**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **24/06/2020** | **Name:** | **Nayanashree K S** |
| **Course:** | **C++** | **USN:** | **4AL16EC042** |
| **Topic:** | **Class and objects**  **More on classes** | **Semester & Section:** | **8 A** |
| **Github Repository:** | **nayana\_online** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **MODULE 5** **What is a Class** Objects are created using classes, which are actually the focal point of OOP.  The class describes what the object will be, but is separate from the object itself. In other words, a class can be described as an object's blueprint, description, or definition. You can use the same class as a blueprint for creating multiple different objects. For example, in preparation to creating a new building, the architect creates a blueprint, which is used as a basis for actually building the structure. That same blueprint can be used to create multiple buildings.  Programming works in the same fashion. We first define a class, which becomes the blueprint for creating objects.  Each class has a name, and describes attributes and behavior.  In programming, the term type is used to refer to a class name: We're creating an object of a particular type. **Abstraction** Data abstraction is the concept of providing only essential information to the outside world. It's a process of representing essential features without including implementation details.  A good real-world example is a *book*: When you hear the term book, you don't know the exact specifics, i.e.: the page count, the color, the size, but you understand the idea of a book - the abstraction of the book. **Constructors** Class constructors are special member functions of a class. They are executed whenever new objects are created within that class.  The constructor's name is identical to that of the class. It has no return type, not even void.  For example:  class myClass { public: myClass() { cout <<"Hey"; } void setName(string x) { name = x; } string getName() { return name; } private: string name; };  int main() { myClass myObj;  return 0; }  //Outputs "Hey”  **MODULE 6**  **More on classes**   C**omposition** Now, our Person class has a member of type Birthday:class Person { public: Person(string n, Birthday b) : name(n), bd(b) { } private: string name; Birthday bd; }; **Creating a New Class** It is generally a good practice to define your new classes in separate files. This makes maintaining and reading the code easier. To do this, use the following steps in CodeBlocks: Click File->New->Class... Give your new class a name, uncheck "Has destructor" and check "Header and implementation file shall be in same folder", then click the "Create" button. Note that two new files have been added to your project: **Destructors** Remember constructors? They're special member functions that are automatically called when an object is created. Destructors are special functions, as well. They're called when an object is destroyed or deleted. |