
- side : double
+ Square(side : double)
+ getSide() : double
+ getArea() : double
+ getPerimeter() : double
+ toString() : String
Octagon
- side : double
+ Octagon(side : double)
+ getSide() : double
+ getArea() : double
+ getPerimeter() : double
+ toString() : String

Implement the `ShapeUtility` class.

a. `getLargestOctagon` method returns the `Octagon` object with the largest area in `shapeList`.
   - This method returns null when `shapeList` is `null` or empty or no matching result.
   - If there is more than one result, return either one.

b. `getAllNonRectangles` method returns all the non-Rectangle objects in `shapeList`.
   - Do not assume that `shapeList` will only contain Rectangle, Square and Octagon objects.
   - This method returns an empty `ArrayList` when `shapeList` is `null` or when there is no matching result.

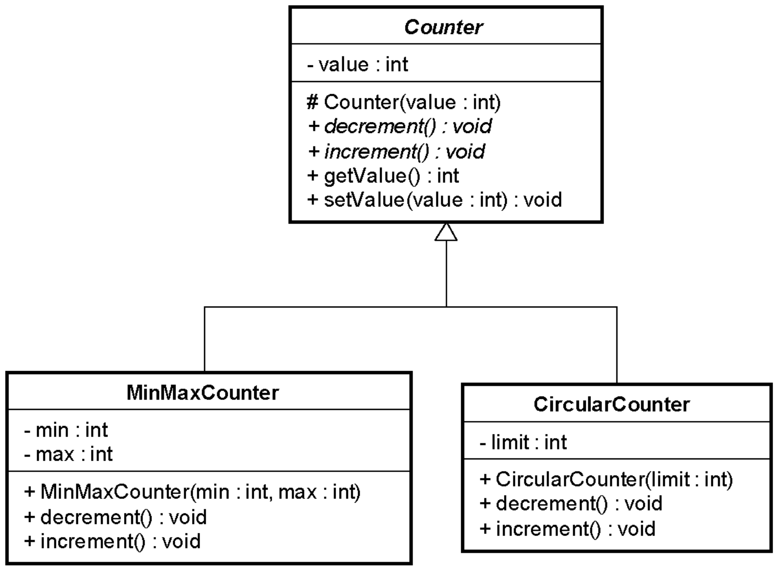
c. `getAllSquareWithMinArea` method returns all the `Square` objects which are of at least the specified `minArea` in `shapeList` (Square, Octagon etc).
   - This method returns an empty `ArrayList` when `shapeList` is `null` or when there is no matching result.

d. `getAllRectangleWithMinLength` method returns all the `Rectangle` objects with length greater or equal to the specified `minlength` in `shapeList`.
   - This method returns an empty `ArrayList` when `shapeList` is `null` or when there is no matching result.

Use the `ShapeUtilityTest` class to check that you have implemented the classes correctly. The output expected is as follows:

```
Test getLargestOctagon:
null
null
Octagon[side=8.0]

Test getAllNonRectangles:
[]
[]
[Octagon[side=3.0], Square[side=3.0], Octagon[side=4.0], Square[side=4.0], Octagon[side=8.0],
Square[side=8.0], Octagon[side=6.0], Square[side=6.0]]

Test getAllSquareWithMinArea:
[]
[]
[Square[side=8.0], Square[side=6.0]]

Test getAllRectangleWithMinLength:
[]
[]
[Rectangle[length=6.0,breadth=4.0], Rectangle[length=8.0,breadth=4.0]]
```

4. **[abstract class]** Write the following 3 classes based on the class diagram shown:

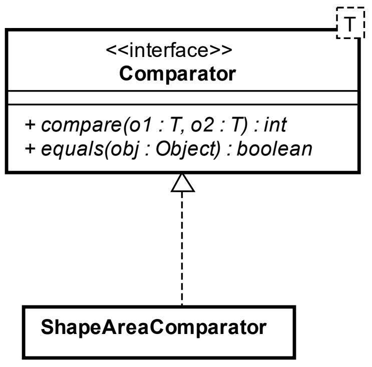


- Each call to `increment()` increases the Counter's `value` by 1
- `MinMaxCounter` will start counting from the `min` value, and stop at the `max` value. Once the value hits the `max` value, a call to `increment()` will not increment the value any more. Similarly once the value hits the min value, a call to `decrement()` will not decrement the value any more.
- `CircularCounter` will count from 0,1,2, .., limit, 0,1,2 ..., limit. . Each call to `increment()` increases `value` by 1. But `value` should never exceed `limit`, so if `value` is already `limit`, a call to `increment()` sets `value` back to 0. When `decrement()` is called after the value is 0, `value` is set to `limit`.

Use the `CounterTest` class to check that you have implemented the classes correctly. The sample output is as follows:

```
Testing CircularCounter:
1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2,

Testing MinMaxCounter (incrementing):
3, 4, 5, 5, 5, 5, 5, 5, 5, 5,
Testing MinMaxCounter (decrementing):
4, 3, 2, 2, 2, 2, 2, 2, 2, 2,
```

5. **[Comparator Interface]** Study the class diagram below and implement the `ShapeAreaComparator` class.


T
<<interface>>
Comparator
+ compare(o1 : T, o2 : T) : int
+ equals(obj : Object) : boolean
ShapeAreaComparator

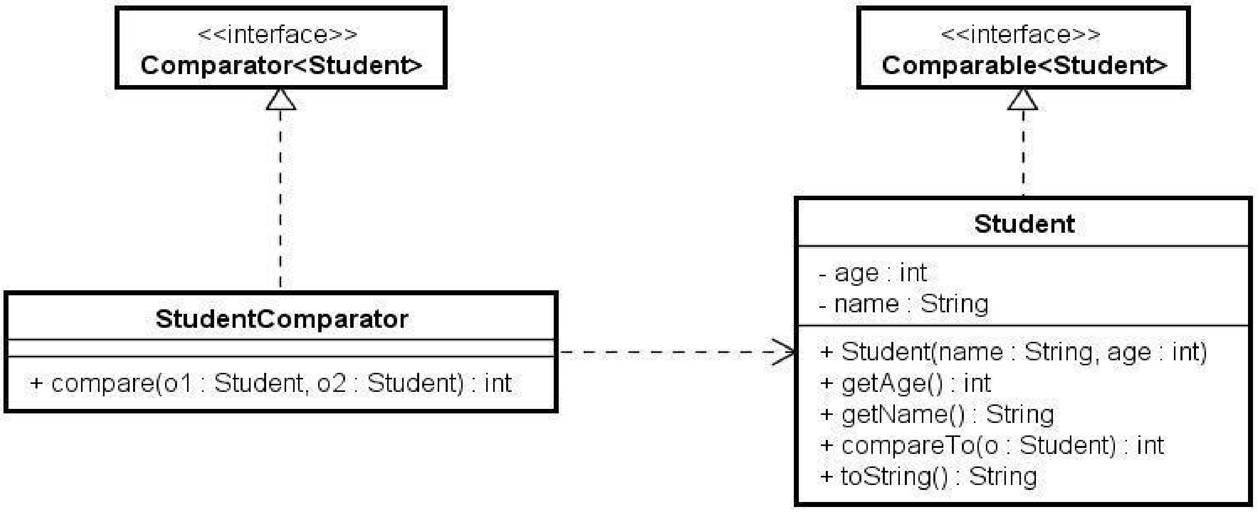
The compare method of `ShapeAreaComparator` will return a negative integer, zero, or a positive integer if `o1` is less than, equal to, or greater than the `o2` in terms of area.

Use the `ShapeAreaComparatorTest` class to check that you have implemented `ShapeAreaComparator` correctly. Study the test class to see how the `ShapeAreaComparator` is being used.

```
Before sorting:
43.46, 9.00, 12.00, 77.25, 16.00, 24.00, 309.02, 64.00, 32.00, 173.82, 36.00, 20.00,

After sorting:
9.00, 12.00, 16.00, 20.00, 24.00, 32.00, 36.00, 43.46, 64.00, 77.25, 173.82, 309.02,
```

OPTIONAL

6. **[**] [Comparator & Comparable]** Study the class diagram & sample output below and implement the `StudentComparator` and a `Student` class.



When `StudentTest` is executed, it should have the following output:

```
[Alex : 10, Alex : 30, Alex : 20, Billy : 1, Charlie : 9, Donkey : 30, Elise : 10,
Fanciful : 45, Gorilla : 43]
[Alex : 10, Alex : 20, Alex : 30, Billy : 1, Charlie : 9, Donkey : 30, Elise : 10,
Fanciful : 45, Gorilla : 43]
```

Note:

1. The first line of output shows that the values are sorted by name only.
2. The second line of output shows that the values are sorted by name, then age.

Below is the code from the Java SDK. The code that uses your Comparable code is bolded.

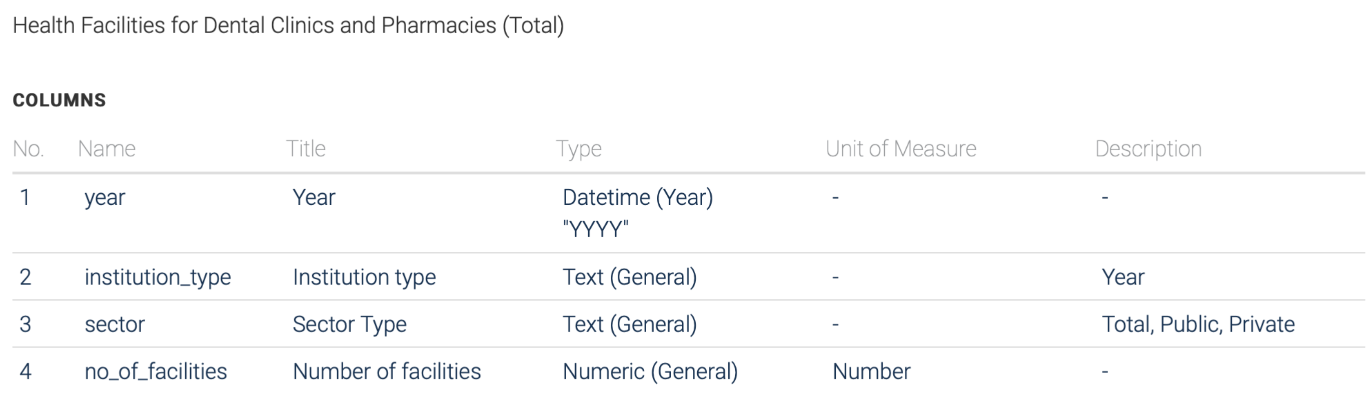
```
for ( ; start < hi; start++) {
        Comparable<Object> pivot = (Comparable) a[start];

        int left = lo;
        int right = start;

        while (left < right) {
            int mid = (left + right) >>> 1;
            if (pivot.compareTo(a[mid]) < 0)
                right = mid;
            else
                left = mid + 1;
        }
        assert left == right;

}
```

7. **[***]** Data.gov.sg was first launched in 2011 as the government's one-stop portal to its publicly-available datasets from 70 public agencies. To date, more than 100 apps have been created using the government's open data. In this exercise, we will explore writing a java application to retrieve data from one of these datasets, specifically (Health Facilities and Beds in Inpatient Facilities) using the APIs provided by Data.gov.sg.

   You can browse the following link to get an overview of the information returned via this API.
   https://data.gov.sg/dataset/health-facilities?resource_id=b698c206-a63c-489f-8c50-b439c349c025

   For successful calls to the API, it will return data in a **JSON** (JavaScript Object Notation) format. This is a lightweight data-interchange format that makes it easy for machines to parse and generate. For more information on JSON, refer to https://en.wikipedia.org/wiki/JSON

   The following link is a sample JSON output for calling this API, every dataset in data.gov.sg is identified by resource_id and limit refers to the number of records to be returned. Study the JSON output.
   https://data.gov.sg/api/action/datastore_search?resource_id=b698c206-a63c-489f-8c50-b439c349c025&limit=5.
   There are 3 main attributes that are returned from this API (help, success and result). if success returns true, result will store an array of information that corresponds to the following structure.

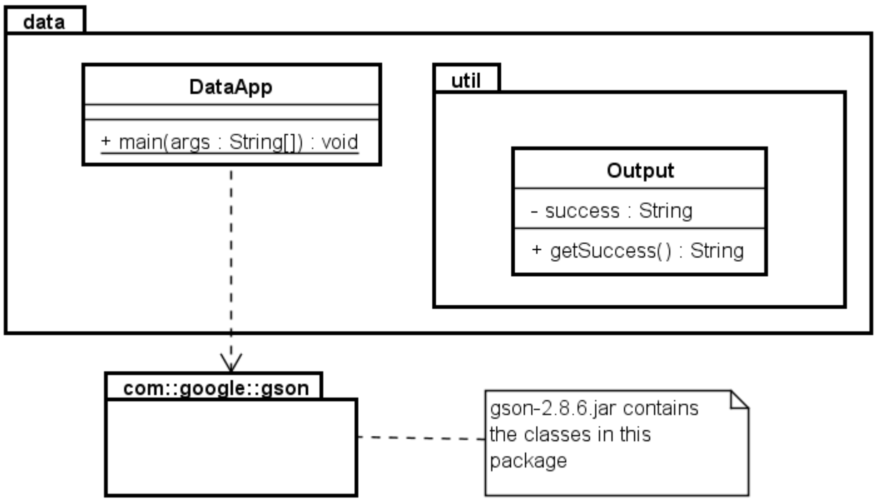


Health Facilities for Dental Clinics and Pharmacies (Total)

COLUMNS

No.	Name	Title	Type	Unit of Measure	Description
1	year	Year	Datetime (Year) "YYYY"	-	-
2	institution_type	Institution type	Text (General)	-	Year
3	sector	Sector Type	Text (General)	-	Total, Public, Private
4	no_of_facilities	Number of facilities	Numeric (General)	Number	-

You are given the following code, DataApp.java, Output.java and gson-2.8.6.jar. (external library used to parse the JSON object)


data
DataApp
+ main(args : String[]) : void
util
Output
- success : String
+ getSuccess() : String
com::google::gson
gson-2.8.6.jar contains the classes in this package

a. Package the Java classes accordingly.

b. Place your Java classes in the proper sub-directories in the `src` folder.

c. Write a one-liner command in compile.bat to compile `DataApp` so that all Java files are compiled and the class files are placed in the `classes` folder.

Write a one-liner command in run.bat to run class `DataApp`.

If the app runs successfully, you will get the following output.

```
Connecting to data.gov.sg......
Retrieve Success Status : true
```

It was mentioned earlier that there are 3 main attributes that are returned from this API (help, success and result). The app has managed to display the success value from the output. Make the necessary changes to the codes to process the result and print the values accordingly. You can make modifications to the codes or add new classes where applicable.

This would be the expected output.

```
Connecting to data.gov.sg......
Retrieve Success Status : true

Number of records : 3
1-Total-2009-822-Dental Clinics
2-Public-2009-240-Dental Clinics
3-Private-2009-582-Dental Clinics
```

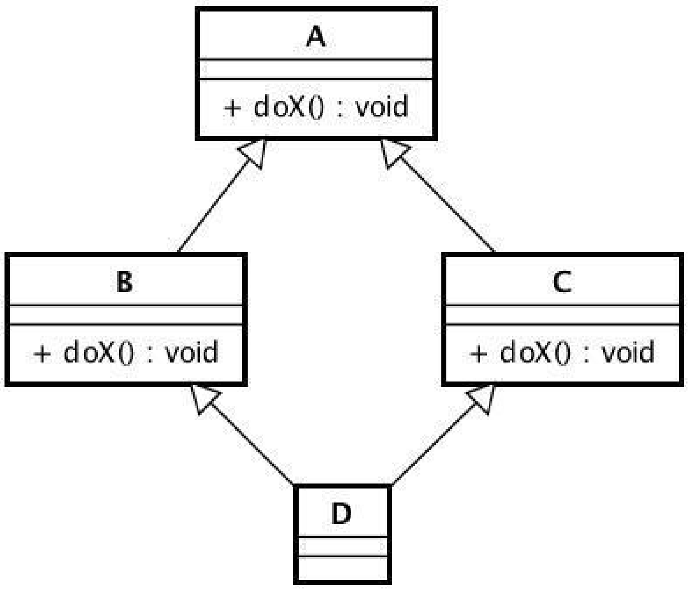
8. **[**]** This question requires the `Box` class from Q1. Compile and execute the `Demo` class. Is the answer what you expect?
   Note: `Set` is the Java version of `Set` in Python. A Set is an unordered collection data type that is iterable, mutable, and has **no duplicate elements**.

```
import java.util.*;

public class Demo {

   public static void main(String[] args) {
       Set<Box> map = new HashSet<>();

       map.add(new Box(12,12,12));
       map.add(new Box(12,12,12));

       System.out.println(map.size());
   }
}
```

Modify your `Box` class with the changes stated and re-run the `Demo` class. What is your observation? Read up the documentation for `Object`'s `equals` method documentation with regards to hash code.

```
public class Box {

    @Override
    public int hashCode() {
        return Objects.hash(length, breadth, height);
    }

}
```

9. **[***]** The "**diamond problem**" (also known as the "deadly **diamond** of death") is an ambiguity that arises when two classes B and C inherit from A, and class D inherits from both B and C. If there is a method in A that B and C have overridden, and D does not override it, then which version of the method does D inherit? D should use the doX() method in B or C?



a. In Java, multiple inheritance is not allowed for classes, only for interfaces. Python supports multiple inheritance. Run the code and observe the output.

```
class A:
   def sayHello(self):
       print('A')

class B:
   def sayHello(self):
       print('B')

class C:
   def sayHello(self):
       print('C')

class D(B, C):
   pass

obj = D()
obj.sayHello()
```

b. Insert the code at the end of the file. Run the code and observe the output.

```
print(D.mro())
```

Reference:

1. http://python-history.blogspot.com/2010/06/method-resolution-order.html
2. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.3910&rep=rep1&type=pdf
   (Read it if you have a strong heart!)

10. **[***]** In Java 8 and later, default methods are added. Run the following classes.

```
public interface Human {

   default void greet() {
      System.out.println("Happy New Year!");
   }
}
```

```
public class Student implements Human{

}
```

```
public class StudentTest {
    public static void main(String[] args) {
        Student s = new Student();
        s.greet();
    }
}
```

If this works, how about the diamond problem that was mentioned in question 8? Try adding the Robot interface. Are you able to compile the new Student class? How do you resolve this issue?
Reference: https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html

```
public interface Robot {

   default void greet() {
      System.out.println("Gong Xi Fa Cai!");
   }
}
```

```
public class Student implements Human, Robot {

}
```

11. The `List` interface contains a `forEach` method. An example is shown below:

```
List<Integer> numbers = Arrays.asList(1,2,3,4,5,6);
// :: is called Method Reference
// (reference to a single method)
numbers.forEach(System.out::println);
```

- END -