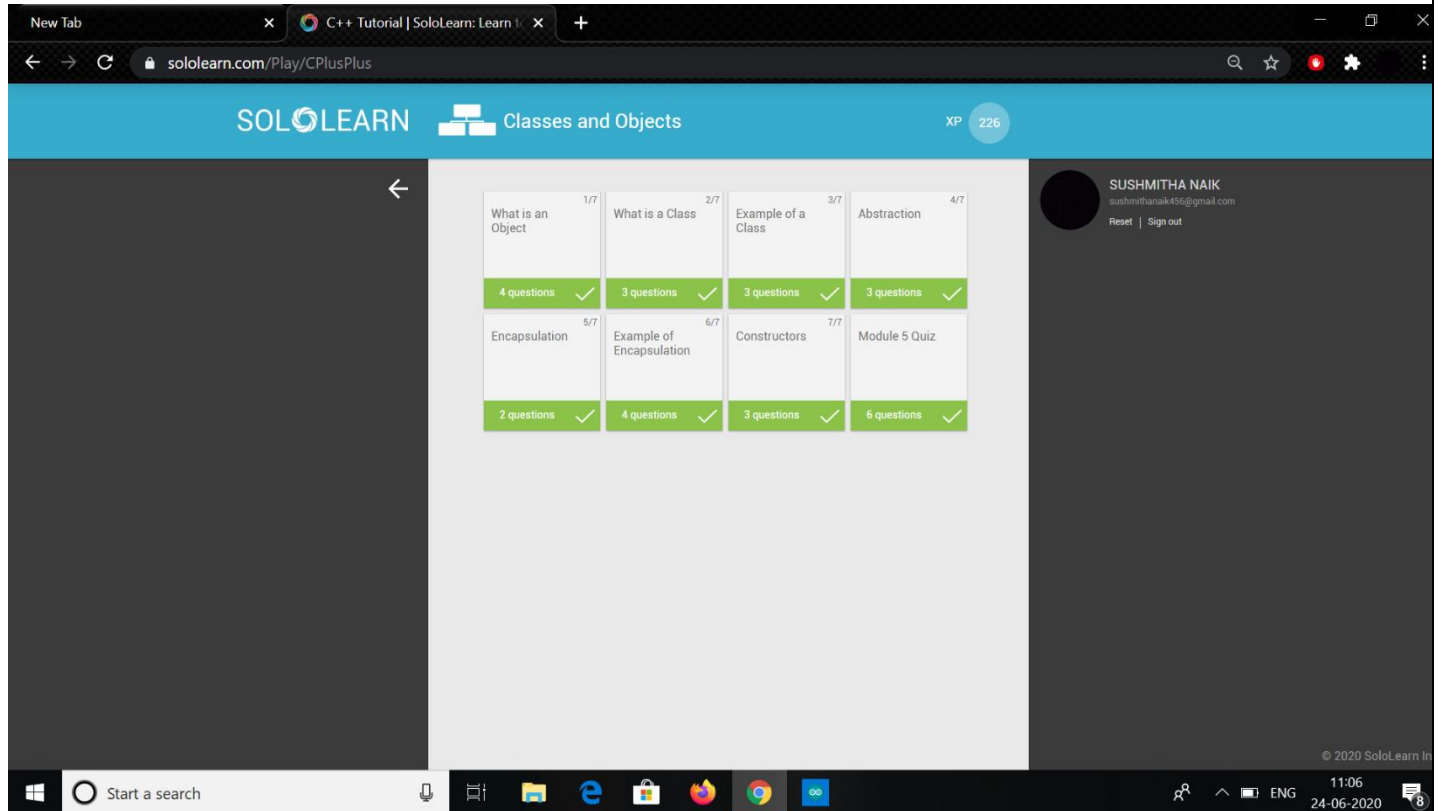


# DAILY ASSESSMENT FORMAT

Date:	24 <sup>th</sup> June 2020	Name:	Sushmitha R Naik
Course:	C++ programming	USN:	4AL17EC090
Topic:	Classes and Objects	Semester & Section:	6 <sup>th</sup> sem & B sec
Github Repository:	Sushmitha_naik		

## FORENOON SESSION DETAILS

### Image of session



## Objects

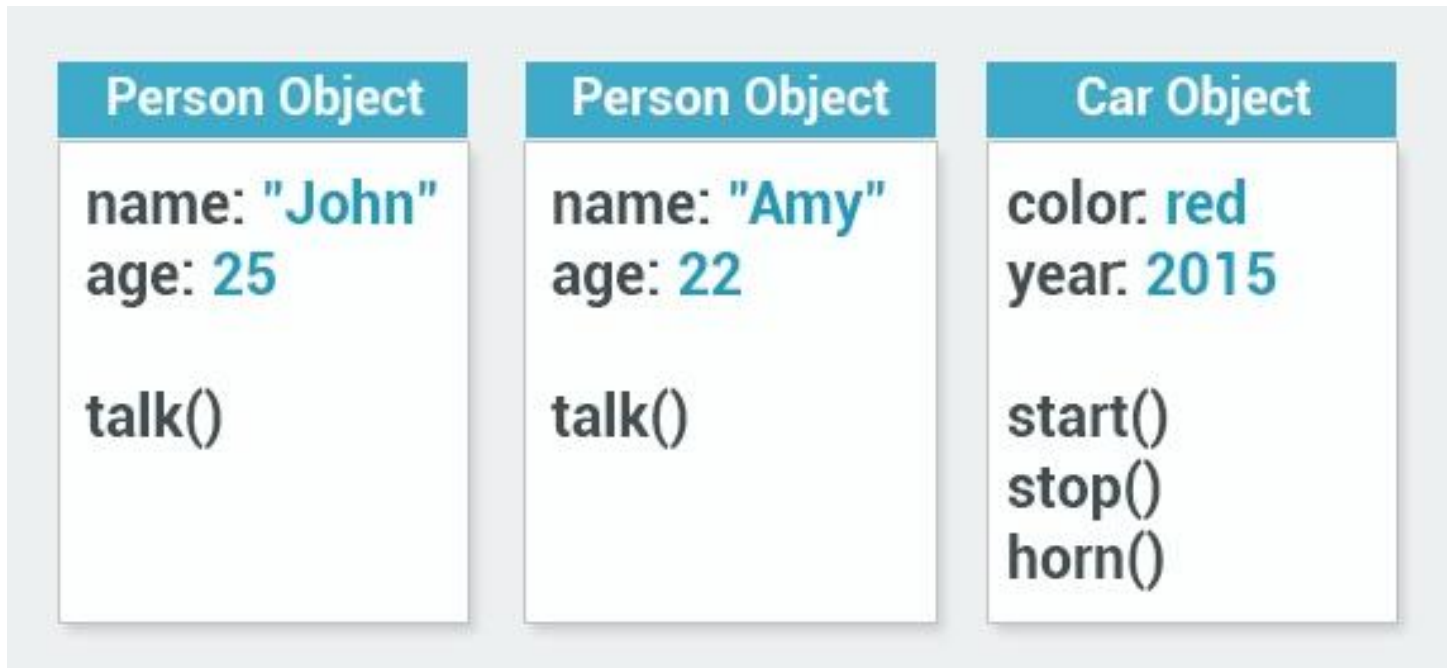
An object might contain other objects but they're still different objects.

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.

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.

## 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.

## Encapsulation:

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. The key principle here is that an object only reveals what the other application components require to effectively run the application. All else is kept out of view.

<b>Date:</b>	<b>24<sup>th</sup> June 2020</b>	<b>Name:</b>	<b>Sushmitha R Naik</b>
<b>Course:</b>	<b>C++ programming</b>	<b>USN:</b>	<b>4AL17EC090</b>
<b>Topic:</b>	<b>More on classes</b>	<b>Semester &amp; Section:</b>	<b>6<sup>th</sup> sem &amp; B sec</b>
<b>Github Repository:</b>	<b>Sushmitha_naik</b>		

### AFTERNOON SESSION DETAILS

#### Image of session

The screenshot displays the 'More On Classes' section of the SOLOLEARN platform. It features a grid of 12 quiz topics, each with a progress indicator (e.g., 1/10, 2/10) and a status bar indicating the number of questions and completion status (e.g., '4 questions' with a green checkmark). The topics include: Separate Files for Classes, Destructors, Selection Operator, Const Objects, Member Initializers, Composition, Part 1, Composition, Part 2, The Friend Keyword, The This Keyword, Operator Overloading, and Module 6 Quiz. A user profile for SUSHMITHA NAIK is visible on the right side of the interface.

## 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.

**Class definition**

Class name:

Arguments:

☐ Has destructor ☐ Has copy ctor

☒ Virtual destructor ☐ Has assignment op.

**Inheritance**

☐ Inherits another class

Ancestor:

Ancestor's include filename:

Scope:

**Member variables**

**Add new:**

☒ Add "Getter" method

☒ Add "Setter" method

☒ Remove prefix:

**Documentation**

☐ Add documentation where appropriate

**File policy**

☒ Add paths to project ☐ Use relative path

☒ Header and implementation file shall be in same folder

Folder:

☐ Header and implementation file shall always be lower case

**Header file**

Folder:

Filename:

☒ Add guard block in header file

Guard block:

**Implementation file**

☒ Generate implementation file

Folder:

Filename:

Header include:

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.

