C++ / JAVA 프로그래밍

10주차 실습

클래스와 객체 기초

인스턴스의 개념

- 지금까지 수업을 진행하면서 다양한 데이터 타입을 이용해 변수를 만들어 왔음
  - int a;
  - double b;

- 여기에 클래스와 객체의 개념을 적용해보면…
  - 클래스를 이용해 데이터 타입을 만들 수 있음
  - 이렇게 만든 데이터 타입 형식으로 변수를 선언한 것이 객체(인스턴스)

- 클래스는 정보를 추상화한 것이고, 인스턴스는 클래스를 실체화 한 것이다?
  - 클래스는 어떤 정보를 가지고 있는지, 어떤 동작을 할 수 있는지에 대한 틀을 서술한 것
  - 인스턴스는 정보를 구체적으로 채워 넣은 것
  - 하나의 클래스를 이용해 다양한 인스턴스를 생성할 수 있음

인스턴스는 속성과 행위를 갖는다?

- 속성과 행위란?
  - 속성 (attribute): 인스턴스가 가지는 특징을 의미
  - 행위 (behavior): 어떤 인스턴스가 스스로 할 수 있는 작업 또는 연산

- C++은 클래스(class)라는 구문을 사용하여 타입(사용자 정의 자료형)을 생성
  - 클래스를 기반으로 만든 변수를 인스턴스 또는 객체라 부름
  - 객체 지향 프로그래밍에서 인스턴스의 속성과 행위는 데이터 멤버와 멤버 함수로 생성

- 데이터 멤버와 멤버 함수란?
  - 데이터 멤버: 속성을 표현하기 위한 변수
  - 멤버 함수: 어떤 행위를 할 수 있는 기능의 모임

객체 지향 패러다임의 구성 요소

- 객체 지향 패러다임에는 다음 3가지 구성요소가 필요함
  - **클래스 정의**: 인스턴스들이 '가질 속성'과 '할 수 있는 행위'의 목록을 정의하는 부분
  - **멤버 함수 정의**: 인스턴스가 할 수 있는 행위를 정의하는 부분
  - **애플리케이션**: 클래스를 기반으로 객체를 만들어서 사용하는 부분

클래스 정의	속성과 행위 선언
멤버 함수 정의	행위 정의
애플리케이션 개발	객체를 인스턴스화하고 사용

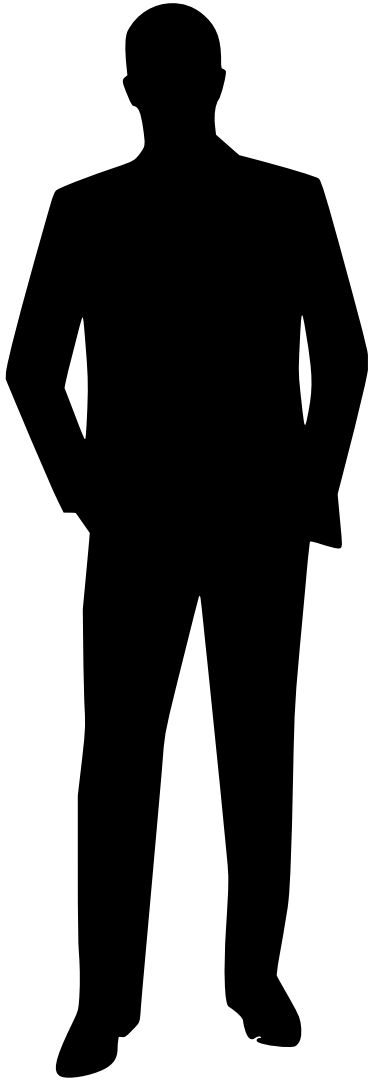
객체지향 패러다임의 3가지 구성요소

클래스 정의 방법

- 새로운 타입을 만들려면 클래스 정의(class definition)를 작성
  - 클래스 정의는 헤더, 본문, 세미콜론이라는 세 부분으로 구성
  - 클래스 헤더(class header)는 class라는 키워드 뒤에 **클래스의 이름**을 붙여서 생성
  - 클래스 본문(class body)은 **데이터 멤버와 멤버 함수의 선언을 가진 블록**(중괄호로 열고 닫히는 영역) 부분
  - 마지막으로 **세미콜론**은 클래스 정의를 종료하겠다고 나타내는 부분
- 접근 제한자(access modifier)
  - 접근 제한자는 접근 권한을 나타낼 때 사용
  - 클래스에서 데이터 멤버와 멤버 함수를 선언하면 기본적으로 private 접근 제한자가 붙음
  - private 접근 제한자가 붙으면 해당 요소로부터 값을 추출할 수도 없고, 값을 변경할 수도 없음
  - 데이터 멤버에 접근해서 어떤 작업을 하려면, 애플리케이션에서 멤버 함수를 사용할 수 있어야 함
  - 따라서 일반적으로 멤버 함수는 public을 적용

접근 제한자	같은 클래스에서의 접근	서브 클래스에서의 접근	모든 곳으로부터의 접근
private	가능	불가능	불가능
protected	가능	가능	불가능
public	가능	가능	가능

Champion class 설계 – 기본 속성

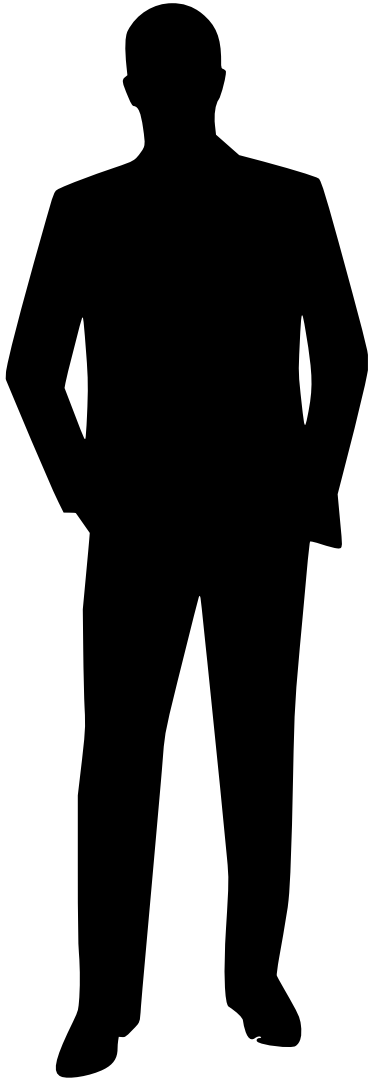


- 게임 속 챔피언을 추상적으로 표현한 클래스를 설계해보자.
- 챔피언은 다양한 속성을 가지고 있다.
  - 챔피언에 대한 기본적인 정보는 이름, 주요 포지션 등이 있다.
  - 챔피언마다 체력, 물리공격력, 마법공격력 등이 다르다.
- 속성을 달리해서 여러가지 챔피언을 만들 수 있다.

Champion class 설계 – 기본 속성

```
01  #include <iostream>       // 이후 생략
02  using namespace std;      // 이후 생략
03
04
05  class Champion            // 챔피언 클래스 정의
06  {
07  private:
08      string name;          // 챔피언 이름
09      int position;         // 0 = top, 1 = jungle, 2 = mid, 3 = bottom, 4 = support
10
11      int hp;               // 체력
12      int pPower;           // 물리 공격력
13      int mPower;           // 마법 공격력
14      int pDef;             // 물리 방어력
15      int mDef;             // 마법 방어력
16      double atkSpd;        // 공격 속도
17      int skilSpd;          // 스킬 가속도
18      int critRate;         // 치명타 확률
19      int moveSpd;          // 이동 속도
20  };
21
```

Champion class 설계 – 속성관리 개선



- 열거체(enumerated)를 이용해 챔피언의 포지션을 숫자가 아닌 문자로 관리해보자.
  - enum Position을 생성하고 그 안에 챔피언의 포지션을 열거한다.
  - Champion class에서는 Position 타입의 변수를 선언한다.

- 속성들을 일괄적으로 관리할 수 있도록 별도의 구조체를 선언해보자.
  - struct Properties를 생성하고 멤버변수로 속성을 정의한다.
  - Champion class에서는 Properties 타입의 구조체 변수를 선언한다.

Champion class 설계 – 속성관리 개선

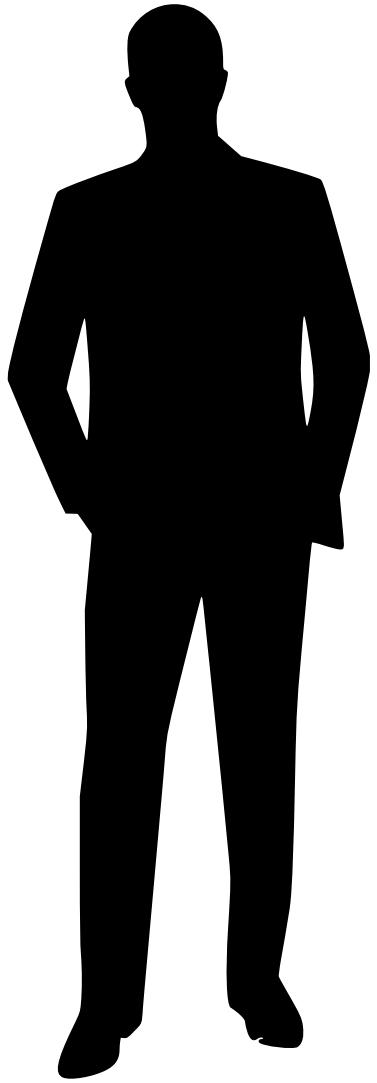
```
01  enum Position {TOP, JUNGLE, MID, BOTTOM, SUPPORT};
02
03  struct Properties {
04      int hp;                 // 체력
05      int pPower;             // 물리 공격력
06      int mPower;             // 마법 공격력
07      int pDef;               // 물리 방어력
08      int mDef;               // 마법 방어력
09      double atkSpd;          // 공격 속도
10      int skilSpd;            // 스킬 가속도
11      int critRate;           // 치명타 확률
12      int moveSpd;            // 이동 속도
13  };
14
15  class Champion              // 챔피언 클래스 수정v1
16  {
17  private:
18      string name;            // 챔피언 이름
19      Position pos;           // 챔피언의 주요 포지션
20      Properties prop;        // 챔피언의 속성
21  };
```

생성자

- 객체가 데이터 멤버를 갖고 어떤 작업을 하려면, 객체를 만든 뒤 데이터 멤버를 초기화하는 작업이 필요
  - 객체를 생성할 때, 생성자(constructor)라고 부르는 특별한 멤버 함수가 호출됨
  - 따라서 생성자를 이용하면 객체의 초기화를 쉽게 처리할 수 있음
  - 생성자는 리턴이 없으며, 클래스와 같은 이름을 사용함

- 객체가 더 이상 필요가 없어지는 경우, 객체가 차지하고 있는 메모리를 비워줘야 함(메모리 재활용)
  - 이때는 소멸자(destructor)라고 부르는 특별한 멤버 함수가 호출함
  - 소멸자 내부에서 객체를 정리하는 작업을 수행하도록 작성함

생성자 Champion() 만들기



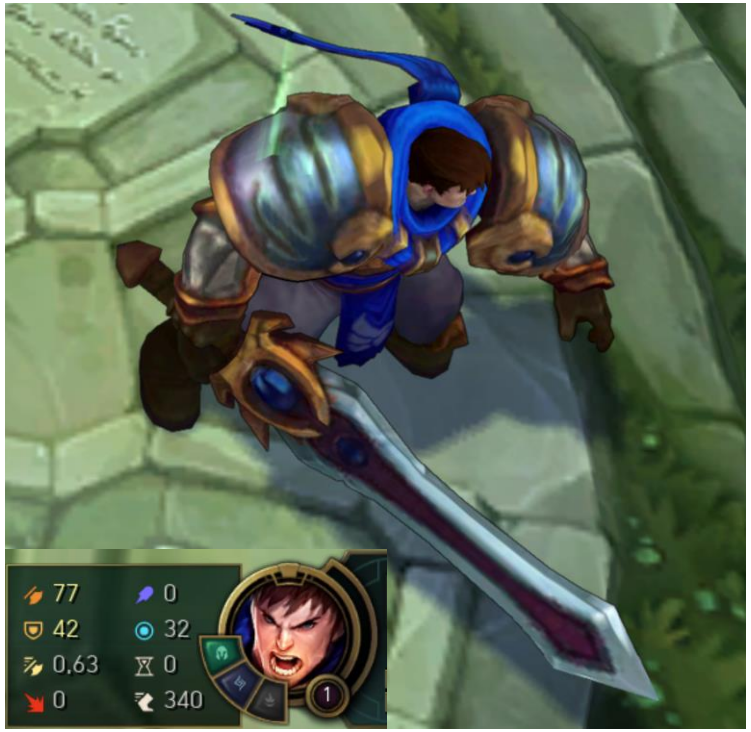
- 생성자에서는 챔피언이 가져야 할 여러가지 속성을 설정하는 역할을 한다.
  - 이름, 스킨, 포지션을 설정한다.
  - 기본 속성 값들을 설정한다.
  - 속성 값이 미리 정의되어 있는 경우, 쉽게 챔피언을 생성할 수 있다.

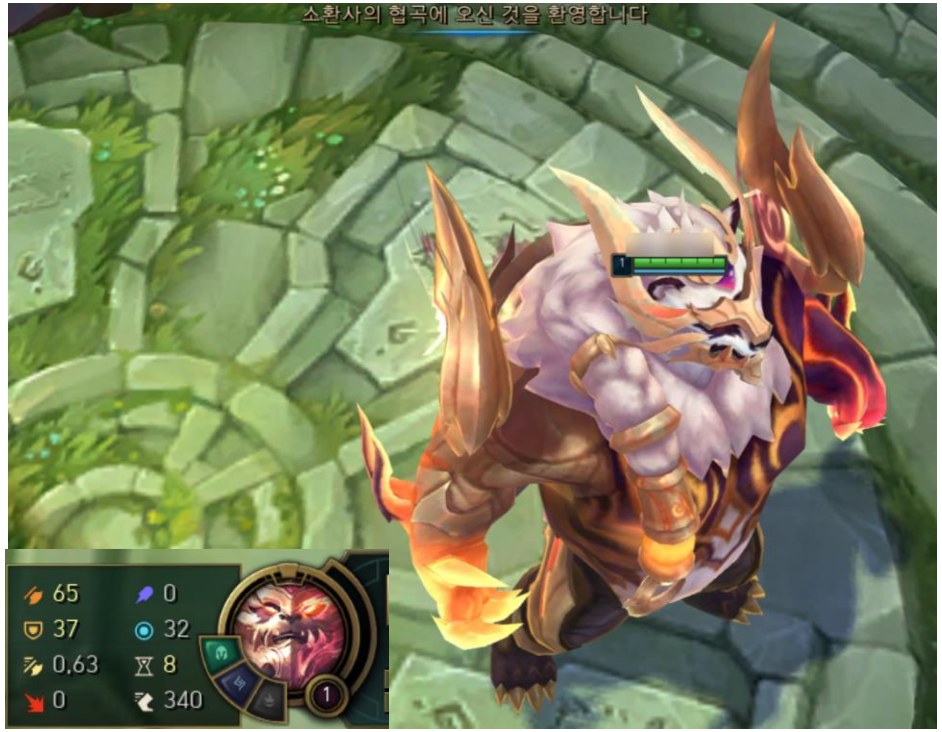
Champion() 생성자 만들기

```
01  enum Position {TOP, JUNGLE, MID, BOTTOM, SUPPORT};
02
03  struct Properties { /* 생략 */ };
04
05  class Champion           // 챔피언 클래스 수정v2
06  {
07  private:
08      string name;         // 챔피언 이름
09      Position pos;        // 챔피언의 주요 포지션
10      Properties prop;     // 챔피언의 속성
11
12  public:
13      Champion(string name, Position pos, Properties prop);         // 생성자 함수 선언
14  };
15
16  Champion::Champion(string name, Position pos, Properties prop){   // 생성자 함수 정의
17      this->name = name;
18      this->pos = pos;
19      this->prop = prop;
20  }
21
```

Champion() 생성자 만들기

```
22  int main() {
23      Properties prop_garen = {620, 77, 0, 42, 32, 0.625, 0, 0, 340};        // 챔피언 속성 선언 및 정의
24      Champion garen = Champion("Garen", TOP, prop_garen);                   // 챔피언 생성
25      Properties prop_volibear = {580, 65, 0, 37, 32, 0.625, 8, 0, 340};     // 챔피언 속성 선언 및 정의
26      Champion volibear = Champion("Volibear", JUNGLE, prop_volibear);       // 챔피언 생성
27  }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```





멤버 함수(=메소드) 정의

- 멤버 함수의 정의는 기본적인 함수 정의와 비슷하지만 2가지 차이점이 있음
  - 멤버 함수에는 한정자 const를 붙일 수 있음
  - 멤버 함수에는 앞에는 클래스 이름이 붙음
  - C++에서는 멤버 함수 이름 앞에 클래스 이름과 클래스 스코프 기호(::)를 성(family name)처럼 붙여야 함

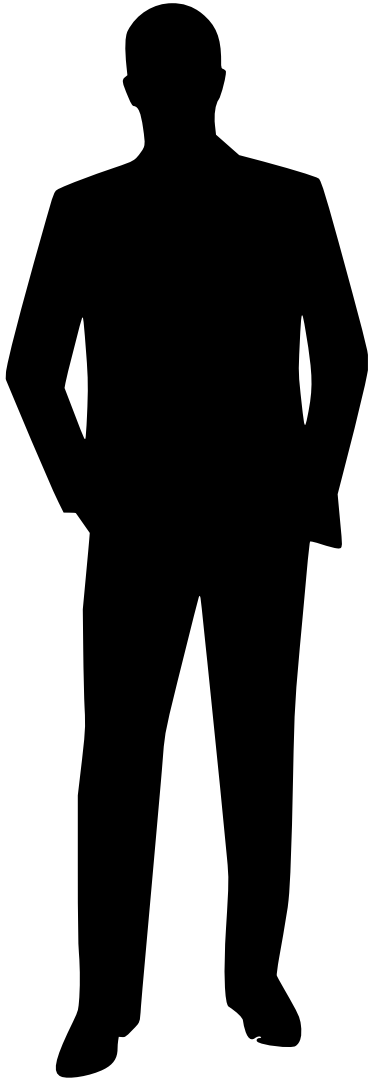
```
class Circle
{
    ...
    public:
        double getRadius() const;
    ...
}
```

클래스를 정의할 때는
멤버 함수 앞에 클래스 이름이 붙지 않음

```
double Circle::getRadius()const;
{
    ...
}
...
```

멤버 함수를 정의할 때는
멤버 함수 앞에 클래스 이름을 붙여야 함

챔피언 정보를 화면에 출력하는 printChampion() 메소드



- printChampion() 메소드는 현재 챔피언의 속성을 화면에 출력한다.
  - Champion() 클래스 안에 printChampion() 메소드를 선언한다.
  - 클래스 밖에서 Champion::printChampion() 메소드를 정의한다.

챔피언 정보를 화면에 출력하는 printChampion() 메소드

```
01  class Champion              // 챔피언 클래스 수정v3
02  {
03  private:
04      string name;            // 챔피언 이름
05      Position pos;           // 챔피언의 주요 포지션
06      Properties prop;        // 챔피언의 속성
07
08  public:
09      Champion(string name, Position pos, Properties prop);          // 생성자 함수 선언
10      void printChampion();                                          // 챔피언의 정보를 화면에 출력
11  };
12
13  void Champion::printChampion(){
14      cout << "이름: " << name << "\t체력: " << prop.hp << endl;
15      cout << "물리 공격력: " << prop.pPower << "\t마법 공격력: " << prop.mPower << endl;
16      cout << "물리 방어력: " << prop.pDef << "\t마법 방어력: " << prop.mDef << endl;
17      cout << "공격 속도: " << prop.atkSpd << "\t스킬 가속도: " << prop.skilSpd << endl;
18      cout << "치명타 확률: " << prop.critRate << "\t이동속도: " << prop.moveSpd << endl;
19  }
20
21
```

챔피언 정보를 화면에 출력하는 printChampion() 메소드

```
22  int main() {     // main 함수 수정v1
23      Properties prop_garen = {620, 77, 0, 42, 32, 0.625, 0, 0, 340};        // 챔피언 속성 선언 및 정의
24      Champion garen = Champion("Garen", TOP, prop_garen);                   // 챔피언 생성
25      Properties prop_volibear = {580, 65, 0, 37, 32, 0.625, 8, 0, 340};     // 챔피언 속성 선언 및 정의
26      Champion volibear = Champion("Volibear", JUNGLE, prop_volibear);       // 챔피언 생성
27
28      garen.printChampion();
29      cout << endl;
30      volibear.printChampion();
31  }
32
33
34
35
36
37
38
39
40
41
42
```

```
이름: Garen 체력: 620
물리 공격력: 77 마법 공격력: 0
물리 방어력: 42 마법 방어력: 32
공격 속도: 0.625    스킬 가속도: 0
치명타 확률: 0  이동속도: 340

이름: Volibear  체력: 580
물리 공격력: 65 마법 공격력: 0
물리 방어력: 37 마법 방어력: 32
공격 속도: 0.625    스킬 가속도: 8
치명타 확률: 0  이동속도: 340
```

예제 1. 동그라미를 만드는 클래스 Circle

- 동그라미를 만드는 클래스를 정의해봅시다.
  - 동그라미는 반지름 값을 갖습니다.
  - 동그라미 마다 서로 다른 반지름을 가질 수 있도록, 반지름을 설정할 수 있습니다.
  - 동그라미는 반지름을 이용해서 면적과 둘레를 계산할 수 있습니다.

- 위의 정의에서 속성과 행위를 구분해봅시다.
  - 속성은 값을 이용해 표현할 수 있어야 합니다.
  - 행위는 계산, 값의 변화, 화면 출력 등 단순히 값으로는 표현할 수 없는 부분입니다.
  - 위의 예제에서 반지름은 속성으로 표현할 수 있습니다.
  - 반지름 설정, 면적계산, 둘레계산 등은 행위로 표현할 수 있습니다.

예제 1. 동그라미를 만드는 클래스 Circle

- Circle은 radius나는 속성을 갖습니다.
- Circle은 다음과 같은 메소드들도 갖습니다.
  - int getRadius(): 반지름을 반환합니다.
  - void setRadius(int radius): 반지름을 설정합니다.
  - double getArea(): 면적을 계산해서 반환합니다.
  - double getPerimeter(): 둘레 길이를 계산해서 반환합니다.

- 위의 기준에 맞춰 Circle 클래스를 정의합니다.
  - 메소드 중 const 키워드를 붙일 수 있는 메소드에 const를 붙입니다.
- 마지막으로 Circle의 멤버 함수를 정의합니다.
  - 클래스에서 선언한 4가지 멤버 함수들을 정의합니다.
  - 둘레와 반지름을 계산하기 위해 상수형 변수 PI를 사용합니다.

예제1-2. 동그라미 클래스 확장하기

- Circle 클래스에 생성자와 소멸자를 추가합니다.

- 생성자는 '**아무것도 받지 않은 경우**'와 '**반지름을 받은 경우**'로 나뉩니다.
  - 아무것도 받지 않는 경우, 아무런 동작을 수행하지 않습니다.
  - 만일 반지름을 받았다면, setRadious() 함수를 호출해 반지름을 설정합니다.

- 소멸자는 별도의 인자를 받지 않습니다.
  - Circle이 소멸될 때는 화면에 "Bye"라고 출력해줍니다.

연습문제1. 은행계좌 클래스 만들기

- 클래스를 이용해 은행 계좌를 표현해봅시다.
  - 은행 계좌는 계좌 번호와 잔금을 속성으로 갖습니다.
  - 계좌에 돈을 입금하거나 출금할 수 있습니다.
  - 다른 계좌로 돈을 송금할 수도 있습니다.
  - 계좌의 정보를 출력할 수 있습니다. (이 함수는 미리 작성된 것을 사용합니다.)
- 계좌번호와 잔금은 정수형으로 표현합니다.
- 입금함수 deposit()은 입금할 금액을 인자로 받습니다.
  - 인자로 받은 정수 만큼의 금액을 현재 계좌의 잔금에 추가합니다.
- 출금함수 withdraw()는 출금할 금액을 인자로 받습니다.
  - 현재 계좌의 잔금에서 인자로 받은 정수 만큼의 금액을 차감합니다.
- 송금함수 transfer()는 송금할 계좌번호와 송금할 금액을 인자로 받습니다.
  - 현재 계좌에서 정수 만큼 금액을 출금하고, 입력 받은 계좌에 정수만큼 금액을 입금합니다.

연습문제1. 은행계좌 클래스 만들기

- 은행계좌 클래스는 생성자와 소멸자를 갖습니다.
  - 계좌를 생성할 때는 계좌번호를 받습니다.
  - 계좌 생성 시 잔금은 0원으로 설정합니다.
  - 계좌를 소멸할 때는 "x번 계좌 소멸됨'이라고 화면에 출력하며, 여기서 x는 계좌번호를 의미합니다.

연습문제1. 은행계좌 클래스 만들기 – 동작 예시

- 계좌 번호가 1인 상태애서 1000원 입금

```
현재 계좌 번호는 1 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계
좌 전환, 5. 종료): 1
입금할 금액을 입력해 주세요: 1000
==========
계좌 번호: 1    잔금: 1000
==========
==========
계좌 번호: 2    잔금: 0
==========
```

- 계좌 전환 후 2번 계좌에 2000원 입금

```
현재 계좌 번호는 1 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계좌 전환, 5. 종료): 4
==========
계좌 번호: 1    잔금: 1000
==========
==========
계좌 번호: 2    잔금: 0
==========
현재 계좌 번호는 2 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계좌 전환, 5. 종료): 1
입금할 금액을 입력해 주세요: 2000
==========
계좌 번호: 1    잔금: 1000
==========
==========
계좌 번호: 2    잔금: 2000
==========
```

- 2번 계좌에서 100원 출금

```
현재 계좌 번호는 2 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계좌 전환, 5. 종료): 2
출금할 금액을 입력해 주세요: 100
==========
계좌 번호: 1    잔금: 1000
==========
==========
계좌 번호: 2    잔금: 1900
==========
```

- 2번 계좌에서 1번 계좌로 500원 송금

```
현재 계좌 번호는 2 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계좌 전환, 5. 종료): 3
송금할 금액을 입력해 주세요: 500
==========
계좌 번호: 1    잔금: 1500
==========
==========
계좌 번호: 2    잔금: 1400
==========
```

- 종료에 의한 소멸자 호출

```
현재 계좌 번호는 2 입니다.
수행할 동작을 입력하세요 (1. 입금, 2. 출금, 3. 송금, 4. 계좌 전환, 5. 종료): 5
2번 계좌 소멸됨
1번 계좌 소멸됨
```


HANYANG
UNIVERSITY