# CS102A: Introduction to Computer Programming
# Lab 13

**Objectives:**

- Practice interface

## [Demo]

First, create two interfaces, one of which represents animals that can speak and the other models movable things. Please put these two interfaces into two separate .java files.

```
public interface Animal {
    public abstract void speak();
}
```

```
public interface Movable {
    public abstract void move();
}
```

Then we modify the Human and Monkey classes defined in the previous lab. We make both classes implement the two interfaces. Please also put the two classes into the corresponding .java files.

```
public class Human implements Animal, Movable {
    @Override
    public void speak() {
        System.out.println("Hello, I am a human :)");
    }

    @Override
    public void move() {
        System.out.println("I jumped three feet.");
    }
}
```

```java
public class Monkey implements Animal, Movable {
    @Override
    public void speak() {
        System.out.println("I am a monkey, aaaa!");
    }

    @Override
    public void move() {
        System.out.println("I climbed up a banana tree.");
    }
}
```

After defining the interfaces and classes, we can write a main method to test them. Interfaces can be used to declare references to point to objects that are of the interface types.

```java
public class Demo1 {
    public static void main(String[] args) {
        Human human = new Human();
        Monkey monkey = new Monkey();
        Movable[] movables = {human, monkey};
        Animal[] animals = {human, monkey};
        System.out.println("*****Animal self-introduction*****");
        for(Animal a : animals) a.speak();
        System.out.println("*****Animal movement*****");
        for(Movable m : movables) m.move();
    }
}
```

The main method would print:

```
*****Animal self-introduction*****
Hello, I am a human :)
I am a monkey, aaaa!
*****Animal movement*****
I jumped three feet.
I climbed up a banana tree.
```

## [Exercise]

**Step1:** Please define an interface `Geometry` to represent geometric objects (rectangles, circles, and etc.). The interface has three abstract methods:

1. A method `area` that takes no argument and returns the area (`double` type) of the geometric object.
2. A method `perimeter` that takes no argument and returns the perimeter (`double` type) of the geometric object.
3. A method `draw` that takes no argument and returns a printable Unicode character that represents the geometric object.

Step 2: Please define a class `Rectangle` that implements the interface `Geometry`. The class has the following fields and methods:

1. A field of `double` type that represents the `height` of the rectangle
2. A field of `double` type that represents the `width` of the rectangle
3. A constructor that takes two arguments of `double` type to initialize the two fields, respectively.
4. The class should implement the `area` method.
5. The class should implement the `perimeter` method.
6. The class should implement the `draw` method. The code point of a black rectangle character ▬ in Unicode table is 25AC (i.e., the literal value of the character is '\u25AC').

Step 3: Please define a class `Circle` that implements the interface `Geometry`. The class has the following field and methods:

1. A field of `double` type that represents the `radius` of the circle
2. A constructor that takes one argument of `double` type to initialize the `radius` field
3. The class should implement the `area` method.
4. The class should implement the `perimeter` method.
5. The class should implement the `draw` method. The code point of a black circle character ● in Unicode table is 25CF (i.e., the literal value of the character is '\u25CF').

Step 4: Create a class `Exercise1` with the following `main` method to test the defined interface and classes:

```
public static void main(String[] args) {
    Rectangle rect = new Rectangle(3.0, 2.0);
    Circle circle = new Circle(3.0);
    Geometry[] geometries = {rect, circle};
    for(Geometry g : geometries) {
        System.out.printf("%c : perimeter = %.2f, area = %.2f\n", g.draw(),
g.perimeter(), g.area());
    }
}
```

If your implementation is correct, the `main` method would print:

```
▬ : perimeter = 10.00, area = 6.00
● : perimeter = 18.85, area = 28.27
```