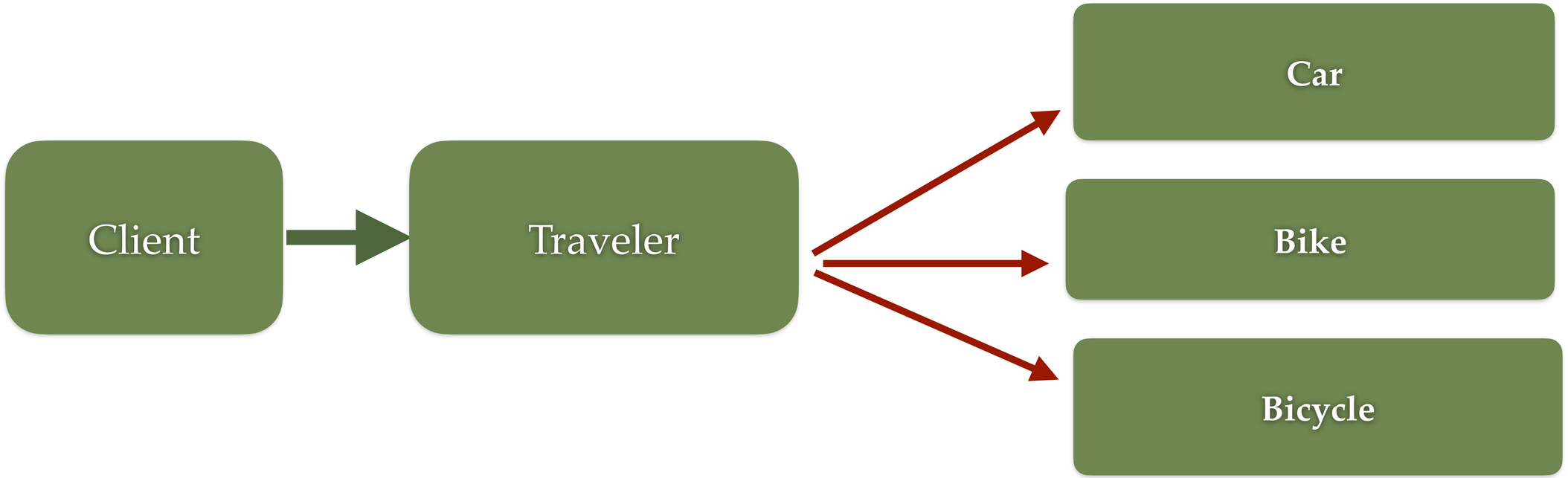
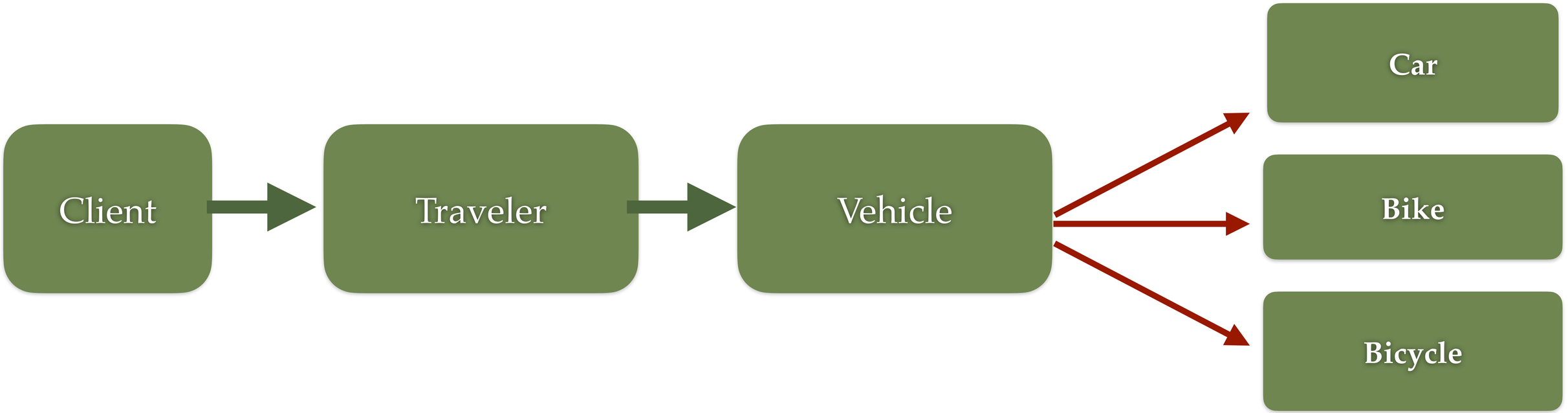
Spring IoC Container

Tight Coupling



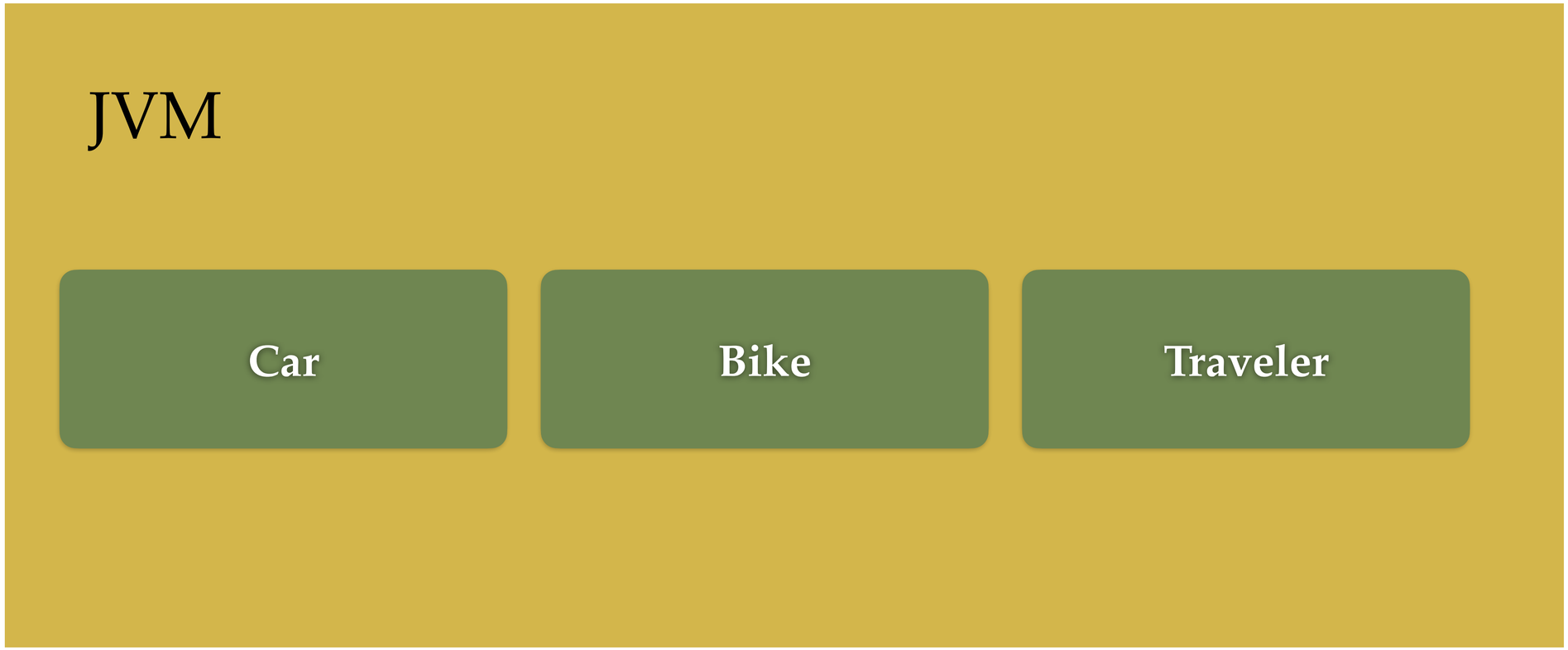
Loose Coupling



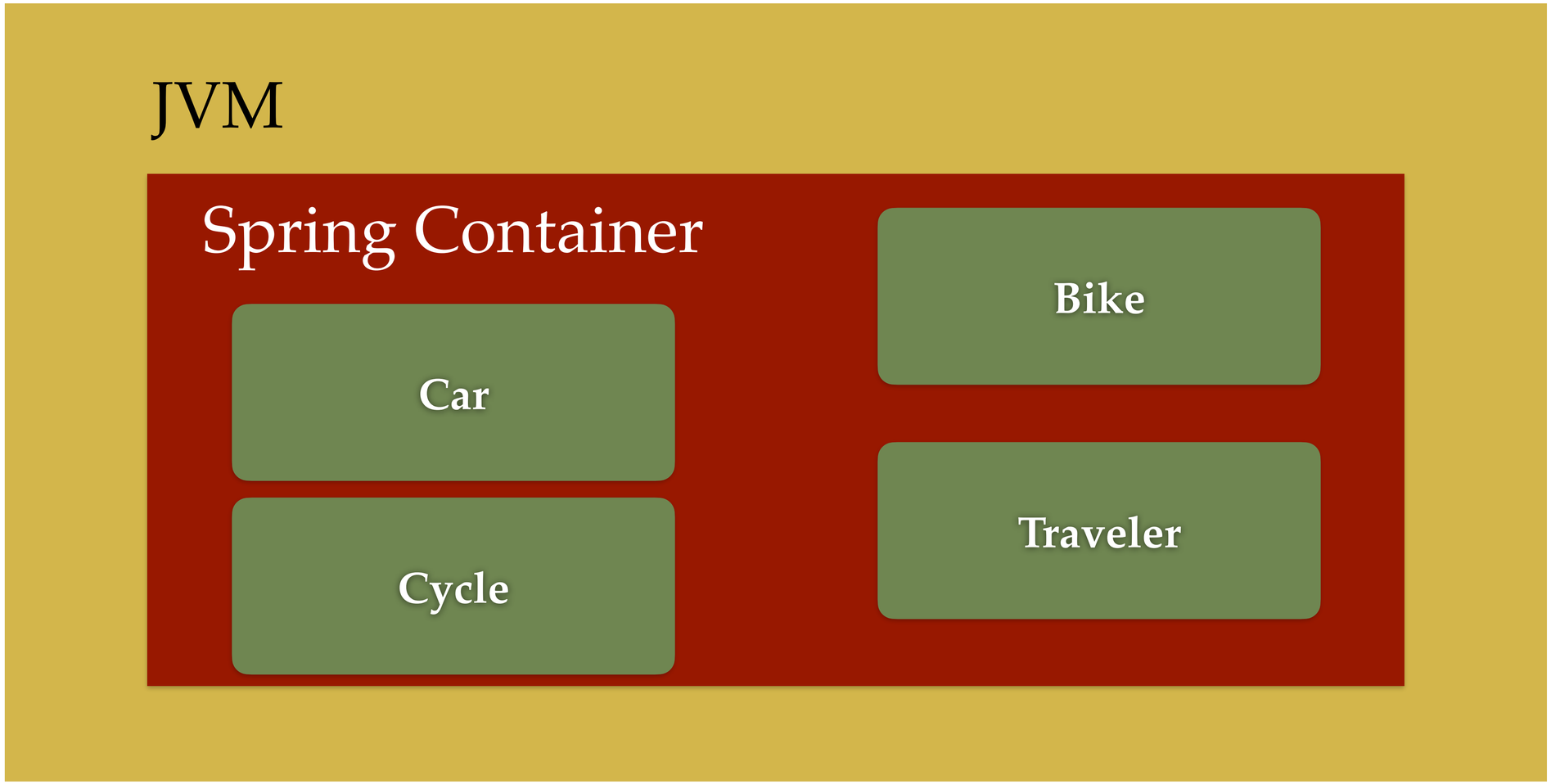
Spring IoC Container Example

1. Tight Coupling
2. Loose Coupling using Interface
3. Using Spring IoC Container to create and manage objects using Java based configuration

Creating Objects Manually



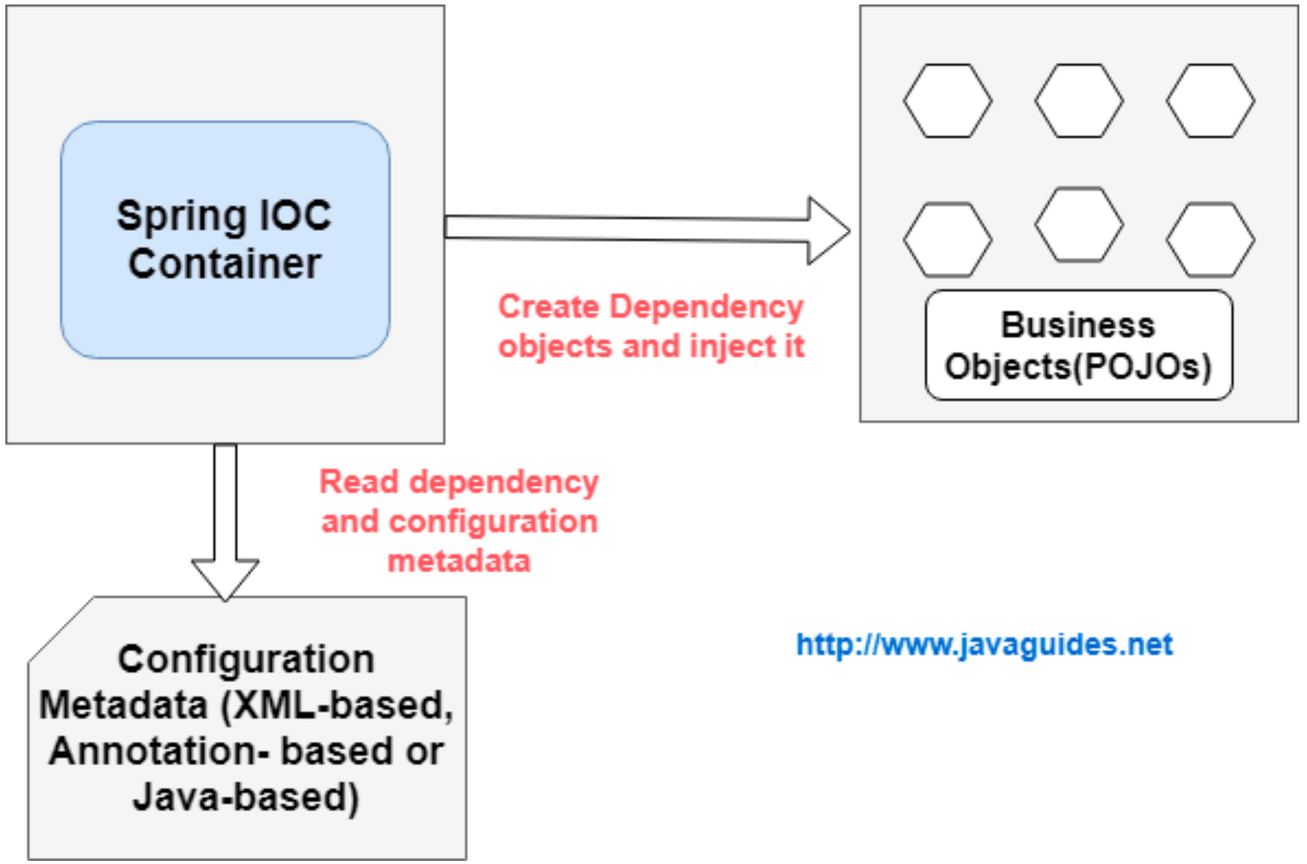
Spring IOC Container



Spring IoC Container

1. Responsible for creating the objects (Spring beans)
2. Responsible for injecting one abject into another object (using DI)
3. Managing the bean's entire life-cycle - It manages the complete lifecycle of an bean from creation to destruction
4. Container uses the information provided in the configuration file (or annotations) to create the objects, and it uses dependency injection to supply the objects with their dependencies.
5. The configuration metadata is represented in XML, Java annotations, or Java code.
6. By using the IoC container, the objects are more loosely coupled and easier to test, since their dependencies can be easily swapped out with mock objects.

Spring IoC Container



Spring IoC Container Types

1. **BeanFactory** container
2. **ApplicationContext** container

BeanFactory Container

1. Responsible for creating the beans, configuring the beans and managing the bean's entire life-cycle.

ApplicationContext Container

1. Responsible for creating the beans, configuring the beans and managing the bean's entire life-cycle.
2. **Enterprise Application Features:**

- It provides messaging (i18n or internationalization) functionality
- Event publication functionality
- Annotation-based dependency injection
- Easy integration with Spring AOP features
- Supports almost all types of bean scopes

@Primary Annotation

We use **@Primary** annotation to give higher preference to a bean when there are multiple beans of the same type.

Steps for Java based configuration

1. Create Configuration class with @Configuration annotation
2. Create method and annotated it with @Bean annotation
3. Create Spring IoC Container (ApplicationContext) and Retrieve Spring bean from Spring IoC container.

1. Create Configuration class with @Configuration annotation

```java
@Configuration
public class AppConfig {

    @Bean
    public Vehicle car(){
        return new Car();
    }

    @Bean
    public Vehicle bike(){
        return new Bike();
    }

    @Bean
    public Vehicle cycle(){
        return new Cycle();
    }

    @Bean
    public Traveler traveler(){
        return new Traveler(bike()); // DI
    }
```

```java
public class Car implements Vehicle {

    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```

```java
public class Bike implements Vehicle{

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```

```java
public class Traveler {

    private Vehicle vehicle;

    public Traveler(Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
```

2. Create method and annotated it with @Bean annotation

```java
@Configuration
public class AppConfig {

    @Bean
    public Vehicle car(){
        return new Car();
    }

    @Bean
    public Vehicle bike(){
        return new Bike();
    }

    @Bean
    public Vehicle cycle(){
        return new Cycle();
    }

    @Bean
    public Traveler traveler(){
        return new Traveler(bike()); // DI
    }
}
```

```java
public class Car implements Vehicle {

    @Override
    public void move(){
        System.out.println("Car is moving ..");
    }
}
```

```java
public class Bike implements Vehicle{

    @Override
    public void move(){
        System.out.println("Bike is moving ..");
    }
}
```

```java
public class Traveler {

    private Vehicle vehicle;

    public Traveler(Vehicle vehicle){
        this.vehicle = vehicle;
    }

    public void startJourney(){
        this.vehicle.move();
    }
}
```

3. Create Spring IoC container and retrieve bean

```
// Creating Spring IOC Container
// Read the configuration class
// Create and manage the Spring beans
ApplicationContext applicationContext = new AnnotationConfigApplicationContext(AppConfig.class);

// Retrieve Spring Beans from Spring IOC Container
Car car = applicationContext.getBean(Car.class);
car.move();

Bike bike = applicationContext.getBean(Bike.class);
bike.move();

Traveler traveler = applicationContext.getBean(Traveler.class);
traveler.startJourney();
```

```
@Configuration
public class AppConfig {

    @Bean
    public Vehicle car(){
        return new Car();
    }

    @Bean
    public Vehicle bike(){
        return new Bike();
    }

    @Bean
    public Vehicle cycle(){
        return new Cycle();
    }

    @Bean
    public Traveler traveler(){
        return new Traveler(bike()); // DI
    }
}
```