University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Lab Practical Two - Classes and Data Abstraction

Objective:
The objectives of this tutorial are to allow students to be able to:

- show how to implement and call the four types of methods a class can have, using the Java and C++ programming languages
- show how to implement a composition relationship in a class in Java and C++

In lab one, you implemented a simple class and wrote a complete program to use that class. In this lab, you will create another class contain attributes and examples of the four types of methods a class can have. You will also create objects of the class and invoke each of the methods of the class.

Exercise One

Requirements

An OfficeWorker has an employee number, a first name, last name, and department.

Perform an object oriented analysis on the above requirements, identifying the class and its corresponding attributes. Using UML, design a model of the above system, showing the individual detailed class diagram. Add a constructor, mutator, accessor and destructor to your class diagram.

Design using UML

OfficeWorker
- EmployeeNumber : integer - FirstName : string - LastName : string - Department : string
+ OfficeWorker(integer, string, string, string) + ~OfficeWorker() + SetEmployeeNumber(integer) : void + GetEmployeeNumber() : integer + Display() : void

Implementation

A sample implementation of the above model is shown in both C++ and Java. below 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
//by David White 2011 Sep 4

//header file needed for screen output
#include <iostream>
using namespace std;

//class for OfficeWorker
class OfficeWorker
{
        private:
                int EmployeeNumber;
                string FirstName;
                string LastName;
                string Department;
        public:
                //constructor - use to initialize object
        OfficeWorker(int en, string fn, string ln, string dep)
        {
                EmployeeNumber  = en;
                FirstName = fn;
                LastName = ln;
                Department = dep;
        }
        //destructor
        ~OfficeWorker()
        {
                cout << "Destructor called." << endl;
        }
        //mutator for employee number
        void SetEmployeeNumber(int en)
        {
                EmployeeNumber  = en;
        }
        //accessor for employee number
        int GetEmployeeNumber()
        {
                return EmployeeNumber;
        }
        //accessor used to display all the data in the  object
        void Display()
        {
                cout << "Employee Number:"  << EmployeeNumber << endl;
                cout << "First Name:"  << FirstName << endl;
        cout << "Last Name:"  << LastName << endl;
        cout << "Department:"  << Department << endl;
    }
}; //class definition ends here

//main function  - execution always start here
int main()
{
    //create an object called Obj1 of the class  OfficeWorker
    //the constructor for  OfficeWorker will be called and will
    //initialize the attributes inside of  the object to the values passed
    OfficeWorker Obj1(10000, "Fred", "Wilson", "Accounts");

    //invoke the Display() method which is an accessor to display
    //the data in the object
    Obj1.Display();

    cout << endl << "now changing the employee number using the mutator" << endl << endl;

    //call the mutator for employee number to set it to 20501
    Obj1.SetEmployeeNumber(20501);
    Obj1.Display(); //and display the data in the object again

    cout << endl << "this is an example of how to use the accessor" << endl << endl;

    //call the accessor for employee number and  display the value returned
    int en;
    en = Obj1.GetEmployeeNumber();
    cout << "This employee's number is:" << en << endl;

    //exit main() - when main() exits Obj1 will be destroyed and the
    //destructor for Obj1 will be called
    return  0;
} //end of main
```

Java Implementation

```java
//Sample program to demonstrate classes in C++
//class to track points for a driver
//by David White 2011 Sep 4

//library file needed for screen output
import java.lang.*;

//class for OfficeWorker
public class OfficeWorker {

      int EmployeeNumber;
      String FirstName;
      String LastName;
      String Department;

      //constructor - use to initialize object
      public OfficeWorker(int en, String fn, String ln, String dep)
      {
            EmployeeNumber  = en;
            FirstName = fn;
            LastName = ln;
            Department = dep;
      }

      //note that the closest method to a destructor in Java is the finalize()
      //method but because Java cleanup up automatically it is a good idea
      //not to use finalize() unless explicitly needed

      //mutator for employee number
      public void SetEmployeeNumber(int en)
      {
            EmployeeNumber  = en;
      }
      //accessor for employee number
      public int GetEmployeeNumber()
      {
            return EmployeeNumber;
      }
      //accessor used to display all the data in the  object
      public void Display()
      {
            System.out.println("Employee Number:"  + EmployeeNumber);
            System.out.println("First Name:"  + FirstName);
            System.out.println("Last Name:"  + LastName);
            System.out.println("Department:"  + Department);
      }

      //main function  - execution always start here
      public static void main(String[] args)
      {
            //create an object called Obj1 of the class OfficeWorker
            //the constructor for  OfficeWorker will be called and will
            //initialize the attributes inside of  the object to the values passed
```

```
        OfficeWorker Obj1 = new OfficeWorker(10000, "Fred", "Wilson",
         "Accounts");

        //invoke the Display() method which is an accessor to display
        //the data in the object
        Obj1.Display();

        System.out.println("\nnow changing the employee number using the
         mutator\n");

        //call the mutator for employee number to set it to 20501
        Obj1.SetEmployeeNumber(20501);
        Obj1.Display(); //and display the data in the object again

        System.out.println("\nthis is an example of how to use the accessor");

        //call the accessor for employee number and  display the value returned
        int en;
        en = Obj1.GetEmployeeNumber();
        System.out.println("\nThis employee's number is:" + en);

        //exit main() - when main() exits Obj1 will be destroyed
        //and the Java garbage collector (GC) will cleanup automatically
    } //end of main

} //class definition ends here
```

Exercise Two

In the code you entered in Exercise One, a specific mutator and accessor was written for the EmployeeNumber attribute. Complete the class by writing a mutator and accessor for each of the other attributes. Show how these methods can be called from objects you create.

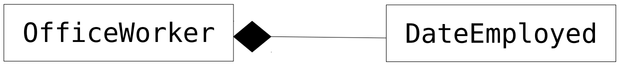
Exercise Three – worked example

In addition to the requirements you were given in Exercise One, you are now told that an OfficeWorker has a date employed. The date employed contains the day, month and year. Draw the UML class diagram representing this new information, and include a constructor, Display method in your diagram. Draw a UML relationship diagram for the relationships in your system.

Class Diagram

DateEmployed
- Day : integer - Month : integer - Year : integer
+ DateEmployed(integer, integer, integer) + Display() : void

Relationship Diagram



Update your program to reflect the new information.

Updated C++ Implementation

```
//Sample program to demonstrate classes in C++
//class to track points for a driver
//by David White 2011 Sep 4

//header file needed for screen output
#include <iostream>
using namespace std;

//DateEmployed class
//note - it is usual to place each class in a separate file
//here there are  placed in the same file for illustration
class DateEmployed
{
        private:
                int Day;
                int Month;
                int Year;
        public:
                DateEmployed(int d, int m, int y)
                {
                        Day = d;
                        Month  = m;
                        Year = y;
                }
                void Display()
                {
                        cout << Day << "-" << Month << "-" << Year << endl;
                }
};

//class for OfficeWorker
class OfficeWorker
{
        private:
                int EmployeeNumber;
                string FirstName;
                string LastName;
                string Department;
                DateEmployed DE; //this is how composition is implemented in C++
    public:
        //constructor - used to initialize object
    //note how the DE attribute is initialised in this constructor
    OfficeWorker(int en, string fn, string ln, string dep, int d, int m, int y)
        : DE(d,m,y)
    {
        EmployeeNumber  = en;
        FirstName = fn;
        LastName = ln;
        Department = dep;
    }
    //destructor
    ~OfficeWorker()
    {
        cout << "Destructor called." << endl;
    }
    //mutator for employee number
    void SetEmployeeNumber(int en)
    {
        EmployeeNumber  = en;
    }
    //accessor for employee number
    int GetEmployeeNumber()
    {
        return EmployeeNumber;
    }
    //accessor used to display all the data in the  object
    void Display()
    {
        cout << "Employee Number:"  << EmployeeNumber << endl;
        cout << "First Name:"  << FirstName << endl;
        cout << "Last Name:"  << LastName << endl;
        cout << "Department:"  << Department << endl;
        cout << "Date Employed:";
            DE.Display(); //display the attributes from the inner class
    }
}; //class definition ends here

//main function  - execution always start here
int main()
{
    //create an object called Obj1 of the class  OfficeWorker
    //the constructor for  OfficeWorker will be called and will
    //initialize the attributes inside of the object to the values passed
    //note the three extra parameters are for the attributes of the inner object
    OfficeWorker Obj1(10000, "Fred", "Wilson", "Accounts", 15, 11, 2010);
```

```
	//invoke the Display() method which is an accessor to display
	//the data in the object
	Obj1.Display();

	cout << endl << "now changing the employee number using the mutator" << endl << endl;

	//call the mutator for employee number to set it to 20501
	Obj1.SetEmployeeNumber(20501);
	Obj1.Display(); //and display the data in the object again

	cout << endl << "this is an example of how to use the accessor" << endl << endl;

	//call the accessor for employee number and  display the value returned
	int en;
	en = Obj1.GetEmployeeNumber();
	cout << "This employee's number is:" << en << endl;

	//exit main() - when main() exits Obj1 will be destroyed and the
	//destructor for Obj1 will be called
	return  0;
} //end of main
```

Updated Java Implementation

```
//Sample program to demonstrate classes in C++
//class to track points for a driver
//by David White 2011 Sep 4

//library file needed for screen output
import java.lang.*;

//DateEmployed class
//note - it is usual to place each class in a separate file
//here there are  placed in the same file for illustration
class DateEmployed
{

            int Day;
            int Month;
            int Year;

            public DateEmployed(int d, int m, int y)
            {
                  Day = d;
                  Month  = m;
                  Year = y;
            }
            public void Display()
            {
                  System.out.println(Day + "-" + Month + "-" + Year);
            }
};


//class for OfficeWorker
public class OfficeWorker {

      int EmployeeNumber;
      String FirstName;
      String LastName;
      String Department;
      DateEmployed DE;

      //constructor - used to initialize object
      //note how the DE attribute is initialised in this constructor
      public OfficeWorker(int en, String fn, String ln, String dep, int d, int m,
       int y)
      {
            EmployeeNumber  = en;
            FirstName = fn;
            LastName = ln;
            Department = dep;
            DateEmployed TempDE  = new DateEmployed(d,m,y);
            DE = TempDE;
      }

      //note that the closest method to a destructor in Java is the finalize()
```

```
    //method but because Java cleanup up automatically it is a good idea
    //not to use finalize() unless explicitly needed

    //mutator for employee number
    public void SetEmployeeNumber(int en)
    {
        EmployeeNumber  = en;
    }
    //accessor for employee number
    public int GetEmployeeNumber()
    {
        return EmployeeNumber;
    }
    //accessor used to display all the data in the  object
    public void Display()
    {
        System.out.println("Employee Number:"  + EmployeeNumber);
        System.out.println("First Name:"  + FirstName);
        System.out.println("Last Name:"  + LastName);
        System.out.println("Department:"  + Department);
        System.out.print("Date Employed:");
        DE.Display(); //display the attributes from the inner class
    }

    //main function  - execution always start here
    public static void main(String[] args)
    {
        //create an object called Obj1 of the class OfficeWorker
        //the constructor for  OfficeWorker will be called and will
        //initialize the attributes inside of  the object to the values passed
        //note the three extra parameters are for the attributes of the inner
        //object
        OfficeWorker Obj1 = new OfficeWorker(10000, "Fred", "Wilson",
         "Accounts", 15, 11, 2010);

        //invoke the Display() method which is an accessor to display
        //the data in the object
        Obj1.Display();

        System.out.println("\nnow changing the employee number using the
         mutator\n");

        //call the mutator for employee number to set it to 20501
        Obj1.SetEmployeeNumber(20501);
        Obj1.Display(); //and display the data in the object again

        System.out.println("\nthis is an example of how to use the accessor");

        //call the accessor for employee number and  display the value returned
        int en;
        en = Obj1.GetEmployeeNumber();
        System.out.println("\nThis employee's number is:" + en);

        //exit main() - when main() exits Obj1 will be destroyed
        //and the Java garbage collector (GC) will cleanup automatically
```

```
      } //end of main

} //class definition ends here
```

Exercise Four - Homework

A customer has a first name, last name, address, and id number.

a) Perform an object oriented analysis on the above requirements, identifying the class and its corresponding attributes.

b) Using UML, design a model of the above system, showing the individual detailed class diagram. Add a constructor, mutator, accessor and destructor to your class diagram.

c) Implement the class in your detailed UML diagram, using both Java and C++.

d) Write separate complete programs in Java and C++ to demonstrate how to create objects from the class you implemented. Using comments, show how the constructor, mutator, accessor and destructor for your class are called.

e) Run your program to show that your code runs are stated in your comments.

In addition to the above requirements, you are supplied with these additional requirements:

A customer also has an agent assigned record. An agent assigned record contains the name of the agent and the agent's telephone number.

f) Draw a detailed UML class diagram for any class you identify in the additional requirements.

g) Draw a UML relationship diagram to depict the relationships between the classes you identify in the complete system.

h) Update you program to reflect this new information and run it again, ensuring it works.