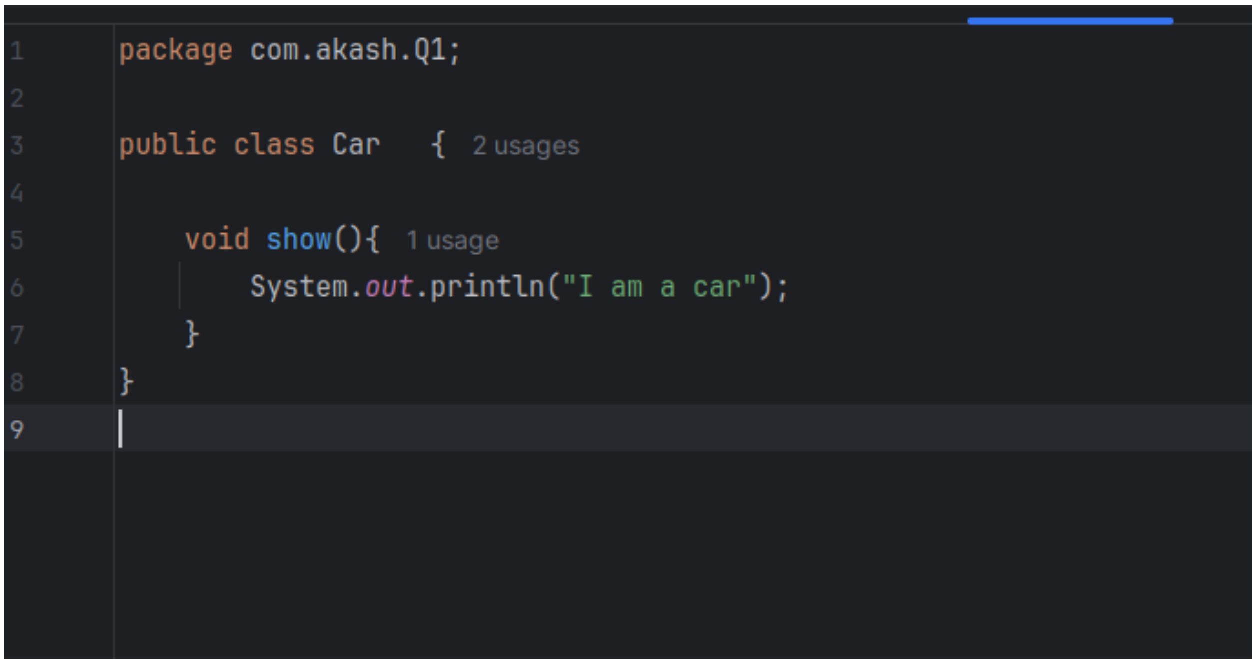
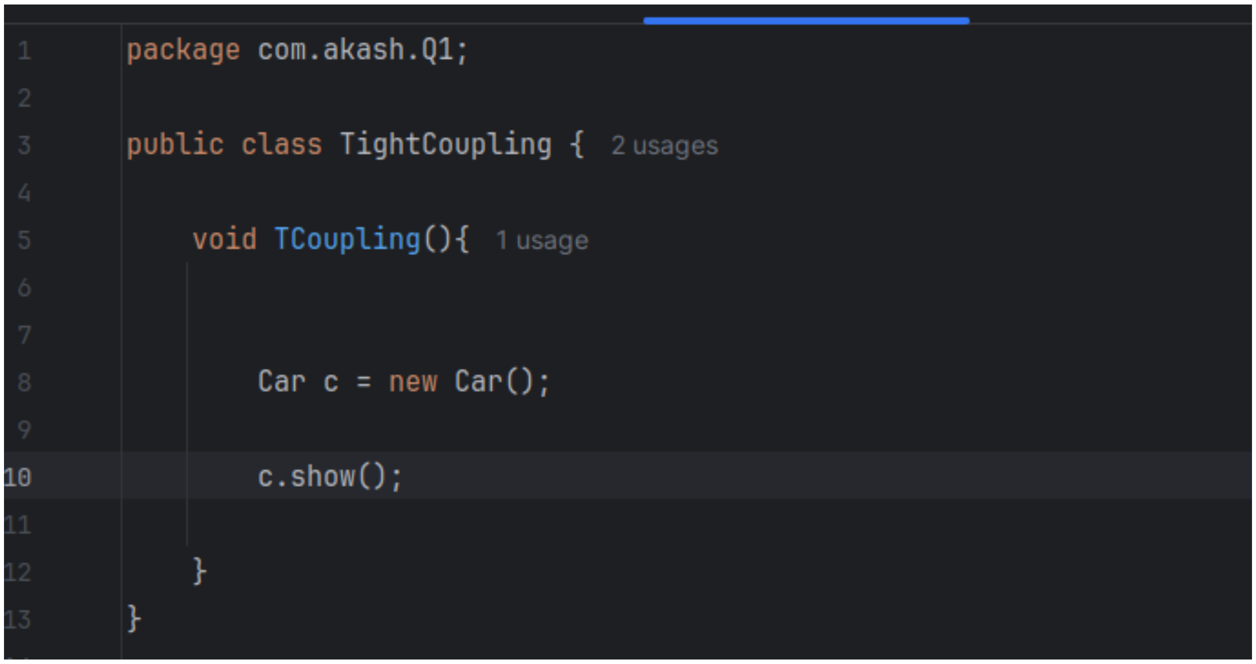
Assignment - Spring Framework - Part 2

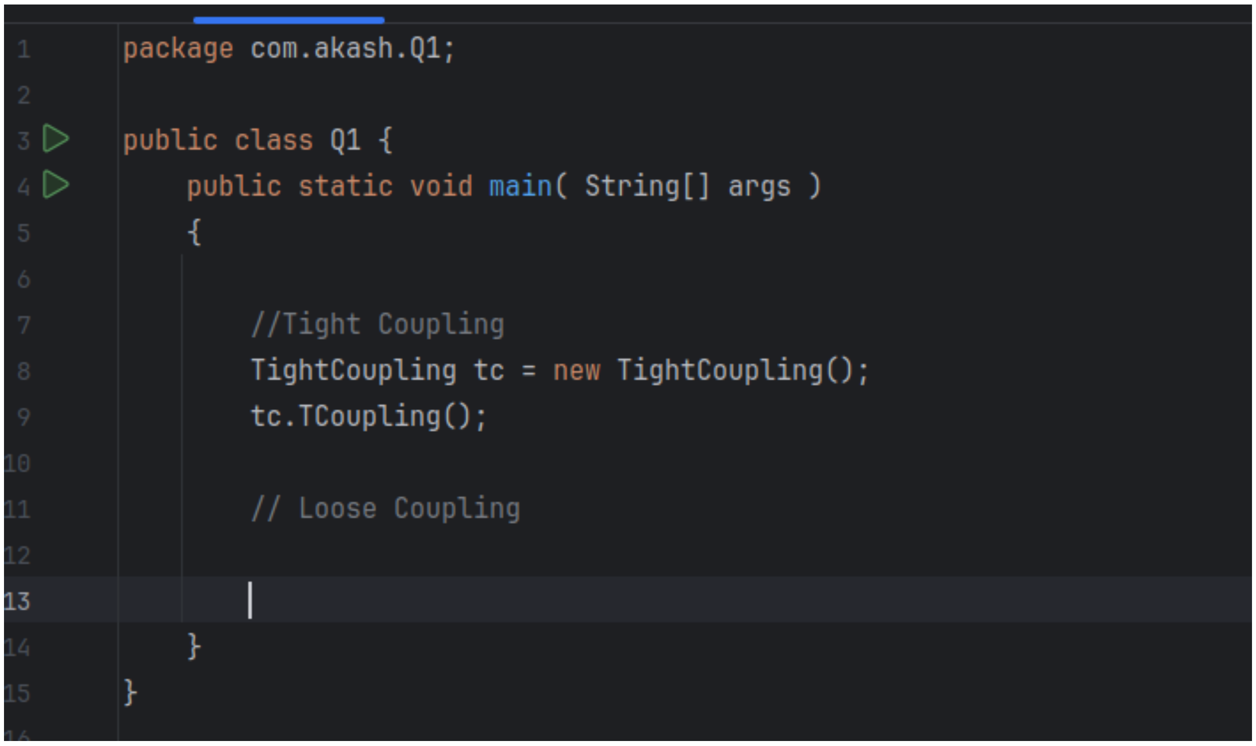
Q1)Write a program to demonstrate Tightly Coupled code.



```java
package com.akash.Q1;

public class Car   {  2 usages

    void show(){  1 usage
        System.out.println("I am a car");
    }
}

```

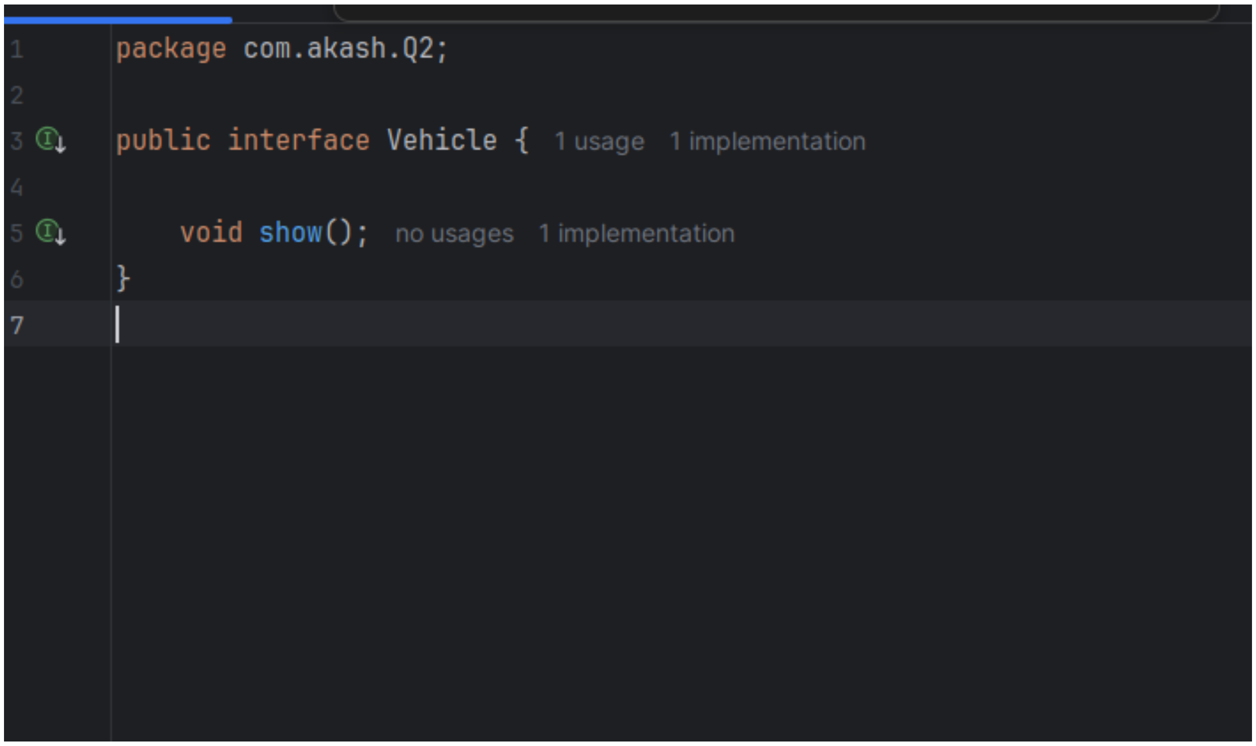


```java
package com.akash.Q1;

public class TightCoupling {  2 usages

    void TCoupling(){  1 usage


        Car c = new Car();

        c.show();

    }
}
```

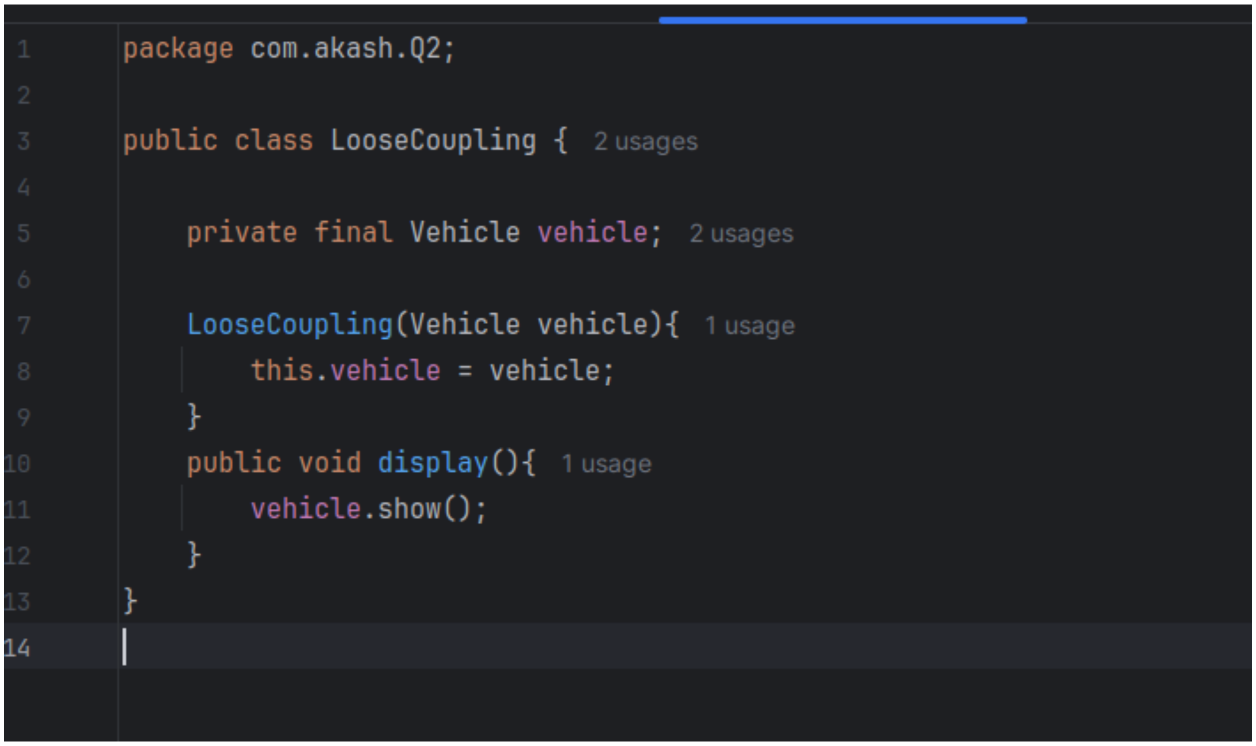


```java
package com.akash.Q1;

public class Q1 {
    public static void main( String[] args )
    {

        //Tight Coupling
        TightCoupling tc = new TightCoupling();
        tc.TCoupling();

        // Loose Coupling


    }
}
```

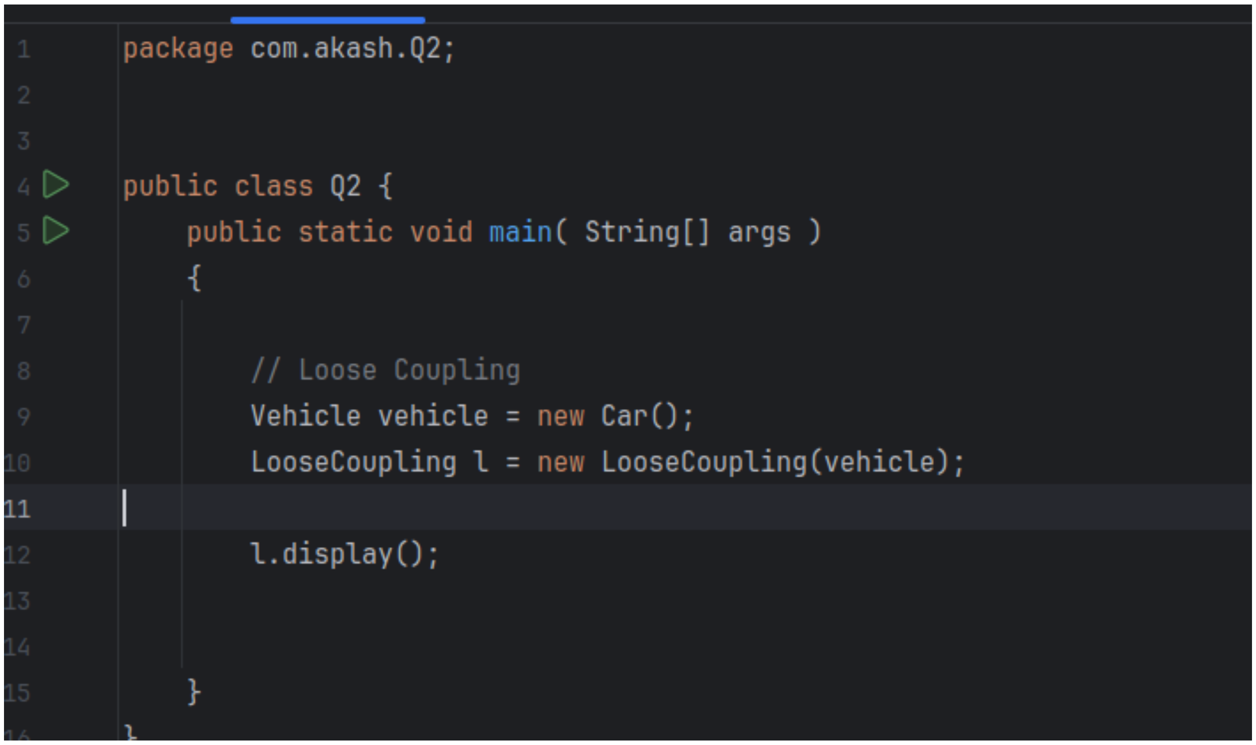
Q2)Write a program to demonstrate Loosely Coupled code.



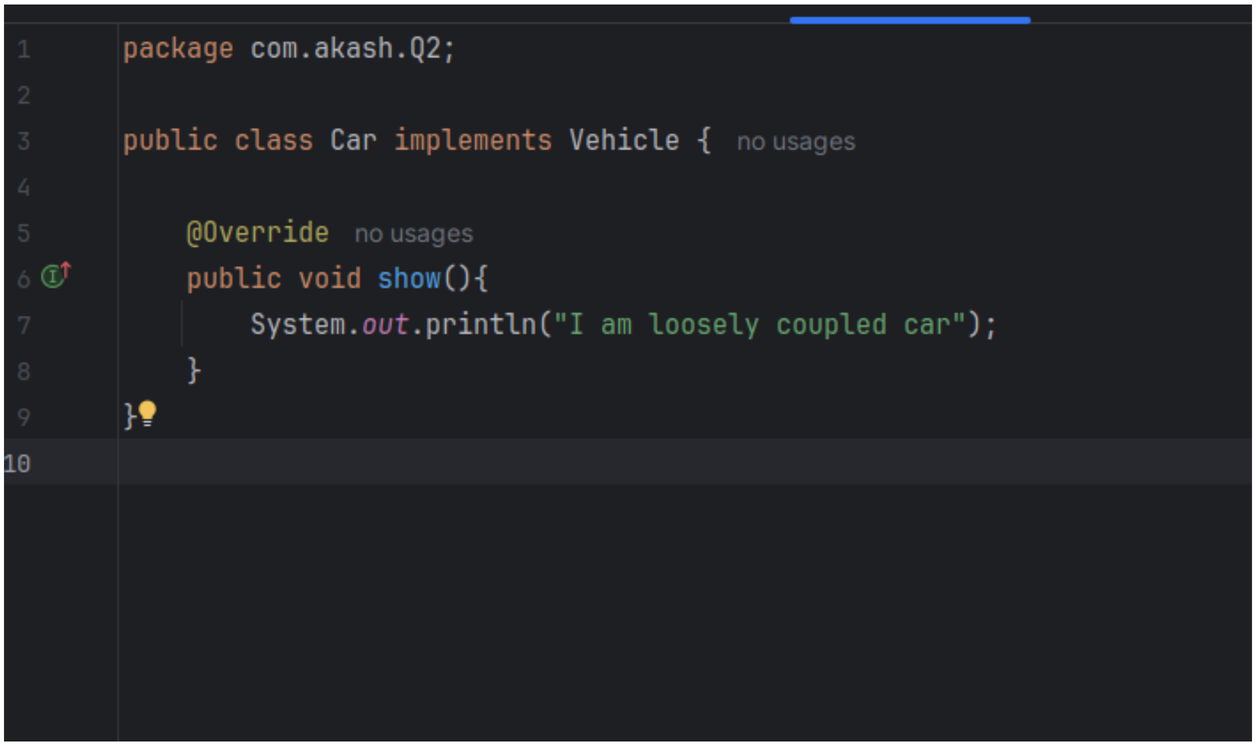
```java
package com.akash.Q2;

public interface Vehicle {  1 usage  1 implementation

    void show();  no usages  1 implementation
}

```



```java
package com.akash.Q2;

public class LooseCoupling {  2 usages

    private final Vehicle vehicle;  2 usages

    LooseCoupling(Vehicle vehicle){  1 usage
        this.vehicle = vehicle;
    }
    public void display(){  1 usage
        vehicle.show();
    }
}

```



```java
package com.akash.Q2;


public class Q2 {
    public static void main( String[] args )
    {

        // Loose Coupling
        Vehicle vehicle = new Car();
        LooseCoupling l = new LooseCoupling(vehicle);

        l.display();


    }
}
```



```java
package com.akash.Q2;

public class Car implements Vehicle {  no usages

    @Override  no usages
    public void show(){
        System.out.println("I am loosely coupled car");
    }
}

```

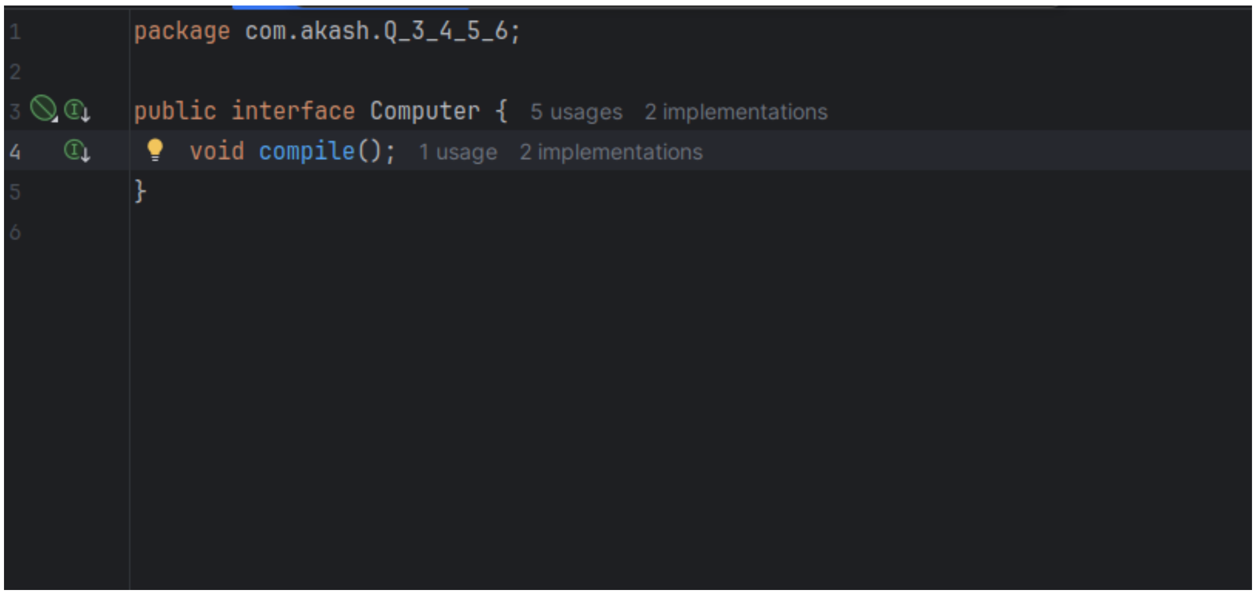
Q)

A)Use @Compenent and @Autowired annotations to in Loosely Coupled code for dependency management

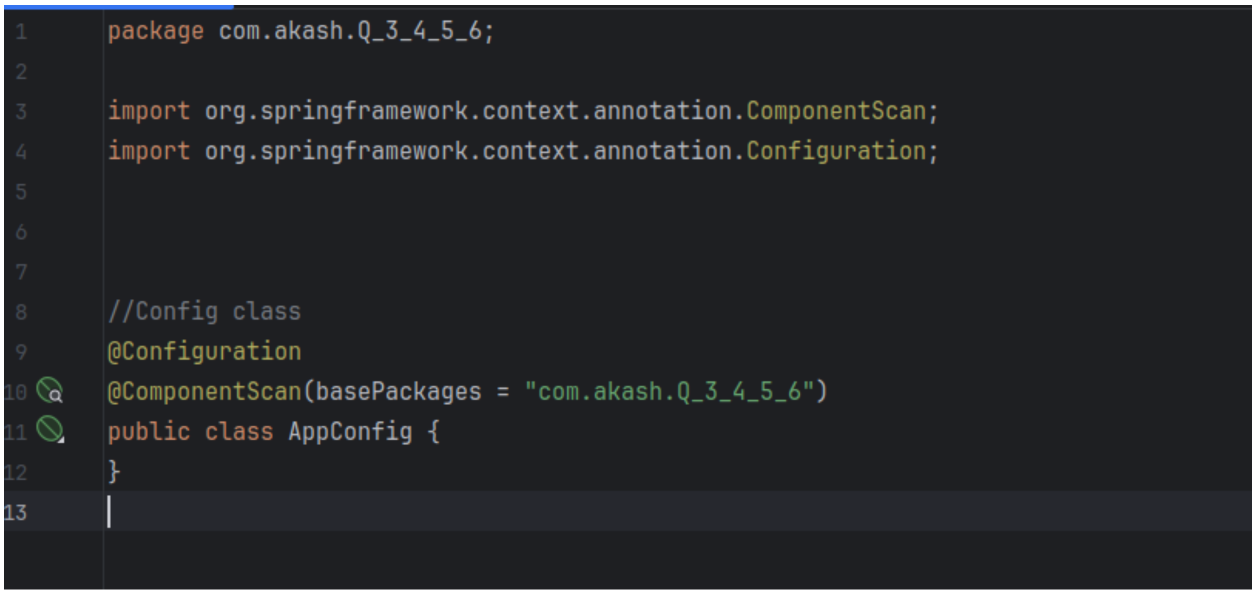
B)Get a Spring Bean from application context and display its properties.

C)Demonstrate how you will resolve ambiguity while autowiring bean

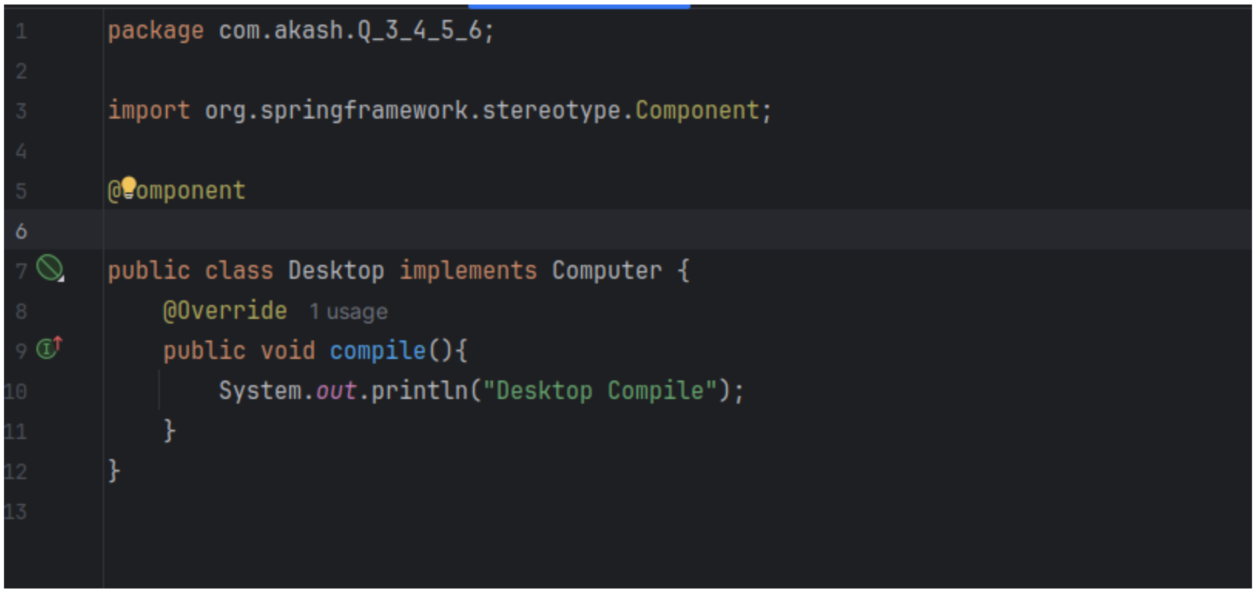
D)Perform Constructor Injection in a Spring Bean

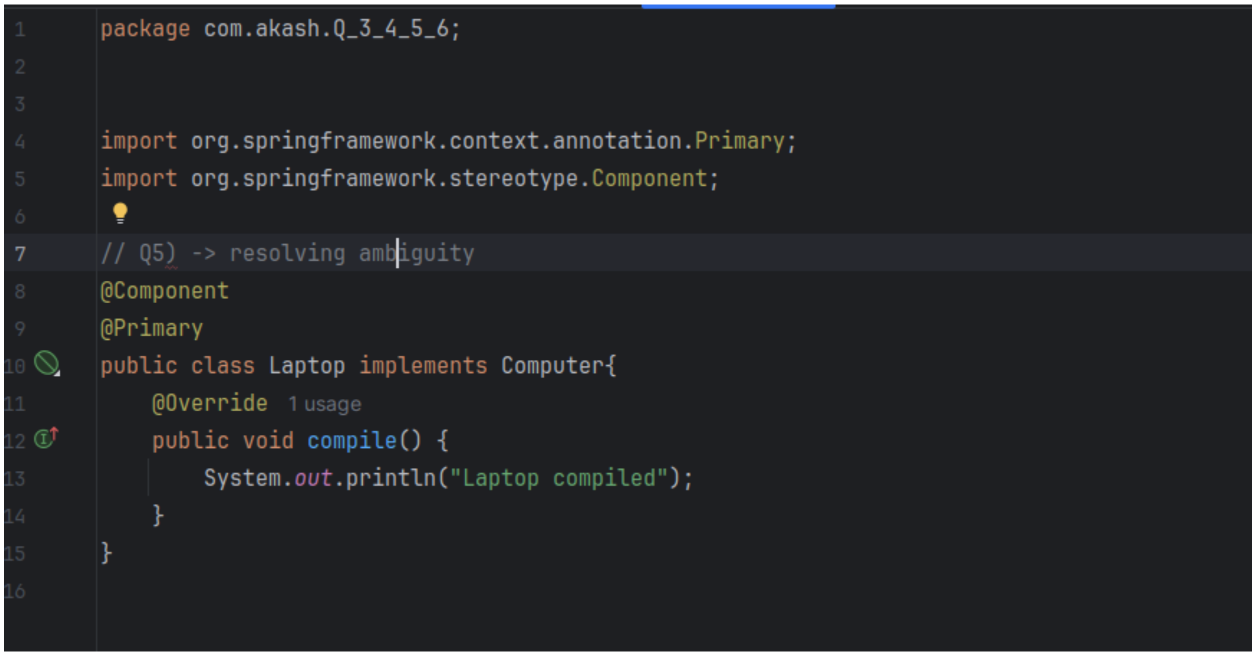


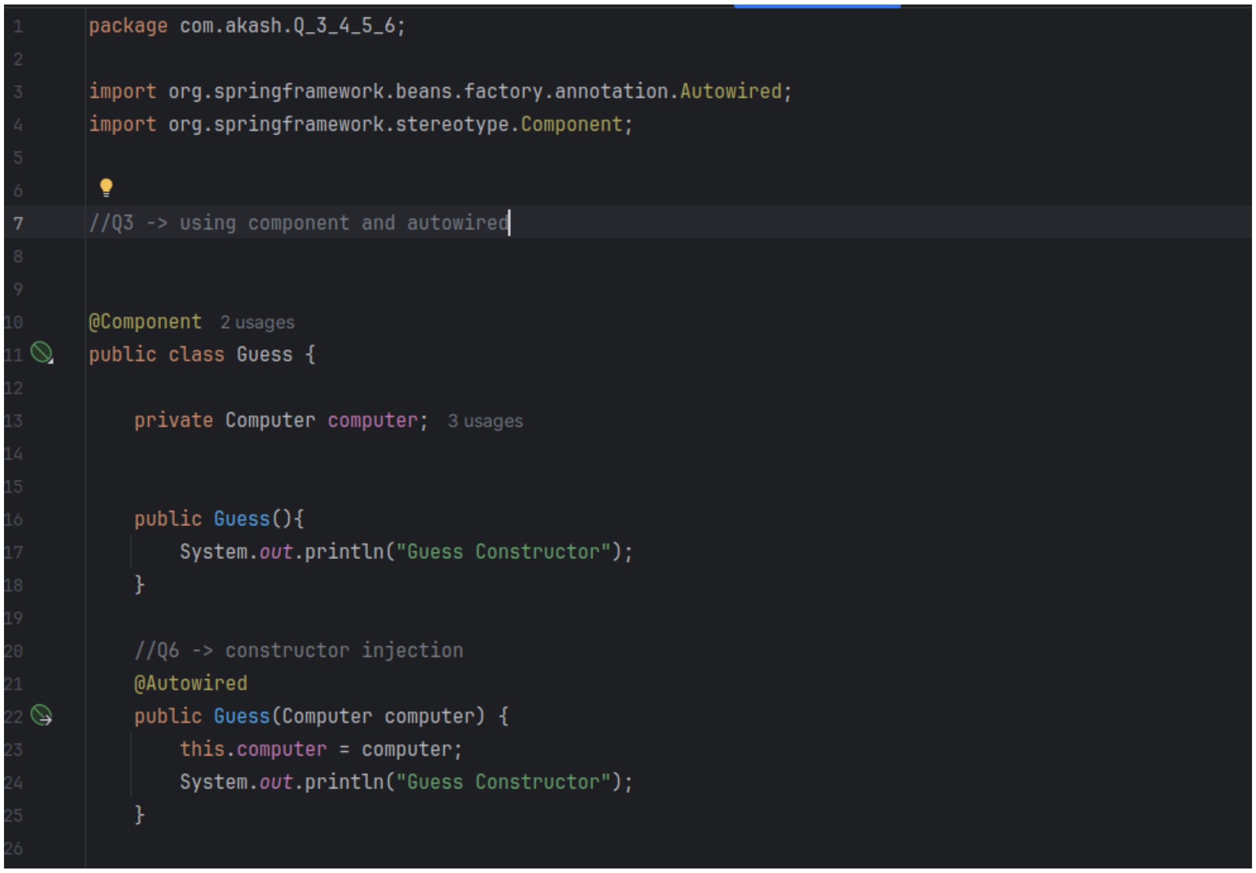
```java
package com.akash.Q_3_4_5_6;

public interface Computer {  5 usages  2 implementations
    void compile();  1 usage  2 implementations
}

```



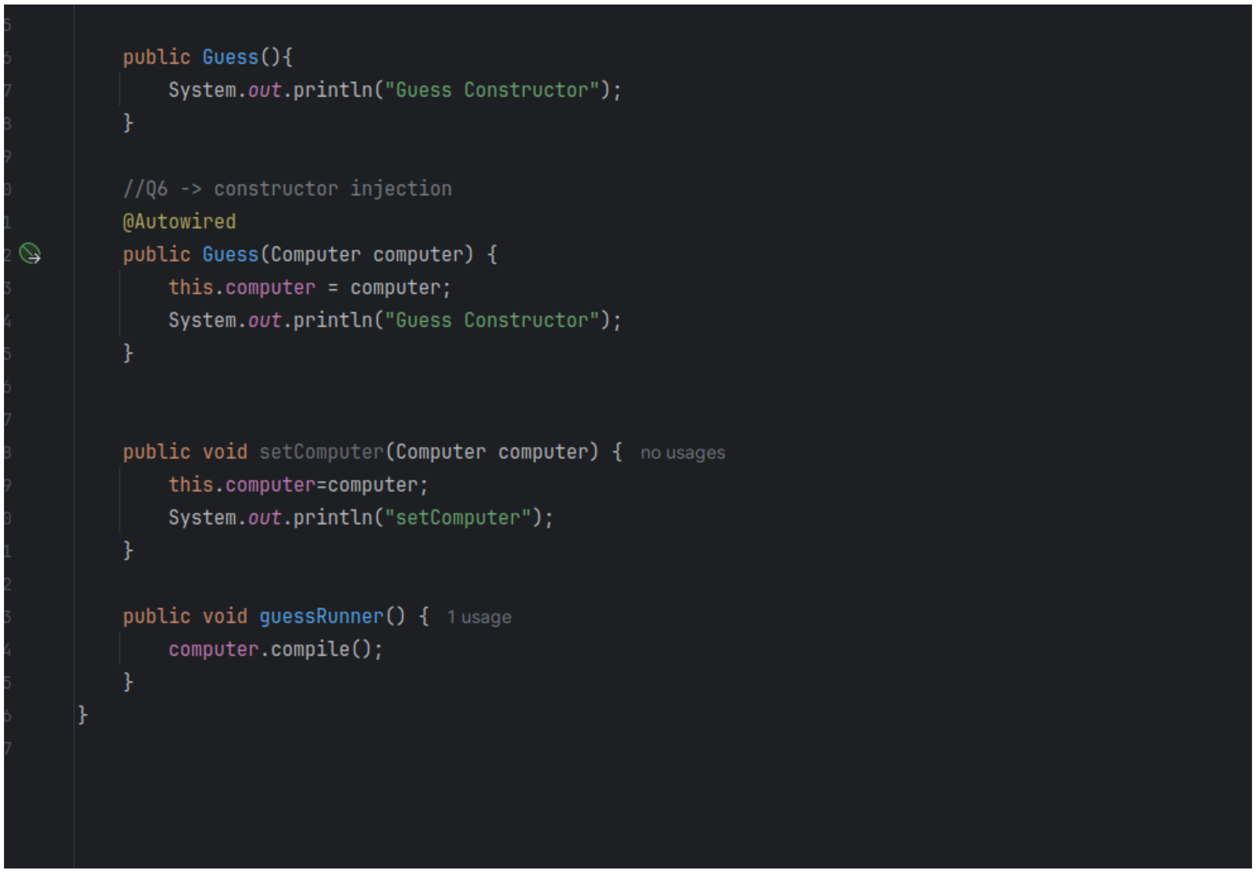
```java
package com.akash.Q_3_4_5_6;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;



//Config class
@Configuration
@ComponentScan(basePackages = "com.akash.Q_3_4_5_6")
public class AppConfig {
}

```


```java
package com.akash.Q_3_4_5_6;

import org.springframework.stereotype.Component;

@Component

public class Desktop implements Computer {
    @Override  1 usage
    public void compile(){
        System.out.println("Desktop Compile");
    }
}
```

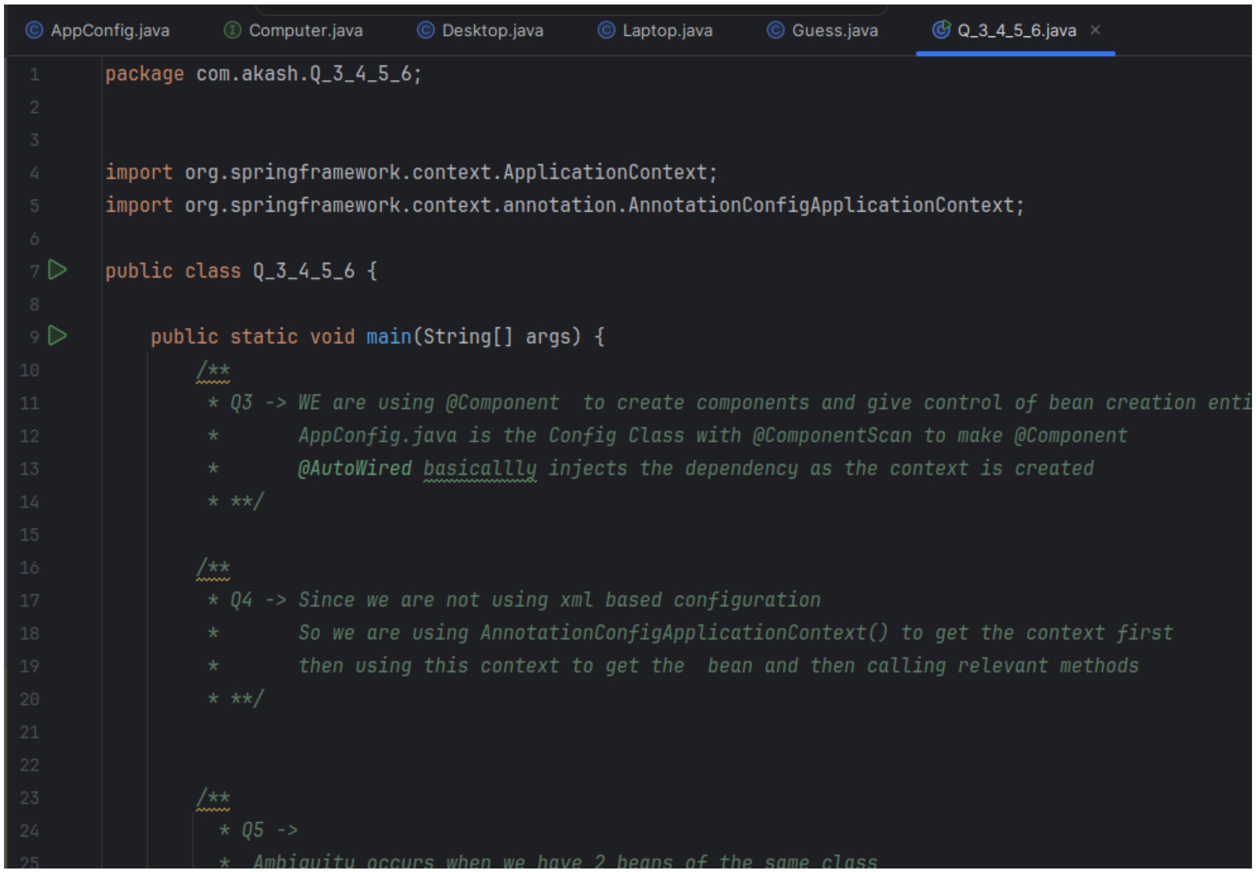

```java
package com.akash.Q_3_4_5_6;


import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;

// Q5) -> resolving ambiguity
@Component
@Primary
public class Laptop implements Computer{
    @Override  1 usage
    public void compile() {
        System.out.println("Laptop compiled");
    }
}
```



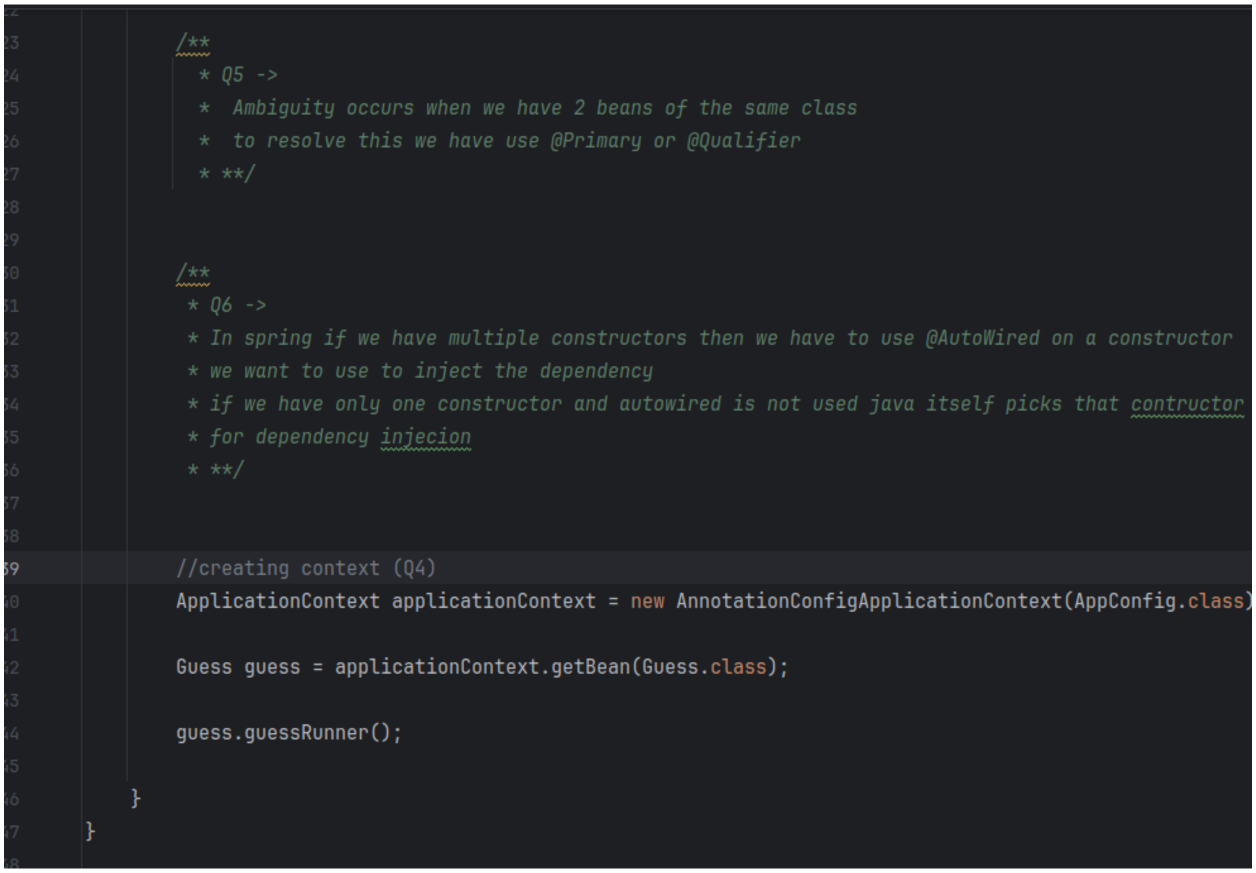
```java
package com.akash.Q_3_4_5_6;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;


//Q3 -> using component and autowired


@Component  2 usages
public class Guess {

    private Computer computer;  3 usages


    public Guess(){
        System.out.println("Guess Constructor");
    }

    //Q6 -> constructor injection
    @Autowired
    public Guess(Computer computer) {
        this.computer = computer;
        System.out.println("Guess Constructor");
    }

```



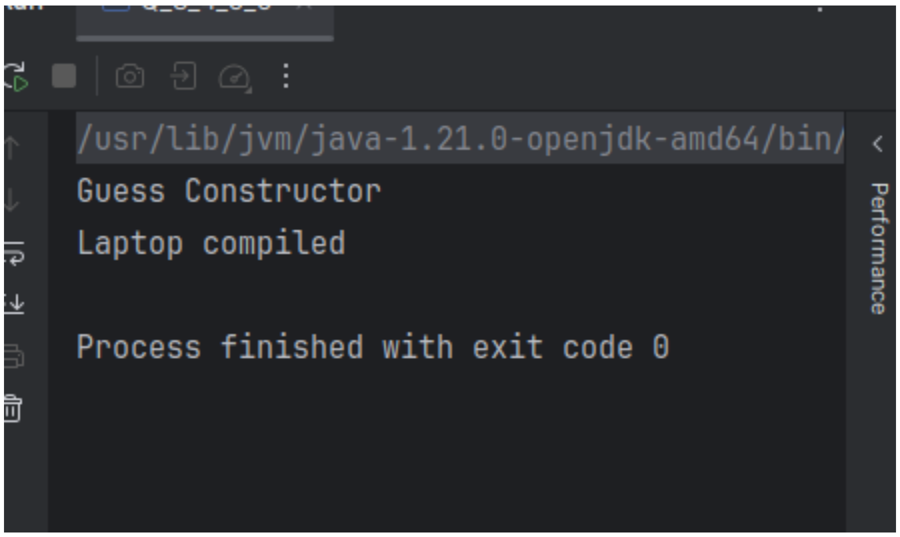
```java
    public Guess(){
        System.out.println("Guess Constructor");
    }

    //Q6 -> constructor injection
    @Autowired
    public Guess(Computer computer) {
        this.computer = computer;
        System.out.println("Guess Constructor");
    }


    public void setComputer(Computer computer) {  no usages
        this.computer=computer;
        System.out.println("setComputer");
    }

    public void guessRunner() {  1 usage
        computer.compile();
    }
}
```


AppConfig.java   Computer.java   Desktop.java   Laptop.java   Guess.java   Q_3_4_5_6.java

```java
package com.akash.Q_3_4_5_6;


import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Q_3_4_5_6 {

    public static void main(String[] args) {
        /**
         * Q3 -> WE are using @Component  to create components and give control of bean creation enti
         *       AppConfig.java is the Config Class with @ComponentScan to make @Component
         *       @AutoWired basicallly injects the dependency as the context is created
         * **/

        /**
         * Q4 -> Since we are not using xml based configuration
         *       So we are using AnnotationConfigApplicationContext() to get the context first
         *       then using this context to get the  bean and then calling relevant methods
         * **/


        /**
          * Q5 ->
          *  Ambiguity occurs when we have 2 beans of the same class
```



```
        /**
          * Q5 ->
          *  Ambiguity occurs when we have 2 beans of the same class
          *  to resolve this we have use @Primary or @Qualifier
          * **/


        /**
         * Q6 ->
         * In spring if we have multiple constructors then we have to use @AutoWired on a constructor
         * we want to use to inject the dependency
         * if we have only one constructor and autowired is not used java itself picks that contructor
         * for dependency injecion
         * **/


        //creating context (Q4)
        ApplicationContext applicationContext = new AnnotationConfigApplicationContext(AppConfig.class)

        Guess guess = applicationContext.getBean(Guess.class);

        guess.guessRunner();

    }
}
```



```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/
Guess Constructor
Laptop compiled

Process finished with exit code 0
```