COP3330C Module 5 Graded Assignment

For this assignment we will we will modify the appointment application by applying the Factory Method pattern in order to delegate the responsibility for instantiating appointments to factory subclasses for each Scrum meeting category. We will also implement a lambda expression which allows the factories to send the reminders.

We will continue using the Scrum meeting types as described in **The 2020 Scrum Guide**, https://scrumguides.org/scrum-guide.html).

1. Sprint Planning Meeting:
2. Daily Scrum:
3. Sprint Review Meeting:
4. Sprint Retrospective Meeting:
5. Backlog Refinement Meeting:

Start with the published Module 4 programming assignment solution (or your solution) and make the following changes. Be sure to add an ID header for each separate .java file, if you use my solution, add your ID header information after mine.

Remember to use incremental development in the following steps! Make a change, test your code. The application should run as before after each change. Keep your build status "green" (or "yellow", at least; there are some IDE warnings that I choose to ignore). If you encounter errors ("red"), fix them before continuing.

Declare the ScrumAppointmentFactory class as abstract and provide an abstract createAppointment method as shown here:

```
public abstract class ScrumAppointmentFactory {

    protected abstract Appointment createAppointment(
            String title, String description, Contact contact,
            ZonedDateTime apptTime);
}
```

Remove all other code from this class.

Note that this is a **parameterized factory method** as we are adding these fields to the Appointment object.

Create five ScrumAppointmentFactory subclasses, one for each meeting type which inherits from the ScrumAppointmentFactory class and implement the createAppointment method. Here is the code for one of those factory subclasses which represents a factory for Scrum planning meetings:

```
// factory subclasses
class Appointment_SprintPlanning_Factory extends ScrumAppointmentFactory {
    @Override
    protected Appointment createAppointment(
            String title, String description, Contact contact,
            ZonedDateTime apptTime) {
        return new Appointment_SprintPlanning(
                title, description, contact, apptTime);
    }
}
```

The createAppointment method implementation in each factory subclass must instantiate the appropriate meeting object based on the factory subclass. The method returns an Appointment superclass, which is a polymorphic reference to the runtime object that is created.

Note that with the factory subclass implementations and removal of the meeting type enum code lines from the parent class, we no longer need the meeting type enum. I have commented it out in my source code, in case we need to bring it back in a future version:

```
//enum ScrumAppointmentType { SPRINT_PLANNING,
//                            DAILY_SCRUM,
//                            SPRINT_REVIEW,
//                            SPRINT_RETROSPECTIVE,
//                            BACKLOG_REFINEMENT }
```

Remove the createAppointment methods in each Appointment subclass, these classes will now only contain a constructor; the factory methods in the factory subclasses now call createAppointment. I have added a debugging println call to each constructor to verify the correct class constructor is being called as the Appointment objects are instantiated (see the first few lines of the output shown below).

Here is what my ScrumPlanning subclass looks like now:

```
// subclasses for Scrum meeting types
// Sprint Planning
class Appointment_SprintPlanning extends Appointment {
    public Appointment_SprintPlanning(String title, String description,
                                      Contact contact, ZonedDateTime apptTime) {
        super(title, description, contact, apptTime);
        System.out.println("building a SprintPlanning appointment");
    }
}
```

Modify the Calendar reminder time by splitting each method out into separate interfaces; this will create two separate functional (SAM) interfaces. You will need to rename the Java source file, but IntelliJ makes this simple by right clicking on the red “squiggly” line that appears when you change the public interface’s name, then selecting “Context Actions” / “Rename file”.

```java
package edu.fscj.cop3330c.calendar;

public interface ReminderBuilder {
    // build a reminder in the form of a formatted String
    public String buildReminder(Appointment appt);

}

interface ReminderSender {
    // send a reminder using contact's preferred notification method
    public void sendReminder(String reminder);
}
```

Modify the Appointment application class to implement the new Reminder interface, but not the ReminderSender interface.

```java
// main application class
public class AppointmentApp implements ReminderBuilder {
```

Comment out the sendReminder method in this class and in its place create a lambda expression in the main method which performs the same task (prints out the reminder). Instead of calling sendReminder, use the lambda.

Generate one appointment of each kind using the factory. For this assignment don't worry about fabricating reminder times to generate a reminder, just force the reminder to show for each appointment when you check the reminder times (in other words, you do not need to see if the reminder time matches the current time).

Submit your solution to the GitHub classroom repo created when you accepted the assignment invitation.

Sample output (note the different meeting descriptions, which correspond to a specific factory/subclass):

```
building a SprintPlanning appointment
building a DailyScrum appointment
building a SprintReview appointment
building a SprintRetrospective appointment
building a BacklogRefinement appointment
Sending the following SMS message to John McReady at (904) 555-1212
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Hello, John McReady!                                       +
+ This is a reminder that you have an upcoming appointment. +
+                                                           +
+ Title: Appointment1                                       +
+ Description: Sprint Planning                              +
+ Date: 3 November, 2023                                    +
+ Time: 05:23 PM America/New_York                           +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sending the following email message to Keith Childs at
KeithChilds@email.com
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Hello, Keith Childs!                                       +
+ This is a reminder that you have an upcoming appointment. +
+                                                           +
+ Title: Appointment2                                       +
+ Description: Daily Scrum                                  +
+ Date: 3 October, 2023                                     +
+ Time: 05:23 PM America/New_York                           +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sending the following SMS message to T.K Nauls at (904) 555-3434
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Hello, T.K Nauls!                                          +
+ This is a reminder that you have an upcoming appointment. +
+                                                           +
+ Title: Appointment3                                       +
+ Description: Sprint Review                                +
+ Date: 3 November, 2023                                    +
+ Time: 05:23 PM America/New_York                           +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
Sending the following email message to Richard Copper at
RichardCopper@email.com
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Hello, Richard Copper!                                      +
+ This is a reminder that you have an upcoming appointment.  +
+                                                            +
+ Title: Appointment4                                        +
+ Description: Sprint Retrospective                          +
+ Date: 3 July, 2024                                         +
+ Time: 05:23 PM America/New_York                            +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sending the following SMS message to Wilford Blair at (904) 555-5656
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Hello, Wilford Blair!                                       +
+ This is a reminder that you have an upcoming appointment.  +
+                                                            +
+ Title: Appointment5                                        +
+ Description: Backlog Refinement                            +
+ Date: 3 October, 2024                                      +
+ Time: 05:23 PM America/New_York                            +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```