Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

Workshop # 4 & 5

Due Date: 29 March 2024

INSTRUCTIONS

- *This workshop must be completed individually without any outside collaboration. All work must be your own. Copying or reproducing the work done by others (in part or in full) or letting others to copy or reproduce your own work is subject to significant grade reduction or getting no grade at all and/or being treated as academic dishonesty under the College's Academic Dishonesty Policy.*

- *Your goal is to finish the design part (using the Scene Builder is optional if you want to use it) during the lab time, unless otherwise specified by your instructor.*

- *The backend coding for your design can be submitted as a part of DIY.*

- *Your application must compile and run upon download to receive any mark.*

- *To submit the workshop, please follow the Submission Guideline provided at the end of this document.*

- *You must submit your workshop by the due date. Late submissions policy is specified in the Academic Procedures for Evaluations document available through the class plan on Blackboard.*

Task:

You are working for a small manufacturing organization that has outgrown its current inventory system. They have been using a spreadsheet program to manually enter inventory additions, deletions, and other data from a paper-based system but would now like you to develop a more sophisticated inventory program.

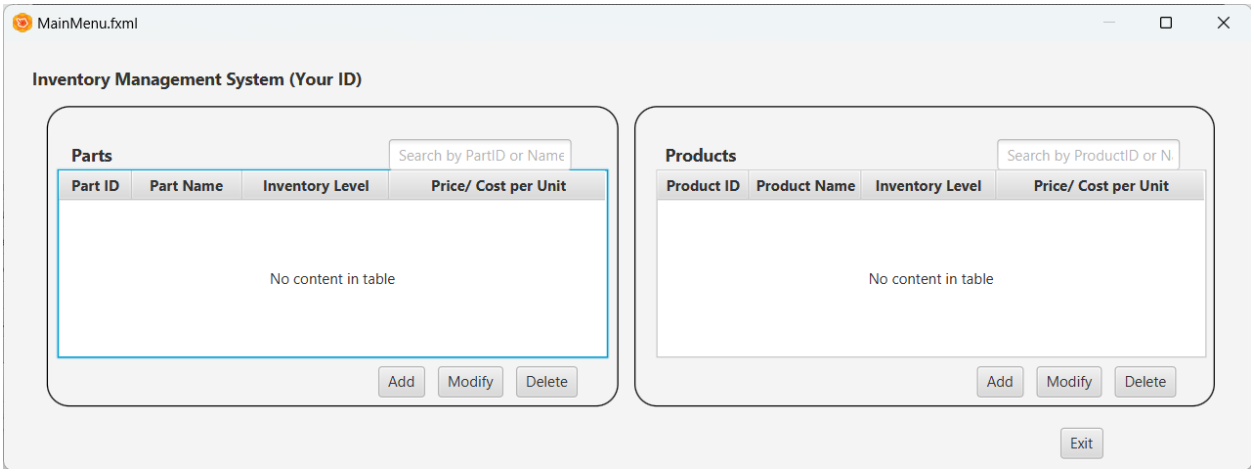
The organization also has specific business requirements that must be included as part of the application. A system analyst from your company created the solution statements outlined in the requirements section based on the manufacturing organization's business requirements. You will use these solution statements to develop your application.

Create a JavaFX application with a graphical user interface (GUI). Write code to display each of the following screens in the GUI:

Front-End

Main Screen

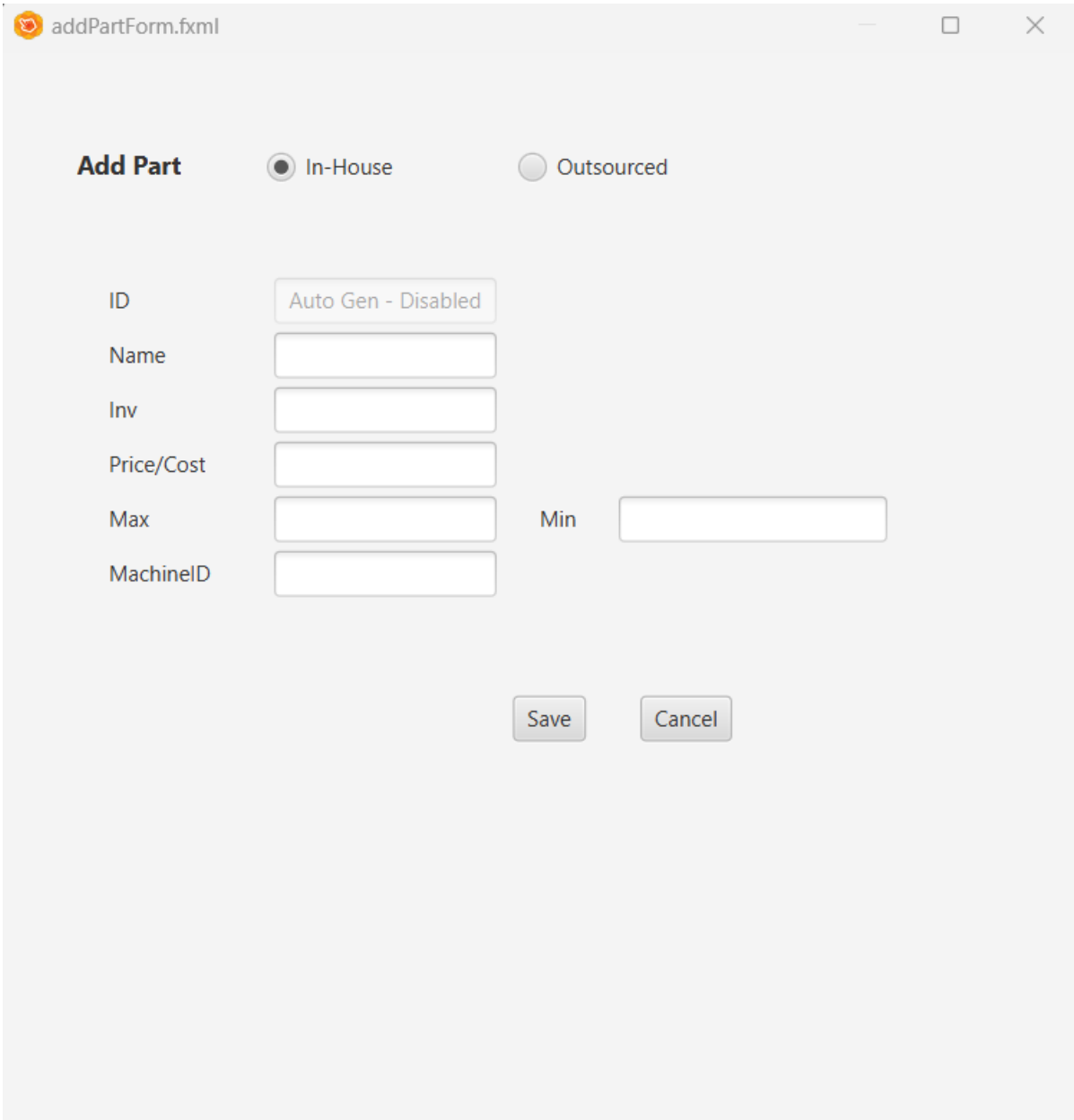
A. A main screen, showing the following controls:
   i. Buttons for "Add", "Modify", "Delete", "Search" for parts and products, and "Exit"
   ii. Lists for parts and products
   iii. Text boxes for searching for parts and products
   iv. Title labels for parts, products, and the application title



Parts Screen

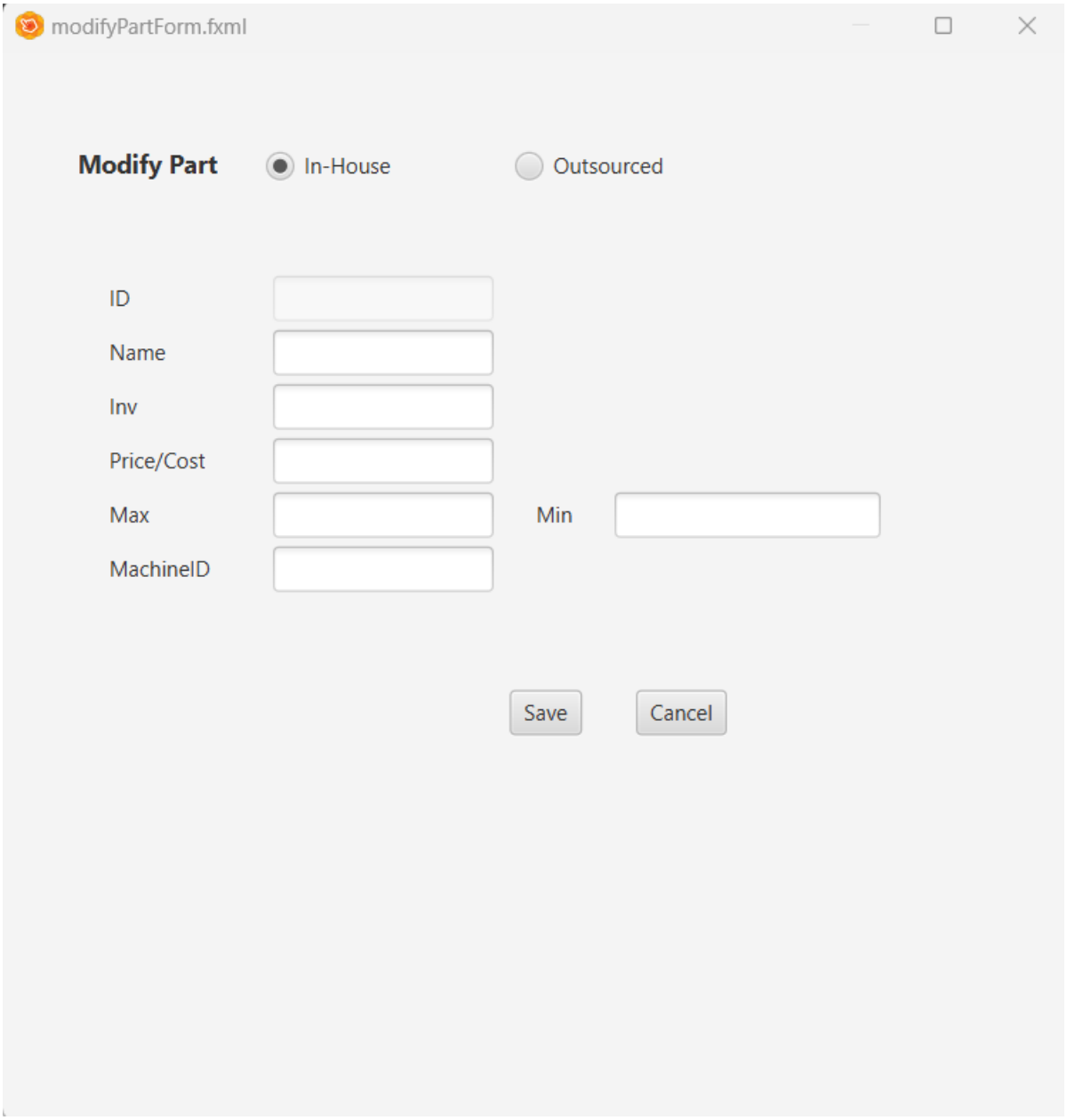
B. An add part screen, showing the following controls:
   i. Radio buttons for "In-House" and "Outsourced" parts
   ii. Buttons for "Save" and "Cancel"

iii. Text fields for ID, name, inventory level, price, max and min values, and company name or machine ID
iv. Labels for ID, name, inventory level, price/cost, max and min values, the application title, and company name or machine ID



C. A modify part screen, with fields that populate with pre-saved data, showing the following controls:
   i. Radio buttons for “In-House” and “Outsourced” parts
   ii. Buttons for “Save” and “Cancel”
   iii. Text fields for ID, name, inventory level, price, max and min values, and company name or machine ID

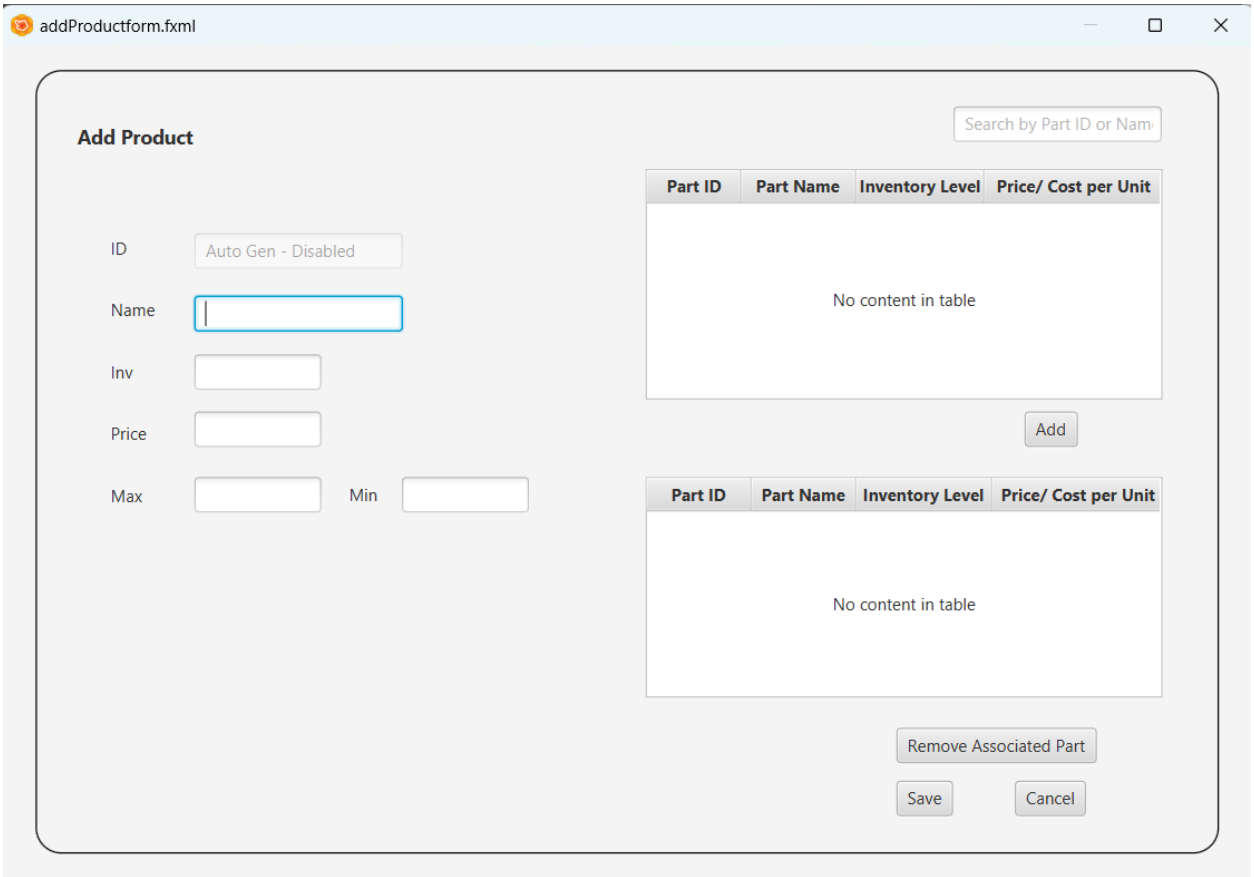
iv. Labels for ID, name, inventory level, price, max and min values, the application title, and company name or machine ID



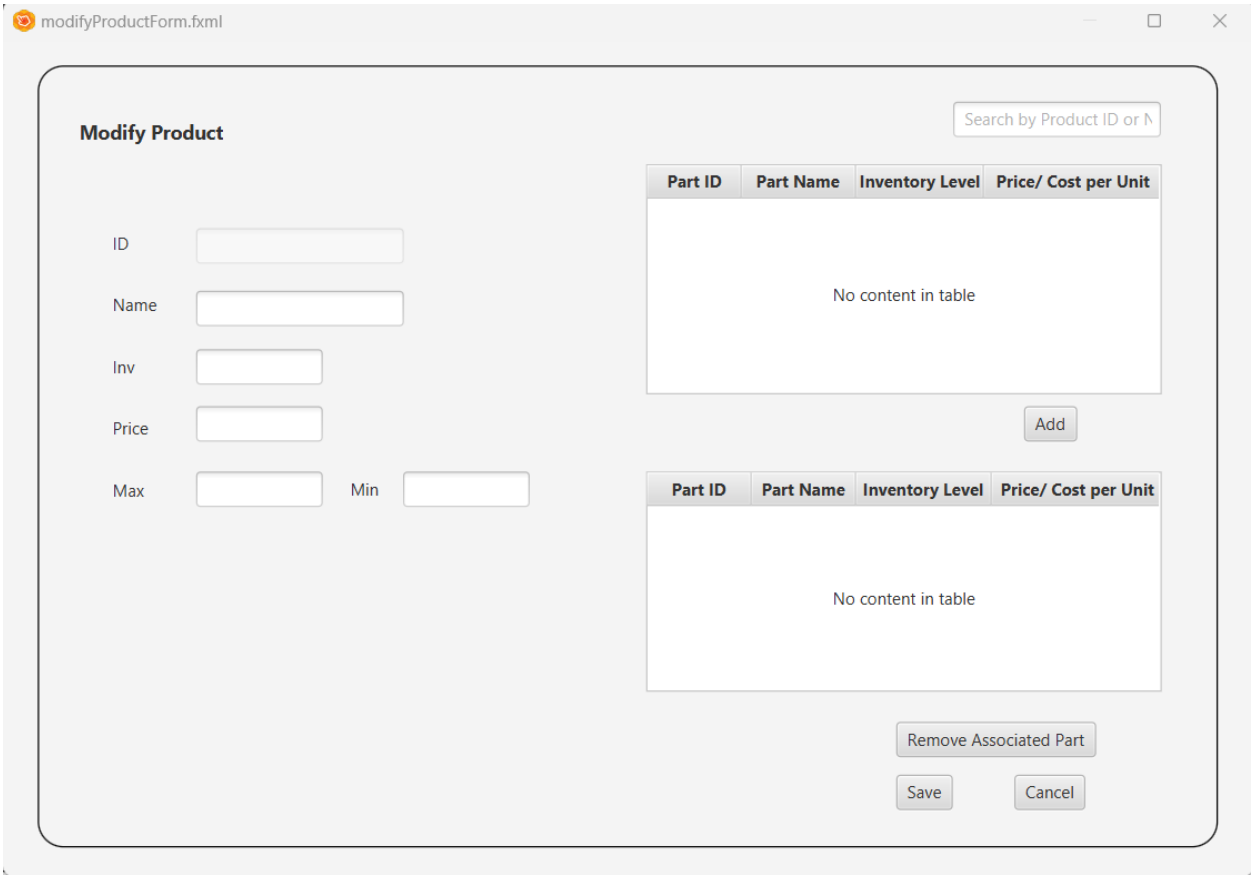
Product Screen

D. An add product screen, showing the following controls:
   i. Buttons for “Save”, “Cancel”, “Add” part, and “Delete” part
   ii. Text fields for ID, name, inventory level, price, and max and min values

iii. Labels for ID, name, inventory level, price, max and min values, and the application
iv. A list for associated parts and their products
v. A “Search” button and a text field with an associated list for displaying the results of the search



E. A modify product screen, with fields that populate with pre-saved data, showing the following controls:
i. Buttons for “Save”, “Cancel”, “Add” part, and “Delete” part
ii. Text fields for ID, name, inventory level, price, and max and min values
iii. Labels for ID, name, inventory level, price, max and min values, and the application
iv. A list for associated parts and their products
v. A “Search” button and a text field with associated list for displaying the results of the search



Backend

Now that you've created the GUI, write code to create the class structure provided in the attached "UML Class Diagram". Enable each of the following capabilities in the application:

F. Using the attached "UML Class Diagram", create appropriate classes and instance variables with the following criteria:

- Five classes with the associated instance variables.
- Variables are only accessible through getter methods.
- Variables are only modifiable through setter methods.

Note: The UML Class Diagram may be altered so long as the aspects of the current UML diagram are intact and the changes applied do not provide a work around for key aspects, such as

- Inheritance.
- Abstraction.
- Encapsulation of the data.

G. Add the following functionalities to the main screen, using the methods provided in the attached "UML Class Diagram":
    i. Redirect the user to the "Add Part", "Modify Part", "Add Product", or "Modify Product" screens
    ii. Delete a selected part or product from the list
    iii. Search for a part or product and display matching results
    iv. Exit the main screen

H. Add the following functionalities to the part screens, using the methods provided in the attached "UML Class Diagram":

    1. "Add Part" screen
        a. Select "In-House" or "Outsourced"
        b. Enter name, inventory level, price, max and min values, and company name or machine ID
        c. Save the data and then redirect to the main screen
        d. Cancel or exit out of this screen and go back to the main screen
    2. "Modify Part" screen
        a. Select "In-House" or "Outsourced"
        b. Modify or change data values
        c. Save modifications to the data and then redirect to the main screen
        d. Cancel or exit out of this screen and go back to the main screen

I. Add the following functionalities to the product screens, using the methods provided in the attached "UML Class Diagram":

    1. "Add Product" screen
        a. Enter name, inventory level, price, max and min values, and company name or machine ID
        b. Save the data and then redirect to the main screen
        c. Associate one or more parts with a product
        d. Remove or disassociate a part from a product
        e. Cancel or exit out of this screen and go back to the main screen
    2. "Modify Product" screen
        a. Modify or change data values
        b. Save modifications to the data and then redirect to the main screen
        c. Associate one or more parts with a product
        d. Remove or disassociate a part from a product
        e. Cancel or exit out of this screen and go back to the main screen

J. Write code to implement exception controls with custom error messages for each of the following sets:

Set 1

- Entering an inventory value greater than the maximum value for a part or product, or lower than the minimum value for a part or product
- Preventing the minimum field from having a value above the maximum field
- Preventing the maximum field from having a value below the minimum field
- Ensuring that a product must always have at least one part

Set 2

- Preventing the user from deleting a product that has a part assigned to it
- Including a confirm dialogue for all “Delete” and “Cancel” buttons
- Ensuring that the price of a product cannot be less than the cost of the parts
- Ensuring that a product must have a name, price, and inventory level (default 0)

Note: UML diagram is attached as a PDF.

Workshop Header

/***
Workshop #
Course:*<subject type> - Semester*
Last Name:*<student last name>*
First Name:*<student first name>*
ID:*<student ID>*
Section:*<section name>*
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:*<submission date>*
***/

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Do Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- Design should be discussed/ created during the lab. 10%.
- Complete the code behind as the part of your DIY. 60%
- Submit a **reflect.txt** file with the submission. 10%
  Questions to be answered for the reflection:
  - If you would've to use the Properties types instead of primitive data types? E.g. SimpleStringProperty instead of String, would it benefit the coding in any way? Would it be useful to use the properties instead of primitive datatypes?
- Video submission explaining core code pointers and showing the full working application (3 – 8 minutes max). 20%
- All submission goes to Black Board.
- Your submission should include
  - Video file with audio
  - Reflect.txt file
  - Complete zipped project.
- Late submissions would result in additional 10% penalties for each day or part of it.

- Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the workshops, but the final solution may not be copied from any source.