抽象类可以用来声明变量，以存储其子类实例的引用。

```
abstract class Shape { // 抽象类
    public abstract double getArea();

    public void display() {
        System.out.println("这是一个形状");
    }
}

class Circle extends Shape { // Circle 类继承自 Shape 抽象类
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape { // Rectangle 类也继承自 Shape 抽象类
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double getArea() {
        return length * width;
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape1 = new Circle(5);    // 抽象类类型的变量 shape1 存储了 Circle 对象的引用
        Shape shape2 = new Rectangle(4, 6); // 抽象类类型的变量 shape2 存储了 Rectangle 对象的引用
        shape1.display(); // 调用父类的方法
        System.out.println("圆的面积：" + shape1.getArea()); // 调用子类重写的方法
        shape2.display();// 调用父类的方法
        System.out.println("矩形的面积：" + shape2.getArea());// 调用子类重写的方法
        //Shape shape3 = new Shape();//报错，抽象类不能被实例化
    }
}
```

抽象类可以并且经常包含实例变量，这些变量在子类对象中是存在的，并且可以通过子类对象进行访问。抽象类不能被实例化，指的是不能直接创建抽象类的对象，而不是指抽象类不能拥有实例变量。

```
abstract class Shape {
    protected String color; // 实例变量，存储形状的颜色

    public Shape(String color) { // 构造方法，用于初始化实例变量
        this.color = color;
    }

    public abstract double getArea(); // 抽象方法，获取面积

    public void displayColor() { // 具体方法，显示颜色
        System.out.println("颜色： " + color);
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(String color, double radius) {
        super(color); // 调用父类构造方法初始化 color
        this.radius = radius;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle("红色", 5);
        circle.displayColor(); // 输出：颜色：红色 (访问了抽象类的实例变量)
        System.out.println("圆的面积： " + circle.getArea());
    }
}
```