Main.java

```
/*
- Inheritance
-
- Inheritance allows a new class to use the properties and methods of an existing class.
-
- Benefits:
-     1. Code Reusability
-     1. Hierarchical Classification (parent-child relationship)
-     1. Extensibility (adding unique attributes and methods)
-
- Parent Class (Superclass):
-     1. Foundation (common attributes and behaviors)
-     1. Generalization (represents a general concept)
-     1. Independent Existence (can exist independently)
-
- Child Class (Subclass):
-     1. Specialization (inherits attributes and methods)
-     1. Inherited Features (automatically acquires non-private properties and methods)
-     1. Multiple Inheritance (combining different behaviors, not supported in Java)
-
- Super Keyword:
-     1. Access Parent Class Features (access methods and properties)
-     1. Constructor Calls (call parent class constructor)
-     1. Avoid Naming Conflicts (distinguish between parent and child class methods/properties)
*/

class Shapes {
  String color;

  public Shapes(String color) {
  this.color = color;
  }

  public void displayColor() {
  System.out.println("My color is " + color);
  }

  public double area() {
  return 0;
  }
  }

  class Circle extends Shapes {
  String name;
  double radius;

  public Circle(String name, String color, double radius) {
  super(color);
  this.name = name;
  this.radius = radius;
  }

```

```
53  @Override
54  public double area() {
55  return 3.14 * (radius * radius);
56  }
57  }
58
59  class Rectangle extends Shapes {
60  String name;
61  int length;
62  int width;
63
64  public Rectangle(String name, String color, int length, int width) {
65  super(color);
66  this.name = name;
67  this.length = length;
68  this.width = width;
69  }
70
71  @Override
72  public double area() {
73  return length * width;
74  }
75  }
76
77  public class Main {
78  public static void main(String[] args) {
79  Circle myCircle = new Circle("myCircle", "pink", 5.8);
80  myCircle.displayColor();
81  System.out.println(myCircle.area());
82
83  Rectangle Rec = new Rectangle("myRec", "Red", 8, 5);
84  System.out.println(Rec.area());
85  Rec.displayColor();
86  }
87  }
88
```