Program 1 : Simple Interface

```
interface Animal {
    void sound();
}

class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }
}

public class Interface1 {
    public static void main(String[] args) {
        Cat c = new Cat();
        c.sound();
    }
}
```

Output :

Cat meows

Program 2 : Interface Reference

```
interface Vehicle {
    void run();
}

class Car implements Vehicle {
    public void run() {
        System.out.println("Car is running");
    }
}
```

```
public class Interface2 {
    public static void main(String[] args) {
        Vehicle v = new Car();
        v.run();
    }
}
```

Output :

Car is running

Program 3 : Multiple Interfaces

```
interface A {
    void showA();
}

interface B {
    void showB();
}

class C implements A, B {
    public void showA() {
        System.out.println("Method A");
    }

    public void showB() {
        System.out.println("Method B");
    }
}

public class Interface3 {
    public static void main(String[] args) {
        C obj = new C();
```

```
        obj.showA();
        obj.showB();
    }
}
```

Output :

Method A

Method B

Program 4 : Interface with Default Methods

```
interface Test {
    default void display() {
        System.out.println("Default method");
    }
}

class Demo implements Test {
}

public class Interface4 {
    public static void main(String[] args) {
        Demo d = new Demo();
        d.display();
    }
}
```

Output :

Default method

Program 5 : Interface with Multiple Classes

```
interface Shape {
    void draw();
}

class Circle implements Shape {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

class Rectangle implements Shape {
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
}

public class Interface5 {
    public static void main(String[] args) {
        Shape s;

        s = new Circle();
        s.draw();

        s = new Rectangle();
        s.draw();
    }
}
```

Output :

Drawing Circle

Drawing Rectangle

Program 6 : Basic Abstract Method

```
abstract class Animal {
    abstract void sound();
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}

public class Abstract1 {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();
    }
}
```

Output :

Dog barks

Program 7 : Abstract Class with Normal Method

```
abstract class Vehicle {
    void start() {
        System.out.println("Vehicle starts");
    }
}

class Bike extends Vehicle {
}

public class Abstract2 {
```

```
    public static void main(String[] args) {
        Bike b = new Bike();
        b.start();
    }
}
```

Output :

Vehicle starts

Program 8 : Abstract + Normal Method Together

```
abstract class Shape {
    abstract void draw();

    void show() {
        System.out.println("Drawing shape");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Circle drawn");
    }
}

public class Abstract3 {
    public static void main(String[] args) {
        Circle c = new Circle();
        c.show();
        c.draw();
    }
}
```

Output :

Drawing shape

Circle drawn

Program 9 : Abstract Class Reference

```
abstract class Bank {
    abstract void interest();
}

class SBI extends Bank {
    void interest() {
        System.out.println("Interest is 6%");
    }
}

public class Abstract4 {
    public static void main(String[] args) {
        Bank b = new SBI();
        b.interest();
    }
}
```

Output :

Interest is 6%

Program 10 : Abstract Class with Constructor

```
abstract class Person {
    Person() {
        System.out.println("Person constructor");
    }
}
```

```
class Student extends Person {
}


public class Abstract5 {
    public static void main(String[] args) {
        Student s = new Student();
    }
}
```

Output :

```
 Person constructor
```