ASD project

You can do this project either by yourself or in a group of 2 (not more)
You have 3 days for this lab:

- Friday August 11 PM
- Saturday August 12 whole day
- Monday August 14 whole day
- Tuesday August 15 AM (Till 12:30 PM)

Part A
Write your own Spring Framework that instantiate all classes that are annotated with @Service. The framework should also apply dependency injection to all fields that are annotated with @Autowired. Write an example application that uses the framework and shows that it works correctly. This application should get a bean out of the context and call a method on it.

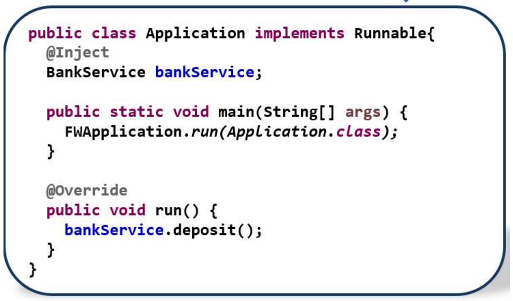
Part B
Modify your framework so that it also supports:

- Field injection by name using the @Qualifier annotation
- Setter injection by placing the @Autowired on the setter method
- Constructor injection by placing the @Autowired on the constructor

Show the correct working in the application that uses the framework.

Part C
Modify your framework so that it also supports value injection by using the @Value annotation on a field. The value that should be injected is specified in the application.properties file.
Show the correct working in the application that uses the framework.

Part D
Modify your framework so that the Application class with the main method looks similar as the Application class in Spring Boot:

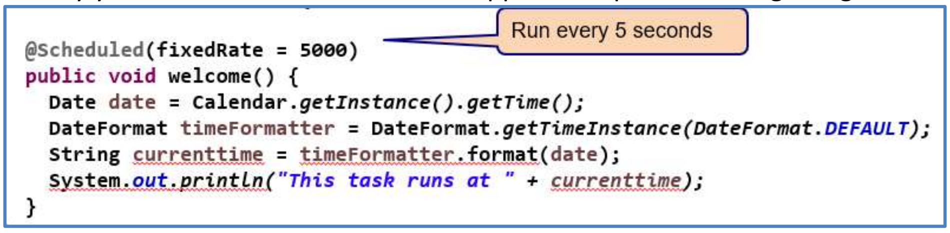


```
public class Application implements Runnable{
  @Inject
  BankService bankService;

  public static void main(String[] args) {
    FWApplication.run(Application.class);
  }

  @Override
  public void run() {
    bankService.deposit();
  }
}
```

Package the framework in a separate jar file
Show the correct working in the application that uses the framework. This application should have the framework jar as dependency.

Part E
Modify your framework so that it also supports profiles like in Spring Boot.
Show the correct working in the application that uses the framework

Part F
Modify your framework so that it also supports simple scheduling using the @Scheduled annotation.



```java
@Scheduled(fixedRate = 5000)
public void welcome() {
  Date date = Calendar.getInstance().getTime();
  DateFormat timeFormatter = DateFormat.getTimeInstance(DateFormat.DEFAULT);
  String currenttime = timeFormatter.format(date);
  System.out.println("This task runs at " + currenttime);
}
```

Show the correct working in the application that uses the framework
(Hint: use java.util.Timer)

Part G
Modify your framework so that it also supports simple scheduling using the @Scheduled annotation with a simple cron expression:
@scheduled(cron = “5 0”) which should run every 5 seconds
@scheduled(cron = “5 1”) which should run every 1 minute and 5 seconds (65 seconds)
@scheduled(cron = “7 5”) which should run every 5 minutes and 7 seconds (307 seconds)
Show the correct working in the application that uses the framework

Part H
Modify your framework so that it also supports events (publish-subscribe) like Spring Boot.
Show the correct working in the application that uses the framework

Part I
Modify your framework so that it also supports asynchronous methods using the @Async annotation like in Spring Boot.
Show the correct working in the application that uses the framework
(Hint: use CompletableFuture.runAsync())

Part J (Extra credit)

1. Modify your framework so that it also supports simple AOP using the @Before and @After annotation like in Spring Boot. You can use a very simple pointcut expression like:
   @After(pointcut="Customer.setName")

2. Modify your framework so that it also supports simple AOP using the @Around annotation like in Spring Boot.
   Show the correct working in the application that uses the framework

For all parts, you do not need to worry about error handling and exceptions. You can assume that the application that uses your framework is configured in the correct way.

We have done 13 labs so far which gives a total of 130 points.
This project counts for an additional 100 points.
You can get a total of 230 points which counts for 5% of the grade (5 points out of 100)
If you complete the extra credit and it works correctly, you get 1 extra point (so you get 6 instead of 5 points for the labs). You only get the 1 extra credit point if it works completely and correctly. If it does not work completely, you do not get the extra credit point.

What to hand in?

1. A text file where you describe which parts (part A – part J) you successfully implemented.
2. A separate zip file for part A
3. A separate zip file for part B
4. A separate zip file for part C
5. A separate zip file for the framework of part D and a separate zip file for the application of part D
6. A separate zip file for the framework of part E and a separate zip file for the application of part E
7. A separate zip file for the framework of part F and a separate zip file for the application of part F
8. Etc.

Submit all solutions in Sakai.
If you do this project in a group of 2, specify clearly with who you did this project. **Both students should submit their solutions in Sakai**.
You have till **Tuesday 12:30 PM** to submit your solutions. If you submit too late, you will lose points.

We will discuss this lab on **Tuesday 1:30 PM** sharp.