University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Lab Practical One - Introduction to OOP

Objective:
The objectives of this tutorial are to allow students to be able to:
– understand how an object-oriented model in UML is implemented in C++ and Java
– identify the components of a complete object-oriented program in C++ and Java
– enter, edit, compile, run and debug a complete program in C++ and Java

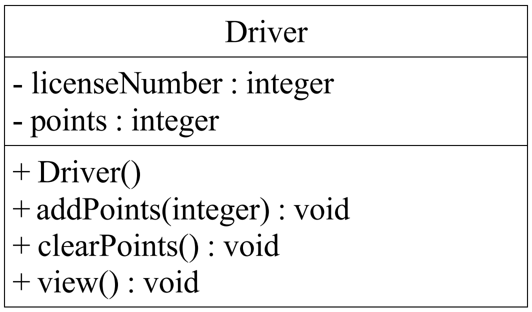
Before you begin these exercises, your lab tutor will introduce you to the C++ and Java integrated development environment (IDE) installed on your lab computer. You can install these IDEs on your laptop too so you can complete the homework outside the lab.

Exercise One

Requirements

The Jamaica Constabulary Force (the Jamaican Police) needs a system to track points accumulated for each driver who has committed an offense. The system keeps track of the driver's license number and total points accumulated by a driver. When a record is created for a new driver, the driver's total points accumulated is set to zero. Points can be added any time an offense is committed, and after one year of offense-free driving, all points are cleared. The driver's total points accumulated can be viewed at any time.

Design using UML



Implementation

A sample implementation of the above model is shown in both C++ and Java. Enter both programs in their respective IDEs or other editors in your lab or on your laptop, save them, compile them, then run them. Your lab tutor will walk you through each program and show you how to debug them.

C++ Implementation

```
//Sample program to demonstrate classes in C++ class to track points for a driver
//Filename: Driver.h
//by David White 2011 Aug 29
//edited by Tyrone Edwards 2016 Jan 14

#ifndef DRIVER_H
#define DRIVER_H

//header file needed for screen output
#include <iostream>

//class for Driver
class Driver
{
        private:
                int licenseNumber;
                int points;
        public:
               //constructor - use to initialise object
               Driver(int ln, int pnts){
                       licenseNumber = ln;
                       points  = pnts;
               }

               //add the specified points to the drivers total
               void addPoints(int pnts)
               {
                       points = points + pnts;
               }

               //clear the total acummulated points
               void clearPoints()
               {
                       points  = 0;
               }

               //Display the DL number and total acummulated points for the driver
               void view()
               {
                       std::cout << "License  No.: " << licenseNumber << std::endl;
                       std::cout << "Total Points Accumulated: " << points << std::endl;
               }
};
#endif
```

```
//main function  - execution always start here
int main()
{
	//create an object called A of class Driver and initialise it's DL number and points
	Driver A(12345, 0);

	A.view ();  //display what the points are

	A.addPoints(5); //add 5 points to the driver's total
	A.view (); //and display the total

	A.addPoints(3); //add 3 more points and
	A.view (); //display the new total

	A.clearPoints(); //clear the total points
	A.view ();  //then display the total now

	Driver B(67890, 0); //create another object called B and initialize it too

	B.view (); //view this driver's total points

	B.addPoints(1); //add 1 point to the driver's  total
	B.view (); //then display the new total  for B

	return  0;
} //end of main
```

Java Implementation

```
//Sample program to demonstrate classes in Java class to track points for a driver
//Filename: Driver.java
//by David White 2011 Aug 29
//edited by Tyrone Edwards 2016 Jan 14

//Default java package
import java.lang.*;

//class for Driver
public class Driver
{
        private int licenseNumber;
        private  int points;

        //constructor - use to initialise object
        public Driver(int ln, int pnts)
        {
                licenseNumber = ln;
                points  = pnts;
        }

        //add the specified points to the drivers total
        public void addPoints(int pnts)
        {
                points = points + pnts;
        }

        //clear the total accumulated points
        public void clearPoints()
        {
                points  = 0;
        }

        //Display the DL number and total accumulated points for the driver
        public void view ()
        {
                System.out.println("License  No.: " + licenseNumber);
                System.out.println("Total Points Accumulated: " +points);
        }
}
```

```
public class Main
{
        //main function  - execution always start here
        public static void main(String args[])
        {
                //create an object called A of class Driver and initialize it's DL number and points
                Driver A = new Driver(12345, 0);

                A.view();  //display what the points are
                A.addPoints(5); //add 5 points to the driver's total
                A.view(); //and display the total


                A.addPoints(3); //add 3 more points and
                A.view (); //display the new total
                A.clearPoints(); //clear the total points
                A.view();  //then display the total now

                Driver B = new Driver(67890, 0); //create another object called B and initialize it too

                B.view(); //view this driver's total points
                B.addPoints(1); //add 1 point to the driver's total
                B.view(); //then display the new total for B

        } //end of main
}
```

Exercise Two

Modify the above code in C++ and Java to add a new method called subtractPoints() which deducts the specified points (passed as a parameter) from the driver's accumulated total. Use your new method in main() to subtract points for a driver. Display the total after the points are deducted.

Exercise Three - Homework

X
- Y : integer
+ X() + incrementY() : void + showY() : void

Implement the above model for class X in both C++ and Java. The constructor method X() takes no arguments and simply sets Y to 0, incrementY() adds the value 1 to Y, while showY() displays the current value of Y on the screen. Write a complete program in both languages to demonstrate how you would use the X class to create and use objects and all the methods in the class. Compile, run and debug your programs.