**Java Software Development**

**Laboratory II**

*The purpose of this laboratory is to review advanced OO design and programming, exception handling and using HashMap to search and data aggregation.*

- Study the sample code to understand the HashMap for search and data aggregation. Use the code from lab I and the same Driver, Car, Address, and PostCode classes. Change the following items:

- Add Class MyDate which has three fields int year, Month month, int day. And Month is defined as an enum

-Add a MyDate LicenceDate to Driver class.

- Implement comparable for MyDate. Hint: date1 is smaller than date2 is date1.year<date2.year. If years are equal then compare months and if months are equal then compare days.

-Comparable for Driver should be done based on Date.compareTo()

-Create a HashMap<Integer, Driver> of drivers in your main (Driver ID as key) and add three drivers to the HashMap (test code). (Note, that you have had ArrayList for week 4 and you only need to change it to HashMap).

-Add a static method HashMap<String, Integer> getTotalDriverCountPerCity (HashMap <Integer, Driver>) which aggregates the total number of drivers for any given city in the system as <City name, Number of drivers for that given city> e.g., we have three drivers from Wollongong, two drivers from Sydney, etc

-Add a static method HashMap<String, Double> getTotalCarPricePerCity (HashMap <Integer, Driver>) which aggregates the total car price for any given city (of drivers) in the system as <City name, Total car price for that given city> e.g., if we have three drivers from Wollongong, the total for Wollongong is the total price of all the cars these drivers own

-Add a method HashMap<String, Double> getTotalCarPricePerCarModel() to Driver which aggregates the total car price for any given car model the driver owns as <Car model, Total car price for that given model>. Hint: If we have three “Toyota Corolla” owning by this driver, the total price for them should be calculated.

-Add a static method HashMap<String, Double> getTotalCarPricePerCarModel (HashMap <Integer, Driver>) which aggregates the total car price for any given car model in the system as <Car model, Total car price for that given model>. Hint: If we have three “Toyota Corolla” owning by all of our drivers in the system, the total price of them should be calculated. Call the getTotalCarPricePerCarModel() method inside the Driver and aggregate all of the HashMaps together

-Add try catch to any section of the code that the program uses Scanner to read from the console and catch the **InputMismatchException** to catch any wrong input (e.g., String entry for a number) and prompt user. Your program should not stop/crash but to continue normally after the catch

-Let’s assume that Drievr ID should be 6 digits starting with 3:

* Add a new Exception Class DrievrException with one field : int ID.
* In the Driver constructor if the given ID does not follow the required pattern:
  + - Generate a random ID with the given pattern 3xxxxxx
    - Throw a DriverException by passing the generated ID as an argument to DriverException constructor.
    - The toString() method in DriverException creates an error message “ The Driver ID was not valid and a new ID (the generated ID) is generated by admin and assigned to the Driver”.
* In your UI catch this exception and print it. Again your program should continue normally after catching this exception.

Update your UI from week 4

Option 1 is almost the same as before to get all the information from the user and creates a Driver object and adds it to the original HashMap. Note: if the Driver ID is already taken, the new User should not be added and a prompt should be displayed.

Option 2 almost the same as before to make a shallow copy and deep copy by using copy constructor and deep copy by using clone() of original HashMap, and store them in three **ArrayLists** and then sorts the shallow copy list by LicenceDate in descending order. You need to iterate over HashMap and add the drivers to the ArrayLists.

Option 3 is almost the same as before to print original HashMap, and shallow copy and deep copy lists on the console.

Option 4 is almost the same as before to ask the user to supply a Driver ID and a city name and then finds the given driver in the HashMap and copies the driver record to a new record (using clone()). Then changes the city name of the driver and prints both objects. If driver ID is not found a proper prompt should be displayed.

Option 5 is almost the same as before to ask the user to supply a Driver ID as well as the information for a new car and then finds the given driver in the HashMap and copies the driver record to a new record (using copy constructor). Then adds the car to the driver record and prints both objects. If driver ID is not found a proper prompt should be displayed.

Option 6 to report a list of all the cities and the total number of drivers per city

Option 7 to report a list of all the cities and the total car price per city

Option 8 to report a list of all the car models and the total car price per car model