




고급프로그래밍

상속

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory
Kwangwoon University
(02-940-5625, mhjeong@kw.ac.kr)

Schedule

week	Topics		Homework	Quiz
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++ 	C++ 프로그래밍의 기본(3, 3/12), 클래스와 객체(4, 3/14)	1	1
3		휴강(3/17), 객체생성과 사용(5, 3/19)	2	
4		함수와 참조(6, 3/26), 복사 생성자와 함수중복(7. 3/28)	3	2, 3
5		static friend 연산자중복(8, 4/2), 연산자중복 상속(9, 4/4)	4	4
6		상속(10, 4/9), 템플릿과 STL, 표준 입출력	5	5
7		파일 입출력		
8	중간고사			
9	C++	예외처리 및 C 사용, 람다식	6	6
10		멀티스레딩	7	7
11		멀티스레딩, 고급문법	8	8
12		고급문법	9	9
13	병렬 프로그래밍	병렬프로그래밍		
14		병렬프로그래밍		
15	기말고사			

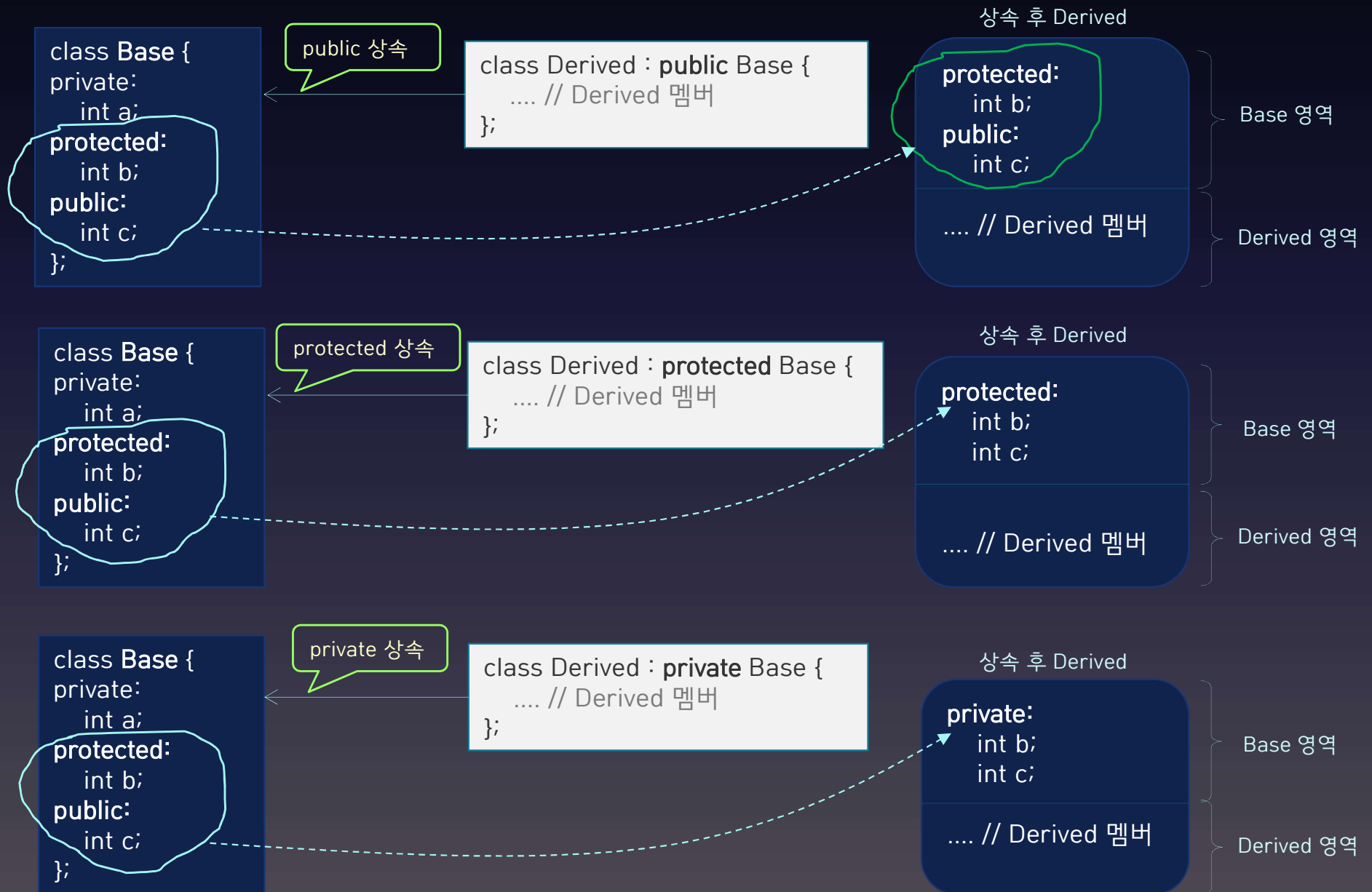
오늘의 학습내용

- 상속 - 계속

상속 지정

- 상속 선언 시 `public`, `private`, `protected`의 3가지 중 하나 지정
- 기본 클래스의 멤버의 접근 속성을 어떻게 계승할지 지정
 - `public` - 기본 클래스의 `protected`, `public` 멤버 속성을 그대로 계승
 - `private` - 기본 클래스의 `protected`, `public` 멤버를 `private`으로 계승
 - `protected` - 기본 클래스의 `protected`, `public` 멤버를 `protected`로 계승

접근 지정 속성 변화



예제 - protected 상속 사례

- 다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : protected Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() { cout << b; }
};
```

```
int main() {
    Derived x;
    x.a = 5;           // ①
    x.setA(10);        // ②
    x.showA();         // ③
    x.b = 10;          // ④
    x.setB(10);        // ⑤
    x.showB();         // ⑥
}
```

예제 - protected 상속 사례

- 다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : protected Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() { cout << b; }
};
```

```
int main() {
    Derived x;
    x.a = 5;           // ①
    x.setA(10);        // ②
    x.showA();         // ③
    x.b = 10;          // ④
    x.setB(10);        // ⑤
    x.showB();         // ⑥
}
```

컴파일 오류

①, ②, ③, ④, ⑤, ⑥

예제 - 상속이 중첩될 때 접근 지정

- 다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : private Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() {
        setA(5);           // ①
        showA();           // ②
        cout << b;
    }
};
```

```
class GrandDerived : private Derived
{
    int c;
protected:
    void setAB(int x)
    {
        setA(x);           // ③
        showA();           // ④
        setB(x);           // ⑤
    }
};
```


예제 - 상속이 중첩될 때 접근 지정

- 다음에서 컴파일 오류가 발생하는 부분을 찾아라.

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

class Base {
    int a;
protected:
    void setA(int a) { this->a = a; }
public:
    void showA() { cout << a; }
};

class Derived : private Base {
    int b;
protected:
    void setB(int b) { this->b = b; }
public:
    void showB() {
        setA(5);           // ①
        showA();           // ②
        cout << b;
    }
};
```

```
class GrandDerived : private Derived
{
    int c;
protected:
    void setAB(int x)
    {
        setA(x);           // ③
        showA();           // ④
        setB(x);           // ⑤
    }
};
```

컴파일 오류

③, ④

실습

- 아래와 같은 코드에서 `Circle`을 상속받은 `NamedCircle`클래스를 작성하시오

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

class Circle {
    int radius;
public:
    Circle(int radius=0) {
        this->radius = radius;
    }
    int    getRadius() { return radius; }
    void   setRadius(int radius) { this->radius = radius; }
    double getArea() { return 3.14*radius*radius; }
};

int main() {
    NamedCircle waffle(3, "waffle"); // 반지름 3, 이름 waffle
    waffle.show();
}
```

다중 상속

```
class MP3 {  
public:  
    void play();  
    void stop();  
};
```

```
class MobilePhone {  
public:  
    bool sendCall();  
    bool receiveCall();  
    bool sendSMS();  
    bool receiveSMS();  
};
```

상속받고자 하는 기본 클래스를 나열한다.

다중 상속 선언

```
class MusicPhone : public MP3, public MobilePhone { // 다중 상속 선언  
public:  
    void dial();  
};
```

다중 상속 활용

```
void MusicPhone::dial( ) {  
    play();    // mp3 음악을 연주시키고  
    sendCall(); // 전화를 건다.  
}
```

MP3::play() 호출

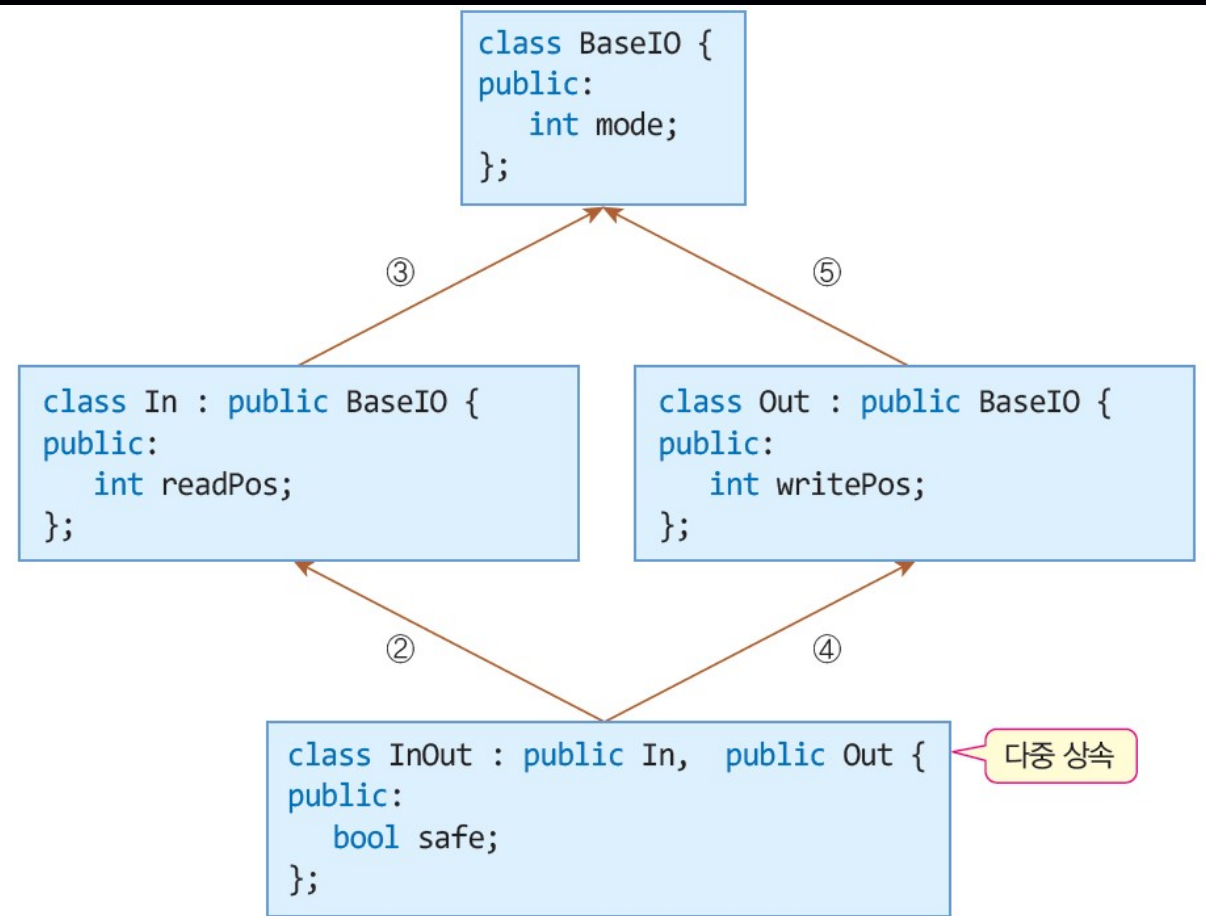
MobilePhone::sendCall() 호출

다중 상속 활용

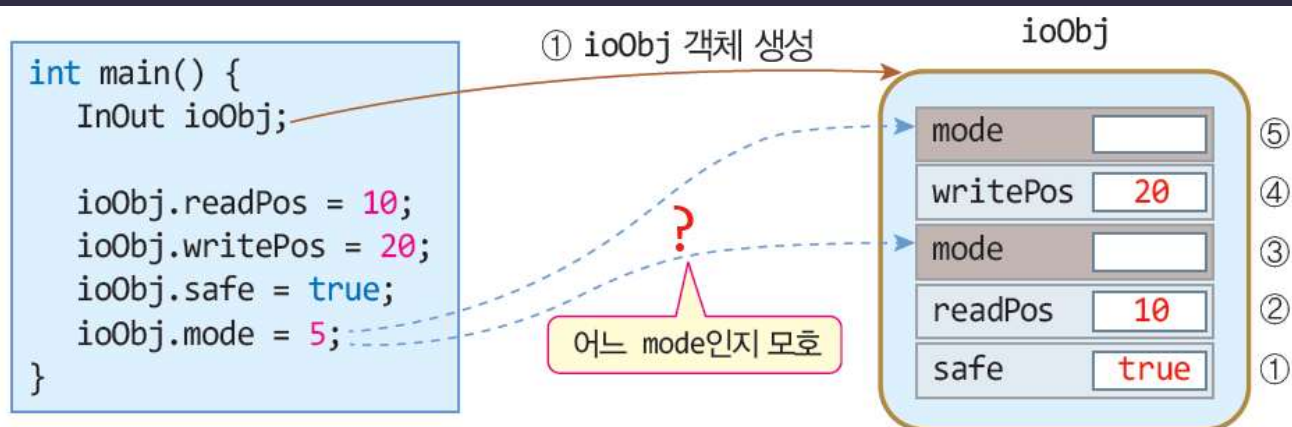
```
int main( ) {  
    MusicPhone hanPhone;  
    hanPhone.play();    // MP3의 멤버 play() 호출  
    hanPhone.sendSMS(); // MobilePhone의 멤버 sendSMS() 호출  
}
```

다중 상속의 문제점

- Base의 멤버가 이중으로 객체에 삽입되는 문제점
- 동일한 x를 접근하는 프로그램이 서로 다른 x에 접근하는 결과를 낳게 되어 잘못된 실행 오류가 발생



(a) 클래스 상속 관계



(b) ioObj 객체 생성 과정 및 객체 내부

가상 상속

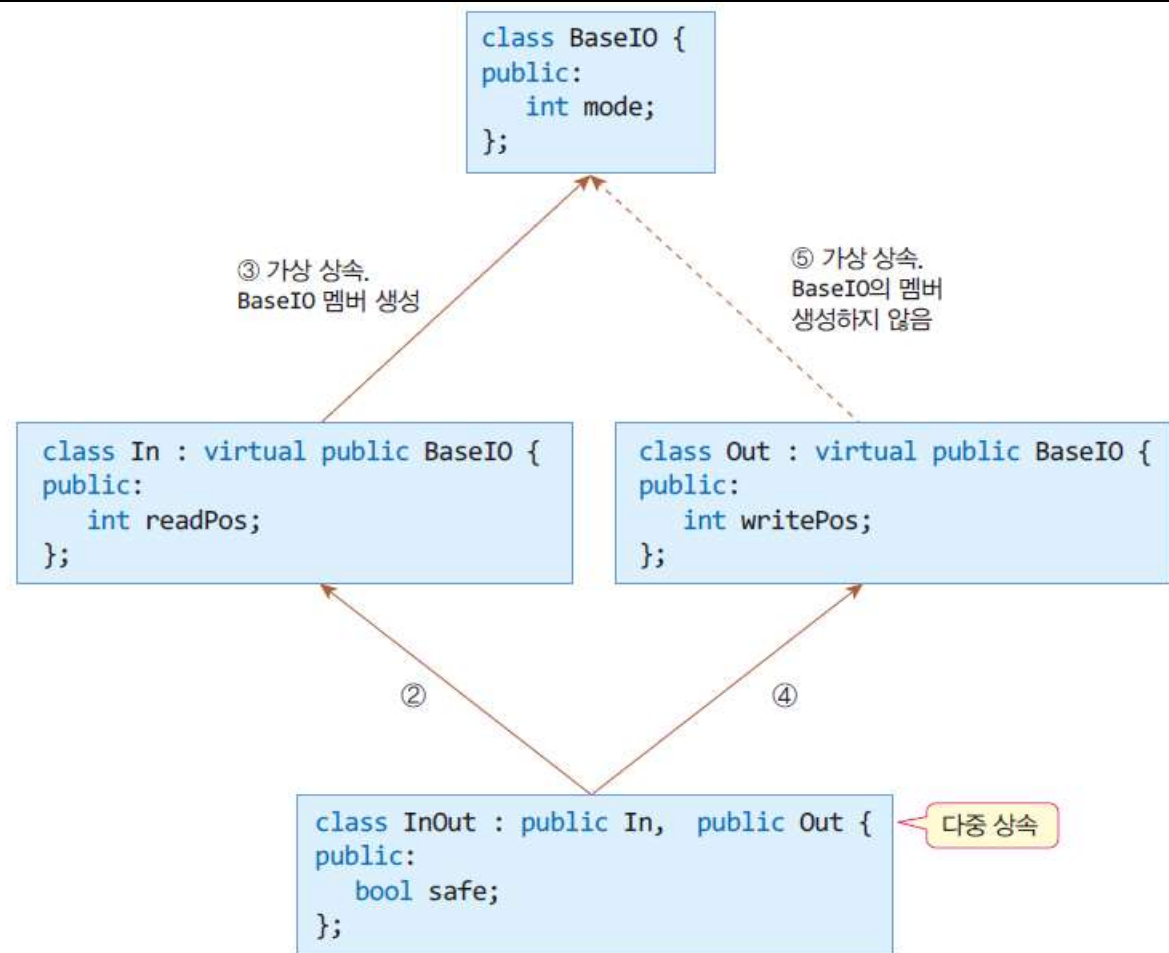
- 다중 상속으로 인한 기본 클래스 멤버의 중복 상속 해결
- 가상 상속
 - 파생 클래스의 선언문에서 기본 클래스 앞에 **virtual**로 선언
 - 파생 클래스의 객체가 생성될 때 기본 클래스의 멤버는 오직 한 번만 생성
 - 기본 클래스의 멤버가 중복하여 생성되는 것을 방지

```
class In : virtual public BaseIO { // In 클래스는 BaseIO 클래스를 가상 상속함
...
};
```

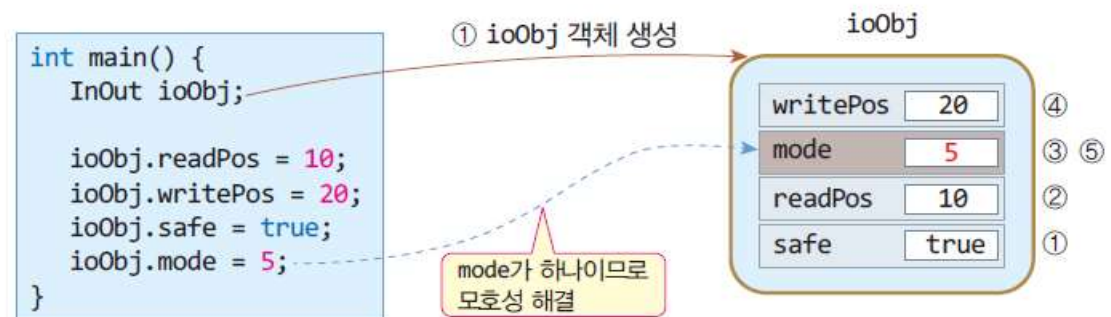
```
class Out : virtual public BaseIO { // Out 클래스는 BaseIO 클래스를 가상 상속함
...
};
```

가상 상속

■ 다중상속의 모호성



(a) 기본 클래스를 가상 상속 받는 클래스 상속 관계



(b) 가상 기본 클래스를 가진 경우, ioObj 객체 생성 과정 및 객체 내부

실습 - 다중 상속

- Adder와 Subtractor를 다중 상속받는 Calculator 클래스를 작성하라.

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

```
class Adder {
protected:
    int add(int a, int b) { return a+b; }
};
```

```
class Subtractor {
protected:
    int minus(int a, int b) { return a-b; }
};
```

```
int main() {
    Calculator handCalculator;
    cout << "2 + 4 = "
        << handCalculator.calc('+', 2, 4)
    << endl;
    cout << "100 - 8 = "
        << handCalculator.calc('-', 100, 8)
    << endl;
}
```

```
2 + 4 = 6
100 - 8 = 92
```

