CSD-3464 “Final Exam” Assignment 10

Submit this paper with your name, and the time that you leave the room at the top.

Starting files are available on GitHub via D2L. Please do not start from scratch. Use the starting files.

To complete this examination, you must build a solution that passes a sufficient amount of the grading criteria below. Some components will be tested automatically (and all unit tests will be provided) while others will be graded by hand.

Problem Description:

You are a Junior Java Developer at ABC Gum Corporation. You have been asked to build out the data model for a new scheduling system. Your team lead started the job, but was called away to a meeting and asked you to finish up. Attached are some design notes, and the lead has provided some work including the unit tests for the data model.

The scheduling system is responsible for setting and tracking working hours for employees across the company.

Every employee of the company falls into three top-level categories: salaried, flexible contractors, and fixed-schedule contractors. Salaried employees work 9:00am to 5:00pm, Monday to Friday. Contractors have hours that vary, and submit timesheets.

You must build an abstract class called **`Schedule`** that extends into three regular classes **`SalarySchedule`**, **`FlexSchedule`** and **`FixedSchedule`**.

Each day’s pair of starting/ending hours are to be stored in a class called **`WorkingHours`** that has only four properties: starting hours and minutes, and ending hours and minutes. **`WorkingHours`** objects should default to starting at 9:00 and ending at 17:00 (5:00pm). Each **`Schedule`** class will have an array of seven **`WorkingHours`** objects, indexed 0=Sunday. Your supervisor has already built the **`WorkingHours`** class, and configured it to throw a **`TimeException`** if an object is created or modified to have an invalid time (eg- 25:94).

Salaried employees are able to accumulate and spend “flex days” (a combination of sick-days and vacation). **`SalarySchedule`** should track the number of these remaining as a field.

Flexible employees do not get sick days or flex days, but instead get a total number of hours to work (eg- 24 hours per week), and an overflow: if an employee doesn’t meet their scheduled commitment one week, they are expected to overflow those hours into the next week. The overflow time is rounded up to the nearest hour and stored as an integer. To aid this operation, there is a method called **`getTimeLeft()`** which uses the **`WorkingHours duration`** method to find out how much time has been used so far in the week, and subtracts that from the total.

Fixed schedule employees have a fixed schedule that is not necessarily 9-5. They do not receive flex days. This is, in all respects, just an implementation of the abstract class **`Schedule`**.

The following validation schemes should be set up for specific fields. If the validation fails, the system should fail silently, and the value should simply not be set.

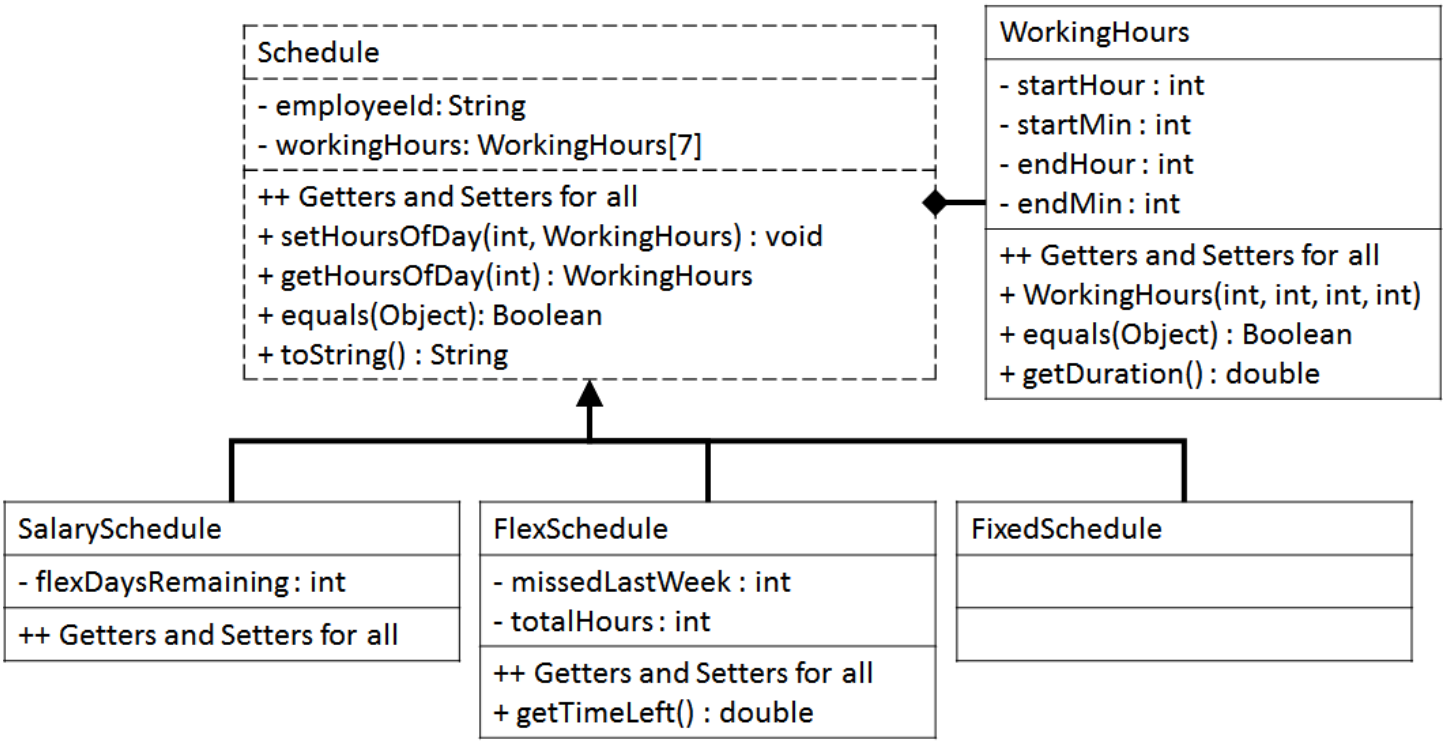
Employee ID is of the form **`abc0######`** where each **`#`** is a digit from 0-9. For example **`abc0123456`**.

Times must be valid. For example, 24:95 should not be an acceptable time. In the case of an invalid time, **`WorkingHours`** will throw a **`TimeException`**. This is already configured for you by your supervisor.

The **`Schedule`** class should include a **`toString`** that displays the schedule as JSON (see appendix.) Note: new **`toString`** implementations are not desired for sub-classes, and extra info beyond employee ID and times are unnecessary.

Each class should include an **`equals`** method. The **`WorkingHours`** class will compare hours and minutes, while the **`Schedule`** class will compare the contents of the **`workingHours`** array (simply iterate the array and use the **`WorkingHours`** equals method). Schedules do not include the employee ID in a check for equality.

Use the following UML diagram as a guide.



Your team lead has also asked you to build a proof-of-concept Schedule record. In the main method of your application, instantiate one object of each regular class: **`SalarySchedule`**, **`FlexSchedule`** and **`FixedSchedule`**. Use their getters and setters to build a sample, and put all of them in an array of **`Schedule`** objects. Iterate over the array and output the JSON objects as a valid JSON array. You can use www.jsonlint.com to validate your JSON objects.

Tasks to Accomplish for **`Schedule`** Class

- Make sure getters and setters are working correctly
- Build the **`equals`** method
- Fix issues with the **`toString`** method
- Make sure the class is **`abstract`** and cannot be instantiated

Tasks to Accomplish for **`SalarySchedule`**

- Create a default no-arg constructor that sets up a week of Mon-Fri 9am-5pm working hours
- Make sure getters and setters are working correctly for new fields

Tasks to Accomplish for **`FlexSchedule`**

- Make sure getters and setters are working correctly for new fields
- Build the **`getTimeLeft`** method that returns a decimal hour (eg 1.5 hours) for the difference between the scheduled hours and the total hours

Tasks to Accomplish for **`FixedSchedule`**

- Make sure that this class properly inherits from the **`Schedule`** abstract class

Grading Scheme

	MARKS
PASSES ALL UNIT TESTS FOR VALID SCHEDULE DATA	8
PASSES ALL UNIT TESTS FOR INVALID SCHEDULE DATA	4
PASSES UNIT TESTS FOR TOSTRING AND EQUALS METHODS	4
PASSES UNIT TESTS FOR DATA IN OTHER CLASSES	5
INSTANTIATES AN OBJECT OF EACH CLASS	3
CREATES A POLYMORPHIC ARRAY OF EACH CLASS AND ITERATES OVER IT TO OUTPUT	5
CORRECTLY IMPLEMENTS ABSTRACT CLASSES	2
USES CORRECT SOURCE FORMATTING	2
INCLUDES JAVADOC COMMENTS FOR ALL NEW METHODS	2
TOTAL	30

Note: There are 35 possible marks. Plan your solution for your strengths.

JSON Samples

```
[{
	"employeeId": "abc0123456",
	"Monday": {
		"startTime": "9:00",
		"endTime": "19:00"
	},
	"Tuesday": {
		"startTime": "9:00",
		"endTime": "19:00"
	},
	"Wednesday": {
		"startTime": "9:00",
		"endTime": "19:00"
	},
	"Thursday": {
		"startTime": "9:00",
		"endTime": "19:00"
	}
}, {
	"employeeId": "abc0456789",
	"Monday": {
		"startTime": "9:00",
		"endTime": "17:00"
	},
	"Wednesday": {
		"startTime": "9:00",
		"endTime": "17:00"
	},
	"Friday": {
		"startTime": "9:00",
		"endTime": "17:00"
	}
}, {
	"employeeId": "abc0234234",
	"Tuesday": {
		"startTime": "9:00",
		"endTime": "12:00"
	},
	"Wednesday": {
		"startTime": "12:00",
		"endTime": "17:00"
	},
	"Thursday": {
		"startTime": "9:00",
		"endTime": "17:00"
	}
}]
```