Lesson 11: Abstraction

11.1. Introduction

In our previous lesson, we discussed about classes and objects. In this lesson, you will learn about abstraction- both functional and data abstraction.

11.2. Lesson objectives

By the end of this lesson, you will be able to:

- Define abstraction
- Explain the importance of abstraction
- Describe ways of implementing abstraction

11.3. Lesson outline

This lesson is structured as follows:

11.1. Introduction
11.2. Lesson objectives
11.3. Lesson outline
11.4. Abstraction
11.5. Data abstraction
11.6. Advantages of data abstraction
11.7. Disadvantages of data abstraction
11.8. Functional abstraction
11.9. Advantages of functional abstraction
11.10. Disadvantages of functional abstraction
11.11. Abstract classes
11.12. Revision questions
11.13. Summary
11.14. Suggested reading

11.4. Abstraction

Abstraction is an object oriented programming feature that allows users to work with functions and/or data without concerning themselves of how the functions are actually implemented (functional abstraction) and/or how the data is actually stored(data abstraction).Using abstraction, you only provide the necessary details and hide the background details. It is also referred to as information hiding.

11.5. Data abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details. Using abstraction, programmers are able to define methods and properties of an object. C++ classes provide great level of data abstraction. They provide sufficient public methods to the outside world to play with the functionality of the object and to manipulate object data, i.e., state without actually knowing how class has been implemented internally. We can One

can regard the notion of an object (from object oriented programming) as an attempt to combine abstractions of data and code. In C++ access specifiers are used to enforce abstraction **Access specifiers:**

- **Public**: This access specifier makes a member to be accessible from anywhere outside the class but within a program. You can set and get the value of public variables without any member function.
- **Protected**: This access specifier allows a member variable or function can be accessed from by members of the same or derived class.
- P**rivate:** This access specifier makes a member variable or function to be accessed only from within the class or by friend functions. Otherwise such a member cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members.

11.6. Advantages of data abstraction:

The advantages of data abstraction include:

- Class members are protected from inadvertent user-level errors, which might corrupt the state of the object.
- The class implementation may be modified in response to changing requirements or bug reports without requiring change in user-level code.
- Data and methods on the data are often specified together and this makes it easier to modify the code in future.

11.7. Disadvantages of data abstraction

- Not all programming languages support data abstraction
- Knowledge of the abstraction used is required in order to use data or function e.g.
  during inheritance and code reuse.

11.8. Function abstraction

In this abstraction, a function is declared and defined in one place in the program but used elsewhere without concern of how it has been implemented (declared and defined).This is facilitated by use of function call and message passing in objects. In object oriented programming, a function or method can be abstracted by using access specifiers e.g. protected, private and public. They can also be abstracted by declaring and defining in one location in the program and used elsewhere by invoking the function name. Each program component should hide (abstract) as much information as possible from the users of that component.

11.9. Advantages of functional abstraction

- Unnecessary details are hidden from the user
- It simplifies the program structure,

- It makes it easier to modify the program in future as need arise
- It reduces chances of making errors and makes debugging easier

11.10. Disadvantages of functional abstraction

- For large programs with several functions naming conventions may give rise to conflicts
- Integrating the modules/functions become a challenge
- Functions may override others and produce unexpected results (for base and derived class functions)

11.11. Abstract class

An abstract class is a class created to be used as a base class. An abstract class contains at least one pure virtual function. A pure virtual function has a pure specifier (= 0) in the declaration of a virtual member function in the class declaration.

The following is an example of an abstract class:

```
class class_name{
public:
  virtual void functionname() = 0;
 };
```

Abstract classes are used in subtype polymorphism.

11.12. Revision questions

[a] Describe the function of the following access modifiers in c++
   i. Private
   ii. Protected
   iii. Public

[b] Discuss three advantages of functional abstraction in C++

11.13. Summary

In this lesson you have learnt about abstraction as used in object oriented programming. In particular, we discussed two types of abstraction: The data abstraction and functional abstraction. For each type we discussed the advantages and disadvantages.

11.14. Suggested reading

[1]. http://www.tutorialspoint.com/cplusplus/cpp_class_access_modifiers.htm

[2]. https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Abstraction_%28computer_science%29.html

[3]. Sams Teach Yourself C++ in 24 hours by Jesse Liberty and Rogers Cadenhead.

[4]. Object oriented programming using c++ by Joyce Farrell 4th.Edition

[5]. Object oriented programming with C++ by E Balagurusamy 3rd ed;
publisher: Tata Mcraw Hill
[6]. Object-oriented programming with c++ by Sourav Sahay.
[7].https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.cbclx01/cplr142.htm