Liskov Substitution Principle

The Liskov substitution principle is a principle in object-oriented programming that states that, in a computer program, if S is a subtype of T, then objects of type T may be replaced with objects of type S without affecting the correctness of the program. In other words, the principle states that subclasses should be able to extend the functionality of their superclasses without changing the behavior of the superclass. This means that subclasses should be able to extend the functionality of their superclasses in a way that is consistent with the expectations of the users of the superclass. For example, if a user expects a certain behavior from an object of a superclass, they should be able to expect the same behavior from an object of a subclass, even if the subclass has additional functionality. This allows for greater flexibility and reusability of code in object-oriented programming.

Example:

As an example, consider a program that has a superclass called **Vehicle** and subclasses called **Car** and **Bicycle**. The Vehicle class has a method called **startEngine**() that starts the engine of the vehicle. The **Car** class extends the Vehicle class and adds additional functionality, such as the ability to open and close the windows and turn on the headlights. The **Bicycle** class also extends the Vehicle class, but it does not have an engine, so it does not have a **startEngine**() method.

In this example, the **Car** class adheres to the Liskov substitution principle because it extends the functionality of the Vehicle class in a way that is consistent with the expectations of the users of the Vehicle class. A user of the Vehicle class expects the **startEngine**() method to start the engine, and this is exactly what the **startEngine**() method does in the **Car** class.

On the other hand, the **Bicycle** class does not adhere to the Liskov substitution principle because it does not have a **startEngine**() method, which goes against the expectations of the users of the **Vehicle** class. A user of the Vehicle class expects the startEngine() method to start the engine, but this is not possible for a Bicycle object because it does not have an engine. To adhere to the Liskov substitution principle, the Bicycle class would need to either provide an alternative implementation of the **startEngine**() method or throw an exception when the method is called.

Here is an example implementation of the Vehicle and **Car** classes in Java:

```java
public class Vehicle {
    public void startEngine() {
        System.out.println("Starting engine...");
    }
}

public class Car extends Vehicle {
    public void openWindow() {
        System.out.println("Opening window...");
    }

    public void closeWindow() {
        System.out.println("Closing window...");
    }

    public void turnOnHeadlights() {
        System.out.println("Turning on headlights...");
    }
}

public class Test{
    public static void main(String args[]){
        Car car = new Car();
        car.startEngine(); // Output: "Starting engine..."
        car.openWindow();  // Output: "Opening window..."
        car.closeWindow(); // Output: "Closing window..."
        car.turnOnHeadlights(); // Output: "Turning on headlights..."
    }
}
```

In this implementation, the **Car** class extends the Vehicle class and adds additional functionality, such as the ability to open and close the windows and turn on the headlights. However, it also maintains the **startEngine**() method from the Vehicle class, so it adheres to the Liskov substitution principle.

As you can see, the **Car** object has all the functionality of a **Vehicle** object, as well as additional functionality that is specific to a **Car** object. This allows us to use a **Car** object wherever a **Vehicle** object is expected, without affecting the correctness of the program.