CS101- Algorithms and Programming I
Lab 08

Lab Objectives: `Classes and Objects.`

For all labs in CS 101, your solutions must conform to the CS101 style guidelines (rules!)

1. Create a class, `Date.java` that stores the day, month, year for a given date. You should NOT use any Java Date classes or functionality, you should define your own functionality for the following methods. All attributes should be private.

   Instance Data Members:
   - int day, int month, int year

   Static Data Members:
   - ArrayList that stores the number of days in each month.

   Methods:
   Methods below should be declared static where appropriate.
   - Constructor: initialize Date to year/month/day int parameters.
   - Overloaded Constructor: initialize Date to String parameter, assuming the date is formatted as: YYYY-MM-DD.
   - Overloaded Copy Constructor.
   - Getters for all private attributes.
   - Set methods for all private attributes. The set method should validate the data, where appropriate. For example, months can only be from 1 to 12. Depending on the month, days will be between 28 – 31. Use the static ArrayList containing the days in each month.
   - `isLeapYear()`: takes a year as a parameter and returns true if it is a leap year, false if not. Any year that is evenly divisible by 4 is a leap year. However, there is still a small error that must be accounted for. To eliminate this error, if a year that is evenly divisible by 100 it is a leap year only if it is also evenly divisible by 400.
   - `toString()`: returns a String representation of a date in the format: YYYY-MM-DD.
   - `daysBetween()`: takes a Date object as a parameter and calculates and returns the number of days between this Date and the given date. Your method should consider leap years where appropriate.
   - `addDays()`: takes an int number of days and add the number of days to the Date. Your method should consider leap years where appropriate.
   - `isBefore()`: takes a Date as a parameter and returns true if this Date is before the given Date, false if not.
   - `initDaysInMonth()`: initializes the ArrayList containing the number of days in each month.

2. Create a class, `Project.java` that defines the following data and behaviours:

   Data Members:

   - **projectId:** id of the project.
   - **projectName:** name of the project.
   - **projectDescription:** brief description of the Project.
   - **projectType:** project type may be: ST (Short Term), IM (intermediate), LT (long term).

- **projectStartDate:** `Date` project begins. Must be after 2000-01-01. Otherwise current date.
- **estimatedEndDate:** projected `Date` the project will finish. Must be after the projectStartDate, otherwise projectStartDate + 30 days is used.
- **actualEndDate:**sets the end date to the Date passed as a parameter. If the parameter is not null, sets the Date to the value passed as a parameter if the parameter end date is after the start date. Sets to null otherwise.
- **isActive:** true if project is active, false if completed/deactivated.
- **static projectCounter:** initialized to 1000.

Methods:

- **Constructor:** initializes projectName, projectDescription, projectedStartDate, projectedEndDate to ***STRING*** parameters. Dates should be initialized using set methods. Set the projectType and projectId using the set methods described below.
- **Getters for all private attributes.**
- **Setters for:** projectedStartDate, projectedEndDate, actualEndDate. Construct the Dates and validate according to the constraints given above.
- **setProjectType():** determines the type based on the following: projects with up to 1 year estimated duration are SHORT TERM, between 1 and 3 years: INTERMEDIATE, and over 3 years, LONG TERM.
- **setProjectId():** for SHORT TERM projects, ST-1000 (use projectCounter and increment), for INTERMEDIATE, IM-1000, and for LONG TERM, LT-1000.
- **deactivateProject():** sets the actualEndDate to String date passed as a parameter, sets İsActive to false.
- **activateProject():** sets the actualEndDate to null, sets isActive to true.
- **pushProject():** takes a number of days as a parameter and adds the number of days to the estimatedEndDate. Re-set the project type. (you do not need to reset the projectId).
- **estimatedDaysToCompletion():** returns the number of days between the start date and estimated end date.
- **equals():** two projects are equal if their names are the same.
- **toString():** returns a String representation of a Project.

3. Create an application, `ProjectApp.java` to test your `Project` class. You should test all of your functionality; some sample output is shown below.

Sample Output:

```
Project 1:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
Start: 2022-5-22 End: 2022-6-21

Project 2:
ID: (ST-1001)
LAB-IMPLEMENT
Update Lab Results Reports
Start: 2023-5-22 End: 2023-9-30
```

****PROJECT 1****
Estimated days to completion: 30

5 days late:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
Start: 2022-5-22 End: 2022-6-26

50 days late:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
Start: 2022-5-22 End: 2022-8-16

5 days late:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
Start: 2022-5-22 End: 2023-9-8

Deactivated:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
COMPLETED: 2023-10-15

Activated:
ID: (ST-1000)
Sys-Renew
Upgrade payroll system and hardware
Start: 2022-5-22 End: 2023-9-8