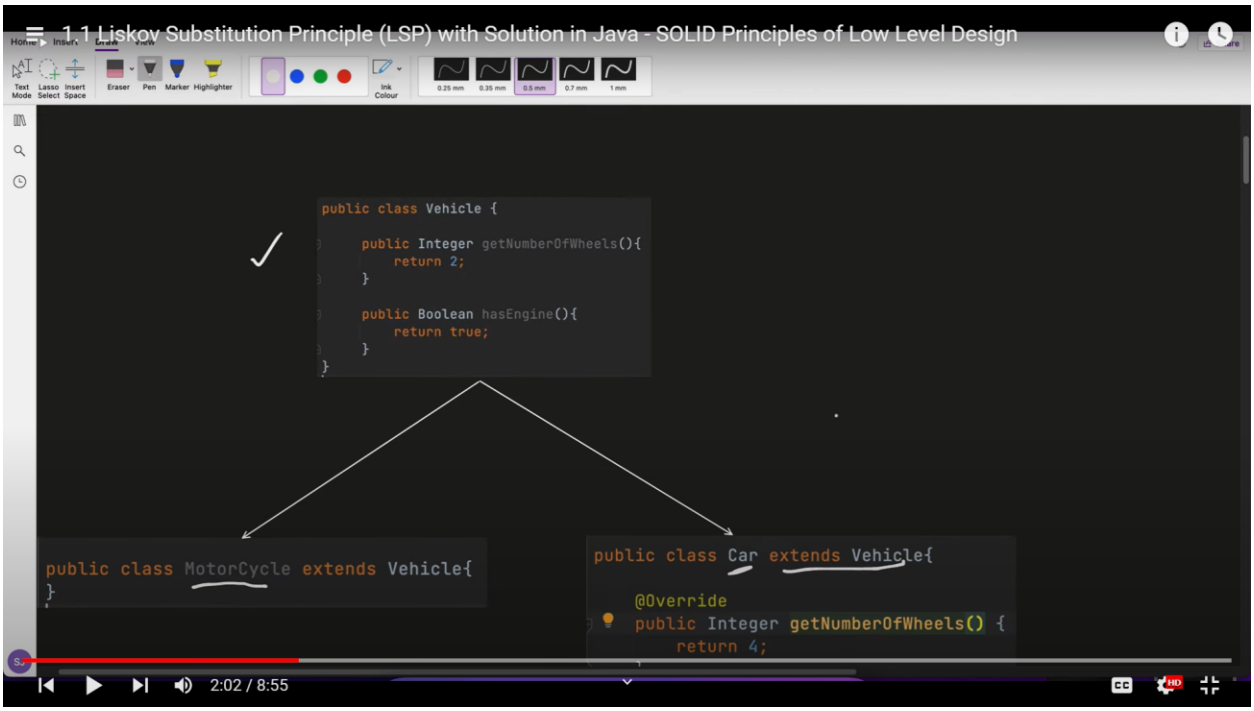
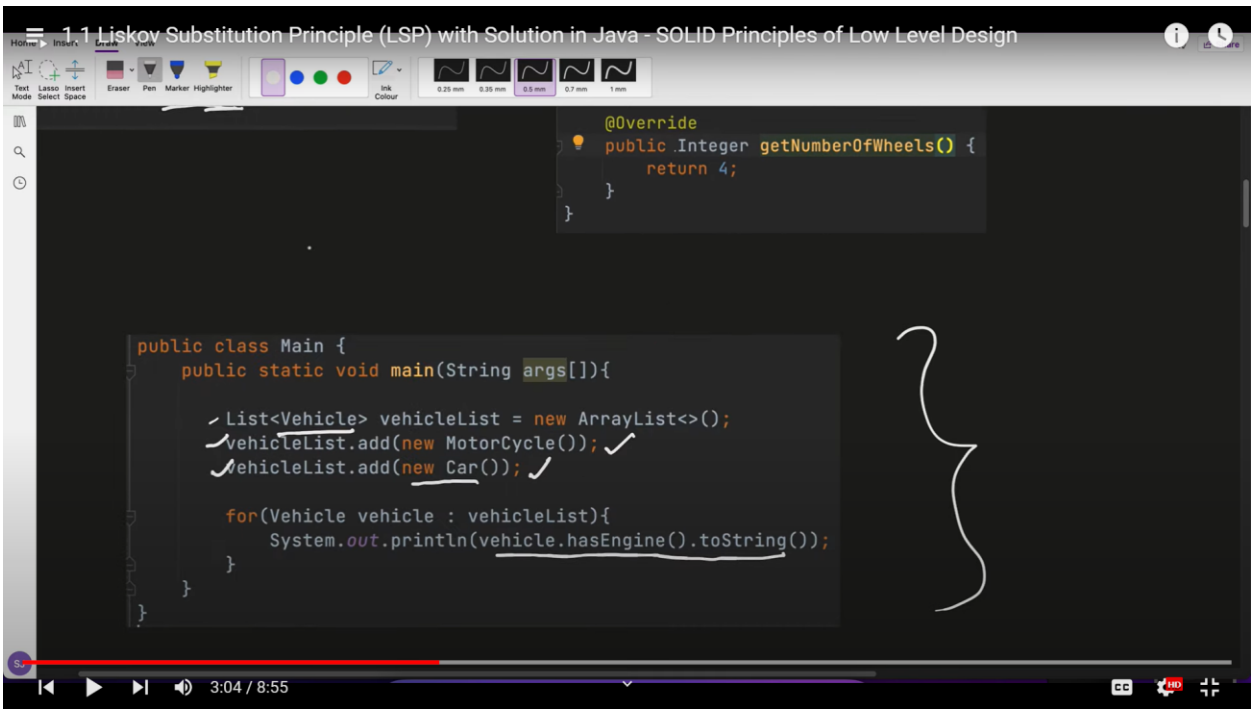
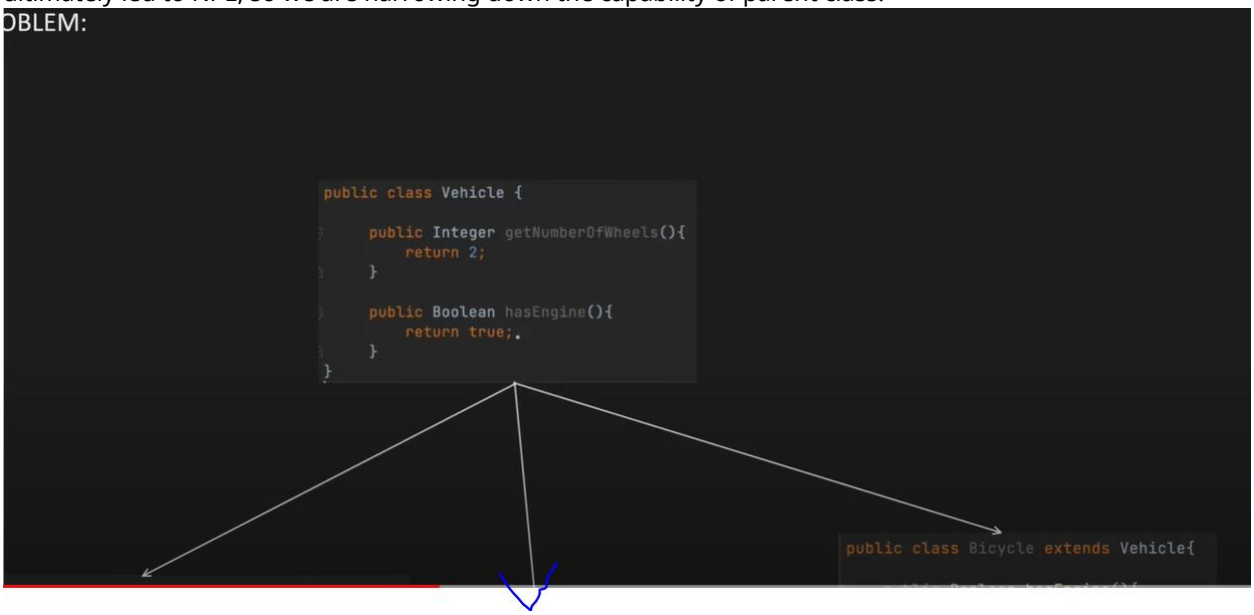
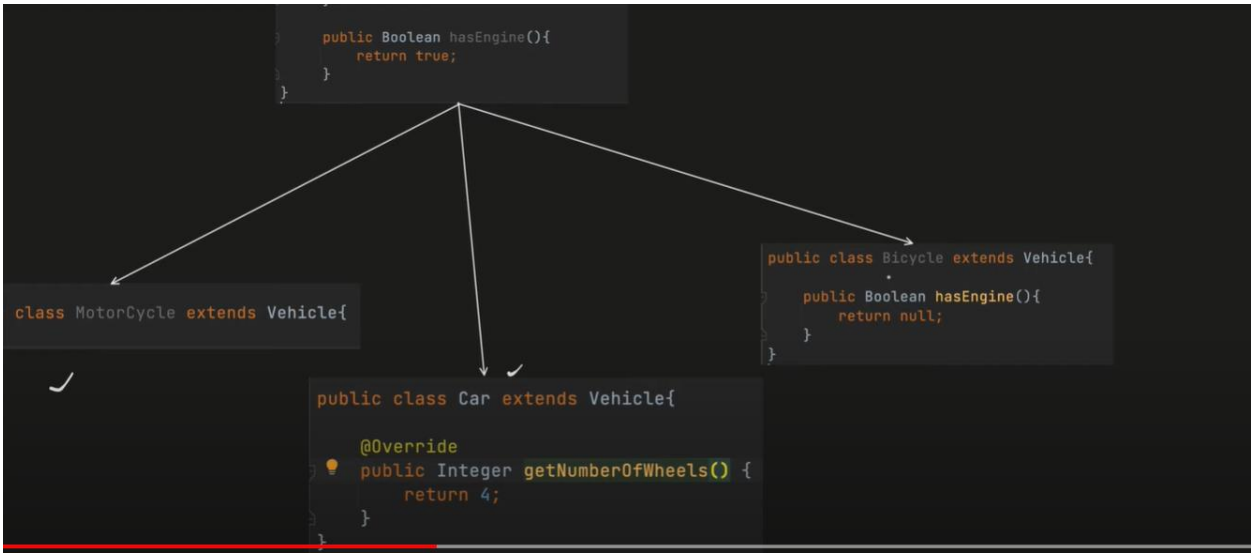
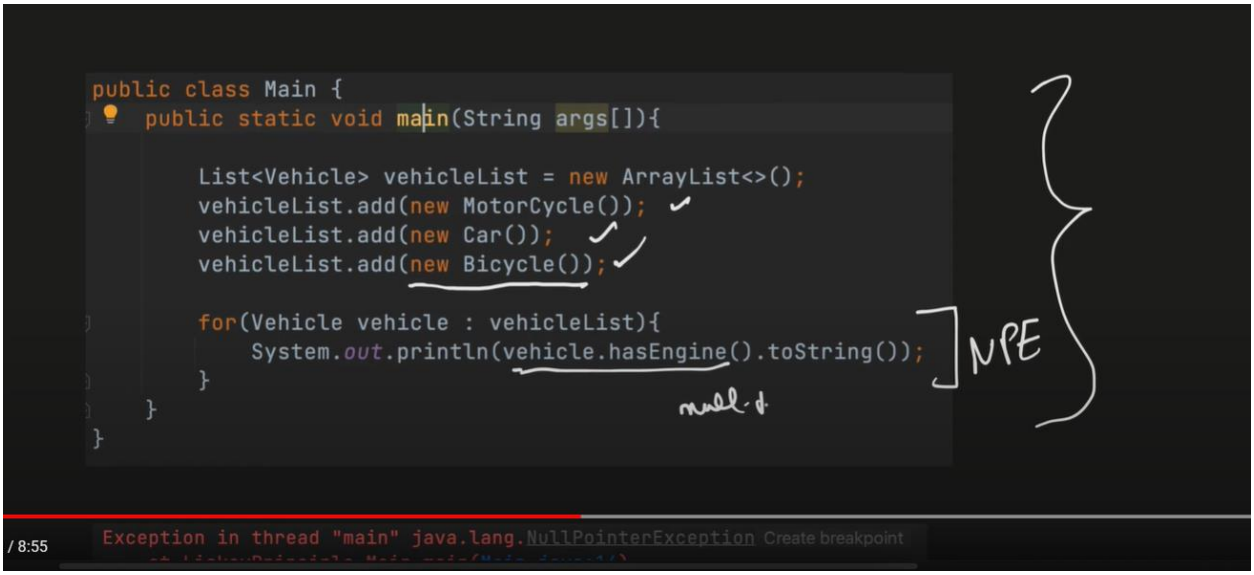

1.1 Liskov Substitution Principle (LSP) with Solution in Java - SOLID Principles of Low Level Design
public class Vehicle {
public Integer getNumberOfWheels(){
return 2;
}
public Boolean hasEngine(){
return true;
}
}
public class MotorCycle extends Vehicle{
}
public class Car extends Vehicle{
@Override
public Integer getNumberOfWheels() {
return 4;
2:02 / 8:55


1.1 Liskov Substitution Principle (LSP) with Solution in Java - SOLID Principles of Low Level Design
@Override
public Integer getNumberOfWheels() {
return 4;
}
}
public class Main {
public static void main(String args[]){
List<Vehicle> vehicleList = new ArrayList<>();
vehicleList.add(new MotorCycle());
vehicleList.add(new Car());
for(Vehicle vehicle : vehicleList){
System.out.println(vehicle.hasEngine().toString());
}
}
}
3:04 / 8:55

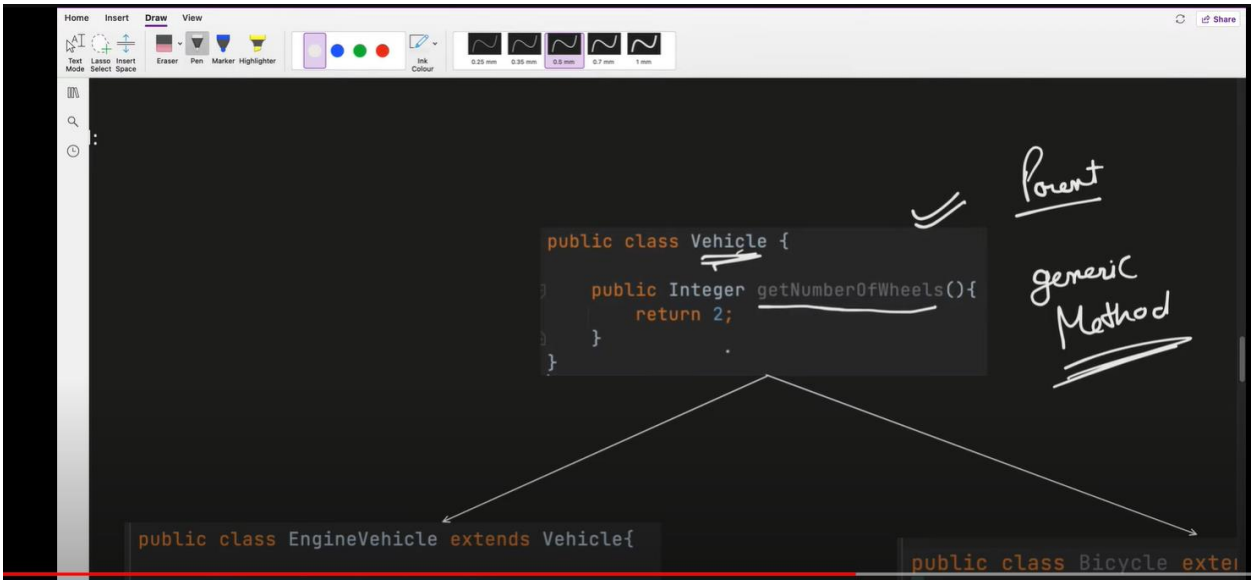
Problem arises ,when we extends Vehicle class to Bicycle class ,in here ,we are returning null, which will ultimately led to NPE, so we are narrowing down the capability of parent class.

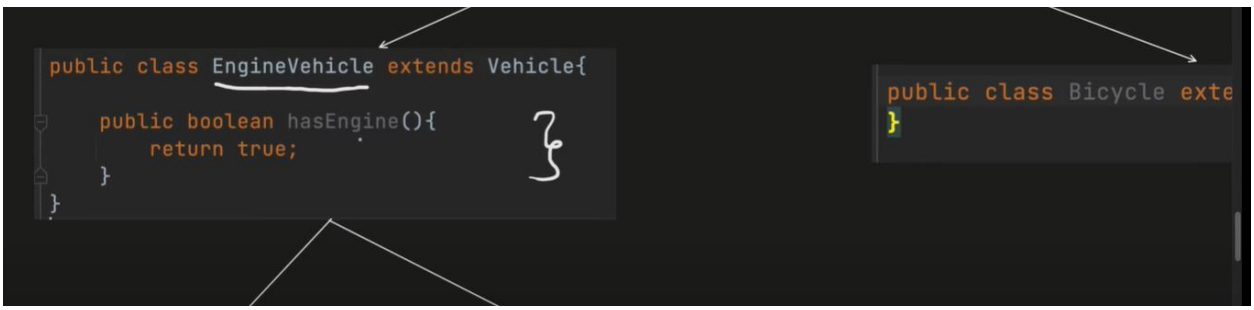


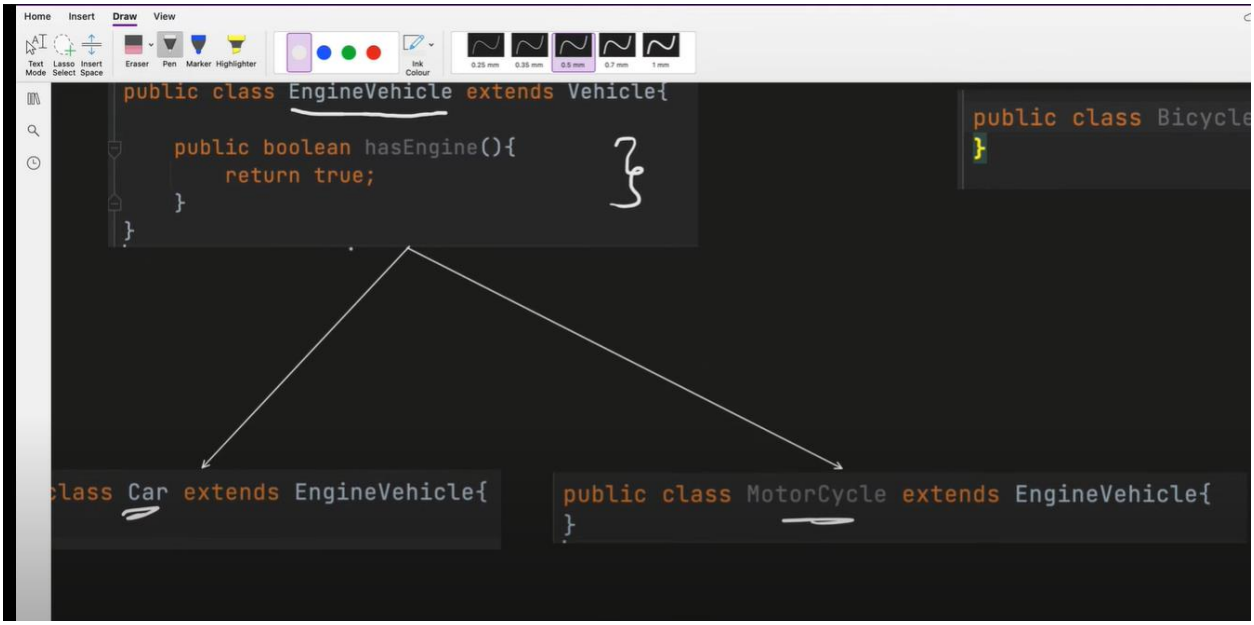






Solution:
Keep only most generic method in Parent class.






```
public class Main {
    public static void main(String args[]){

        List<Vehicle> vehicleList = new ArrayList<>();
        vehicleList.add(new MotorCycle());
        vehicleList.add(new Car());
        vehicleList.add(new Bicycle());

        for(Vehicle vehicle : vehicleList){
            System.out.println(vehicle.getNumberOfWheels().toString());
        }
    }
}
```

```
public class Main {
    public static void main(String args[]){

        List<EngineVehicle> vehicleList = new ArrayList<>();
        vehicleList.add(new MotorCycle());
        vehicleList.add(new Car());
        vehicleList.add(new Bicycle());

        for(EngineVehicle vehicle : vehicleList){
            System.out.println(vehicle.hasEngine());
        }
    }
}
```