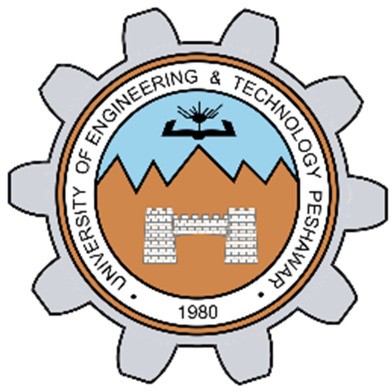
**Lab Title**

## LAB # 01



**Fall 2020**

**CSE208L Object Oriented Programming Lab**

Submitted by: **Fawad Ali** Registration No. : **19PWCSE1845** Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature:

Submitted to:

## Engr. Sumayyea Salahuddin

November 26, 2020

Department of Computer Systems Engineering University of Engineering and Technology, Peshawar

# Objectives of the Lab:

Objectives of the lab are to:

* Clearly understand the purpose and advantages of OOP
* Understand the concept of a Class and Objects
* Develop a basic class containing Data Members and Member Functions
* Use access specifiers to access Class Members
* Make Simple and Overloaded Constructor
* Use the Class Objects and Member Functions to provide and extract data from Object
* Practice with Classes and Objects

**Activity # 01**

**Title:**

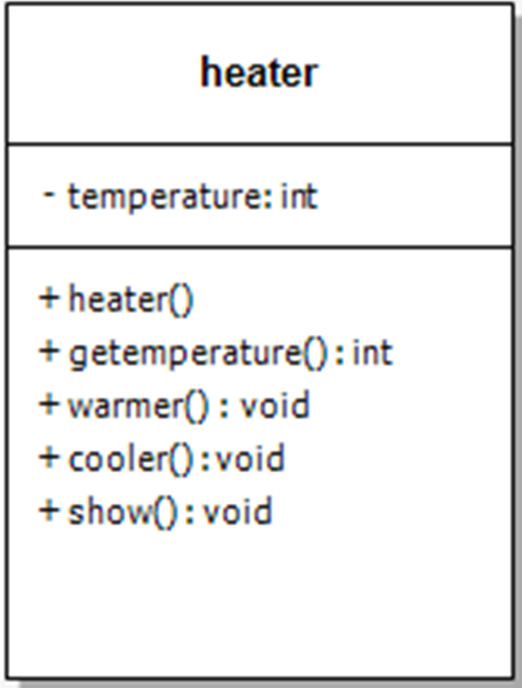
Make a class for heater and model it using temperature.

# Problem analysis:

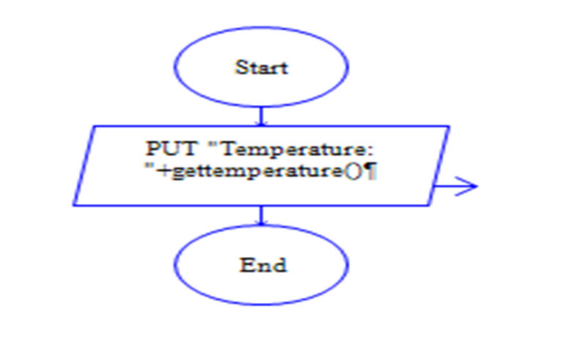
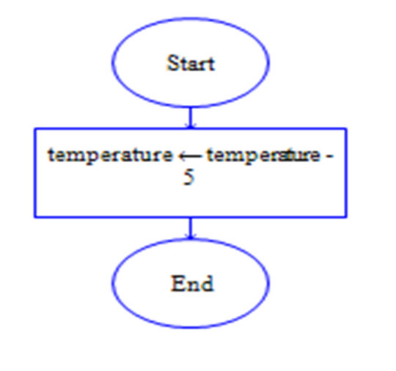
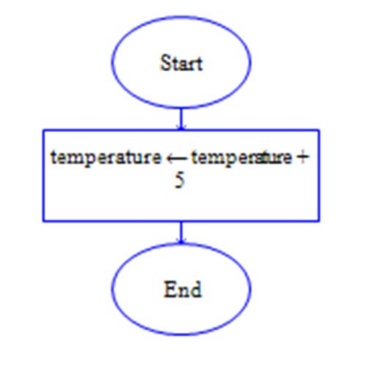
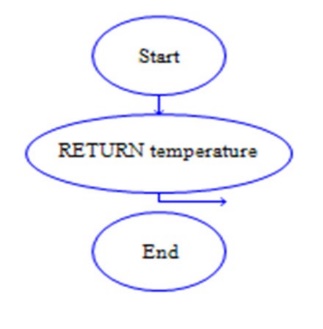
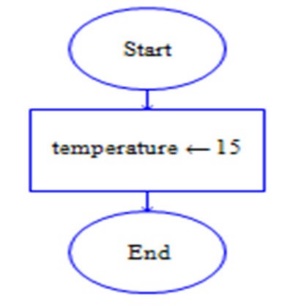
Create a class, **Heater** that contains a single integer field, **temperature**. Define a constructor that takes no parameters. The **temperature** field should be set to the value 15 in the constructor. Define the mutators **warmer** and **cooler**, whose effect is to increase or decrease the value of the temperature by 5 respectively. Define an accessor method to return the value of **temperature**. Demonstrate the use of Heater class.

# Algorithm:

UML diagram for the above problem is given below:

* First make class heater
* Declare temperature as private integer field
* Define no argument constructor to set value of temperature to 15
* Define gettemperature method to return value of temperature
* Define warmer and cooler method to increase and decrease temperature by 5 respectively
* Define show function to display the output
* In main function, make objects of heater to demonstrate the use of heater
* Call each function one after the other and display the show function as shown in the flow chart.

# Flowchart:



Start

Private: int temperature

heater()

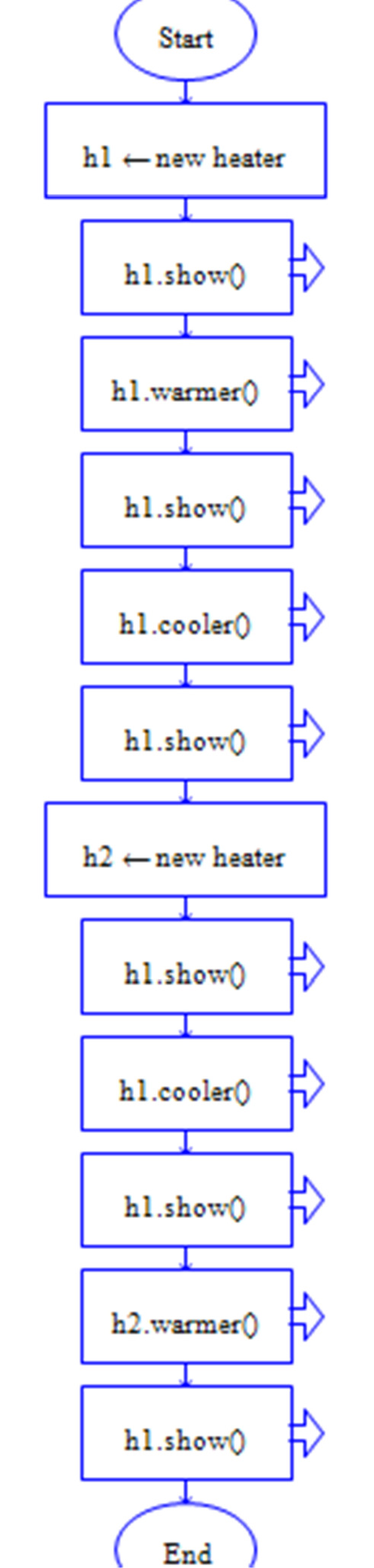
Int gettemperature()

void warmer()

Void cooler()

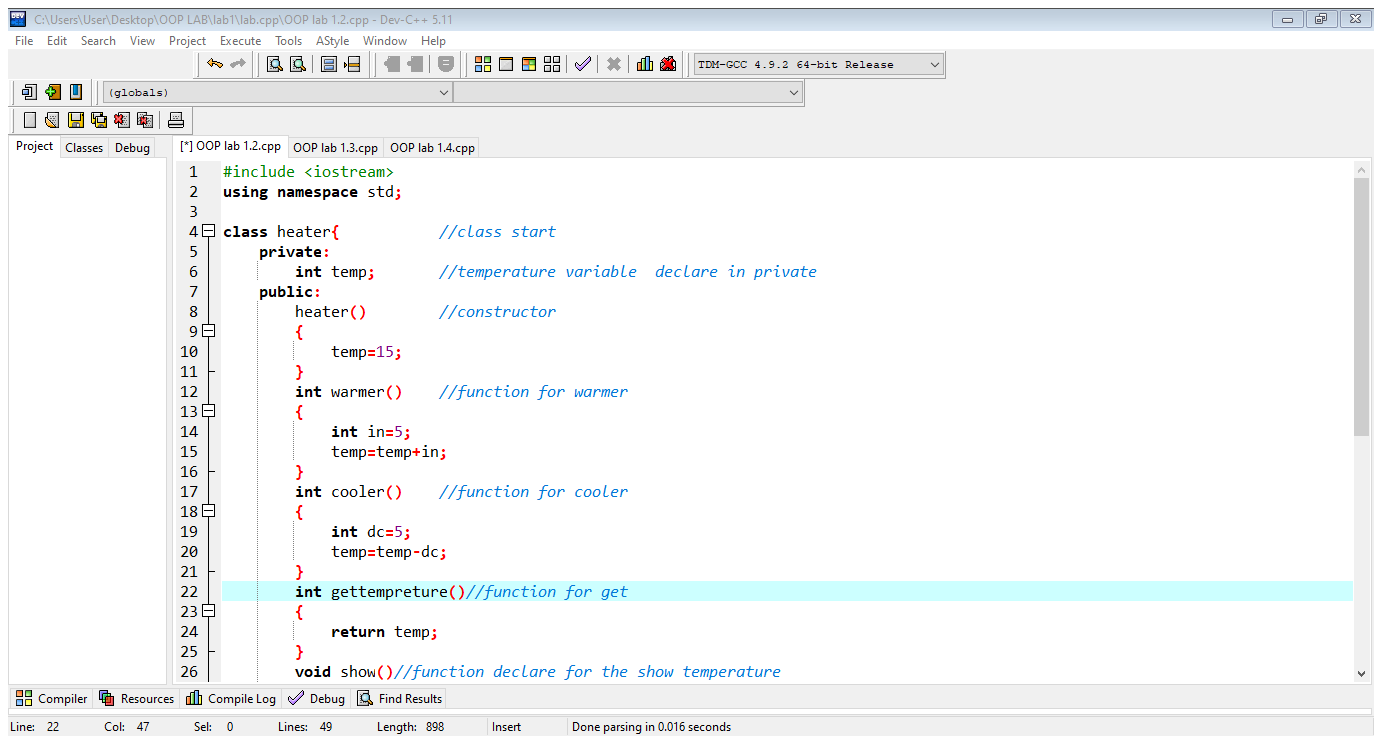
void show()

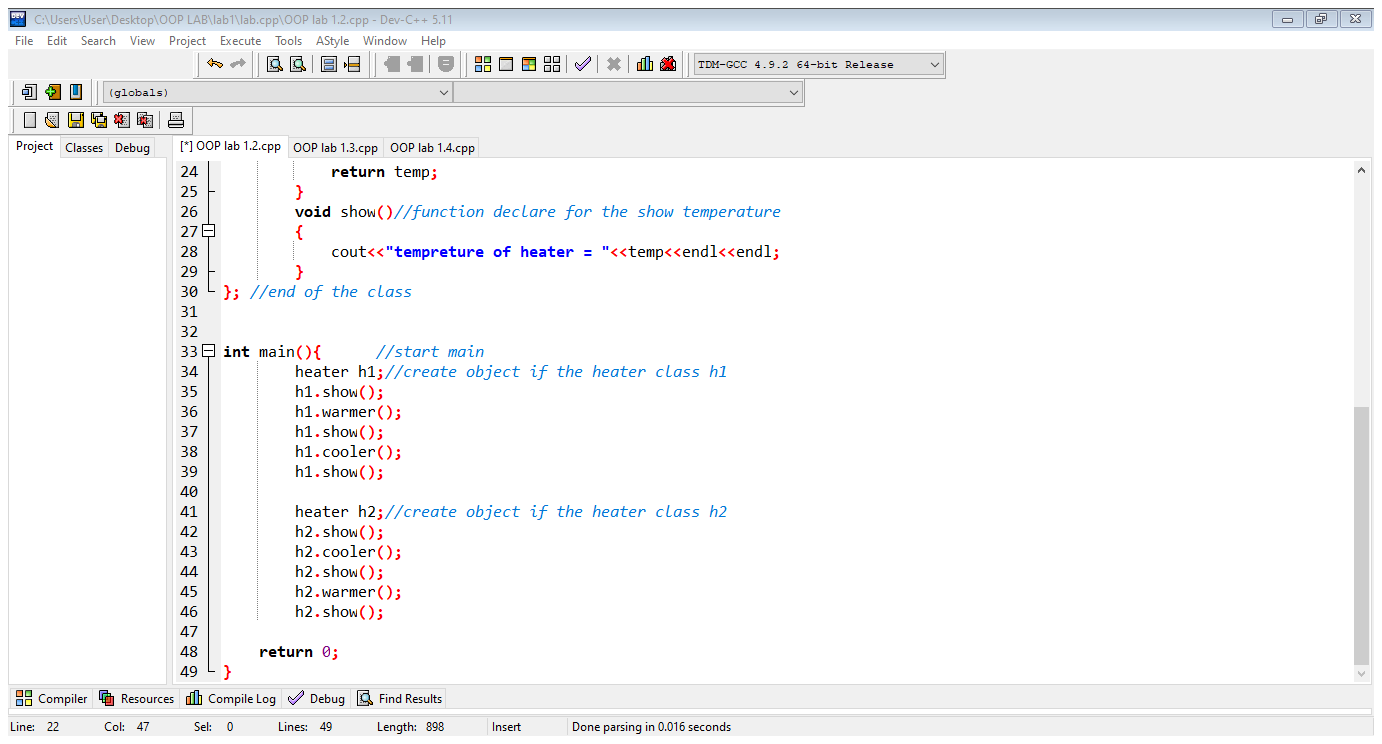
End



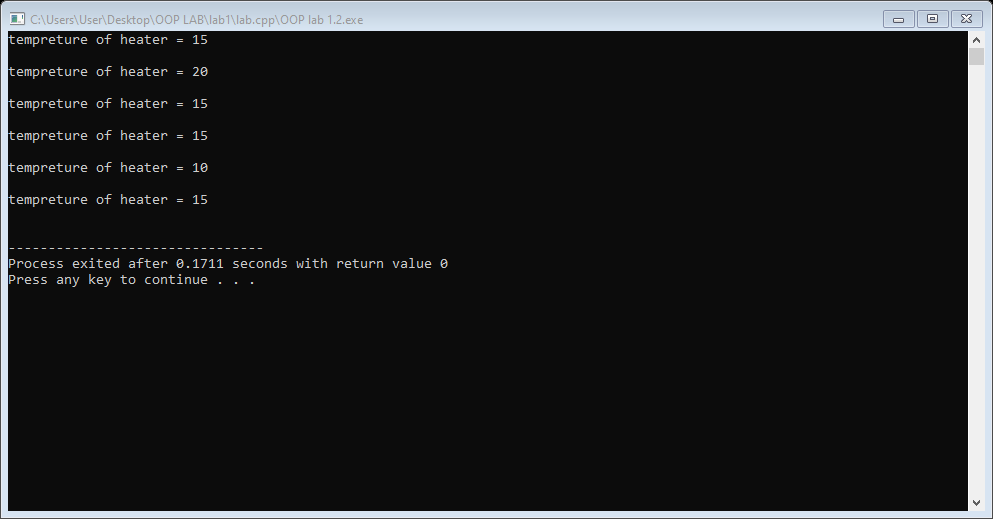
**In C++**

**Source code:**

****

****

**Output:**

****

**Activity # 02**

**Title:**

Make a class for point and model it using x and y coordinates of a point.

# Problem analysis:

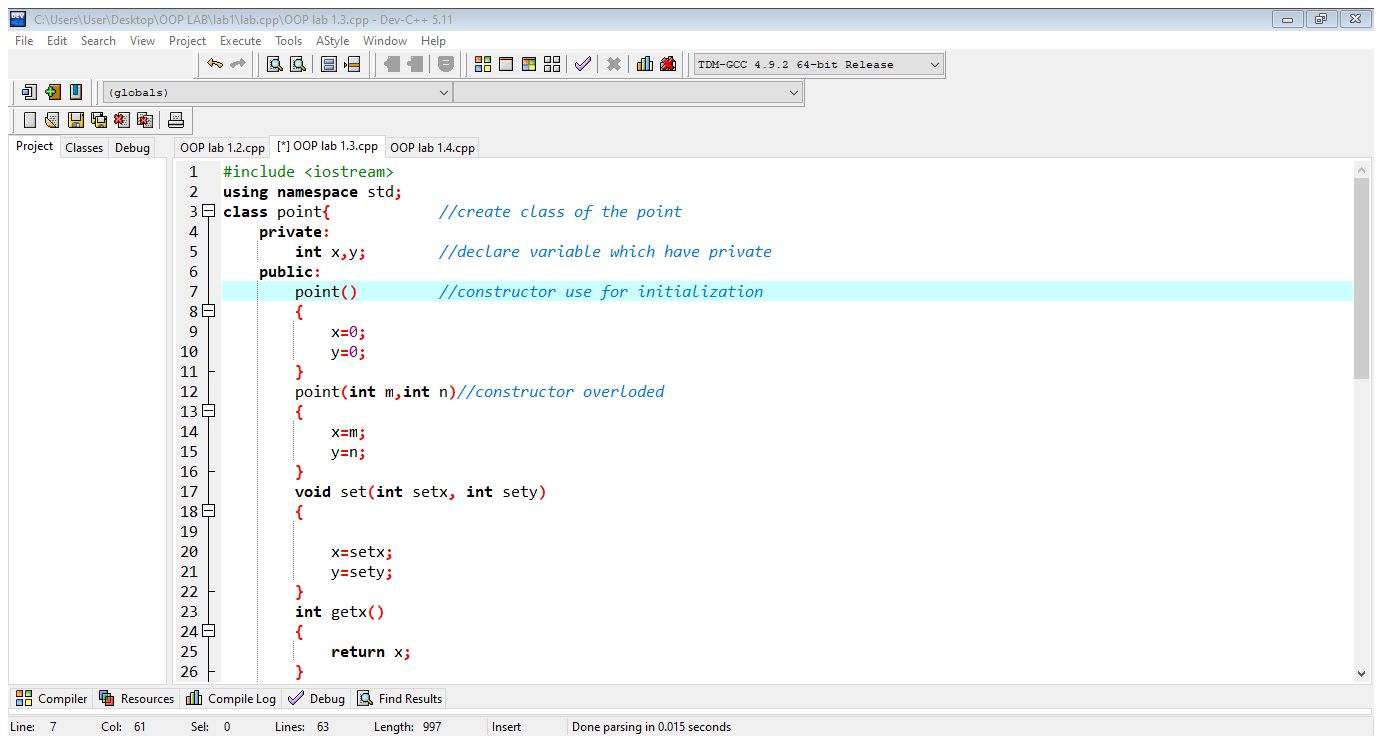
# Create a class called Point that has two data members: x‐ and y‐coordinates of the point. Provide a no‐ argument and a 2‐argument constructor. Provide separate get and set functions for the each of the data members i.e. getX, getY, setX, setY. The getter functions should return the corresponding values to the calling function. Provide a display method to display the point in (x, y) format. Make appropriate functions const.

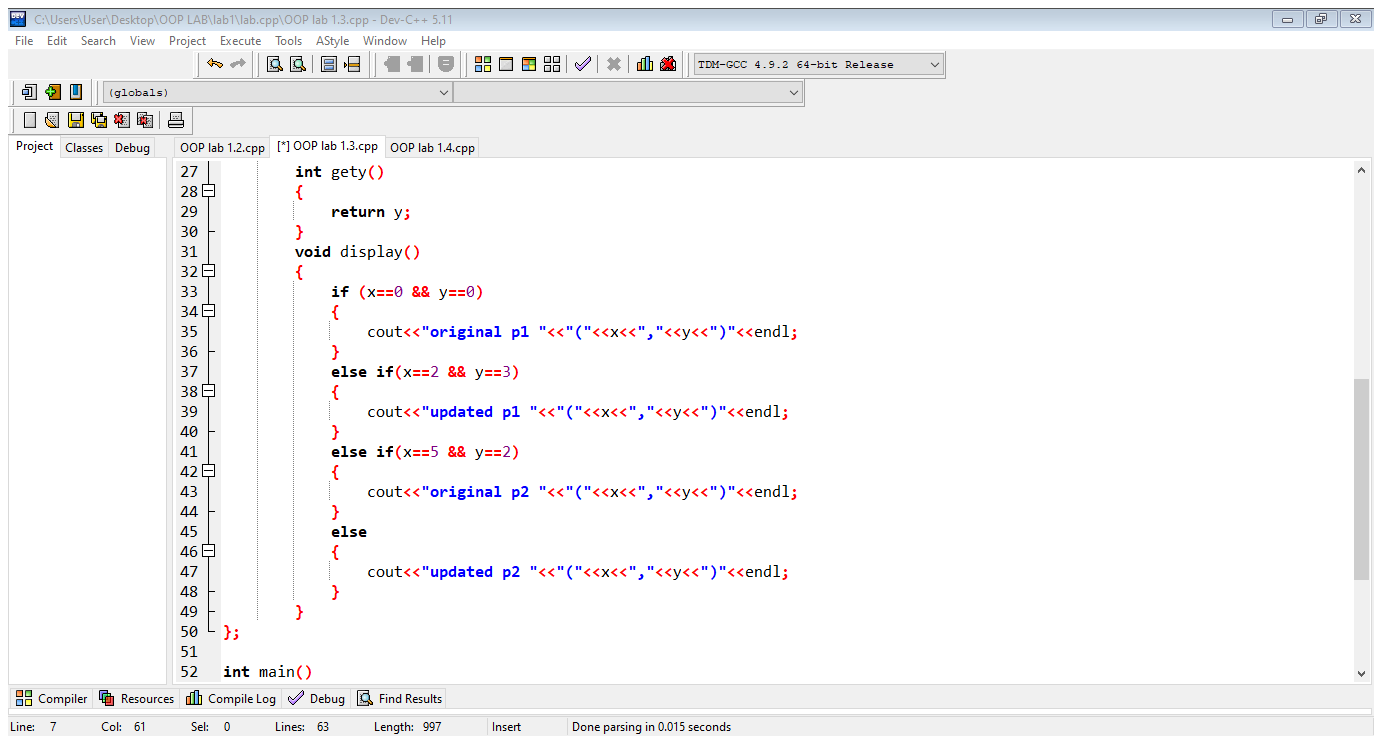
# Algorithm:

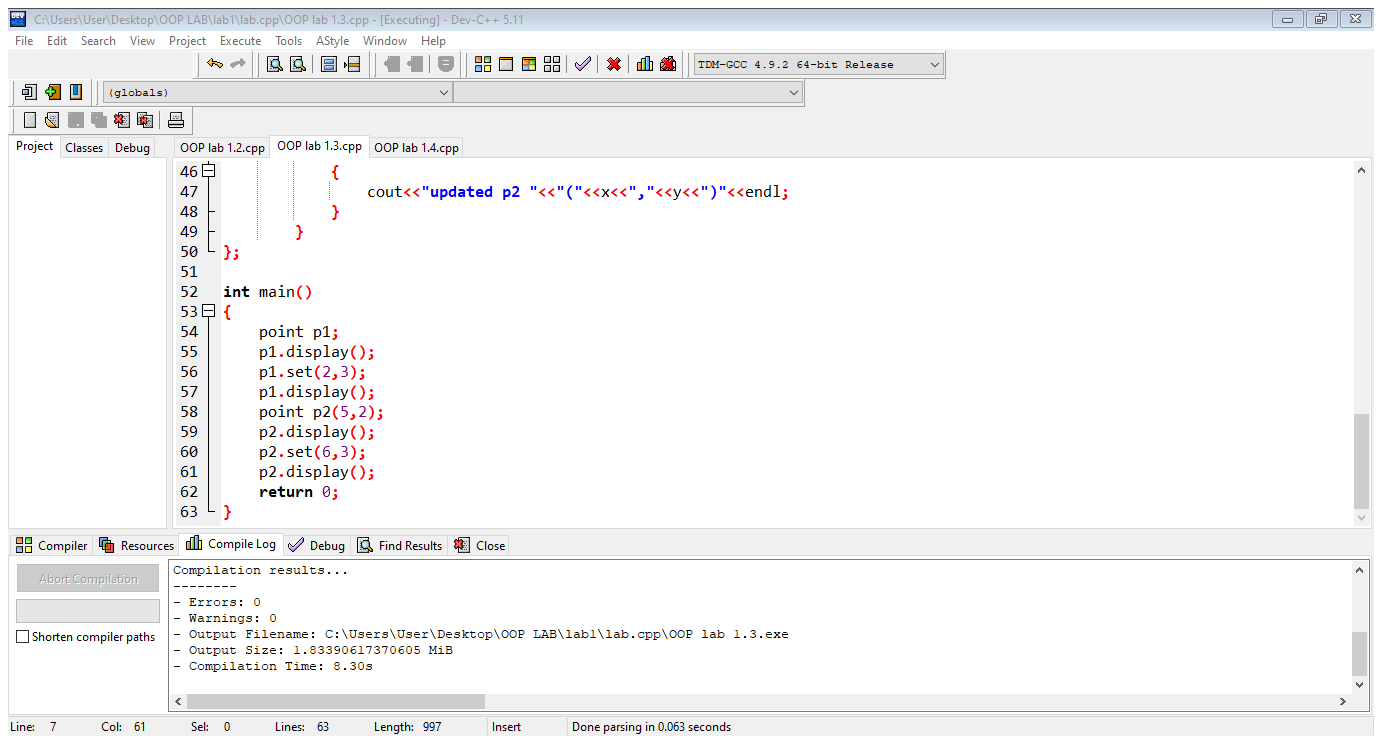
* First make class point.
* Declare x and y as private integer field.
* Define no argument constructor to set value of x and y is zero and can constructor overload.
* Define getx and gety method to return value of x and y.
* Define set method to set the value of x and y.
* Define display function to display the output and use the conditional statements.
* In main function, make objects of point to demonstrate the use of point.
* Call each function one after the other and display the display function as show

**In C++**

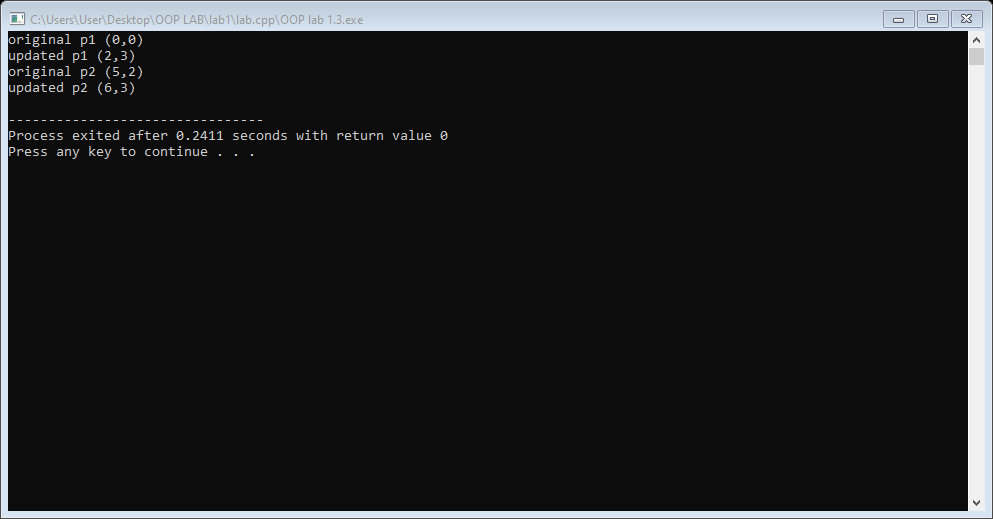
**Source code:**





****

**Output:**

****

**Activity # 03**

**Title:**

Make a class bankaccount that model to checking the bank account at a bank.

# Problem analysis:

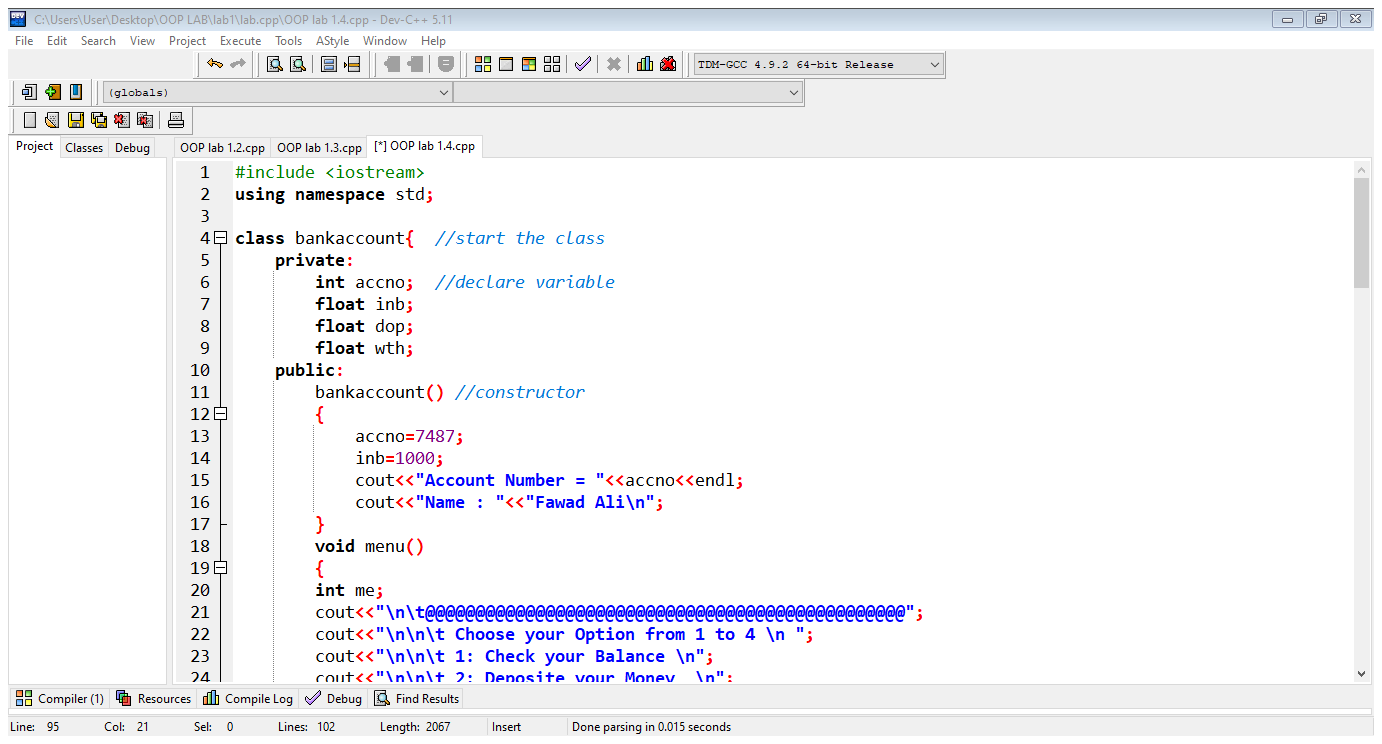
# Create a class called BankAccount that models a checking account at a bank. The program creates an account with an opening balance, displays the balance, makes a deposit and a withdrawal, and then displays the new balance. Note in withdrawal function, if balance is below Rs. 500 then display message showing insufficient balance otherwise allow withdrawal.

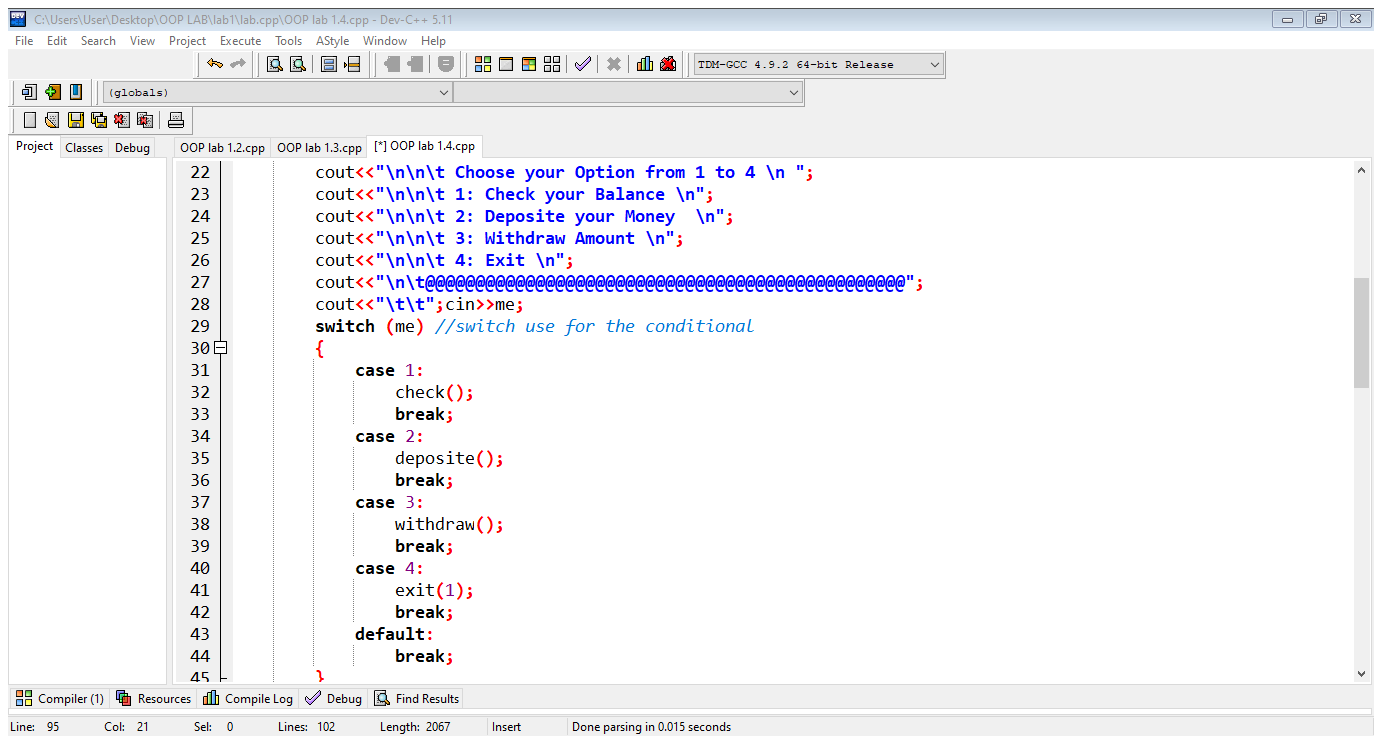
# Algorithm:

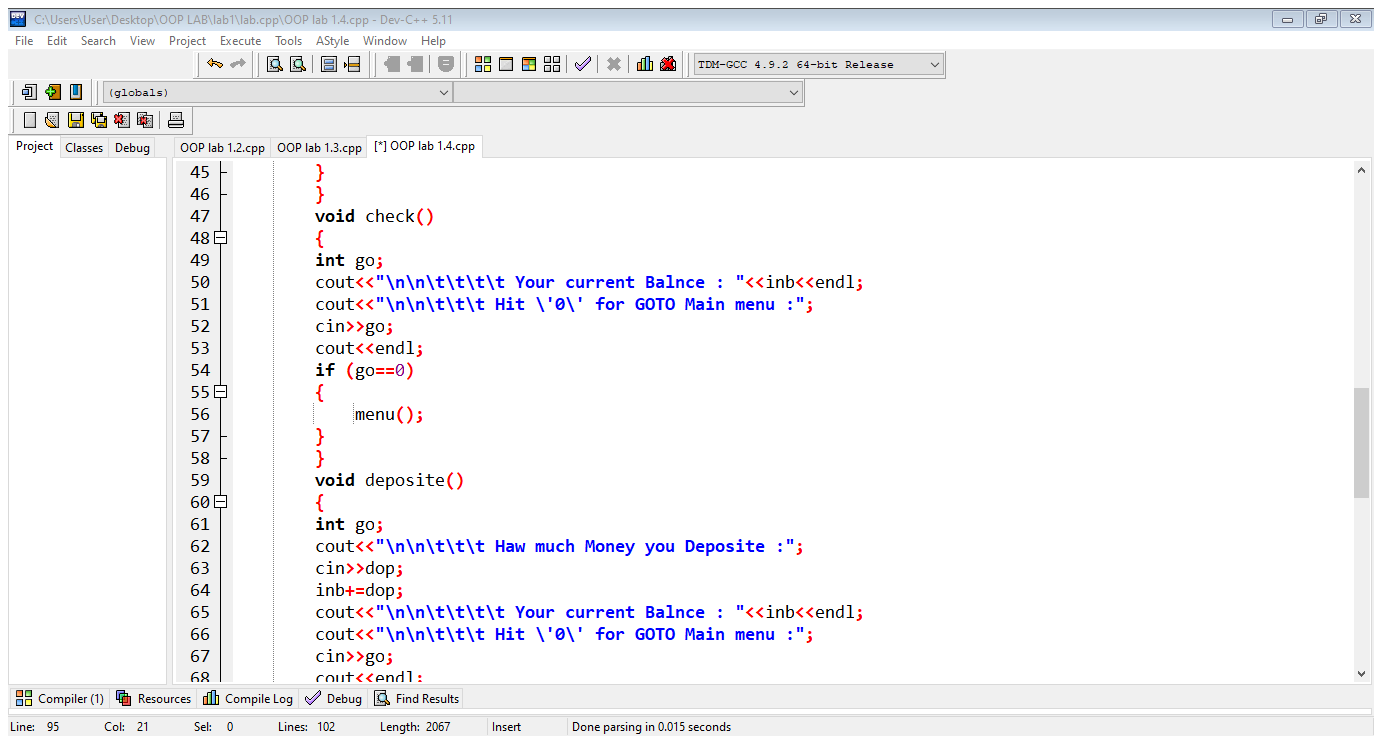
* First make class bankaccount.
* Declare variable for the account number, initial balance, deposit & withdraw as private integer field.
* Define no argument constructor.
* Define the function for menu. Then use the switch.
* Define the check for the checking balance.
* Define the deposit.
* Define withdraw when balance is less then to not complete the withdraw use conditional statement.
* Create object in the main in call the menu.

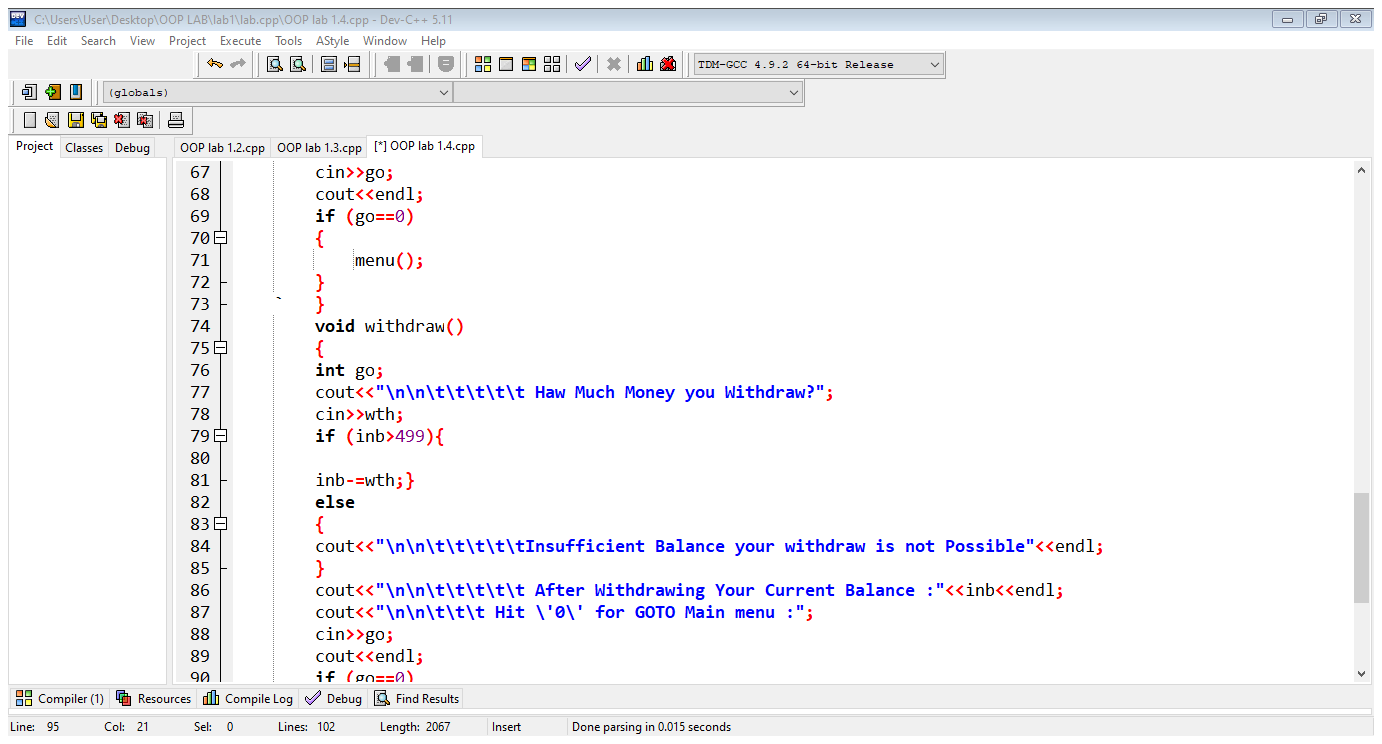
**In C++**

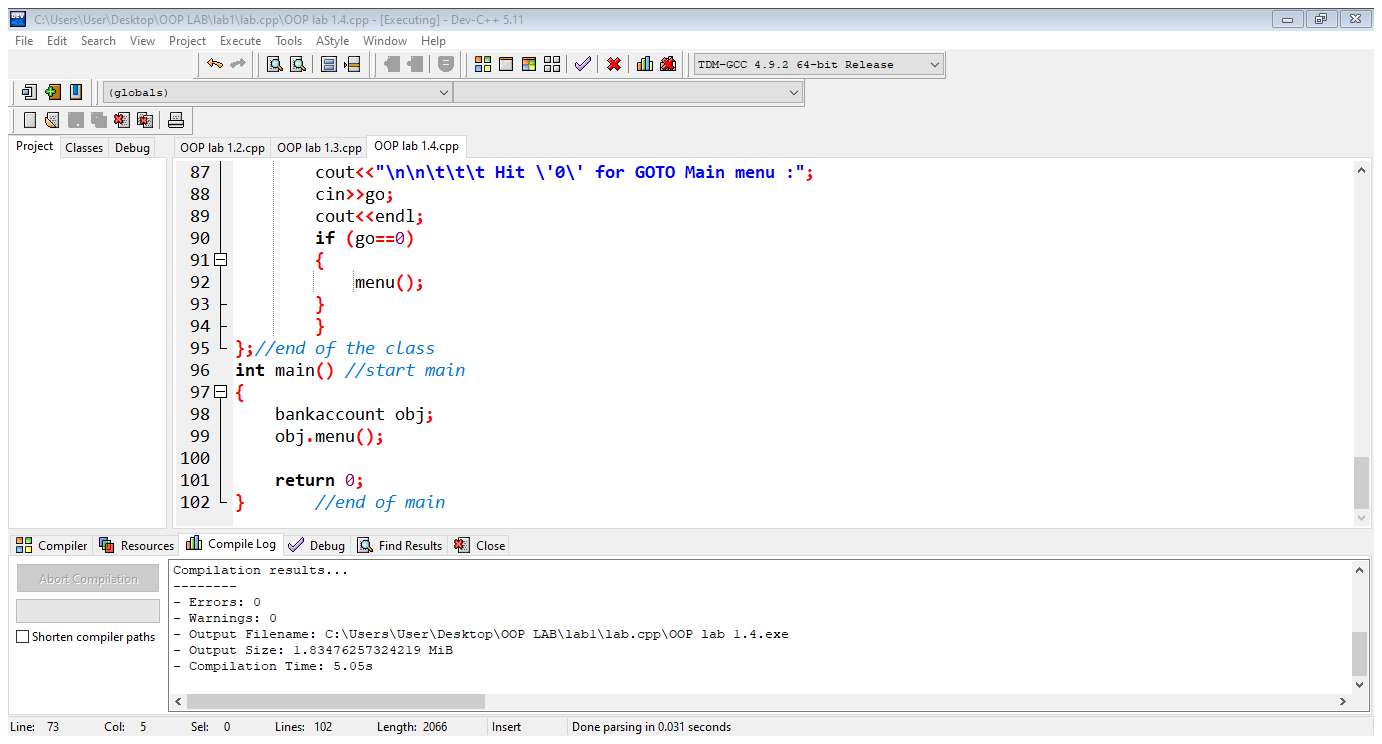
**Source code:**

****

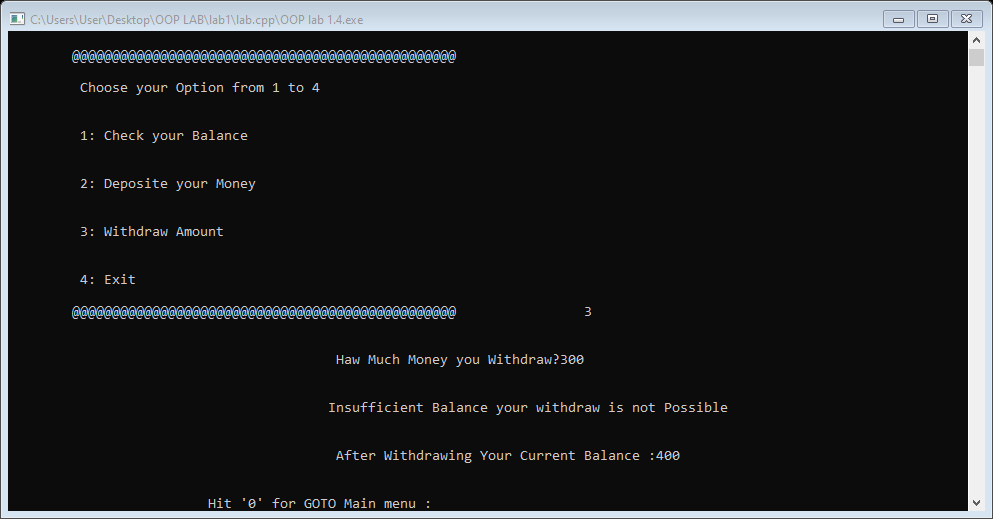
****

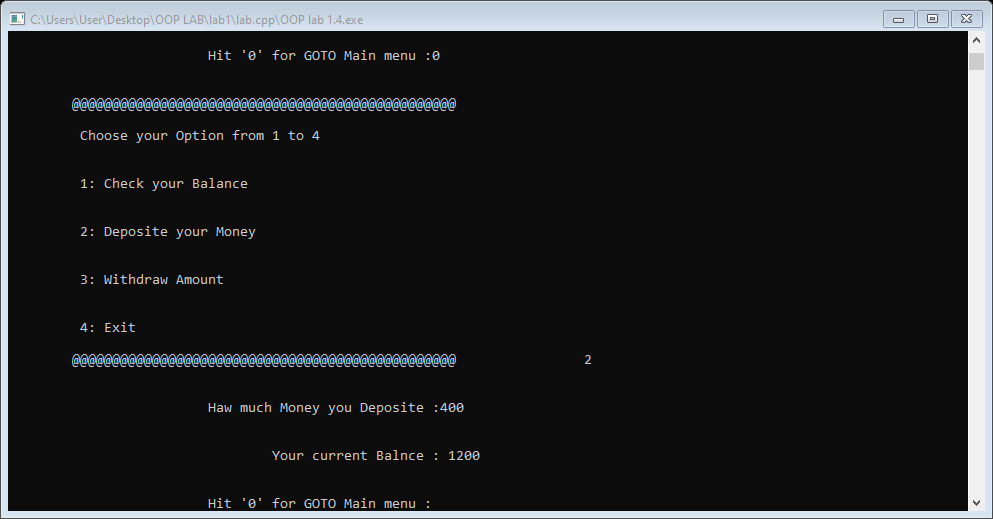
****

****

****

**Output:**

****

****

**Conclusion:**

This program helps us in understanding the basic concepts of classes and objects in different languages. It acts as a base for us and helps us in preparing ourselves for the higher level of programming. We get to know about the constructor and method in OOP with the help of this program. We get to know constructor overloaded and can value return to the function .

**Registration #:**

**Name & Section:**

**Date:**

## CSE 208L – OBJECT ORIENTED PROGRAMMING LAB LAB 01 ASSESSMENT RUBRICS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dimension** | **Exemplary** | **Acceptable** | **Developing** | **Unsatisfactory** | **Student Score out of 10 Marks** |
| **10** | **9‐7** | **6‐5** | **4‐0** |
| **Submission** | Report is submitted on time and in correct format. | Report is submitted on time with slight incorrect format. | Report is submitted on time in incorrect format. | Report is not submitted. |  |
| **Overall Impression of Lab Report** | Report is complete, well written, and organized appropriately with additional elements that enhance it. Task titles and output screenshots are included. Purpose for each concept, input requirements and output results is noted. | Report is complete, briefly written, and organized. Lacks additional elements. Task titles and output screenshots are included. Purpose for each concept, input requirements and output results is noted. | Report is mostly complete, loosely written, and fairly organized. Basic documentation including descriptions of all concepts. Specific purpose is noted for each concept. Task titles and output screenshots are  included and good formatting. | Report is incomplete, sloppy, and/or disorganized.  No documentation included.  No task titles, no output screenshots, poor  formatting. |  |
| **Ability to Code Required Class/Classes** | Able to code required class, use objects effectively, and produces desired results. | Able to code required class, use objects effectively, and produces most results. | Able to code required class, somewhat use of objects, and some results are produced. | Unable to code required class or unable to use objects. |  |
| **Compilation, Execution, and Results** | Program compiles with no errors and no warnings.  Executes without errors, excellent user prompts, good use of symbols, and spacing in output.  Thorough and organized testing has  been completed and output from test cases is included. | Program compiles with no errors and some warnings. Executes without errors.  User prompts are understandable, minimum use of symbols or spacing in output. Most of the testing has been  completed. | Program compiles with no errors and lots of warnings.  Executes without errors.  User prompts are understandable, minimum use of symbols or spacing in output.  Some testing has been  completed. | Program fails to compile. Does not execute due to errors.  User prompts are misleading or non‐ existent.  No testing has been d. |  |

**Marks**: /4 =

**Teacher Remarks and Signature:**

Department of Computer Systems Eng. UET Peshawar Prepared By: Engr. Sumayyea Salahuddin.