In Java, **an interface specifies the behavior of a class by providing an abstract type**. As one of Java's core concepts, [abstraction](https://www.simplilearn.com/tutorials/java-tutorial/abstract-class-in-java), [polymorphism](https://www.simplilearn.com/tutorials/java-tutorial/java-polymorphism), and multiple inheritance are supported through this technology. Interfaces are used in Java to achieve abstraction. By using the implements keyword, a java class can implement an interface.

In general terms, an interface can be defined as a container that stores the signatures of the methods to be implemented in the code segment. It improves the levels of Abstraction.

Following the brief introduction to Interface in Java, we will now be exploring why we need it and why we should prefer it over the conventional way of using an abstract class.

Need for Interface in Java

So we need an Interface in Java for the following reasons:

* Total Abstraction
* Multiple Inheritance
* Loose-Coupling

Total Abstraction

Abstraction is the critical concept of Object-Oriented programming techniques. An interface only stores the method signature and not the method definition. Method Signatures make an Interface achieve complete Abstraction by hiding the method implementation from the user.

Multiple Inheritance

Without Interface, the process of multiple inheritances is impossible as the conventional way of inheriting multiple parent classes results in profound ambiguity. This type of ambiguity is known as the Diamond problem. Interface resolves this issue.

Loose Coupling

The term Coupling describes the dependency of one class for the other. So, while using an interface, we define the method separately and the signature separately. This way, all the methods, and classes are entirely independent and archives Loose Coupling.

Now, let us check out how we can practically implement it—starting from its syntax to a real-time example.

Java Interface Syntax

So the Syntax of an Interface in Java is written as shown below.

Interface <Interface Name> {

//Declare Constant Fields;

//Declare Methods;

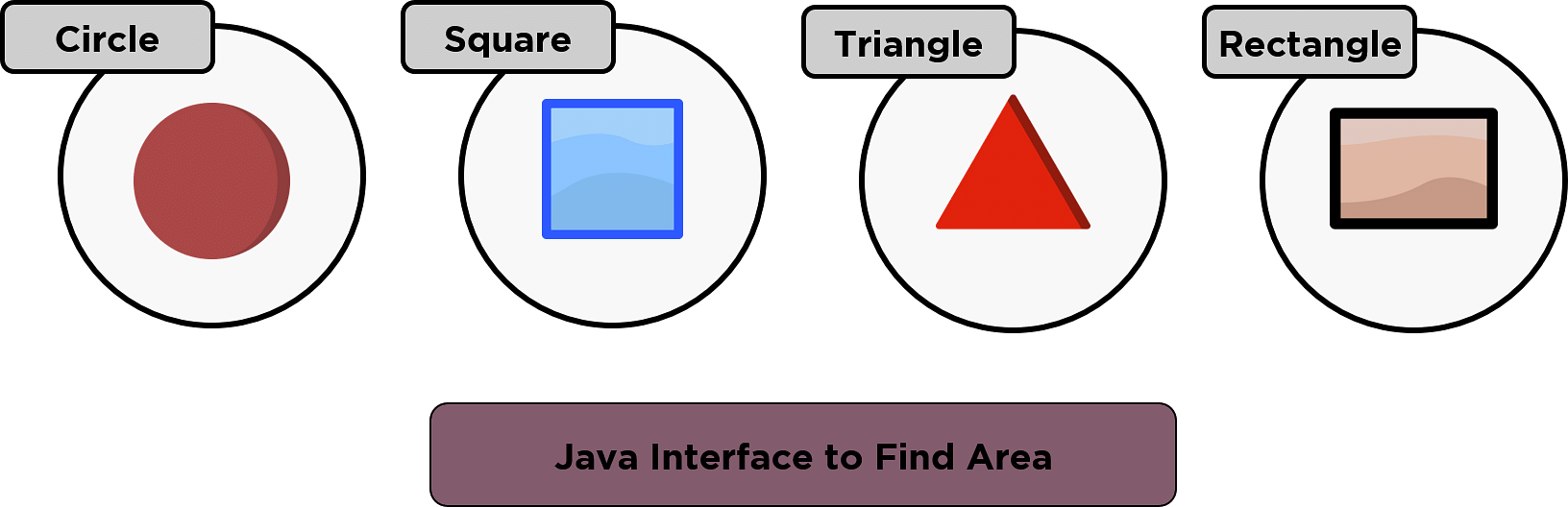
//Default Methods;

 }

With the syntax explained, let us now move ahead onto the next part, where we go through an example.

Java Interface Example

Following is an ideal example of Interface in Java. Here we try to calculate the area of geometrical shapes, and for each shape, we have different methods. And all the methods are defined, independent of each other. Only method signatures are written in the Interface.



//Interface

package simplilearn;

public interface Area {

public void Square();

public void Circle();

public void Rectangle();

public void Triangle();

}

//Class

package simplilearn;

import java.util.Scanner;

public class shapeArea implements Area {

public void Circle() {

Scanner kb = new Scanner(System.in);

System.out.println("Enter the radius of the circle");

double r = kb.nextInt();

double areaOfCircle = 3.142 \* r \* r;

System.out.println("Area of the circle is " + areaOfCircle);

}

@Override

public void Square() {

// TODO Auto-generated method stub

Scanner kb2 = new Scanner(System.in);

System.out.println("Enter the length of the side of the square");

double s = kb2.nextInt();

double areaOfSquare = s \* s;

System.out.println("Area of the square is " + areaOfSquare);

}

@Override

public void Rectangle() {

// TODO Auto-generated method stub

Scanner kb3 = new Scanner(System.in);

System.out.println("Enter the length of the Rectangle");

double l = kb3.nextInt();

System.out.println("Enter the breadth of the Rectangle");

double b = kb3.nextInt();

double areaOfRectangle = l \* b;

System.out.println("Area of the Rectangle is " + areaOfRectangle);

}

@Override

public void Triangle() {

// TODO Auto-generated method stub

Scanner kb4 = new Scanner(System.in);

System.out.println("Enter the base of the Triangle");

double base = kb4.nextInt();

System.out.println("Enter the height of the Triangle");

double h = kb4.nextInt();

double areaOfTriangle = 0.5 \* base \* h;

System.out.println("Area of the Triangle is " + areaOfTriangle);

}

public static void main(String[] args) {

shapeArea geometry = new shapeArea();

geometry.Circle();

geometry.Square();

geometry.Rectangle();

geometry.Triangle();

}

}

//ExpectedOutput:

Enter the radius of the circle

15

Area of the circle is 706.9499999999999

Enter the length of the side of the square

12

Area of the square is 144.0

Enter the length of the Rectangle

10

Enter the breadth of the Rectangle

25

Area of the Rectangle is 250.0

Enter the base of the Triangle

25

Enter the height of the Triangle

30

Area of the Triangle is 375.0

Followed by the example, we shall learn about another example of Nesting interfaces and why is it important.

Nesting Interface in Java

The purpose of using the procedure of the Nesting Interface is to resolve the namespace issues by grouping related interfaces or related interfaces with class.

So this particular example is based on nesting interfaces. Here, we are trying to print the first ‘n’ prime numbers using interface nesting.

//Interface Nesting

//Find First 15 Prime numbers

package InnerInterface;

public interface InterfaceOuter {

void display();

interface InterfaceInner {

void InnerMethod();

}

}

//class

package InnerInterface;

import InnerInterface.InterfaceOuter.InterfaceInner;

public class NestedInterface implements InterfaceInner {

public void InnerMethod() {

int iteration = 0, num = 0, x = 1, y = 1;

while (num < 15) {

y = 1;

iteration = 0;

while (y <= x) {

if (x % y == 0)

iteration++;

y++;

}

if (iteration == 2) {

System.out.printf("%d ", x);

num++;

}

x++;

}

}

public static void main(String args[]) {

NestedInterface obj = new NestedInterface();

obj.InnerMethod();

}

}

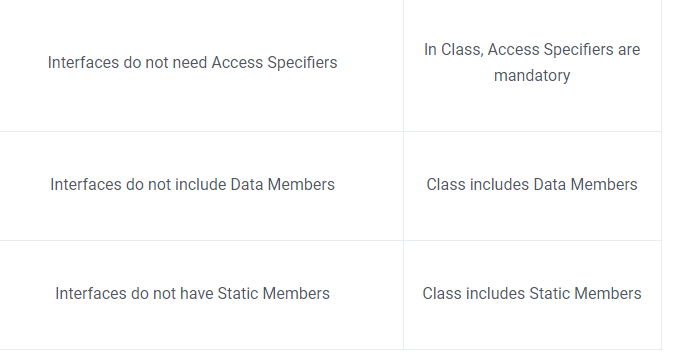
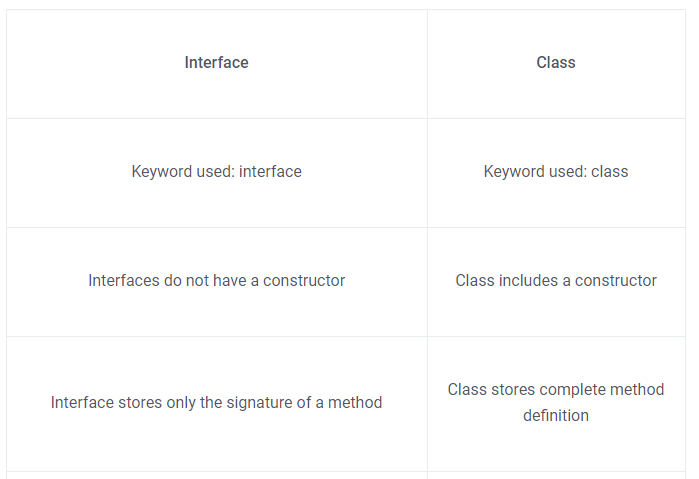
//Expected Output:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

So with the particle part done, we will now move ahead and learn the differences between an interface and a class.

Interface vs Class

Though a class and an Interface look a little similar, they are extremely different from each other. Some of the primary differences between them are enlisted below as follows.

So, with the differences discussed, let us now move ahead to the next part where we discuss the disadvantages of an Interface in Java.

## Disadvantages of Interface in Java

* An interface in real-world projects is used either extensively or not at all.
* The use of Interface can reduce the execution speed.

So these were a couple of disadvantages of using an interface in Java.

With this, we have now arrived at the end of the "Interface in Java' .

The interface is only one of the crucial concepts of Object-Oriented Programming in Java. It is highly recommended to read all the Object-Oriented Programming Concepts in Java