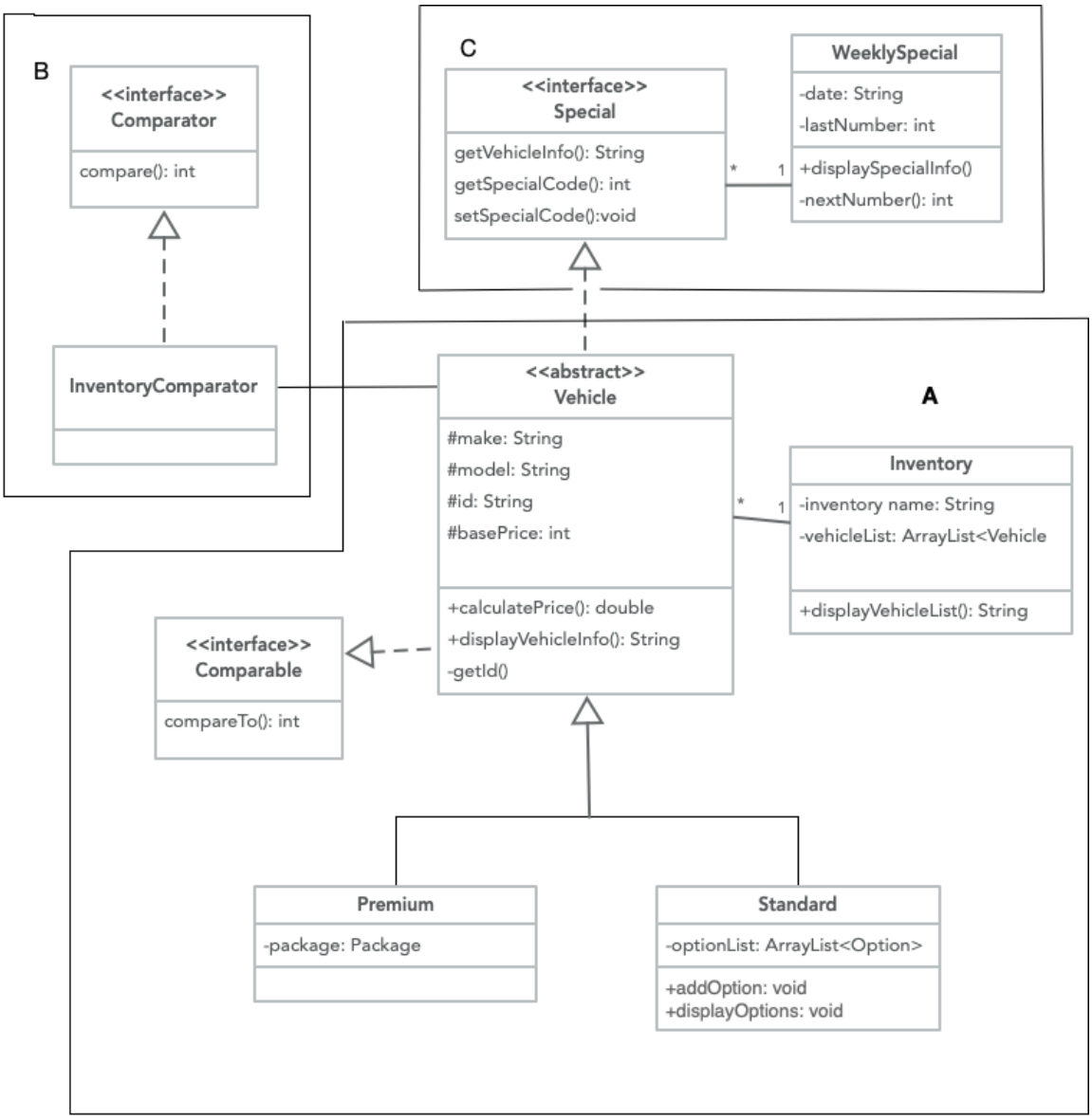
ACS-1904 W2024

Assignment #3

Due by Sunday, March 31, 2024, at 11:59 pm

- Submit your `.java` files (together in a .zip file called `A3.zip`) only the .java files please, via Nexus
- Include your name and student number in each file as a comment.
- As per the Course Outline, you may submit your work up to one day late with a penalty of 20% applied.



*Note that this assignment requires reading/writing data to/from **XML files**. Encoding/decoding XML has requirements that need to be fulfilled:*

- *All classes in the object graph must include a no-arg constructor and all getters and setters*

PART A:

Create the Hierarchical Class structure shown in Section A of the partial UML diagram.
Note that:

Fields and/or methods may/will be required in addition to those shown on the partial UML diagram above.

1) *Vehicle* is an abstract class with toString overridden to display the make and model of the vehicle in the format;

   <*make*>, <*model*> e.g. Ford, Mustang

   a) The `calculatePrice()` method is abstract. It will be implemented in the two sub-classes that extend `Vehicle`.

   b) The vehicle's id field is a catenation of the first two letters of the make, and model followed by an automatically generated 4-digit number. The number part of the vehicle's id should start at 1045. For example, `FoMu-1234`. The id should be composed and assigned in both the no-arg and full-arg constructors.

   c) The `displayVehicleInfo()` method will return the vehicle's information in the following format:

   <make> <model>: <id> i.e, `Ford Mustang: FoMu-1234`

   Use StringBuilder() to compose the String.

   d) The `basePrice` field contains the cost per day to rent the vehicle.

   e) *Vehicle* implements the `Comparable` interface (from the Java Class Library), *Vehicle* objects have a natural ordering by id (see the output in A-3 below)

2) Your project will require two Enum classes, not shown on the partial UML diagram.

   a) PremiumPackage: defines the different optional packages available to Premium rentals.

      i) Four package types:

Premium Package	**Display**	**Price per Day**
NONE	None	0
SPORT	Sport	15
LUXURY	Luxury	12
BUSINESS	Business	14

      ii) Note that these packages are not available to standard vehicle rentals.

      iii) Only one package may be purchased.

b) Option: defines the optional extras that can be added to a standard vehicle rental.

i) Five options

Option	**Display**	**Price per Day**
NONE	None	0
SATNAV	SatNav	5
BLUETOOTH	Bluetooth	7
KEYLESS	Keyless Entry	4
ROADSIDE	Roadside Assistance	7

ii) These options are not available to `Premium` vehicle rentals.

iii) More than one option can be added to a `Standard` vehicle rental (i.e up to 4 options can be added)

iv) Obviously, the NONE option cannot be combined with any other options.

3) The `Premium` and `Standard` classes

a) both implement the `calculatePrice()` abstract method from Vehicle. The price is simply the base price plus the package price or the base price plus the sum of the optional extras.

b) Note that the `Standard` object options will be added using the `addOption()` method after the object has been instantiated.

c) Include the fields and methods indicated on the partial UML diagram and any other that may be required.

4) Implement the associations in the following manner:

a) When a `Vehicle` is added to the Inventory it, the Inventory, is automatically added to the appropriate field in the `Vehicle`. This should happen in the `addVehicle()` method in the `Inventory` class.

b) The Inventory list defaults to alphabetical ordering by `id` and updates with every new `Vehicle` added. Use `Collections.sort` to achieve this.

Note: The `this` keyword can be passed as an argument in the method call: use this to set the second part of these associations

2) Download the file `A3ADriver.java` from this GitHub repository, (you can download the file or the whole project) and add it to your BlueJ project. The driver code instantiates several `Vehicle` objects and adds them to an ArrayList of type `Vehicle` called `vehicles`:

   `A3ADriver` also instantiates and populates an `Inventory` object called `inventory` and applies the associations. Running this code should display the following output:

```
St Mary's Hot Rod Rentals Vehicle Inventory:

Audi R8: AuR8-1058 $35
Chevrolet Camaro: ChCa-1054 $37
Chevrolet Camaro: ChCa-1055 $34
Chevrolet Camaro: ChCa-1056 $37
Chevrolet Corvette: ChCo-1057 $34
Ford Cortina: FoCo-1048 $33
Ford Mondeo: FoMo-1049 $32
Ford Mustang: FoMu-1045 $25
Ford Mustang: FoMu-1046 $40
Ford Mustang: FoMu-1047 $40
Honda Civic: HoCi-1066 $28
Honda Civic: HoCi-1067 $33
Lotus Elan: LoEl-1051 $38
Lotus Europa: LoEu-1050 $37
Lotus Seven: LoSe-1052 $22
Maserati Quatra: MaQu-1053 $38
Nissan Micra: NiMi-1061 $37
Nissan Micra: NiMi-1065 $19
Renault Megane: ReMe-1059 $29
Renault R5: ReR5-1060 $24
Toyota Yaris: ToYa-1062 $15
Toyota Yaris: ToYa-1063 $16
Toyota Yaris: ToYa-1064 $26

-----------------------------------------
Print the inventory name for one Vehicle object.
Audi R8: AuR8-1058
St Mary's Hot Rod Rentals
```

   Complete the driver code:

   a) Write the `vehicles` ArrayList to an XML file called `As3-vehicles.xml`. Include a message confirming that the ArrayList has been written to the file. Note that you are writing the whole ArrayList, not each Vehicle object individually.
   b) Write the Inventory object to an XML file called `As3-inventory.xml`. Include a message confirming that the Inventory object has been written to the file.

PART B

1. Create the `InventoryComparator` class that implements the `Comparator` interface (from the Java Class Library). The alternate ordering of vehicles in the inventory list is in descending order by price. Note that this is the total price, not just the base price. There is no secondary ordering.
2. In `A3BDriver.java (you create A3BDriver.java)`, read the data from `As3-vehicles.xml`. Create a new ArrayList for each of the *`Premium`* ***and*** *`Standard`* objects named *`premiums`* ***and*** *`standards`*. Iterate through *`vehicles`* and add objects of type *`Premium`* to the *`premiums`* list and objects of type *`Standard`* to the *`standards`* list.
3. Sort both lists using the `InventoryComparator` to be ordered by price and display the output in the following manner using the *`displayVehicleInfo()`* and *`calculatePrice()`* methods:

```
Premium Vehicles by Price:

Ford Mustang: FoMu-1047 $40
Ford Mustang: FoMu-1046 $40
Maserati Quatra: MaQu-1053 $38
Lotus Elan: LoEl-1051 $38
Lotus Europa: LoEu-1050 $37
Chevrolet Camaro: ChCa-1056 $37
Chevrolet Camaro: ChCa-1054 $37
Audi R8: AuR8-1058 $35
Chevrolet Covette: ChCo-1057 $34
Chevrolet Camaro: ChCa-1055 $34
Ford Cortina: FoCo-1048 $33
Ford Mondeo: FoMo-1049 $32
Ford Mustang: FoMu-1045 $25
Lotus Seven: LoSe-1052 $22

Standard Vehicles by Price:

Nissan Micra: NiMi-1061 $37
Honda Civic: HoCi-1067 $33
Renault Megane: ReMe-1059 $29
Honda Civic: HoCi-1066 $28
Toyota Yaris: ToYa-1064 $26
Renault R5: ReR5-1060 $24
Nissan Micra: NiMi-1065 $19
Toyota Yaris: ToYa-1063 $16
Toyota Yaris: ToYa-1062 $15

end of program
```

Write both the `premiums` and `standards` array lists to a file named `As3B-vehicles.xml`. Include a message confirming that the ArrayList has been written to the file.

```
*** Registration info has been written to As3B-vehicles.xml ***
```

PART C:

Continue to develop the program to include a listing of the weekly specials, a list of vehicles that are to be rented at a reduced rate.

1. Create the `WeeklySpecial` class and the `Special` interface and make any changes to the `Vehicle` class so that objects of type `Vehicle` can be added to the Weekly Specials list.
    a. Note that the `getVehicleInfo()` method in the `Special` interface is different from the `displayVehicleInfo()` method in the abstract class `Vehicle`. See the sample output below.
    b. Add necessary features suggested for `WeeklySpecial` such as a date (this is a `String` in the form `March 23`, a method to add a vehicle to the specials list, and a method to display the date of the special and the list of vehicles on special.
    c. Include a static variable `lastNumber` that is used to assign a unique number to each of the vehicles as they are added to the list of specials: when a vehicle is added, assign the next number of the `WeeklySpecial` (in sequence starting at 1045).
2. Implement a Driver (`A3CDriver.java`) class with a main method that;
    a. Instantiates a `WeeklySpecial` object.
    b. Reads the contents of `As3B-vehicles.xml`.
    c. From the vehicles read from the file, add all of the Fords, Lotuses and Nissans to the list of vehicles on special this week.
    d. Displays information about this week's specials using the `displaySpecialInfo()` method. See the sample output below.
    e. Write the `WeeklySpecial` object to an XML file called `As3C-specialInfo.xml`. Include a message confirming that the object has been written to the file.

Sample output:

```
Date: March 20
Special ID     Vehicle
1045           Ford Mustang
1046           Ford Mustang
1047           Lotus Elan
1048           Lotus Europa
1049           Ford Cortina
1050           Ford Mondeo
1051           Ford Mustang
1052           Lotus Seven
1053           Nissan Micra
1054           Nissan Micra

Weekly special info written to file.

end of program
```

Notes:

- Take care to format your code appropriately.
- Assign descriptive names to methods, variables, fields, and constants.
- Use camelCase for variable and method names, ALLCAPS for the names of constants, and CapitalizeEachWord for class names.
- Include comments where appropriate to make your code more readable and self-documenting.
- Use appropriate line spacing and indenting, again, to enhance the readability of your code.
- Be sure to conform to the programming standards and restrictions outlined in class. For example, don't use break except in switch statements, don't use while(true), don't use exit(0), and so on.

Submit your `A3.zip` file that includes all assignment files (`Vehicle.java, Premium.java, Standard.java, Inventory.java, InventoryComparator.java, WeeklySpecial.java, Special.java, A3ADriver.java, A3BDriver.java, A3CDriver.java, PremiumPackage.java, Option.java`) via Nexus

Note that you are not required to submit any data files.