**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **24/06/2020** | **Name:** | **Abhishek Vasudev Mahendrakar** | |
| **Course:** | **C++ Tutorial by SOLOLEARN** | **USN:** | **4AL17EC003** | |
| **Topic:** | 1. **Classes and Objects.** 2. **More on Classes** | **Semester & Section:** | **6th-‘A’** | |
| **Github Repository:** | **ECEAbhishekVMahendrakar** | **E-mail:** | **abhi2244mahendrakar@gmail.com** | |
| **FORENOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**   1. **Classes and Objects:** 2. **What is an Object?**  * **O**bject **O**riented **P**rogramming is a programming style that is intended to make thinking about programming closer to thinking about the real world. * In programming, **objects**are independent units, and each has its own **identity**, just as objects in the real world do. * An object might contain other objects but they're still different objects. * Objects also have **characteristics**that are used to describe them. For example, a car can be red or blue, a mug can be full or empty, and so on. These characteristics are also called **attributes**. An attribute describes the current **state**of an object. * Objects can have multiple attributes (the mug can be **empty**, **red**and **large**). * Each object has its own **attributes**, which describe its current state. Each exhibits its own **behavior**, which demonstrates what they can do.  1. **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. * Each class has a **name**, and describes **attributes**and **behavior**. * **Method**is another term for a class' behavior. A method is basically a **function**that belongs to a class.  1. **Example of a class:**  * Define all **attributes**and **behavior**(or members) in the body of the class, within curly braces. * A member that has been defined using the **public**keyword can be accessed from outside the class, as long as it's anywhere within the scope of the class object.  1. **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**. * **Abstraction**means, that we can have an idea or a concept that is completely separate from any specific instance.  1. **Encapsulation:**  * In object orientation, encapsulation means more than simply combining attributes and behavior together within a class; it also means restricting access to the inner workings of that class.  1. **Example of Encapsulation:**  * Access specifiers are used to set access levels to particular members of the class. The three levels of access specifiers are **public**, **protected**, and **private**. * A **public**member is accessible from outside the class, and anywhere within the scope of the class object. * A **private**member cannot be accessed, or even viewed, from outside the class; it can be accessed only from within the class. * A **public**member function may be used to access the **private**members.  1. **Constructors:**  * Class **constructors**are special member functions of a class. They are executed whenever new objects are created within that class. * **Constructors**can be very useful for setting initial values for certain member variables. * A default constructor has no parameters. However, when needed, parameters can be added to a constructor.  1. **More on Classes:** 2. **Separate Files for Classes:**  * It is generally a good practice to define your new classes in separate files. This makes maintaining and reading the code easier. * The header file (.h) holds the function declarations (prototypes) and variable declarations. It currently includes a template for our new **MyClass**class, with one default constructor. * The **double colon** in the source file (.cpp) is called the **scope resolution operator**, and it's used for the constructor definition: * The scope resolution operator is used to define a particular class' member functions, which have already been declared. Remember that we defined the constructor prototype in the **header file**.  1. **Destructors:**  * **Destructors**are special functions, as well. They're called when an object is **destroyed**or **deleted**. * The name of a **destructor**will be exactly the same as the class, only prefixed with a **tilde (~)**. A destructor can't return a value or take any parameters. * Each class will have just **one**destructor.  1. **Selection Operator:**  * **ifndef**stands for "if not defined". * **endif**ends the condition. * We can also use a **pointer**to access the object's members. * The **arrow member selection operator (->)** is used to access an object's members with a pointer.  1. **Const Objects:**  * A **constant**is an expression with a fixed value. It cannot be changed while the program is running. Use the **const**keyword to define a constant variable. * Attempting to call a regular function from a constant object results in an error.  1. **Member Initializers:**  * C++ provides a handy syntax for initializing members of the class called the **member initializer list** (also called a **constructor initializer**). * The member initialization list may be used for regular variables, and must be used for constant variables.  1. **Composition,Part 1:** 2. **Composition, Part 2:** 3. **The friend keyword:**  * declaring a **non-member**function as a **friend**of a class allows it to access the class' private members. This is accomplished by including a declaration of this external function within the class, and preceding it with the keyword **friend**.  1. **The this keyword:**  * Every object in C++ has access to its own address through an important pointer called the **this**pointer. * The **this**keyword has an important role in **operator overloading.**  1. **Operator Overloading:**  * Overloaded operators are functions, defined by the keyword **operator**followed by the symbol for the operator being defined. * An overloaded operator is similar to other functions in that it has a **return type** and a**parameter list**. | | | |