OOP3200 – Object Oriented Programming II

Java Lab 2
Classes

Due: Week 10 (Sunday November 22, 2020) @ midnight

Value 6%

Classes **Maximum Mark: 100**

Overview In this lab you and your lab partner will create a **Java version** of the **`WorkTicket`** class that we previously created in C++.

Objects of this class could be used in an IT support tracking application to store information about client requests for support. Add this new class to your project.

Instructions:
Section 1: Requirements for StandardDeck Class
(70 Marks: Functionality)

1. **Attributes.:** An object of class **`WorkTicket`** has the following attributes. Include a **`set`** (mutator) and **`get`** (accessor) method for each attribute. (35 Marks: Functionality)
    a. **Work Ticket Number** – A whole, positive number. If a work ticket number is set to a zero or a negative number, an **`IllegalArgumentException`** exception should be thrown, with an appropriate message. (10 Marks)
    b. **Client ID** – The alpha-numeric code assigned to the client. No special validation is needed in the set. (5 Marks)
    c. **Work Ticket Date** – The date the ticket was opened, stored as a **`java.time.LocalDate`** object. The **`LocalDate`** class will handle most of the validation, throwing a **`DateTimeException`** when an attempt to set the day of month to an invalid value occurs. You need to add class specific validation for the year, which must be in the 21st Century (between 2000 and 2099). An **`IllegalArgumentException`** exception should be thrown, with an appropriate message if the year is set out of range. (15 Marks)
    d. **Issue Description** – A description of the issue the client is having. No special validation is needed in the set. (5 Marks)

2. **Default and parameterized constructor(s).** (15 Marks: Functionality).
    a. If parameters are not specified, ensure that the work ticket number is **zero** and all other attributes are set to **`null`**. (5 Marks)
    b. The parameterized version should set the object's attributes to the parameters specified if they are valid. (5 Marks)
    c. If any of the parameters are not valid, an **`IllegalArgumentException`** should be thrown. (5 Marks)

3. **`SetWorkTicket()`:** a mutator method to set all the attributes of the object to the parameters if the parameters are valid (10 Marks: Functionality).
    a. ALL the parameters must be valid in order for ANY of the attributes to change. Validation rules are explained above for work ticket number and date. (5 Marks)
    b. Client number and Description must be at least one character long. If no problems are detected, return TRUE. Otherwise return FALSE. (5 Marks)
4. **`toString()`:** Override the **`toString()`** method inherited from **`Object`**. Return a string that shows all the object's attributes formatted neatly. This version of the class will not have a **`ShowWorkTicket()`** method. (10 Marks: Functionality).

Section 2: Program Requirements

(10 Marks: Functionality)

1. Your `main()` function should demonstrate each of the features to provide the appropriate input, data validation and output as described above. How you do this is up to you (10 Marks: Functionality).

Section 3: General Requirements

(5 Marks: Internal Documentation, 5 Marks: Version Control, 10 Marks: Demo Video)

1. Include **Internal Documentation** for your site **(5 Marks: Internal Documentation):**
    a. Ensure you include a **comment header** for your **C++ Files** that indicate: the **File name**, **Student's Name**, **StudentID, and Date,** (2 Marks: Internal Documentation).
    b. Ensure you include **method and function headers** for all of your **class methods,** and any **utility functions** (1 Marks: Internal Documentation)
    c. Ensure all your code uses **contextual variable names** that help make the files human-readable and follow the C++ style guide (1 Marks: Internal Documentation).
    d. Ensure you include **inline comments** that describe your program's functionality where appropriate. **Note:** Please avoid "over-commenting" (1 Marks: Internal Documentation)
2. Share your files on **GitHub** to demonstrate Version Control Best Practices **(5 Marks: Version Control).**
    a. Create an appropriately named GitHub Repository that you and your partner will use for this lab. Only one repository is required (1 Mark: Version Control)
    b. Your repository must include **your code** and be well structured (2 Marks: Version Control).
    c. Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).
3. Create a Short Video presentation on **YouTube** or another streaming provider. You must include a short **PowerPoint** (or Google Slides) Slide Deck that includes a **single slide** to start your video (10 Marks: Video)
    a. The **first** (and only) **Slide** of your Slide Deck must include **current images** of you and your partner (no avatars allowed) that are displayed appropriately on the page. You must also include your **Full Names**, **Student IDs**, the **Course Code**, **Course Name,** and your **Assignment** information. (2 Marks: video)

b. You or your partner will **demonstrate** your program's functionality. You must show your program working properly in the console (2 Marks: Video)
c. You or your partner will **describe** the code in your files that drives the functionality of your program (2 Marks Video).
d. Sound for your Video must at an appropriate level so that your voices may be **clearly heard,** and your screen resolution should be set so that your program's code and console details are **clearly visible** (2 Marks: Video).
e. Your Short Video should run no more than 5 minutes (2 Marks: Video).

SUBMITTING YOUR WORK

Your submission should include:

1. A zip archive of your program's Project files uploaded to DCConnect.
2. A working link to your appropriately named GitHub repository
3. A working link to your demo video posted on YouTube or another streaming provider

Evaluation Criteria

Feature	Description	Marks
Functionality	Program deliverables are me and site functions are met. No errors, including submission of user inputs.	80
Internal Documentation	File headers present, including file & student names & description. Functions and classes include headers describing functionality & scope. Inline comments and descriptive variable names included and follow style guide.	5
Version Control	GitHub repo created with commit history demonstrating regular updates. 2 marks for initial commit. 2 marks for working with GitHub throughout the development process	5
Video Presentation	Your short video must demonstrate your site and describe your code. Your audio must be at an appropriate level and your screen must be clearly seen.	10
Total		**100**

This assignment is weighted **6%** of your total mark for this course.

Late submissions:

- 20% deducted for each day late.

External code (e.g. from the internet or other sources) can be used for student submissions within the following parameters:

1. The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
2. It encompasses a maximum of 10% of your code (any more will be considered cheating).
3. You must understand any code you use and include documentation (comments) around the code that explains its function.
4. You must get written approval from me via email.