Documentation for Project1 unit2: Car Configuration

Part A

It's the class diagram about our project 1 unit2.

Here are additional files we add base on the unit1:

BuildAuto.java: It's a empty class, just extends the ProxyAutomobile class and implements some interface function. But it has not over write any function from interface.

CreateAuto.java: It's an interface, has function: buildAuto and printAuto.

FixAuto.java: it an interface, also has function: fix.

ProxyAutomobile.java: It a abstract class, implements all function which from UpdateAuto, CreateAuto and FixAuto.

UpdateAuto.java: It's an interface, has function: updateoptionsetname and undateoptionprice.

AutoException.java: it's a customer exception class.

ExceptionEnumerator.java: it's support a enum type to identify the number of exception.

Fix1to100.java: it's just supports a set of function to fix the error which throws by exception.

Test Case

In this part, we wrote the API, created the interface: CreateAuto, UpdateAuto and FixAuto. Also, we implement the customer exception class and support the fix function to fix specific exception. Here are the some test cases I did.

From the handout of the project1 unit2, we have such test need to do.
1. Are you able to create and print an Auto instance through CreateAuto interface?
2. Are you able to update one of OptionSet's name or Option Price for the Auto instance, created in the previous step.

According to above question, we do the test.

Test case 1: we use BuildAuto to create an instance. And then by using CreateAuto interface to create and pint the Auto instance.
The test result detail just in the file: **testCreat_output.txt**

Test case 2: Update the option set name and option price through instance, which created in the previous step.
The test result detail just in the file: **testUpdate_output.txt**

And then, we wrote some class to handle exceptions, and we also provided the fix function to fix the specific error. Here are the test results we got.

Test case 3: Handle the exception about cannot found the file.
The test result detail just in the file: **test_fileNotFound.txt**

Test case 4: Handle the exception about when we load the txt file to create an Auto, the file just miss the name of option set.
The test result detail just in the file: **test_missoptionsetname.txt**

Test case 5: Handle the exception about cannot we load the txt file to create an Auto, the file just miss the name of option.
The test result detail just in the file: **test_missoptionname.txt**

Test case 6: Handle the exception about cannot we load the txt file to create an Auto, the file just miss the price of option.
The test result detail just in the file: **test_missoptionprice.txt**

Test case 7: Handle the exception about cannot we load the txt file to create an Auto, the file just miss the basic price of the car
The test result detail just in the file: **test_nobaseprice.txt**

Part B

In part b, I modified some code. The detail just shows below.

1. Add two String into the Automobile.class: make and model. The name just make + model.
2. Modify the option and option set as ArrayList.
3. Add function: setOptionChoice("transmission", "standard");
   getOptionChoice(): Option
   setOptionChoice(optionName: String): void

The part b routine has two important improvements. First, it can create and print multiple car information. Second, the user can set choice through the routine, and the routine would compute the final price and print what choice you select and print the final price.

Test Case

Here are some test cases for the part b.

Test case 8: Use two configuration files, FordZTW.txt and GTR.txt file to create and print them together.
The test result detail just in the file: **test_morecar_out.txt**

Test case 9: We add some code, use the method of Automobile: setoptionchoice to set the choice, and the routine would print the choices and compute the final price according to each choice and base price.
The test result detail just in the file: **test_pricecompute_out.txt.**