Programming with C++ - UE19CS208B

Assignment - 2020

Name	SRN	Section
Sumukh Raju Bhat	PES1UG19CS519	H

1) Is it possible to have a constructor in private section of the class? If yes, give programming examples to illustrate the object creation of such a class.

Answer: Yes. We can think of ways where the constructor is defined in private section and using some indirect methods and making use of it by utilizing some basic c++ concepts.

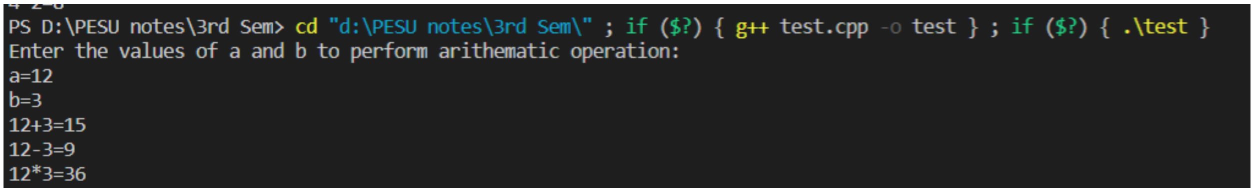
1. By using a static member function (As the call of static member functions do not need object of the class to be created in main()), we return a pointer of type class pointing to a new object of that class. As private members(including constructor) can be accessed inside a class, we can create a new object which interns invokes the private constructor in public section inside the definition of a static member function returning a object pointer. As this is done inside class itself, there won’t be any error. The returned pointer is used in main() to access its members.

Eg:

```
#include<iostream>
using namespace std;

class Arithematic{
    private:
        int a,b;
    //parameterized constructor in private section of the class
    Arithematic(int c_a,int c_b):a(c_a),b(c_b){};
    public:
        void add(){
            cout<<a<<'+'<<b<<'='<<a+b<<endl;
        }
        void subtract(){
            cout<<a<<'-'<<b<<'='<<a-b<<endl;
        }
        void divide(){
            if(b==0){
                cout<<"infinity"<<endl;
            }
            else{
                cout<<a<<'/'<<b<<'='<<a/b<<endl;
            }
        }
        void multiply(){
            cout<<a<<'*'<<b<<'='<<a*b<<endl;
        }

        static Arithematic* returnObject(int a,int b){
            Arithematic *objPointer = new Arithematic(a,b);
            return objPointer;
        }
};

int main()
{
    int a,b;
    cout<<"Enter the values of a and b to perform arithematic operation:"<<endl;
    cout<<"a=";
    cin>>a;
    cout<<"b=";
    cin>>b;
    Arithematic *objPtr = Arithematic::returnObject(a,b);
```

```
            objPtr->add();

            objPtr->subtract();

            objPtr->multiply();

            objPtr->divide();

            return 0;

        }
```

OUTPUT:



```
PS D:\PESU notes\3rd Sem> cd "d:\PESU notes\3rd Sem\" ; if ($?) { g++ test.cpp -o test } ; if ($?) { .\test }
Enter the values of a and b to perform arithematic operation:
a=12
b=3
12+3=15
12-3=9
12*3=36
```

<u>OR</u>

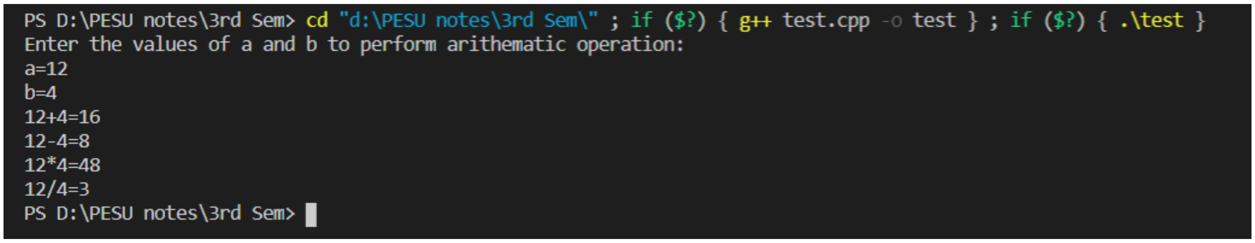
2. By using friend of a class, we can access the private members and hence declare a class object in the friend class and hence invoking even private constructor of it.

Eg:

```
#include <iostream>

using namespace std;

class Arithematic{

    private:

        int a,b;

        //private paramterized constructor

        Arithematic(int c_a,int c_b):a(c_a),b(c_b){};

        friend class friendArithematic;

    public:

        void add(){

            cout<<a<<'+'<<b<<'='<<a+b<<endl;

        }

        void subtract(){

            cout<<a<<'-'<<b<<'='<<a-b<<endl;
        }
        void divide(){
            if(b==0){
                cout<<"infinity"<<endl;
            }
            else{
                cout<<a<<'/'<<b<<'='<<a/b<<endl;
            }
        }
        void multiply(){
            cout<<a<<'*'<<b<<'='<<a*b<<endl;
        }

};
//friend of Arithematic class
class friendArithematic{
    int a,b;
    public:
        friendArithematic(int c_a,int c_b):a(c_a),b(c_b){
            Arithematic obj(a,b);
            obj.add();
            obj.subtract();
            obj.multiply();
            obj.divide();
        }
};
int main(){
    int a,b;
```

```
        cout<<"Enter the values of a and b to perform arithematic operation:"<<endl;

        cout<<"a=";

        cin>>a;

        cout<<"b=";

        cin>>b;

        friendArithematic obj(a,b);

        return 0;

}
```

OUTPUT:



```
PS D:\PESU notes\3rd Sem> cd "d:\PESU notes\3rd Sem\" ; if ($?) { g++ test.cpp -o test } ; if ($?) { .\test }
Enter the values of a and b to perform arithematic operation:
a=12
b=4
12+4=16
12-4=8
12*4=48
12/4=3
PS D:\PESU notes\3rd Sem> 
```

2) How many virtual tables will be created for the following program? Explain your answer.

```
#include <iostream>

class A { public: virtual void f() { } };

class B : public A { };

class C : public B { };
```

Answer:

In theory, we can say that 3 virtual tables are created in the above example (Compiler might create none). One virtual table for class A, one for class B and one for class C.

It is noteable that as virtual function f() is not overridden in any of the classes, So all the virtual tables point to the same copy of virtual function f().