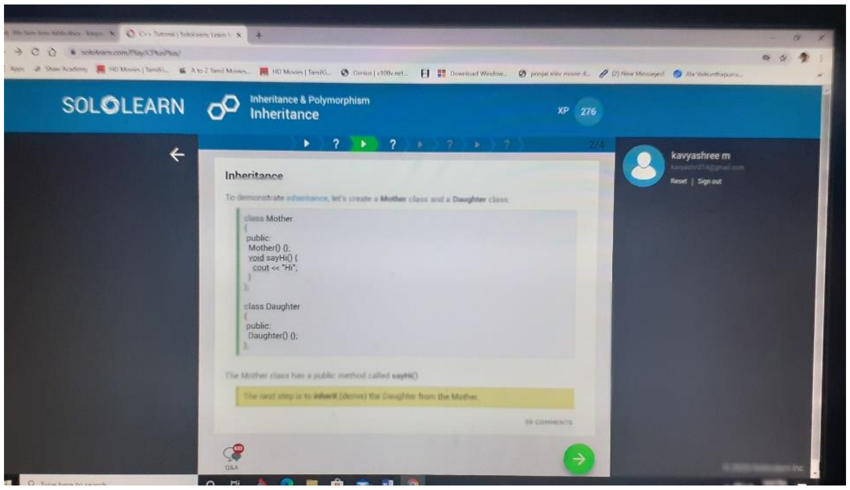
DAILY ASSESSMENT

Date:	**25-6-2020**	**Name:**	**Kavyashree m**
Course:	**C++**	**USN:**	**4al15ec036**
Topic:	**Inheritance and polymorphism Templates, exception and files**	**Semester & Section:**	**8th A**
GitHub Repository:	**kavya**		

FORENOON SESSION DETAILS



Inheritance

To demonstrate inheritance, let's create a Mother class and a Daughter class:

```
class Mother
{
 public:
  Mother() {};
  void sayHi() {
   cout << "Hi";
  }
};

class Daughter
{
 public:
  Daughter() {};
};
```

The Mother class has a public method called sayHi()

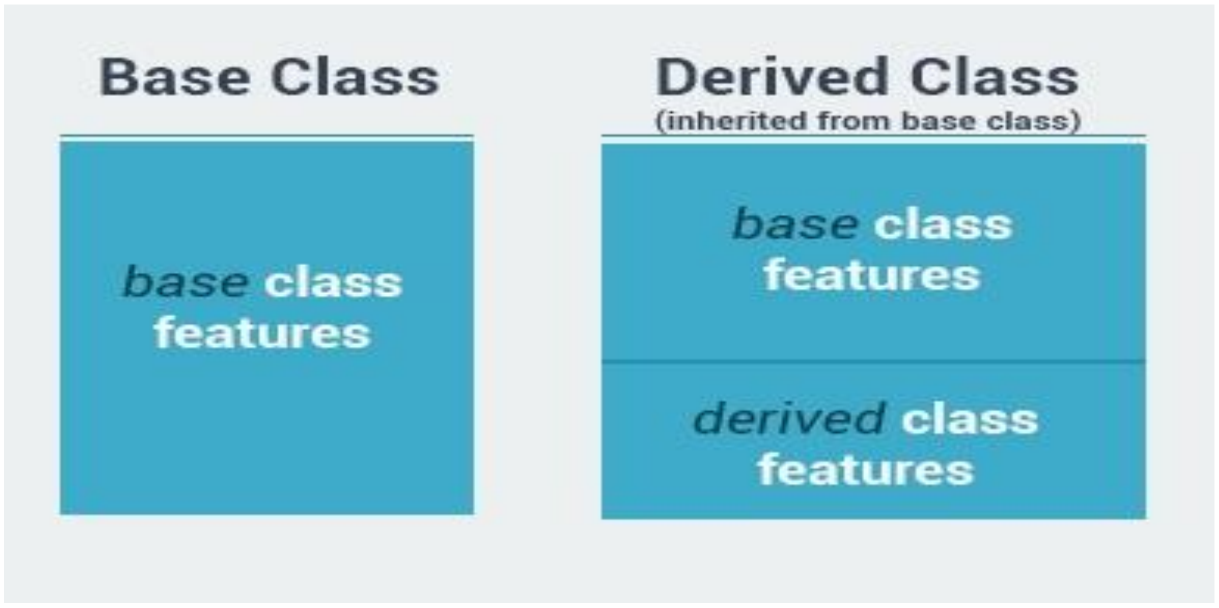
The next step is to inherit (derive) the Daughter from the Mother.

Inheritance

Inheritance is one of the most important concepts of object-oriented programming. Inheritance allows us to define a class based on another class. This facilitates greater ease in creating and maintaining an application.

The class whose properties are inherited by another class is called the Base class. The class which inherits the properties is called the Derived class. For example, the Daughter class (derived) can be inherited from the Mother class (base).

The derived class inherits all feature from the base class, and can have its own additional features.



To demonstrate inheritance, let's create a Mother class and a Daughter class:class Mother

```
{
public:
Mother() {};
void sayHi() {
cout << "Hi";
}
};

class Daughter
{
public:
Daughter() {};
};
```

This syntax derives the Daughter class from the Mother class.class Daughter : public Mother

```
{
public:
Daughter() {};
};
```

The Base class is specified using a colon and an access specifier: public means, that all public members of the base class are public in the derived class.

As all public members of the Mother class become public members for the Daughter class, we can create an object of type Daughter and call the sayHi() function of the Mother class for that object:

```
#include <iostream>
using namespace std;
class Mother
{
public:
Mother() {};
void sayHi() {
cout << "Hi";
}
};

class Daughter: public Mother
```

```
{
public:
Daughter() {};
};

int main() {
Daughter d;
d.sayHi();
}
//Outputs "Hi"
```

Polymorphism in C++

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Real life example of polymorphism, a person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person possess different behavior in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object Oriented Programming.

Virtual functions

A virtual function is a member function which is declared within a base class and is re-defined(Overridden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

- Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
- They are mainly used to achieve Runtime polymorphism
- Functions are declared with a virtual keyword in base class.
- The resolving of function call is done at Run-time.

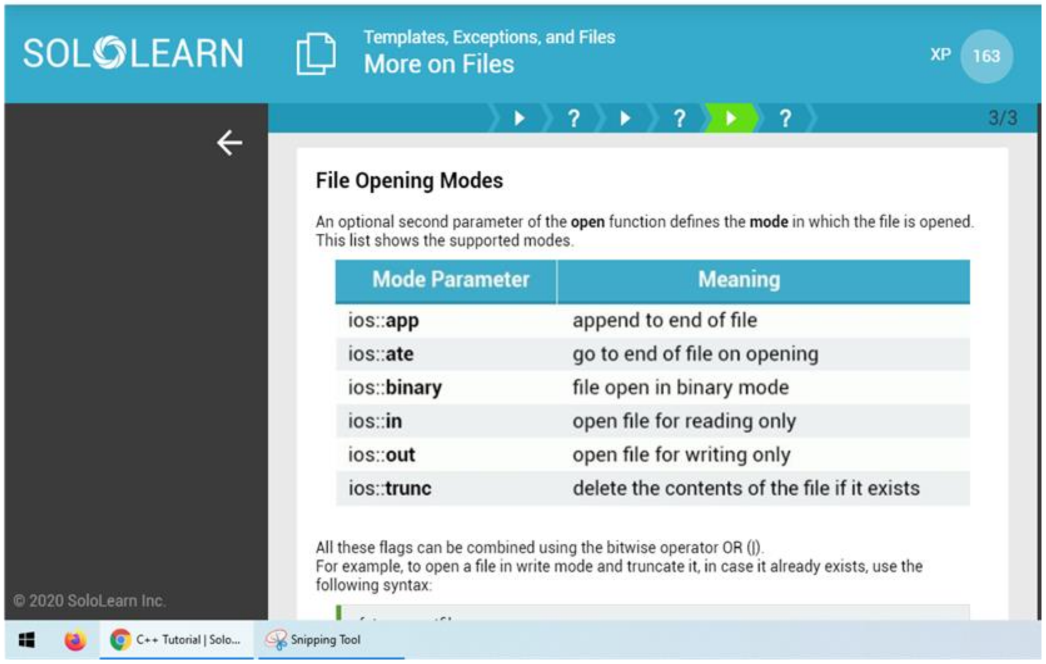
Abstract classes

Abstract Class is a class which contains at least one Pure Virtual function in it. Abstract classes are used to provide an Interface for its sub classes. Classes inheriting an Abstract Class must provide definition to the pure virtual function, otherwise they will also become abstract class.

AFTERNOON SESSION DETAILS

Date:	**25-6-2020**	**Name:**	**Kavyashree m**
Course:	**C++**	**USN:**	**4al15ec036**
Topic:	**Function template ,class template ,template specialization ,exception**	**Semester & Section:**	**8th A**
GitHub Repository:	**kavya**		

Image of session



File Opening Modes

An optional second parameter of the **open** function defines the **mode** in which the file is opened. This list shows the supported modes.

Mode Parameter	Meaning
ios::**app**	append to end of file
ios::**ate**	go to end of file on opening
ios::**binary**	file open in binary mode
ios::**in**	open file for reading only
ios::**out**	open file for writing only
ios::**trunc**	delete the contents of the file if it exists

All these flags can be combined using the bitwise operator OR (|).
For example, to open a file in write mode and truncate it, in case it already exists, use the following syntax:

Function template

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type. In C++ this can be achieved using template parameters.

Class template

Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates. Templates are often used in larger codebase for the purpose of code reusability and flexibility of the programs.

Template Specialization

In many cases when working with templates, you'll write one generic version for all possible data types and leave it at that--every vector may be implemented in exactly the same way. The idea of template specialization is to override the default template implementation to handle a particular type in a different way.

For instance, while most vectors might be implemented as arrays of the given type, you might decide to save some memory and implement vectors of bools as a vector of integers with each bit corresponding to one entry in the vector. So you might have two separate vector classes. The first class would look like this.

```
template <typename T>

class vector

{

    // accessor functions and so forth

    private:
```

```
    T*  vec_data;   //  we'll  store  the  data  as  block  of
dynamically allocated
              // memory
    int length;    // number of elements used
    int vec_size;  // actual size of vec_data
};
```

Exception

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. Exceptions provide a way to transfer control from one part of a program to another.

try: represents a block of code that can throw an exception.

catch: represents a block of code that is executed when a particular exception is thrown.

throw: Used to throw an exception. Also used to list the exceptions that a function throws, but doesn't handle itself.

More on files

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.