Unit 2 – Objects and Classes Program 4 – Activity Class

Background: You are designing a new calendar application that will allow a user to create events and keep track of upcoming activities. Such a project would require a LOT of work, so you wisely decide to implement the project in smaller pieces. This assignment represents the first steps towards making a full calendar application. You will write a portion of the class representing items on the calendar and a small runner to test the various methods.

Before submitting, make sure ALL of the provided tests pass and that your runner (main method) also works. Programs with non-functional or incomplete runner will not receive full credit. Please submit a zip file of the ENTIRE DIRECTORY to Canvas when you are done.

1) Part 1 – Create the Activity Class

IMPORTANT: Please follow the directions below EXACTLY in terms of the variable and method names to be used. Failure to do so means you will NOT pass the provided tests.

Create a classed named `Activity`. This class represents any meeting, sporting event, appointment, etc. that might appear on a calendar. Each `Activity` has a `name` (which describes the event) and a `day` the event occurs on (just "Monday" through "Sunday" for now – we won't be including dates for this project). Both `name` and `day` are `Strings`. Each `Activity` has additional instance variables for a `startTime` and a `duration` in whole minutes. Both `startTime` and `duration` should be created as `ints`. In addition, `startTime` will be kept in 24-hour format. For example, 6:15 am would be represented as 615 while 7:30 pm would be represented as 1930. NOTE: the leading zero that can appear before morning times (like 0615) will NOT be implemented.

The `Activity` class should have a pair of overloaded constructors: a fully parameterized constructor that takes the `name`, `day`, `startTime` and `duration` *in that order*, and a default constructor which sets the `name` to "Unknown", the `day` to "Monday", the `startTime` to 5:oo pm and the `duration` to 1 hour.

Simple accessors and mutators (getters and setters) should be provided for each of the instance variables since calendar events often change. These methods should use the standard naming convention for methods of this type (`getName`, `setStartTime`, etc.).

Additionally, a `toString` method should be created that returns a `String` including the `name`, `day`, `startTime` and `duration` each separated by a tab. Do not include anything else (no extra spaces, no labels, no punctuation). For example:

```
Practice	Monday	1700	60
```

Two additional methods will complete phase 1 of your project. The first method, `getEndTime` takes no parameters. The method calculates the ending time of the event using the object's `startTime` and `duration` values. The ending time should be in proper 24-hour format and be returned as an `int`. You can assume that the ending time will be on the same day as the starting time, so you do not have to worry about time roll-over at midnight. Think about what needs to happen here. This is NOT a simple addition. For example, an `Activity` with a `startTime` of 730 and a `duration` of 90 would have an end time of 900.

The final method is called `overlap`. This method takes a single parameter which is another `Activity` object. The `overlap` method compares the input to the current `Activity` and determines if the events occur at the same time (either fully or partially). This method returns a `boolean` value indicating if overlap occurs (true if they overlap, false otherwise). You might want to draw some pictures here to make sure you cover the various overlap possibilities. Think about an easy way to tell if one event overlaps with another based on the day they occur, their start times and their end times (which you can get with from the method above).

2) Part 2 – Create a short runner program to test your class

To test your class, you decide to write a short runner program as well. Your `main` method should be in a class called `ActivityRunner` and complete the following actions:

1. Create `Activity a1`: "Homework" on Friday at 5:00pm that lasts for 1 hour
2. Print each of `a1`'s instance variables using the getters. DO NOT USE `toString` on this part! The purpose here is to test the getter methods.
3. Create `Activity a2` using the default constructor
4. Print activity `a2` using the `toString` method
5. Call the `overlap` method to see if these `Activities` overlap (they should not since they are on different days). Note that you will need to print out the result of the method call or you won't know if this worked or not.
6. Change `a2`'s `name` to "Football Game", `day` to "Friday", `startTime` to 6:00pm and `duration` to 3 hours (use the setters). Print activity `a2` again to verify all the values changed.
7. Determine the end time of the football game using the `getEndTime` method (should be 2100 – that's 9:00pm) and print it out.
8. Determine if these two updated `Activities` occur at the same time using the `overlap` method (they should not).
9. Finally, update `Activity a2`'s `startTime` to be 5:30pm and call `overlap` one more time. This time, the `Activities` *should* overlap.