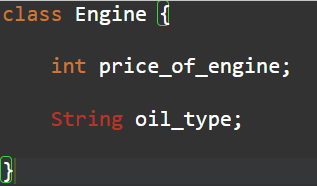
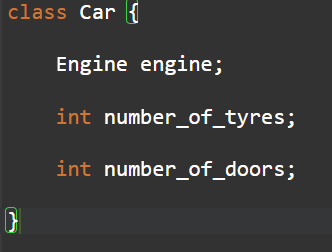
**Spring framework**

* Spring is a java framework which is used to solve the problem of **“Dependency injection”**.
* Spring consists of many different projects. Some of them are given below
  + Spring core
  + Spring MVC
  + Spring boot
  + Spring security
  + Spring Data JPA
  + Spring cloud
* **What is a dependency?**
  + To understand dependency, Let’s take an example.
  + Suppose we have two classes. “Car” and “Engine”
  + 
  + In the above example, class Car requires an object of class Engine
  + Therefore, we can say that class Car is dependent on class Engine. It means class Car has dependency on class Engine.
* **What is dependency injection?**
  + Dependency injection is nothing but providing an object for the dependency.
  + In the above example, If class Car needed to access data present in the class Engine, the object of class Engine needed to be created in class Car. Creation of this object is done by “Dependency Injection”.
* **Need of spring framework**
  + To remove **tight coupling** from the application.
  + Let us understand Tight coupling with the help of an example given below.

.

class Engine {

private int price\_of\_engine=5000;

public int getPrice() {

return price\_of\_engine;

}

}

public class Car {

public static void main (String[] args) {

Engine engine = new Engine();

int price = engine.getPrice();

System.out.println("Price of an Engine is: "+price);

}

}

* In the above example, Car has a dependency of Engine, To access the method getPrice() present inside class Engine class, an object of Engine needs to be created.
* Now in future, If multiple types of Engines are added in the code which implements Engine interface. Refer to the example given below

interface Engine {

public int getPrice();

}

class flatEngine implements Engine {

public int getPrice() {

return 4000;

}

}

class vEngine implements Engine {

public int getPrice() {

return 5000;

}

}

public class Car {

public static void main (String[] args) {

Engine engine = new flatEngine();

int price = engine.getPrice();

System.out.println("Price of an Engine is: "+price);

}

}

* In the above code, the engine object is created for flatEngine(). In future if I want to change it to vEngine(), I need to change the code again.
* Due to this behavior, class Car is tightly coupled with Engine’s dependency.
* **Spring IOC container**
  + Spring IoC Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle.
  + The objects stored in the IOC Container are called Bean.
* **What is ApplicationContext?**
  + ApplicationContext is an interface which is used to retrieve or fetch beans/objects present in the IOC container.
  + Below is an example of ApplicationContext .
    - ApplicationContext context = new AnnotationConfigApplicationContext(<ConfigClass>);
    - Car car = context.getBean(Car.class);
* **Some important annotations (To understand all annotations refer the given example here)**
  + @Configuration
  + @Bean
  + @Autowired
  + @Component
  + @ComponentScan(basePackages = {<package>})
  + @Qualifier
  + @Scope
  + @Value
  + @ProprtySource(“app.properties”)
  + @Lazy