104824 – INTRODUCTION TO SYSTEMS PROGRAMMING – Spring 2025 ..... Practice Sheet
Practice Sheet 4

This practice sheet is not meant for submission, and will not have any grade. You can ask me about any questions you have, including checking your answers.

Warning Even though this guide will not have any grade, it is important to avoid copying/plagiarism. Avoid the usage of LLMs (such as ChatGPT, DeepSeek, etc). When solving a graded assignment, this warnings will be actual rules that must be followed.

Task 1 On Linux you can install *java* and *javac* with `sudo apt install openjdk-11-jdk` for *Java* version 11. For Windows and macOS you will need to look on how to install *Java JDK (Java Development Kit)* version 11 or above.

Task 2 Consider a project *dates*. Create a class `Date` for this project, this class represents dates on a calendar, with *days*, *month*, and *year*; also create a class `MonthCalendar`, which for a particular *month* in a particular *year* will hold every free day, and every busy day. Finally, create a class `MonthOrganizer` that will represent a *Java* program that can allow an user to interact with a `MonthCalendar`. *Use packages to organize your classes, and do not use BlueJ.*

A `Date` class must implement these methods:

SubTask 1 An empty constructor that sets the `Date` to the minimum date *15/10/1582*.

SubTask 2 A constructor that allows to create a new `Date` with a specific day, month, and year (greater or equal than the default `Date`).

SubTask 3 A `public int daysInMonth()` method, to return how many days the month of a `Date` has.

SubTask 4 A `private int daysInMonth(int month)` method, to return how many days a particular month has.

SubTask 5 A `public boolean isLeapYear()` method, to return if the current date belong to a leap year.

SubTask 6 A `private boolean isLeapYear(int year)` method, to return if a specific year is a leap year.

SubTask 7 Methods to get and change the *day*, *month*, and *year* of a `Date`.

SubTask 8 A `public String toString()` method, to return a string representation of a `Date`, the format must be *¡day¿/¡month¿/¡year¿* with two digits for the month.

SubTask 9 A `public boolean equals(Object other)` method, to check if this `Date` is equal to another.

SubTask 10 A `public int compareTo(Date other)` method, to check if this date is less (-1), equal (0), or greater (1) than the other `Date`.

SubTask 11 A `public int differenceInDays(Date other)` method, to calculate the difference in days between another `Date` which must be less or equal than the current one.

SubTask 12 A `public boolean repOk()` method, to validate the class invariant.

A `MonthCalendar` class must implement these methods:

SubTask 1 A constructor taking a *year* and *month* arguments; the *year* must be greater or equal than 1582; and the month must be between 1 and 12, but if the *year* is 1582 then the *month* must be greater or equal than 10. This constructor must start with all valid `Date`s as free.

SubTask 2 A `public void markAsBusy(Date date)` method, that marks a valid, free, `Date` as busy.

SubTask 3 A `public void freeDate(Date date)` method, that marks a valid, busy, `Date` as free.

SubTask 4 A `public String toString()` method, that will return a string representation of a `MonthCalendar`, the format must be *Month calendar: ¡month_name¿/¡year¿* followed by *Busy dates:* and a list (one per line) of `Date`s marked as busy; followed by *Free dates:* and a list (one per line) of `Date`s markes as free.

The class `MonthOrganizer` must implement a `public static void main(String args)` method to run a program that allows to interact with a `MonthCalendar`.

Task 3 Add all necessary class invariants, pre, and postconditions with their corresponding checks.