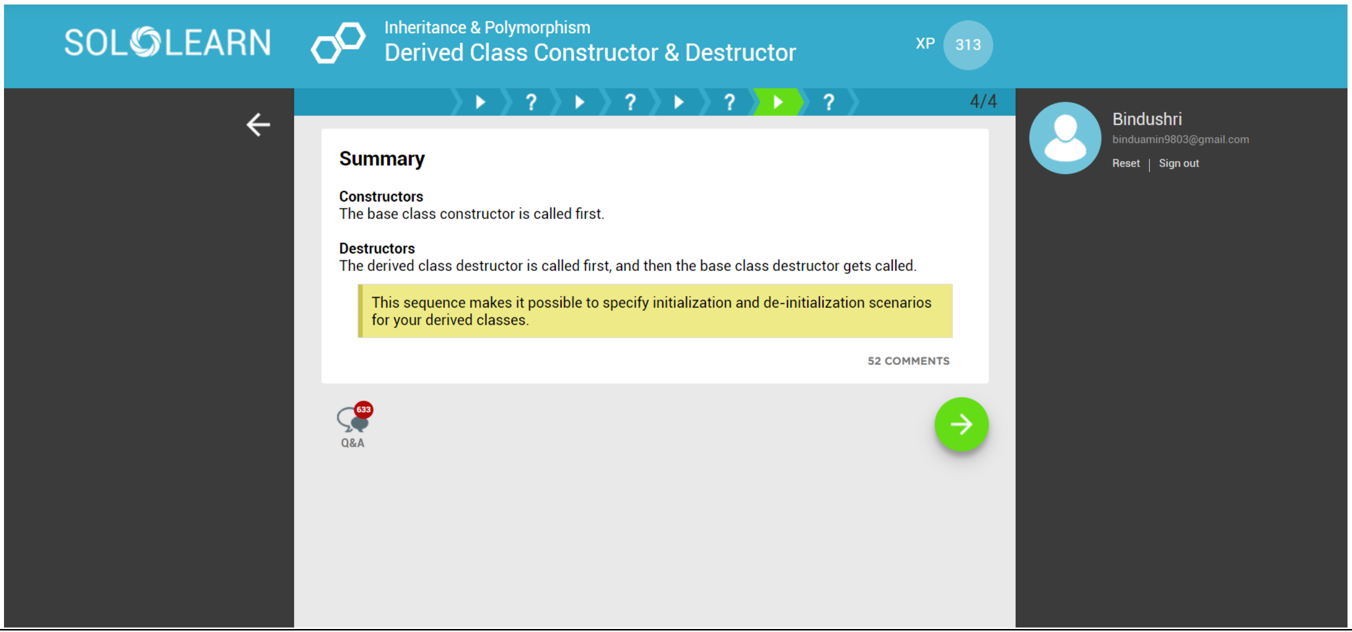
DAILY ASSESSMENT FORMAT

Date:	**25-06-2020**	**Name:**	**BINDUSHRI**
Course:	**C++ programming**	**USN:**	**4AL17EC011**
Topic:	**Inheritance files**		**6th A**
Github Repository:	**Bindushri**		

FORENOON SESSION DETAILS

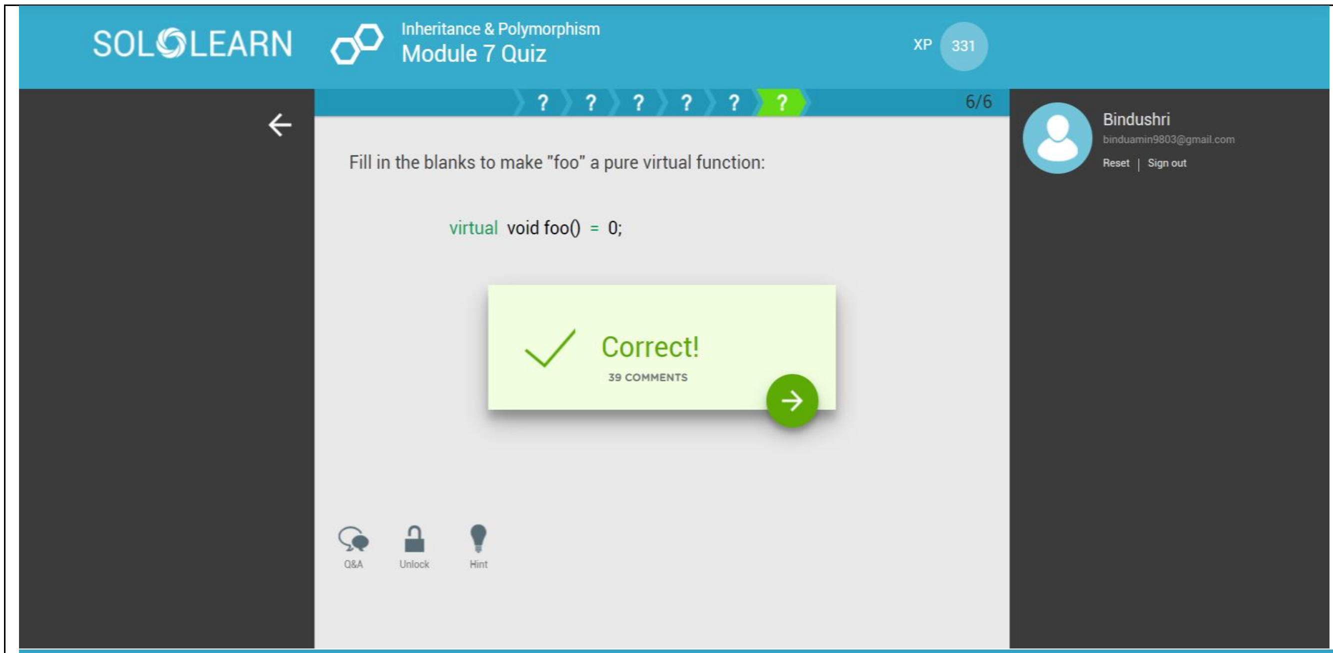


Summary

Constructors
The base class constructor is called first.

Destructors
The derived class destructor is called first, and then the base class destructor gets called.

This sequence makes it possible to specify initialization and de-initialization scenarios for your derived classes.

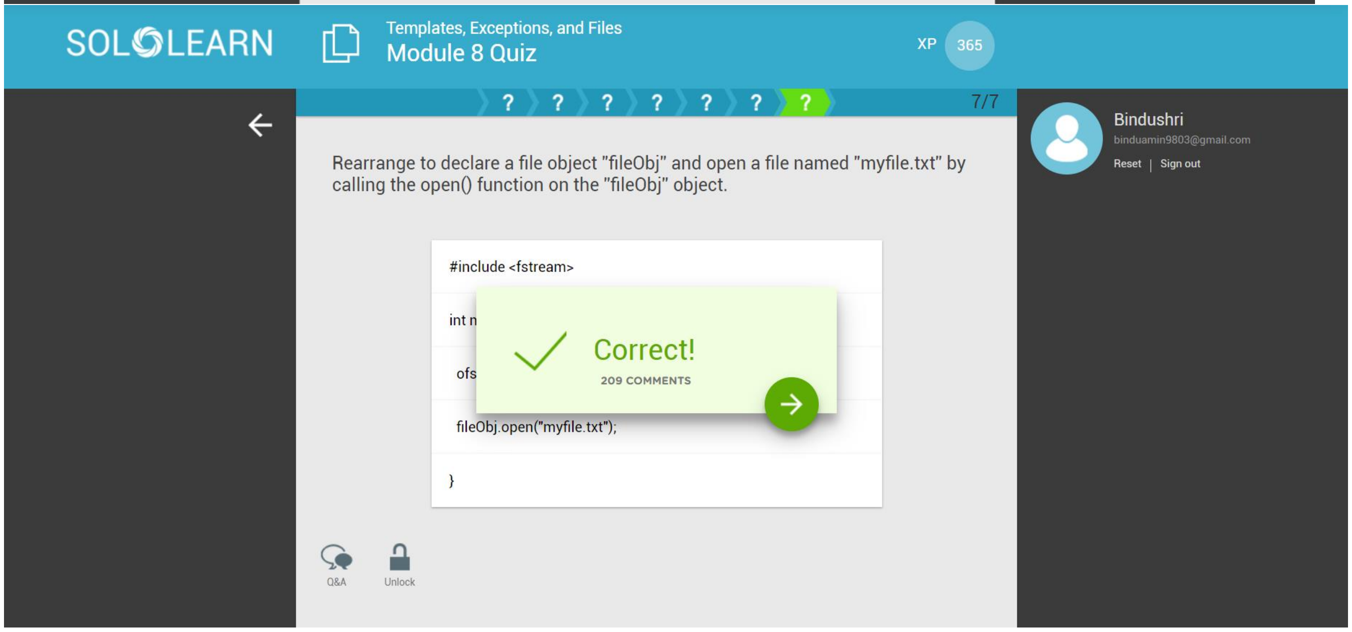


Fill in the blanks to make "foo" a pure virtual function:

```
virtual void foo() = 0;
```

Correct!

39 COMMENTS

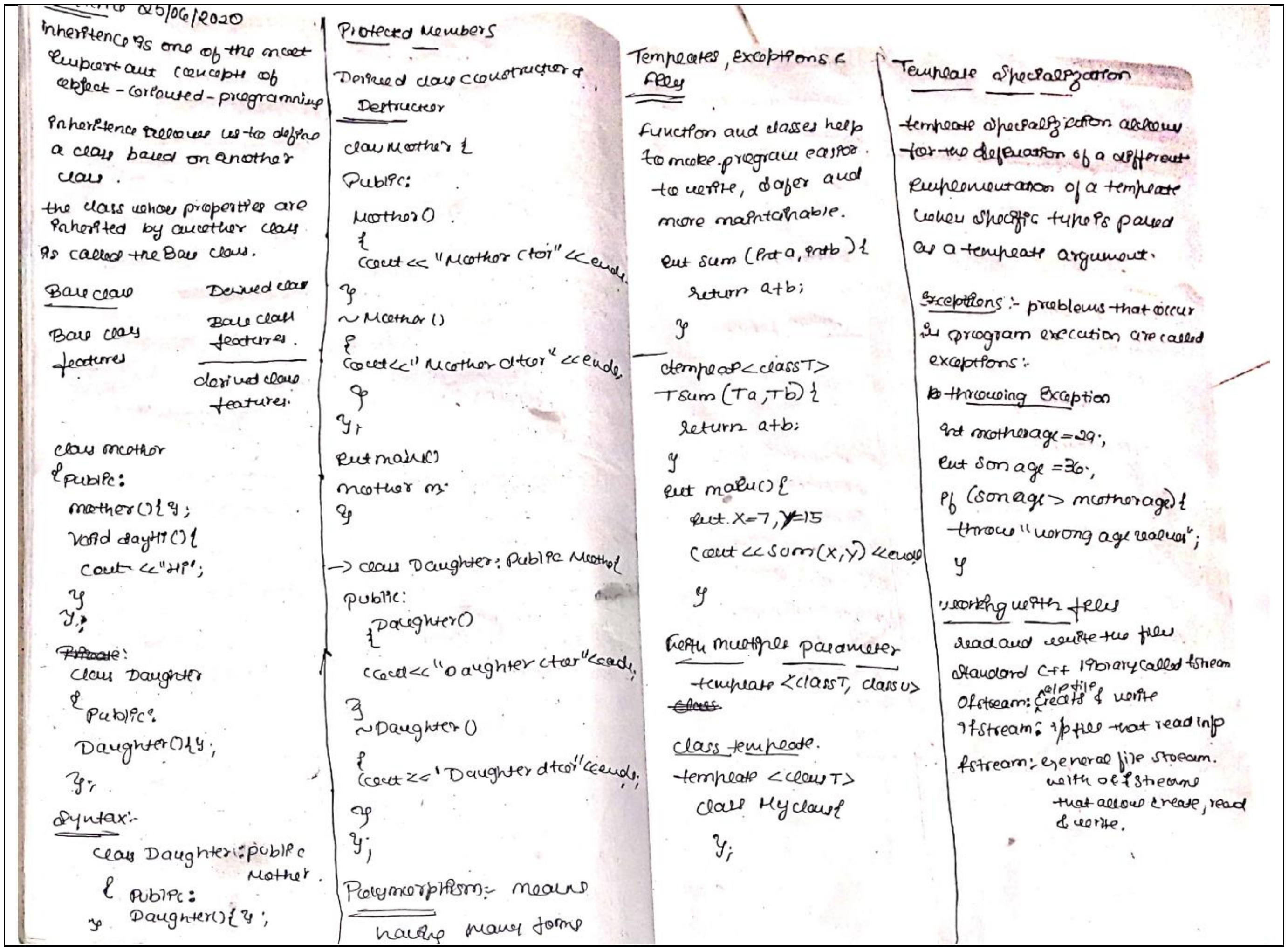


Rearrange to declare a file object "fileObj" and open a file named "myfile.txt" by calling the open() function on the "fileObj" object.

```
#include <fstream>
int n
 ofs
 fileObj.open("myfile.txt");
}
```

Correct!

209 COMMENTS



25/06/2020

Inheritance is one of the most important concepts of object-oriented-programming

Inheritance allows us to define a class based on another class.

the class whose properties are inherited by another class is called the Base class.

Base class	Derived class
Base class features	Base class features.
	derived class features.

```
class mother
{public:
  mother(){};
  void sayHi(){
   Cout <<"Hi";
  }
};
~~Private~~:
 Class Daughter
 {
  Public:
  Daughter(){};
 };
```

Syntax:-

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

Protected Members

Derived class Constructor & Destructor

```
class Mother {
Public:
 Mother()
 {
 Cout << "Mother ctor" << endl;
 }
 ~Mother()
 {
 Cout<<"Mother dtor" << endl;
 }
};
int main()
mother m;
}

-> class Daughter : Public Mother{
 public:
   Daughter()
  {
  Cout<<"Daughter ctor"<<endl;
 }
 ~Daughter()
 {
 Cout << "Daughter dtor"<<endl;
 }
};
```

Polymorphism:- means having many forms

Templates, Exceptions & Files

function and classes help to make program easier to write, safer and more maintainable.

```
int sum (int a, int b){
   return a+b;
 }

template<class T>
T Sum (T a, T b){
 return a+b;
}
int main(){
  int X=7, ~~Y~~=15
  Cout << sum(X,Y) <<endl
 }
```

With multiple parameter

```
 template <class T, class U>
~~class~~
```

class template.

```
template <class T>
 class Myclass{
 };
```

Template specialization

template specialization allows for the definition of a different implementation of a template when specific type is passed as a template argument.

Exceptions :- problems that occur in program execution are called exceptions:

throwing Exception

```
 int motherage=29;
 int son age =36;
 if (sonage> motherage){
  throw "wrong age values";
 }
```

working with files

read and write the files

standard C++ library called fstream

Ofstream: creates o/p file & write

ifstream: i/p file that read info

fstream: general file stream. with of & streams that allow create, read & write.