Lab 12

Part A: Scheduling

In the Bank application, add a new class BankStatementPrinter, that prints out every 20 seconds the details of all accounts to the console.

Part B: Events

In the Bank application, for every change in an account, do the following 2 things using events:

1. Send an email message to the account holder. We are not going to send a real email, but just do a System.out.println() to the console. (Don't worry about the email address because the Customer does not have an email address).
2. Write a trace record to the database using JPA with the following data:
    - Date & time
    - AccountNumber
    - Operation done on the account
    - Amount of the operation

Part C: Spring configuration

a. Write a Spring Boot application where you provide the following data in **application.properties**:

   - Application name
   - Version
   - Server URL
   - Server name
   - User firstname
   - User lastname
   - User username
   - User password
   - A list of countries

   The application writes the configured values to the console.

b. Change the application.properties file to a application.yml file

c. Add validation to the properties:
   - Application name — not blank
   - Version — not blank
   - Server URL — not blank
   - Server name
   - User firstname
   - User lastname
   - User username — not blank, between 8 and 15 characters
   - User password — not blank, between 8 and 15 characters
   - A list of countries

Part D: Answer the following practice exam questions

Question 1

Suppose you need to develop a REST interface for the functionality of the shopping cart in a webshop. The REST API needs to implement the following functionality:

- **getShoppingCart(cartNumber)** *//returns the content of the shopping cart*
- **addProductToShoppingCart(cartNumber, productNumber, quantity)** *//adds <quantity> product(s) to the shopping cart*
- **removeProductFromShoppingCart(cartNumber, productNumber, quantity)** *//removes <quantity> product(s) from the shopping cart*
- **changeQuantityInShoppingCart(cartNumber, productNumber, quantity)** *//changes the quantity of the given product in the shopping cart*
- **clearShoppingCart(cartNumber)** *//removes all products from the shopping cart*
- **checkoutShoppingCart(cartNumber)** *//place an order with the content of the shopping cart*

Write the **signature of all methods** in the ShoppingCartController, including the **necessary annotations.**
A signature of a method contains the following content:
return-type method-name (parameter-type parameter-name, …)

An example of a method signature with corresponding annotation is:

@Getmapping(...)
Customer getCustomer(int customerNumber)

Do **NOT** write the body of these methods.
Follow the API best practices we have learned in this course.

For this question it is important that you show for each method the **mapping annotation** and the **signature** of every method.

Question 2

Suppose we write a JMS listener in the following way:

```
@Component
public class CalculatorMessageListener {
  int currentvalue;

  @JmsListener(destination = "calcQueue")
  public void receiveMessage(final String calcAsString) {
    ObjectMapper objectMapper = new ObjectMapper();
    try {
      Calculation calculation = objectMapper.readValue(calcAsString,
Calculation.class);
      if (calculation.action.equals('+'))
          currentvalue = currentvalue + calculation.value;
      if (calculation.action.equals('-'))
          currentvalue = currentvalue - calculation.value;
      System.out.println(currentvalue);
    } catch (IOException e) {
      System.out.println("JMS receiver: Cannot convert : " + calcAsString +" to a
Calculation object");
    }
  }
}
```

Suppose there are 1000 client applications that send around 100 calculation messages to this listener every minute.

a. Explain if the given implementation of the JMS listener is correct or not. Does it print the correct currentvalue of the calculator according the messages it receives? If it is not correct, explain the issue with this implementation.
b. If it is not correct, how can we solve it so that the listener does print the correct values. (do not write code, write in your own words what we can do to solve the problem.

Question 3 [15 points] {20 minutes}

Given is the following code:

```
@Component
public class FileReader {

   public int countHowOfterStringAppearsInFilesOfDirectory(String searchString, String
directory) {
      ...
   }
}

@Component
public class StringCounter {
   @Autowired
   private FileReader fileReader;
   @Autowired
   private CountProcessor processor;
   @Autowired
   private Logger logger;
   @Autowired
   private EmailSender emailSender;

   public void countStringInFiles() {
      int count = fileReader.countHowOfterStringAppearsInFilesOfDirectory("searchString",
"directory");
      processor.process(count);
      logger.log("Count = "+count);
      emailSender.sendEmail("Count = "+count, "receiver@gmail.com");
   }
}
```

If we call the method countStringInFiles() on the StringCounter class, it first calls the method countHowOfterStringAppearsInFilesOfDirectory() on the FileReader. This methods goes to the file system and counts how often a given String appears in all files of the given directory. This method takes a lot of time. It can take up to 5 minutes before this method finishes.

Once the count is returned, the processor will process the count value. This process() method on the processor is a very slow method and can also take up to 5 minutes to finish.

The problem our project has with this code is 2 fold:

1. The method call to countHowOfterStringAppearsInFilesOfDirectory() on the FileReader is a blocking call. The thread will be blocked by this call for about 5 minutes. This is a

big problem in our application. We need to change the code so that the thread is not blocked when we call the method.

2. Once we have the count value, we do 3 things: we process the value, we log it and we email it. Now this goes in a serial way, we call the logger once the processor is finished. We want to call all 3 methods (process, log and sendEmail) in parallel. So once we have the count value all 3 methods should be called at the same time.

Rewrite the given code such that both problems are solved. Show **ALL** necessary code and annotations that show your solution.