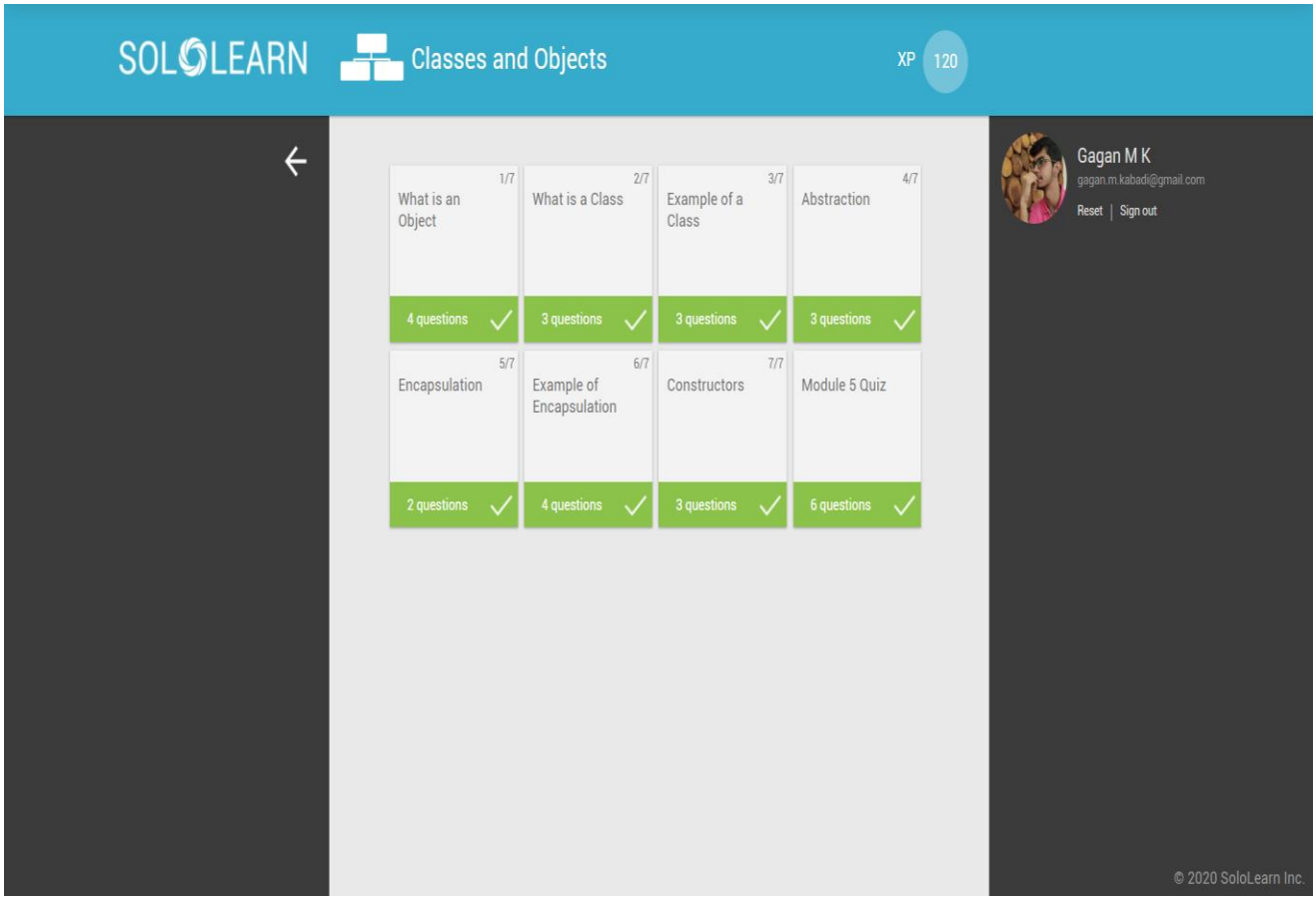
DAILY ASSESSMENT REPORT

Date:	**24 June 2020**	**Name:**	**Gagan M K**
Course:	**C Plus Plus**	**USN:**	**4AL17EC032**
Topic:	• **Classes and Objects** • **More on Classes**	**Semester & Section:**	**6th sem & 'A' sec**
GitHub Repository:	**Alvas-education-foundation/Gagan-Git**		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Classes and Objects:

- **Object Oriented Programming 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 apple is an object; so is a mug. Each has its unique identity. It's possible to have two mugs that look identical, but they are still separate, unique objects.**
- **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).**
- **An object's state is independent of its type; a cup might be full of water, another might be empty.**
- **In the real world, each object behaves in its own way. The car moves, the phone rings, and so on.**
- **The same applies to objects - behavior is specific to the object's type.**
- **So, the following three dimensions describe any object in object oriented programming: identity, attributes, behavior.**
- **In programming, an object is self-contained, with its own identity. It is separate from other objects.**
- **Each object has its own attributes, which describe its current state. Each exhibits its own behavior, which demonstrates what they can do.**
- **In computing, objects aren't always representative of physical items.**
- **For example, a programming object can represent a date, a time, a bank account. A bank account is not tangible; you can't see it or touch it, but it's still a well-defined object - it has its own identity, attributes, and behavior.**
- **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.**
- **Method is another term for a class' behavior. A method is basically a function that belongs to a class.**

- **Begin your class definition with the keyword class. Follow the keyword with the class name and the class body, enclosed in a set of curly braces.**
- **Define all attributes and behavior (or members) in the body of the class, within curly braces.**
- **You can also define an access specifier for members of the class.**
- **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.**
- **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.**
- **It is one of the fundamental building blocks of object oriented programming.**
- **Part of the meaning of the word encapsulation is the idea of "surrounding" an entity, not just to keep what's inside together, but also to protect it.**
- **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.**
- **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 private member cannot be accessed, or even viewed, from outside the class; it can be accessed only from within the class.**
- **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.**

More on Classes:

- **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.**
- **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.**
- **Objects are destroyed when they go out of scope, or whenever the delete expression is applied to a pointer directed at an object of a class.**
- **Since destructors can't take parameters, they also can't be overloaded.**
- **Each class will have just one destructor.**
- **Defining a destructor is not mandatory; if you don't need one, you don't have to define one.**
- **ifndef stands for "if not defined". The first pair of statements tells the program to define the MyClass header file if it has not been defined already. endif ends the condition.**
- **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.**

- **All const variables must be initialized when they're created. In the case of classes, this initialization is done via constructors. If a class is not initialized using a parameterized constructor, a public default constructor must be provided - if no public default constructor is provided, a compiler error will occur.**
- **Once a const class object has been initialized via the constructor, you cannot modify the object's member variables. This includes both directly making changes to public member variables and calling member functions that set the value of member variables.**
- **Only non-const objects can call non-const functions.**
- **A constant object can't call regular functions. Hence, for a constant object to work you need a constant function.**
- **Recall that constants are variables that cannot be changed, and that all const variables must be initialized at time of creation.**
- **In the real world, complex objects are typically built using smaller, simpler objects. For example, a car is assembled using a metal frame, an engine, tires, and a large number of other parts. This process is called composition.**
- **Normally, private members of a class cannot be accessed from outside of that class.**
- **However, 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.**
- **In the example below, someFunc(), which is not a member function of the class, is a friend of MyClass and can access its private members.**
- **Typical use cases of friend functions are operations that are conducted between two different classes accessing private members of both.**
- **Every object in C++ has access to its own address through an important pointer called the this pointer.**
- **You may be wondering why it's necessary to use the this keyword, when you have the option of directly specifying the variable.**
- **Most of the C++ built-in operators can be redefined or overloaded.**
- **Thus, operators can be used with user-defined types as well (for example, allowing you to add two objects together).**

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new[]	delete	delete[]