컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 51강 - C++의 다형성 기법

FASTCAMPUS
Copyright FASTCAMPUS Corp. All Rights Reserved

Fast campus

학습 목표

C++의 다형성 기법

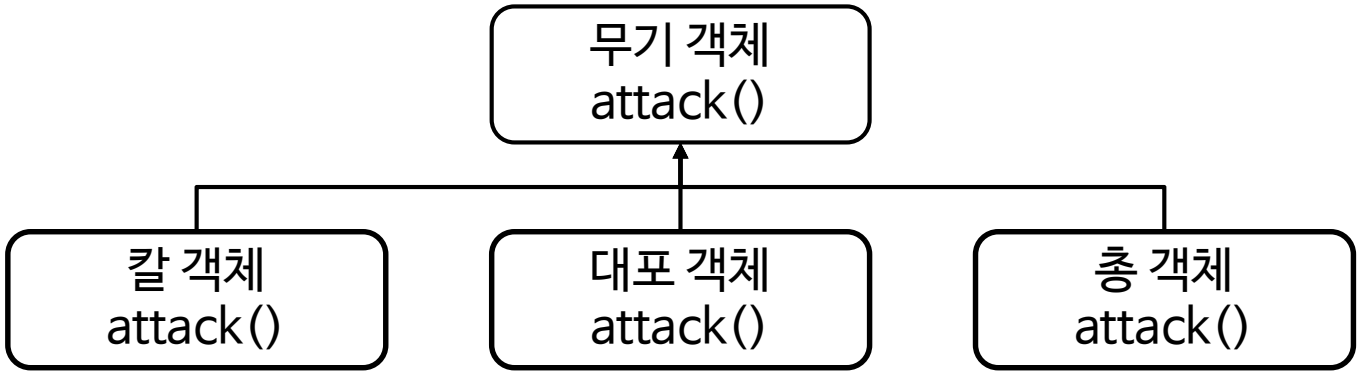
1) C++의 다형성 기법에 대해서 이해하고 이를 바르게 활용할 수 있습니다.

FASTCAMPUS
Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

다형성

다형성(Polymorphism)이란 여러 개의 서로 다른 객체가 동일한 기능을 서로 다른 방법으로 처리할 수 있는 기능을 의미합니다. 예를 들어 게임 프로그램에서 칼, 대포, 총 등의 무기들은 공통적으로 '공격'이라는 동일한 기능을 수행할 수 있습니다.

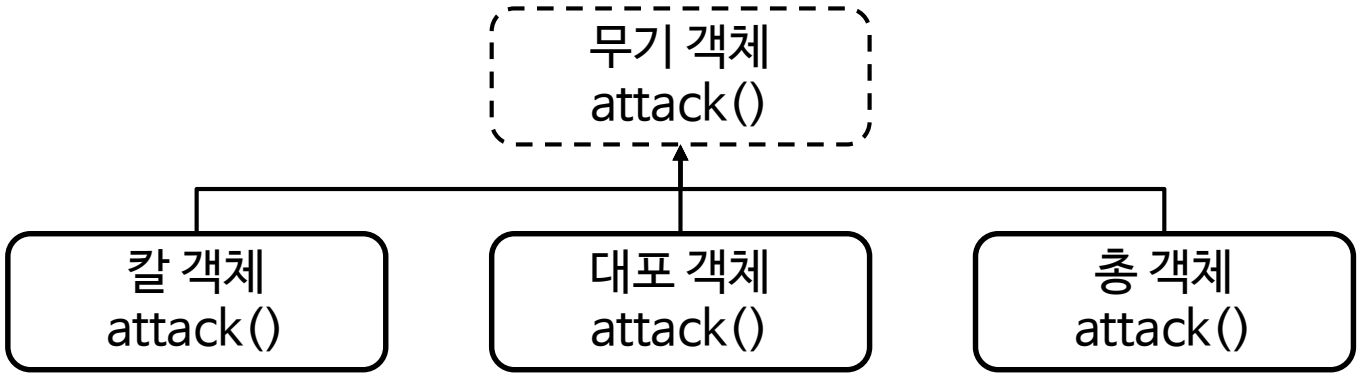


Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

다형성

아래와 같은 구성에서는 무기 객체에서 attack() 함수를 실질적으로 구현할 필요가 없습니다. 이럴 때 무기 객체를 추상 클래스(Abstract Class)로 구현하면 효과적으로 설계를 할 수 있습니다.



Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

자식 클래스에서 오버라이딩의 문제점

자식 클래스에서 멤버 함수를 재정의하여 사용하는 것은 일반적으로 정상적으로 동작합니다. 하지만 포인터 변수로 객체에 접근할 때는 예상치 못한 결과가 발생할 수 있습니다.

C++ 컴파일러는 포인터 변수가 가리키고 있는 변수의 타입을 기준으로 함수를 호출하지 않고, 포인터의 타입을 기준으로 함수를 호출합니다. 따라서 A라는 객체를 가리키는 포인터 변수는 A 객체의 멤버 함수만을 호출할 수 있습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

일반적인 함수의 정적 바인딩

```
#include <iostream>

using namespace std;

class A
{
public:
  void show() { cout << "A 클래스입니다." << endl; }
};

class B : public A
{
  void show() { cout << "B 클래스입니다." << endl; }
};

int main(void)
{
  A* p;
  A a;
  B b;
  p = &a; p->show();
  p = &b; p->show(); // 여전히 A 클래스의 show() 함수를 호출합니다.
  system("pause");
}
```


Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

가상 함수

가상 함수(Virtual Function)란 자식 클래스에서 재정의할 수 있는 멤버 함수입니다. virtual 키워드를 이용해 가상 함수를 선언할 수 있으며 자식 클래스에서 가상 함수를 재정의하면 재정의된 멤버 함수 또한 가상 함수로 분류됩니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

동적 바인딩

C++는 특정한 함수를 호출할 때 해당 함수의 루틴이 기록된 메모리 주소를 알아야 합니다. 특정한 함수를 호출하는 소스코드에서 실제로 함수가 정의된 메모리 공간을 찾기 위해서는 바인딩(Binding) 과정이 필요합니다.

일반적으로 함수의 호출은 컴파일 시기에 고정된 메모리 주소를 이용합니다. 이러한 방식을 정적 바인딩(Static Binding)이라고 말하는데, C++의 일반적인 멤버 함수는 모두 이러한 정적 바인딩을 사용합니다.

다만 가상 함수는 프로그램이 실행될 때 객체를 결정한다는 점에서 컴파일 시간에 객체를 특정할 수 없습니다. 그래서 가상 함수는 실행 시간 때 올바른 함수가 실행될 수 있도록 동적 바인딩(Dynamic Binding)을 사용합니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

가상 함수: 동적 바인딩

```
#include <iostream>

using namespace std;

class A
{
public:
  virtual void show() { cout << "A 클래스입니다." << endl; }
};

class B : public A
{
  virtual void show() { cout << "B 클래스입니다." << endl; }
};

int main(void)
{
  A* p;
  A a;
  B b;
  p = &a; p->show();
  p = &b; p->show();
  system("pause");
}
```


Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

가상 함수

C++ 컴파일러는 가상 함수 테이블(Virtual Function Table)을 이용해 가상 함수를 다루게 됩니다. C++ 컴파일러는 각각의 객체마다 가상 함수 테이블을 가리키는 포인터를 저장하기 위한 멤버를 하나씩 저장합니다.

가상 함수 테이블에는 특정한 클래스의 객체들을 위해 선언된 가상 함수들의 주소가 저장됩니다. 따라서 가상 함수를 호출하면 C++ 프로그램은 가상 함수 테이블에 접근하여 자신이 필요한 함수의 주소를 찾아 호출하게 됩니다.

이러한 과정은 말 그대로 동적 바인딩을 통해 이루어지므로 컴퓨팅 리소스를 소모하게 됩니다. 그래서 C++은 기본적으로 정적 바인딩을 채택하고 있습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

가상 함수

따라서 자식 클래스가 재정의할 가능성이 있는 멤버 함수를 가상 함수로 선언하는 것이 좋습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

가상 클래스의 소멸자

C++에서는 상속 관계가 있으며 메모리 해제를 해야 하는 경우 반드시 부모 클래스의 소멸자를 가상 함수로 선언해야 합니다.

만약 다형성을 이용할 때 소멸자를 가상 함수로 선언하지 않으면 자식 클래스의 소멸자는 호출되지 않고, 부모 클래스의 소멸자만 호출되기 때문에 자식 클래스의 객체는 여전히 정상적으로 해제되지 않습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

순수 가상 함수

C++의 가상 함수는 기본적으로 반드시 재정의할 필요는 없습니다. 하지만 순수 가상 함수(Pure Virtual Function)는 자식 클래스에서 반드시 재정의를 해주어야 하는 함수입니다.

그러므로 일반적으로 순수 가상 함수는 부모 클래스에서 함수 동작의 본체를 정의하지 않습니다. 자식 클래스에서 반드시 이를 정의해야 사용할 수 있습니다.

순수 가상 함수는 '=0' 키워드를 붙여서 선언할 수 있습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

순수 가상 함수

```
#include <iostream>

using namespace std;

class A
{
public:
  virtual void show()=0 { cout << "A 클래스입니다." << endl; }
};

class B : public A
{
  // show() 함수를 재정의하지 않으면 B 클래스의 객체를 사용할 수 없습니다.
};
```


Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

추상 클래스

추상 클래스(Abstract Class)란 하나 이상의 순수 가상 함수를 포함하는 클래스를 의미합니다. 추상 클래스를 활용하면 다형성을 효과적으로 프로그램 상에서 구현할 수 있습니다.

따라서 자식 클래스는 추상 클래스를 상속 받은 이후에 반드시 순수 가상 함수를 모두 오버라이딩 해야 비로소 해당 객체를 사용할 수 있습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

C++의 다형성 기법

C++의 다형성 기법

1) C++에서 추상 클래스를 활용하여 다형성을 효과적으로 구현할 수 있습니다.

Copyright FASTCAMPUS Corp. All Rights Reserved