FACTFILE:
GCE DIGITAL TECHNOLOGY

Rewarding Learning

AS1 Approaches to Systems Development: Programming – Object Oriented Programming Terminology: Objects, Classes, Methods and Inheritance

Object Oriented Programming(OOP) Terminology

Learning Outcomes

Students should be able explain the following terms related to object-oriented programming:

- Object
- Class
- Method
- Inheritance

Content in Object Oriented Programming Terminology Fact File

- ✔ Explain the term Object
- ✔ Explain the term Class
- ✔ What is a Method?
- ✔ Explain the term Inheritance
- ✔ Evaluating the use of the object oriented approach

Explain the term Object

Object-oriented programming makes use of objects when designing and building applications.

An object is an element of a program that can perform actions and interact with other elements of the program. When designing a program or application objects are considered first. During the design process objects are defined formally.

Once all of the objects have been defined, the building of an application can begin. Objects occur in the real world, for example a car could be described as an object. The car would have certain attributes such as colour, registration number, make, model, engine size and so on. In OOP the attributes that describe objects care called properties.

A car can perform actions for example start ignition, accelerate, brake, stop the engine. In OOP these actions are called methods. Methods allow the objects to do things and allow the properties of the object to be manipulated.

Objects therefore describe real world elements through the use of methods and properties.

Explain the term Class

A class is the basic building block of object oriented programming. The class specifies the attributes and behaviours that relate to the real world object. A class is a template, plan or blueprint that describes the details of an object. An object is constructed from a class.

A simple example of the class car could be defined as having the data: Registration Number, Make, Model and Year of Manufacture. A method associated with the class could be to calculate the age of the car on 1st January.

Note that a class is only a definition of objects of the same kind. It is an abstract design. That becomes real when Objects are constructed or instantiated from that class. An object is a specific instance of a class that contains real values. An example of an object for this class would be: DUI 78900, Ford, Fiesta, 2000.

A class is defined by its name, its data members and its member functions (e.g. methods). A class encapsulates or hides these elements within its structure (encapsulation). The class can be re-used within a number of applications. Access to data is provided through member functions.

Explain the term Method

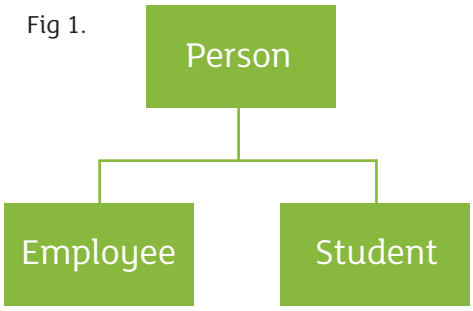
As already indicated, a method is a section of code or procedure contained within a class which defines an action that an object of that class can perform. A class can have a number of methods. The methods only have access to data within their own class. A method is always associated with a class. Examine the code on page 4 and identify the methods associated with the classes.

Explain the term Inheritance

Inheritance is a fundamental feature of OOP. A new class can inherit the attributes and behaviours of an existing class. In this case the new class is known as a derived, child or sub class. The existing class is known as a base, parent or super class. The derived class inherits all the attributes and behaviours of the base class. The benefit of inheritance is that new attributes and behaviours can be defined for the derived class. In addition, the base class can be re-used to define new more specialised classes.

Consider the diagram (Fig 1.). This could represent an inheritance hierarchy. The base class is Person. Both Employee and Student are derived from the base class Person. Both will have the attributes and behaviours associated with a Person. Each will have their own specialised behaviours and attributes.

Fig 1.



The code below represents a simple implementation of this inheritance diagram. It can be implemented in a c# console application. Review the code for the difference between the classes Employee and Student.

Note that the ShowDetails method is used by both objects but is only written once. Therefore the solution is more efficient as code is re-used.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication2
{
    class Inheritance
    {
        class Person
        {
            public string name;
            public int age;

            public void ShowDetails()
            {
                Console.WriteLine("Name and age are " +
                name + " and " + age);
            }
        }

        // Employee is derived from Person.
        //Derived class or Child class.
        class Employee : Person
        {
            public string position; // position of Employee in the company

            public void ShowPosition()
            {
                Console.WriteLine("Position is " + position);
            }
        }
        // Student is also derived from Person.
        class Student : Person
        {
            public string course; // course studied by student
            public void ShowCourse()
            {
                Console.WriteLine("Position is " + course);
            }
        }

        static void Main(string[] args)
        {
            Employee t1 = new Employee();
            Student t2 = new Student();
            t1.name = "John";
            t1.age = 30;
            t1.position = "Manager";
            t2.name = "Mary";
            t2.age = 21;
            t2.course = "BSc Computing";
            Console.WriteLine("Info for t1: ");
            t1.ShowDetails();
            t1.ShowPosition();
            Console.WriteLine();
            Console.WriteLine("Info for t2: ");
            t2.ShowDetails();
            t2.ShowCourse();
            Console.ReadKey();
        }
    }
}
```

Evaluating the use of the Object Oriented approach

The Object Oriented approach is an efficient way of developing applications for the following reasons:

- Classes and methods created for one object oriented application can be reused in other programs and applications. In addition objects can be extended to include other behaviours and attributes. Re-use enables faster development and lowers the overall cost.
- Encapsulation means that once an object is created, it can be used without knowledge of how it is implemented or coded. In addition, objects can hide aspects of themselves from programmers which means that certain parts of the code cannot be altered.
- Due to the modular nature of OOP, designers go through an extensive planning phase. This means better designs with fewer flaws and less time spent on program maintenance.
- The OO approach can also improve software quality. Firstly, a class can be tested in isolation, away from the application in which it is intended to be used. Secondly, with the use of inheritance, the testing of a derived class can assume the base class is correct and can focus on the additional attributes and behaviours.

Object oriented programming has some drawbacks which include:

- Concepts including inheritance and encapsulation can be difficult to understand in the first instance meaning that there is a steep learning curve. This means that programmers must learn extensive class libraries before even beginning to program simple applications.
- Object oriented programs can incur run time cost due to dynamic dispatching. This is where the process selects which particular version of a method to call (polymorphism)

Questions

1 Explain the following terms related to object oriented programming:

a) Object
b) Method
c) Inheritance
d) Class

[12 marks]

2 Investigate the object oriented approach and write a report entitled "An evaluation of the Object Oriented Approach to Application Development". The report should be at least 500 words and include technical detail and simple code examples. [10 marks]

Practical Task

1. Implement the program above in c# or re-create the program in a language of your choice. Run the program and test its outcome by changing the values for t1 and t2.

2. Modify the code used in task 1 above to include the following:
   a) The class Person should now include an address of type string;
   b) A new class is required called Pupil which is derived from Person. A pupil will have a name, address, age and class of type string.

   Hint:
   - ✔ Ensure your code will show all of the details for each class.
   - ✔ You will need to instantiate an object of type Pupil. Use the code to help you. The line Employee t1 = new Employee(); instantiates an object of type Employee.
   - ✔ Ensure you modify ShowDetails() to include the new attribute address.

Bibliography

BCS Academy Glossary Working Party, *2013, BCS Glossary of Computing and ICT*, 13th Edition, Swindon, BCS Learning and Development Ltd

© CCEA 2017