OOP3200 – Object Oriented Programming II

Java Lab 3
Inheritance

Due: Week 12 (Sunday December 6, 2020) @ midnight

Value 6%

Inheritance **Maximum Mark: 100**

Overview In the previous lab you developed a class called **`WorkTicket`**. In this lab you will use your **`WorkTicket`** class to derive a new class called **`ExtendedWorkTicket`**.

The class **`ExtendedWorkTicket`** inherits all the attributes of **`WorkTicket`** and adds a Boolean attribute called **`myOpen`** to indicate if the work ticket is open or closed.

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

1. **Accessors and Mutators.:** (15 Marks: Functionality)
    - **a.** Include an accessor called **`isOpen()`** to get the **`myOpen`** value. (5 Marks)
    - b. Include mutators called **`openTicket()`** and **`closeTicket()`** that will set the **`myOpen`** attribute. (10 Marks)

2. **Default and parameterized constructor(s).** (30 Marks: Functionality).
    - a. The default constructor should initialize all the inherited attributes using the default constructor from the base class and the **`myOpen`** attribute should be initialized to true. (10 Marks)
    - b. A parameterized constructor should initialize all the inherited attributes using the parameterized constructor from the base class and the **`myOpen`** attribute should be initialized to the last parameter. (10 Marks)
    - c. Include another parameterized constructor for **`ExtendedWorkTicket`** that accepts a **`WorkTicket`** object and a boolean as parameters and initializes the object appropriately. (10 Marks)

3. **`SetWorkTicket():`** (15 Marks: Functionality).
    - a. Overload **`SetWorkTicket()`** to include an additional parameter to set the **`myOpen`** attribute. (5 Marks)
    - b. Call the base class version of **`SetWorkTicket()`** to set all the inherited attributes. (5 Marks)

c. Like the base class version of **`SetWorkTicket()`**, this method should only set the attributes and return true if all the parameters were valid. If not, none of the attributes should be set and it should return false. (5 Marks)

4. **`toString():`** (10 Marks: Functionality).
    a. Override the **`toString()`** method so that it calls the base class version to build a String with the inherited attributes (5 Marks)
    b. It should then append either "Ticket: OPEN" or "Ticket: CLOSED" depending on the myOpen attribute (5 Marks)

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 **Java 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 Java 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.