OOPS CONCEPT

The object-oriented programming has the following concepts:

- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- Association:
    - I. Composition
    - II. Aggregation

Abstraction:

Hiding internal details and showing functionality is known as **Abstraction**. We can hide the unnecessary from the user and expose only that data that is of interest to the user.
Abstraction is separating the functions and properties that logically can be separated to a separate entity on which the main type depends on.
Example:
A car has engine wheel any many other parts. When we write all the properties of the Car, Engine and Wheel in a single Class. It would look like this way.

```
public class Car {

	int price;
	String name;
	String color;
	int enjineCapacity;
	int enjineHorsePower;

	String wheelName;
	int wheelPrice;

	void move() {
		//Move Forward
	}

	void rotate() {
		//Wheels rotate
	}

	void internalCombustion() {
		//Engine Method
	}
}
```

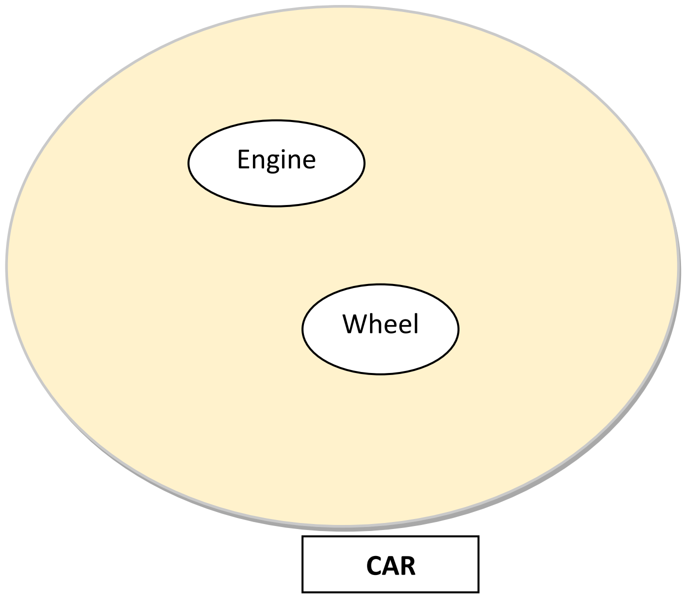
In this example, the attributes of the Wheel and Engine are added to the Car type. As per pogramming it will not create any issues . But when it comes to maintenance of the application, this become more Complex.

Advantages of Abstraction:

1) By the Abstraction, we can separate things that can be grouped to another type.
2) Frequently changing properties and methods can be grouped to a separate type so the main type need not undergo changes.
3) Simplifies the Representation of the Domain Models.
   Applying **Abstraction** with **Composition**, the above example can be modified as given bellow:

```java
public class Car {

	int price;
	String name;
	String color;

	Engine engine=new Engine();
	Wheel wheel=new Wheel();

	void move() {
		//Move Forward
	}
}
class Wheel{
	String wheelName;
	int wheelPrice;

	void rotate() {
		//Wheels rotate
	}
}
class Engine{
	int enjineCapacity;
	int enjineHorsePower;

	void internalCombustion() {
		//Engine Method
	}
}
```

You can see the attributes and methods related to the Engine and Wheel are moved to the respective classes.
Engine and Wheel are referred from the Car type. Whenever an instance of Car is created both Engine and Wheel will be available for the car and when there are changes of this Types(Engine and Wheel) changes will only be confirmed to this classes and will not affect the Car class.


Engine
Wheel
CAR