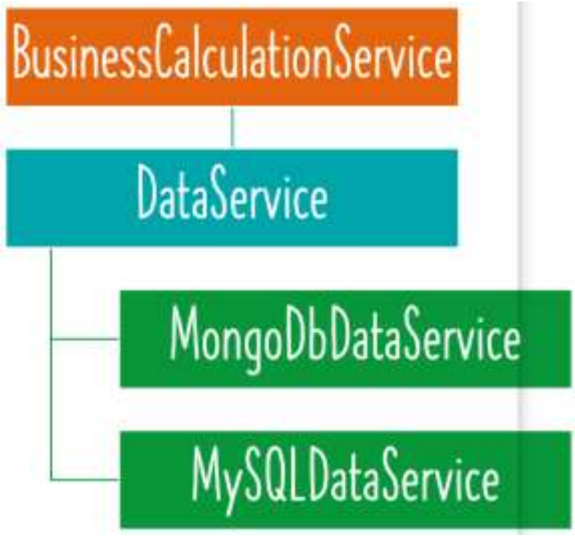
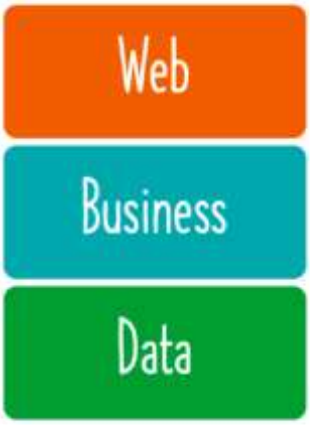
WHY DO WE HAVE A LOT OF DEPENDENCIES?

1. In Game Application example, we have very few classes.
2. But Real – World applications are much more complex:
    a. Multiple Layers (Web, Business, Data etc)
    b. Each layer is dependent on the layer below it!
        i. **Example**: Business Layer class talks to a Data Layer class.
            1. Data Layer class is a dependency of Business Layer class.
        ii. There are thousands of such dependencies in every application!
3. With Spring Framework:
    a. Instead of focusing on objects, their dependencies and wiring,
        i. You can focus on the business logic of your application!
    b. Spring Framework manages the lifecycle of objects:
        i. Mark components using annotations: @Component (and others...)
        ii. Mark dependencies using @Autowired.
        iii. Allow Spring Framework to do its magic!





EXERCISE:

1. Create classes and interfaces as needed.
    a. Use constructor injection to inject dependencies.
    b. Make MongoDbDataService as primary.
    c. Create a Spring Context
        i. Prefer annotations.
        ii. Retrieve BusinessCalculationService bean and run findMax method.

SOLUTION

1. **Create classes and interfaces as needed.**

MongoDbDataService.java

```
package com.naveen.learnspringframework.DependencyExercise;

public class MongoDbDataService implements DataService {

    @Override
    public int[] retrieveData() {
        return new int[] {50, 60, 70, 80};
    }

}
```

MySQLDataService.java

```
package com.naveen.learnspringframework.DependencyExercise;

public class MySQLDataService implements DataService {

    @Override
    public int[] retrieveData() {
        return new int[] {10, 20, 30, 40};
    }

}
```

BusinessCalculationService.java

```
package com.naveen.learnspringframework.DependencyExercise;

import java.util.Arrays;

public class BusinessCalculationService {

    private DataService dataService;

    public int findMax() {
        return Arrays.stream(dataService.retrieveData()).max().orElse(0);
    }
}
```

a) Use constructor injection to inject dependencies.

BusinessCalculationService.java

```
package com.naveen.learnspringframework.DependencyExercise;

import java.util.Arrays;

import org.springframework.stereotype.Component;

@Component
public class BusinessCalculationService {

    private DataService dataService;

    public BusinessCalculationService(DataService dataService) {
        super();
        this.dataService = dataService;
    }

    public int findMax() {
        return Arrays.stream(dataService.retrieveData()).max().orElse(0);
    }
}
```

b) Make MongoDbDataService as primary.

MongoDbDataService.java

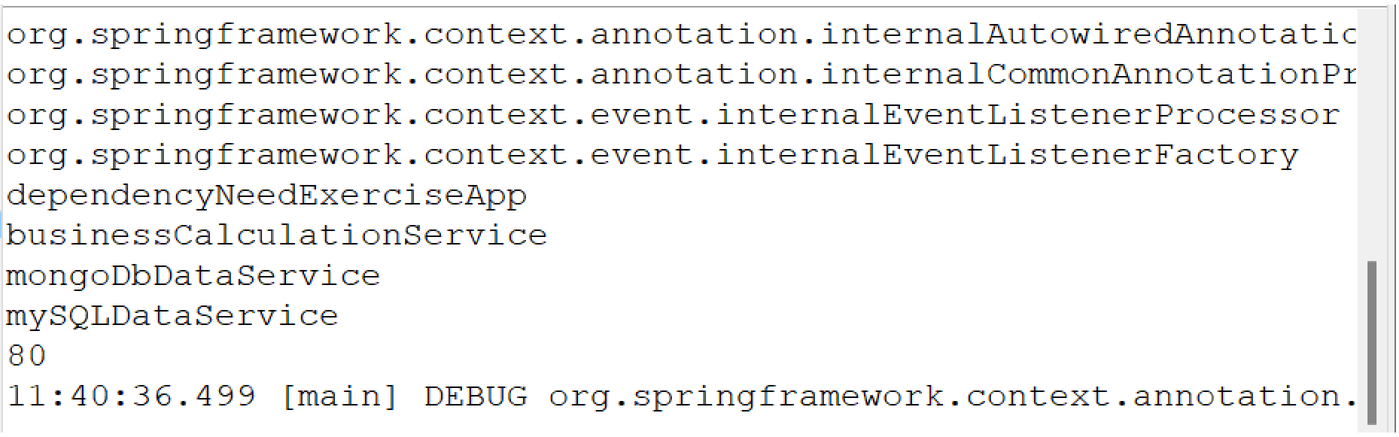
```
package com.naveen.learnspringframework.DependencyExercise;

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


@Component
@Primary
public class MongoDbDataService implements DataService {

    @Override
    public int[] retrieveData() {
        return new int[] {50, 60, 70, 80};
    }

}
```

c) Create a Spring Context.
   a. Prefer annotations.
   b. Retrieve BusinessCalculationService bean and run findMax method.

DependencyNeedExerciseApp.java

```java
package com.naveen.learnspringframework.DependencyExercise;

import java.util.Arrays;

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

@Configuration
@ComponentScan
public class DependencyNeedExerciseApp {

    public static void main(String[] args) {
        try (var context
                = new AnnotationConfigApplicationContext
                    (DependencyNeedExerciseApp.class)) {

            Arrays.stream(context.getBeanDefinitionNames())
                .forEach(System.out::println);

            System.out.println(
            context.getBean(BusinessCalculationService.class).findMax());
        }
    }

}
```

Output:



```
org.springframework.context.annotation.internalAutowiredAnnotatic
org.springframework.context.annotation.internalCommonAnnotationPr
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
dependencyNeedExerciseApp
businessCalculationService
mongoDbDataService
mySQLDataService
80
11:40:36.499 [main] DEBUG org.springframework.context.annotation.
```