cpp_04_클래스와객체

□ 개념 확인 학습

1. 다음 질문에 O, X로 답하세요.

A. 소멸자는 여러 번 구현할 수 있다. ( )

B. 생성자는 반환값이 없다. ( )

C. 프로그램이 종료하면 객체가 생성된 순서로 소멸된다. ( )

D. private 접근 권한은 생성자에 사용할 수 없다. ( )

E. 인라인 함수는 inline 키워드로 선언된 함수이다. ( )

F. 생성자는 객체 생성 시 한 번만 호출할 수 있다. ( )

G. 소멸자는 매개변수를 가질 수 없다. ( )

H. 생성자와 소멸자의 실행 순서는 동일하다. ( )

I. 인라인 선언은 크기가 큰 함수의 경우 효과적이다. ( )

J. 인라인 함수를 사용하면 전체 프로그램의 크기가 작아져서 효과적이다. ( )

2. 빈 괄호를 채워 넣으세요.

A. ( )란 객체를 만들기 위해 정의된 설계도이다.

B. C++에서 클래스는 클래스 선언부와 클래스 ( )로 구성된다.

C. 클래스 선언 시 멤버에 대한 접근 권한이 생략되면 디폴트는 ( ) 이다.

D. ( )는 객체가 생성되는 시점에서 자동으로 호출되는 멤버함수이다.

E. 생성자가 선언되어 있지 않으면 ( )에 의해 기본 생성자가 자동으로 생성된다.

F. 클래스 멤버의 디폴트 접근 권한은 ( )이며 구조체 디폴트 접근 권한은 ( )이다.

G. 한정자 ( )를 사용하여 멤버를 상수화 할 수 있다.

H. ( )는 함수 호출에 따른 오버헤드를 줄이기 위해 함수를 호출하는 곳에 함수의 코드를 삽입한다.

I. 함수를 인라인으로 하려면 키워드 ( )을 사용한다.

J. ( ) 키워드는 상수화 된 대상에 대한 쓰기 작업을 허용하기 위한 목적으로 사용한다.

3. 다음과 같은 생성자와 소멸자가 있다고 할 때 어떤 것이 디폴트 생성자인지, 소멸자인지, 복사 생성자인지 구분하세요.

```
Account ();
~Account ();
Account (const Account & acc);
```

4. 다음과 같이 Rectangle 클래스의 생성자를 선언하면 오류가 발생합니다. 이유를 설명하세요.

```
int Rectangle (int length, int height);
```

5. 다음과 같이 Rectangle 클래스의 소멸자를 선언하면 발생합니다. 이유를 설명하세요.

```
int ~Rectangle (int var);
```

6. 다음과 같은 생성자를 초기화 리스트를 사용하는 코드로 변경하세요.

```
Rectangle :: Rectangle (int len, int wid) {
    length = len;
    height = wid;
}
```

7. 다음과 같은 클래스 정의가 있을 때 객체 생성 코드 중 오류가 발생하는 라인을 제시하세요.

```
class Sample {
private :
    int x;
public:
    int getX() const;
};

int main() {
    Sample s1(4);
    Sample s2();
    Sample s3{};
}
```

8. value가 클래스의 데이터 멤버라고 할 때, 다음 클래스의 멤버 함수 정의에서 발생하는 오류를 수정 하세요.

```
Member::int getValue() {
  return value;
}
```

9. 다음과 같은 클래스 선언에 대하여 생성자와 접근자 함수를 구현 하세요

```
class Hallym{
   string dept;
public:
   Hallym(string d);  //매개변수값을 멤버 변수로 초기화
   string getDept();  //멤버 변수 값 반환
}
```

10. 다음 클래스 선언에서 오류가 발생하는 코드를 수정하세요.

```
class Object {
   double x;
public;
   double const getX();
}
```

□ 응용 프로그래밍

11. 아래의 main()이 실행화면처럼 동작하도록 Person 클래스를 선언하세요.

```
int main() {
   Person baby;
   Person child("benny", 10);
   cout << "baby name = " << baby.getName() << endl;
   cout << "child name = " << child.getName() << endl;
}
```

```
생성자 수행 Anonymous,0
생성자 수행 benny,10
baby name = Anonymous
child name = benny
소멸자 수행 benny
소멸자 수행 Anonymous
```

```
>> class Person
 - 데이터 멤버 : private string name, private int age
 - default 생성자는 initializer를 사용한 매개변수가 있는 생성자 호출
 - 소멸자
 - 접근자 멤버 함수 : getName(), getAge()
```

12. 제시된 main() 함수를 참고하여 실행 결과와 같이 동작하도록 계좌 관리를 위한 클래스 Account를 구현하세요. 단, 클래스 구현과 선언은 분리하여 작성합니다.

```
PS C:₩yanges₩lecture₩lecture_src₩cpp> g++ account.h account.cpp account_main.cpp
PS C:₩yanges₩lecture₩lecture_src₩cpp> ./a
-----------------------------------------
menu : 1. 입금, 2. 출금  3. 조회  4. 종료 >> 1
입금액 >> 1000
C++의 입금 액은 1000
C++의 잔액은 1000
-----------------------------------------
menu : 1. 입금, 2. 출금  3. 조회  4. 종료 >> 2
출금액 >> 300
C++의 출금 액은 300
```

```
C++의 잔액은 700
-----------------------------------------
menu : 1. 입금, 2. 출금  3. 조회  4. 종료 >> 3
C++의 잔액은 700
-----------------------------------------
menu : 1. 입금, 2. 출금  3. 조회  4. 종료 >> 4
C++: 객체 소멸
PS C:₩yanges₩lecture₩lecture_src₩cpp>
```

```
//
//account_main.cpp
//
#include <iostream>
#include "account.h"

using namespace std;
using Bank::Account;
using Bank::MENU;

int main() {
    Account a("C++", 0);
    int menu, money;

    do {
        cout << "-----------------------------------------" << endl;
        cout << "menu : 1. 입금, 2. 출금  3. 조회  4. 종료 >> ";
        cin >> menu;

        if(menu == static_cast<int>(MENU::QUIT))
            break;

        switch (menu) {
        case (static_cast<int>(MENU::DEPOSIT)):
            cout << "입금액 >> ";
            cin >> money;
            a.deposit(money);
            cout << a.getOwner() << "의 입금 액은 " << money << endl;
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
            break;
        case (static_cast<int>(MENU::WITHDRAW)):
            cout << "출금액 >> ";
            cin >> money;
            cout << a.getOwner() << "의 출금 액은 " << a.withdraw(money) << endl;
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
            break;
        case (static_cast<int>(MENU::CHECK)):
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
        }
    } while (true);

    return 0;
}
```

13. 제시된 main() 함수를 참고하여 실행 결과와 같이 동작하도록 프로그램을 작성하세요.

```
- 큰 값과 작은 값 범위 내의 난수 생성
- 객체 생성 시 난수의 범위는 매개변수로 전달 (매개변수 있는 생성자 필요).
- 난수 생성 결과를 출력하는 멤버 함수 print() 작성 (변경 작업이 필요 없으므로 const)

- 클래스 구현과 선언은 분리하여 작성 : randint.h, randint.cpp, randint_main.cpp

//C++ 난수 생성 방법
#include <random>
random_device rd;  //시드값을 얻기 위한 random_device 생성
mt19937 gen(rd()); //random_device를 통해 난수 생성 엔진 초기화
uniform_int_distribution<int> dis(low, high);    //low~high 사이의 난수 및 분포 정의
Int value = dis(gen);      //난수 엔진을 전달하여 범위 내 생성된 난수를 value에 저장
```

```
PS C:₩yanges₩lecture₩lecture_src₩cpp> g++ randint.h randint.cpp randint_main.cpp
PS C:₩yanges₩lecture₩lecture_src₩cpp> ./a
Random number between 100 and 200 : 156
Random number between 400 and 600 : 512
Random number between 1500 and 2000 : 1781
r3객체 소멸
r2객체 소멸
r1객체 소멸
PS C:₩yanges₩lecture₩lecture_src₩cpp>
```

```
//
//randint.h
//
#ifndef RANDINT_H
#define RANDINT_H
#include <string>
using namespace std;

class RandInt {
private:
   string objname;
   int low, high, rannum;
public:
    RandInt(int low, int high, string cn); //생성자
    ~RandInt();  //소멸자
    RandInt(const RandInt& random) = delete;  //복사 생성자를 생성하지 않음
    void print() const;  //범위, 난수 출력
};
#endif
```

```
//
//randint.cpp 구현
//
```

```
//
//randint_main.cpp
//
```

```
#include "randint.h"

int main() {
   RandInt r1{ 100, 200, "r1" };
   r1.print();

   RandInt r2(400, 600, "r2");
   r2.print();

   RandInt r3(1500, 2000, "r3");
   r3.print();
   return 0;
}
```