**COMP167 Major Programming Assignment2**

Fall 2021

# Introduction

This assignment requires you to create a Java application that will be used to inventory mowers in a mower store. In addition to the classes that store the mower data, you will create a GUI to manage the inventory.

# UML Domain/Structure Diagram

*Mower*

Engine

LawnTractor

CommercialMower

GasPoweredMower

*WalkBehindMower*

PushReelMower

MowerWareHouse

# Classes

You are required to implement the following classes at a minimum. You may add other classes (and methods) if you need them.

## Engine Class

|  |  |
| --- | --- |
| Engine | |
| -manufacturer : String  -horsePower : double  -cylinders : integer | Engine manufacturer  Horse power of engine  Number of cylinders |
| +Engine()  +Engine( //all Engine properties )  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## *Mower Class* (Abstract)

|  |  |
| --- | --- |
| *Mower* | |
| -manufacturer : String  -year : integer  -serialNumber : String | Mower manufacturer  Year of manufacture  Serial number of mower |
| +Mower()  +Mower( //All properties )  +//Getters and setters  +toString() | Put each property on a separate line. |

## LawnTractor Class

This class will define each grade level that can be obtained in the course (D to A).

|  |  |
| --- | --- |
| LawnTractor | |
| -engine : Engine  -model : String  -deckWidth : double | Mower engine  Model of Lawn Tractor  Width of mower deck |
| +LawnTractor()  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## CommercialMower Class

This class will track the points received and the maximum points for each class assignment.

|  |  |
| --- | --- |
| CommercialMower | |
| -operatingHours : double  -zeroTurnRadius : boolean | Points earned on the assignment  Maximum points (e.g. Most labs have maxPoints equal to 20) |
| +CommercialMower()  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## *WalkBehindMower* Class (Abstract)

|  |  |
| --- | --- |
| *WalkBehindMower* | |
| -cutWidth : double  -wheelDiameter : double | Blade width of mower  Diameter of the mower wheels |
| +WalkBehindMower()  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## GasPoweredMower

|  |  |
| --- | --- |
| GasPoweredMower | |
| -engine : Engine  -selfPropelled : boolean | Mower engine  Is the mower self-propelled |
| GasPoweredMower()  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## PushReelMower

|  |  |
| --- | --- |
| PushReelMower | |
| -numWheels : integer | Number of wheels on the mower |
| +PushReelMower()  +//Accessor and Mutators  +toString():String | Put each property on a separate line. |

## MowerWareHouse

|  |  |
| --- | --- |
| MowerWareHouse | |
| -storeName : String  -mowers : ArrayList<Mower> | Name of the mower store |
| +MowerWareHouse()  +//Accessor and Mutators  +readMowerData( inputFileName : String) : void  +saveMowerData( outputFileName :String):void  +toString():String | Put each property on a separate line. |

# Handling ArrayLists

Each ArrayList should have five associated methods to perform: getNum, add, remove, get and set. So if you have an ArrayList named *widgets* that stored items of type Widget, then the associated UML behaviors would be:

+getNumWidget**s**() : int //Return the number of items in the ArrayList widgets.

+getWidget(index:int) : Widget //get the Widget at location index in ArrayList widgets

+setWidget(index:int, item:Widget):void //store item at location index in the ArrayList widgets.

+addWidget(item:Widget):void //Append the Widget to the ArrayList.

+removeWidget( index:int ) : Widget //remove and return the Widget at location index

# Input File

The name of the input file will be supplied using command-line arguments. If no command-line argument is supplied, then your program should prompt the user for the input file using the JFileChooser class. Here is the format of the input file:

Store name

Mower Class Properties

Mower Subclass Type (L, C, G, P)

Subclass Properties

Note: Each properties will be on a separate line in the same order listed in the UML diagrams.

# Output File

The format for the output file should be identical to that of the input file. In other words, after writing your output file, you should be able to read it back in as an input file. The toString() methods of your classes are designed to make file output simple.

Figure 1: Input File Format

# Graphical User Interface

**Viewing:** The View window should display the name of each type of concrete mower (LawnTractor, CommericalMower, GasPoweredMower and PushReelMower). Also include a JButton next to each of these mower types that will initiate the population of a JList showing all the properties of each mower of that type. The JList can be included on the View window or as a separate JDialog.

# Grading

If your project does not compile, it receives a grade of zero. If you do not document your program according to the documentation guidelines, the graders have been instructed to deduct **up to 25%.**

**Level 1 (40%):** Implement all the classes except the MowerWareHouse class.

**Level 2 (60%):** Implement the MowerWareHouse class except the readMowerData() and saveMowerData() methods. Use the main method to create one of each type of concrete class and add them to the ArrayList. Output the contents of the MowerWareHouse object to a JOptionPane using the toString() method of the MowerWareHouse class.

**Level 3 (85%):** Modify your main so that it uses command-line arguments to provide the input file name. Add the logic to obtain the input file name from a JFilechooser if no command-line argument is provided. Implement the readMowerData and saveMowerData methods. To prove your code works, read the input file and add additional mowers by creating objects in the main method and adding them to the MowerWareHouse object. Save the updated MowerWareHouse object using the saveMowerData() method.

**Level 4 (100%):** Implement the GUI for viewing data