//Template Design

Template Design Pattern

```
#include <iostream>

using namespace std;

class AbstractClass
{
public:
        void templateMethod() {
                primitiveOperation1();
                primitiveOperation2();
                concreteOperation();
                hook();
        }
        virtual ~AbstractClass() {};
        virtual void primitiveOperation1() = 0;
        virtual void primitiveOperation2() = 0;
        void concreteOperation() {
                cout << "Mandatory Operations for all ConcreteClasses" <<
endl;
        }
        virtual void hook() {}
};

class ConcreteClassA : public AbstractClass
{
public:
  void primitiveOperation1() {
    cout << "primitiveOp1 A" << endl;
  }
  void primitiveOperation2() {
    cout << "primitiveOp2 A" << endl;
  }
};

class ConcreteClassB : public AbstractClass
{
public:
  void primitiveOperation1() {
```

```
    cout << "primitiveOp1 B" << endl;
  }
  void primitiveOperation2() {
    cout << "primitiveOp2 B" << endl;
  }
  void hook() {
    cout << "hook() B" << endl;
  }
};

int main()
{
  ConcreteClassA ca;
  ConcreteClassB cb;
  ca.templateMethod();
  cb.templateMethod();

  return 0;
}
```

output

```
primitiveOp1 A
primitiveOp2 A
Mandatory Operations for all ConcreteClasses
primitiveOp1 B
primitiveOp2 B
Mandatory Operations for all ConcreteClasses
hook() B
```