Template Method in C++

Back to **Template Method** description

Template Method design pattern

1. Standardize the skeleton of an algorithm in a base class "template method"
2. Steps requiring peculiar implementations are "placeholders" in base class
3. Derived classes implement placeholder methods

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

class Base
{
    void a()
    {
        cout << "a  ";
    }
    void c()
    {
        cout << "c  ";
    }
    void e()
    {
        cout << "e  ";
    }
    // 2. Steps requiring peculiar implementations are "placeholders" in base class
    virtual void ph1() = 0;
    virtual void ph2() = 0;
  public:
    // 1. Standardize the skeleton of an algorithm in a base class "template method"
    void execute()
    {
        a();
        ph1();
        c();
        ph2();
        e();
    }
};

class One: public Base
{
    // 3. Derived classes implement placeholder methods
     /*virtual*/void ph1()
    {
        cout << "b  ";
    }
     /*virtual*/void ph2()
    {
        cout << "d  ";
    }
};

class Two: public Base
{
     /*virtual*/void ph1()
    {
        cout << "2  ";
    }
     /*virtual*/void ph2()
    {
        cout << "4  ";
    }
};

int main()
{
  Base *array[] =
  {
     &One(), &Two()
  };
  for (int i = 0; i < 2; i++)
  {
    array[i]->execute();
    cout << '\n';
  }
}
```

Output

```
a  b  c  d  e
a  2  c  4  e
```

Read next

Learn more

Code examples

Java	Template Method in Java	
C++	Template Method in C++	Template Method in C++: Before and After
PHP	Template Method in PHP	
Delphi	Template Method in Delphi	

Design Patterns
AntiPatterns
Refactoring
UML

My account
Forum
Contact us
About us

Terms / Privacy policy