ASD project

You need do this project in a group of 2 students. Both students need to be from a different country.

You have 3.5 days for this lab:

- Friday August 9 whole day (after short session in the morning)
- Monday August 12 afternoon
- Tuesday August 13 whole day
- Wednesday August 14 whole day

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.

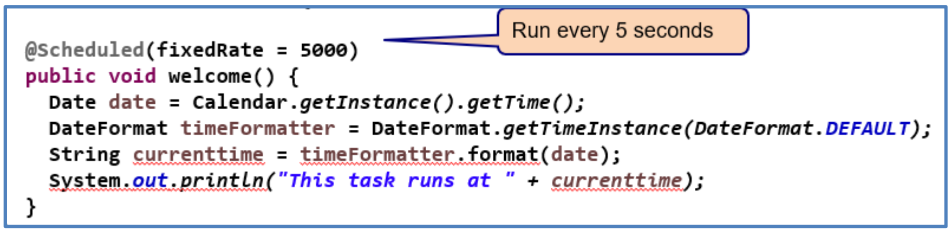
Part D
Modify your framework so that the Application class with the main method looks similar as the Application class in Spring Boot:

```
public class Application implements Runnable {

   @Autowired
   private ICustomerService customerService;

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

   @Override
   public void run(String... args) throws Exception {
     customerService.addCustomer();
   }
}
```

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.



```
@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

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 reading related configuration properties from the application.properties file using the @ConfigurationProperties annotation like in Spring Boot.
Show the correct working in the application that uses the framework

Part J
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

Part K Extra credit (0.5 point)
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")
Show the correct working in the application that uses the framework

Part L Extra credit (0.5 point)
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

Presentation
Record a presentation where you present your solution in a similar way like the presentations we do every morning to discuss the labs.
In the presentation explain the important aspects of your solution and demo your code.

In your presentation do NOT repeat or explain the topics we already studied in this course. Assume your audience know all these topics. Do not repeat the lab question. Only show how you solved the problems and demo that it works.

You are not allowed to use code from another team. It is important that every team writes the code themselves.

The due date for this project is **Thursday at 9:45 AM**. On Thursday some students may present their presentation in class. During the project you don't need to come to class.

What to hand in?
1. A text file where you describe which parts (part A – part J) you successfully implemented.
2. Link to the recorded presentation (save it in Google drive or some other cloud service). Make sure I have access to this recording. **Make it accessible to everyone.**
3. A separate zip file for the framework and a separate zip file for the application

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 **Thursday 9:45 AM** to submit your solutions. If you submit too late, you will lose points.