Interface

In Java, an interface is a blueprint for a class that defines a set of abstract methods that the implementing class must provide. Interfaces allow you to achieve abstraction and define a contract that classes must adhere to, promoting code flexibility and maintainability. Here are two code examples to illustrate the concept of Java interfaces:

Example 1: Basic Interface

```java
// Define an interface named Drawable
interface Drawable {
    void draw(); // Abstract method (no method body)
}

// Implement the Drawable interface in a class
class Circle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Rectangle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a rectangle");
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle();
        Rectangle rectangle = new Rectangle();

        circle.draw(); // Output: Drawing a circle
        rectangle.draw(); // Output: Drawing a rectangle
    }
}
```

In this example, we define an interface `Drawable` with a single abstract method `draw()`. We then have two classes, `Circle` and `Rectangle`, that implement the `Drawable` interface and

provide their own implementations of the `draw()` method. In the `main` method, we create objects of these classes and call their `draw()` methods.

Example 2: Using Interface for Multiple Inheritance

Java allows a class to implement multiple interfaces, which is a way to achieve multiple inheritance through interfaces. Here's an example:

```java
// Define two interfaces
interface Shape {
    void draw();
}

interface Color {
    void setColor(String color);
}

// Implement both interfaces in a single class
class ColoredCircle implements Shape, Color {
    private String circleColor;

    @Override
    public void draw() {
        System.out.println("Drawing a colored circle");
    }

    @Override
    public void setColor(String color) {
        this.circleColor = color;
    }

    public void printColor() {
        System.out.println("Circle color: " + circleColor);
    }
}

public class Main {
    public static void main(String[] args) {
        ColoredCircle coloredCircle = new ColoredCircle();
        coloredCircle.draw(); // Output: Drawing a colored circle
        coloredCircle.setColor("Red");
        coloredCircle.printColor(); // Output: Circle color: Red
    }
```

}
```

In this example, we define two interfaces, `Shape` and `Color`, each with their own methods. Then, we create a class `ColoredCircle` that implements both interfaces. This class provides implementations for both `draw()` and `setColor(String color)` methods. In the `main` method, we demonstrate how this class can be used to draw a colored circle and set its color.

These examples showcase how Java interfaces are used to define contracts for classes to implement and achieve abstraction and multiple inheritance in Java.
```