Liskov's Substitution Principle

Liskov's substitution principle is considered one of the important principles among the SOLID principles. It is also slightly difficult to understand.

It works on the parent and child classes' relationship and behavior. If I want to define this principle formally,it will be defined as per below:

Suppose there are 2 classes. Class A and Class B. Here Class B is the subclass of Class A which means that Class A is the parent class of Class B.

If Class B is the subclass of Class A, then changing the object of Class A with Class B should not affect the behavior of the program.

In other words, the child class should extend the capability of the parent class,it should not narrow down the capabilities or features.

We will understand this principle with a example as per below:

Suppose there is an interface implementation which is named Bike.

```
//interface implementation of Bike
public interface Bike
{
    //there are 2 methods which must be implemented by class
    void turnOnEngine();
    void accelerate();
}
```

There is a class named Motorbike which implements the interface of bike

```
//class declaration
class MotorBike implements Bike
{
    boolean isEngineOn;
    int speed;

    void turnOnEngine()
    {
        isEngineOn = True;
    }

    void accelerate()
    {
        speed = speed + 10;
    }
}
```

Which is perfectly fine as it doesn't violate Liskov's substitution rule till now.
Now let us suppose there is another class named Bicycle which also implements the interface.

```
//class declaration
class Bicycle implements Bike
{

    int speed;

    void turnOnEngine()
    {
        throw assertion error ("There is no engine in the bicycle");
    }

    void accelerate()
    {
        speed = speed + 10;
    }
}
```

Now here the Bicycle class is the subclass but here we can say that it doesn't implement all methods from the interface due to lack of engine which is definitely a narrow down of capabilities from the parent class of MotorBike.
This is why it doesn't follow the **Liskov's Substitution Principle** as we are changing the object of class motorbike to class bicycle which is breaking the program behavior.

This is how we can understand the importance of Liskov's Substitution Principle.