

# 프로그래밍 방법론

## Project 2

서울대학교

담당조교: 김종민, 도완주

# SNU Music

PM 2021: Project 2

- 간편결제 시스템을 탑재한 음악 다운로드 서비스
- 간편결제 시스템 SNU PAY
  - 은행계좌 잔고로부터 SNU Money 충전
  - 체크카드 등록
- 음악 관리 시스템
  - 앨범 별로 음악을 관리
  - 시스템으로부터 음악별, 앨범별로 다운로드 가능
  - 앨범을 통째로 사면 30% 할인!

# 파일 구성

PM 2021: Project 2

- bank.cpp / bank.h
  - 은행 계좌 클래스 BankAccount
- card.cpp / card.h
  - 체크카드 클래스 Card
- client.cpp / client.h
  - 고객 클래스 Client
- music.cpp / music.h
  - 음악 클래스 Music & 앨범 클래스 Album
- SNUMusicSystem.cpp / SNUMusicSystem.h
  - 관리 시스템 클래스 **SNUMusicSystem**
- main.cpp / musicList.txt
  - 건드릴 필요 없음

# class BankAccount

PM 2021: Project 2

- Member Variable
  - `const std::string bankName`  
= 은행 계좌 이름
  - `int balance`  
= 은행 잔고
- 구현해야 될 기능
  - Constructor로 은행 이름과 잔고를 자유롭게 초기 설정 가능
  - Member Variable의 Getter 함수 → 조회 시
  - 은행 예금/출금 기능
  - \* – 출금의 경우 잔고가 부족할 시 실패!     바인더스 통장 넘

# std::string

PM 2021: Project 2

- 이번 프로젝트에서 다양한 이름을 저장하는 데 쓰이는 type
- constructor 또는 GetName() 과 같은 함수를 제외하곤 쓰일 일이 별로 없음
- std::string은 문자열(텍스트)를 저장하는 데 이용됨
- std::string은 다른 type처럼 자유롭게 사용이 가능하다.
- 예시

```
#include <string>
```

```
std::string name1 = "Gildong Hong";
```

```
std::cout << name1 << std::endl;    // Output: Gildong Hong
```

```
std::string name2("John Doe");
```

```
return name2;
```

# class Card

PM 2021: Project 2

- Member Variable

- `const std::string cardName`  
= 카드 이름

- `BankAccount* withdrawalAccount` (pointer → dynamic X)  
= 출금 계좌 (class BankAccount)

은행 계좌 → 카드

- 구현해야 될 기능

- 이 클래스는 신용카드가 아니라 체크카드를 상징합니다
  - Constructor로 카드 이름과 출금 계좌를 지정해야 한다
  - Member Variable의 Getter 함수
  - 결제 기능
  - 은행계좌와 마찬가지로 출금 계좌의 잔고가 부족하면 결제가 실패!

카드 없으면 결제  
사라지려 X .

# class Client

PM 2021: Project 2

- Member Variable

- const std::string clientName  
= 고객 이름
- int SNUMoney  
= 은행계좌로부터 충전해서 사용하는 온라인 캐시 **SNU Money** (카드로 충전 불가능)
- BankAccount\* accountList [5]  
= 등록한 은행계좌 리스트, 5개까지 가능
- Card\* cardList [5]  
= 등록한 카드 리스트, 5개까지 가능
- Album albumList [100] *pointer 아님.*  
= **SNU Music**에서 구매한 앨범 리스트  
= 앨범 전체를 구매하지 않아도 음악은 해당 리스트에 앨범 별로 보관됩니다
- int numAlbums  
= 고객이 구매한 앨범의 개수  
= 앨범 전체를 구매하지 않고, 한 앨범의 한 트랙만 가지고 있어도 카운트합니다

) null ptr 로 채워줌 .

# class Client

PM 2021: Project 2

- 구현해야 될 기능
  - Constructor
  - Getter
  - 간편 결제 시스템 SNU PAY
  - 등록한 은행 계좌로부터 SNU Money 충전
  - 은행 계좌 등록
  - 카드 등록
  - SNU Money를 이용한 앨범 및 음악 결제
  - 카드를 이용한 앨범 및 음악 결제
  - 가지고 있는 앨범 및 음악 리스트 관리
  - 은행 계좌에서 바로 결제할 수는 없습니다
    - = SNU Money를 충전하여서 이용하세요
  - 카드로는 SNU Money 충전이 불가능합니다
  - 잔고 부족 시 결제 실패

Add bank account  
(dynamic 하게)  
Add card도 dynamic 하게



# class Music

PM 2021: Project 2

- Member Variable

- `std::string musicName`  
= 음악 트랙의 제목
- `Album* album`  
= 이 음악이 들어있는 앨범의 포인터

- `bool owned`  
= 고객이 이 음악을 가지고 있는지 여부

*false default*

- 구현해야 될 기능

- Constructor
- 이 클래스는 앨범 안에 들어 있는 한 트랙을 상징합니다
- `owned`는 client의 `albumList` 안에서만 의미가 있습니다  
= `SNUMusicSystem` 안에서는 항상 `false`
- 이 클래스는 복사될 수 있음에 유의하세요  
= ex) `SNUMusicSystem`에서 client의 `albumList`로 복사

# class Album

PM 2021: Project 2

- Member Variable

- std::string albumName

- = 앨범의 이름

- std::string artistName

- = 아티스트의 이름

- int numMusic

- = 앨범의 수록곡 개수

- ~~\*\*\*~~ Music\*\* musicList

- = 앨범 수록곡의 dynamic array

- = Music\*의 array

7251  
0

pointer의 dynamic

# class Album

↓  
초기화 필요

PM 2021: Project 2

2'05

- 구현해야 될 기능

- Constructor

- = Album의 constructor는 수록곡이 몇 곡인지 확인하여 musicList의 공간을 확보해 놓아야 합니다.
    - = 하지만 musicList array의 각 항목 (각각의 Music\* type 앨범 수록곡)까지 초기화하지는 않습니다.
    - = 그러므로 Album의 musicList에 수록곡 목록을 채우고 싶으면, constructor에서가 아닌 추가적인 작업이 필요합니다.

- 이 클래스는 복사될 수 있습니다

- = Ex) **SNUMusicSystem**에서 client의 albumList로 복사되는 경우
    - = 복사 시에 (앨범 수록곡 중 하나만 사는 경우라도) 앨범의 전체 musicList가 복사되어야 합니다
    - = Rule of three 구현

- Operator == *overloading*

- = 앨범 이름과 아티스트 이름으로 같은 앨범인지 == 으로 구분할 수 있어야 합니다.

- 앨범 또는 곡 구매시 해당 곡들의 owned 설정 함수

↳ own하면 true로 변경

# class SNUMusicSystem

PM 2021: Project 2

- Member Variables
  - Client\* client  
= 고객 (단 한 명만 이용)
  - Album\*\* albumList  
= SNUMusicSystem에 등록된 전체 앨범 리스트 (owned = false)  
= Album\*의 <sup>dynamic</sup>array
  - int numAlbum  
= SNUMusicSystem에 등록된 전체 앨범의 개수
- albumList와 numAlbum은 자동으로 등록되므로 (musicList.txt 사용) 구현을 걱정하지 않으셔도 됩니다.

# class SNUMusicSystem

PM 2021: Project 2

- 구현해야 될 기능 : 10가지 서비스 제공
  - 0: Show entire album list
    - = 시스템에 존재하는 전체 앨범 리스트 출력 (이미 거의 구현)
  - 1: Purchase an entire album
    - = 앨범 전체를 구매
    - = 가격:  $700 * (\text{앨범 전체 곡 개수}) ₩$
    - = 단,  $1000 * (\text{앨범 수록곡 중 아직 갖고 있지 않은 곡의 수}) ₩$  이 더 저렴할 경우, 해당 가격으로 결제
  - 2: Purchase a song
    - = 곡 한 개를 구매: 1000원
  - 3: View my song list
    - = 내가 가지고 있는 앨범 및 곡 목록 출력 (이미 거의 구현)
  - 4: View my **SNU Money** balance
    - = 내가 가지고 있는 **SNU Money** 충전금 출력 (이미 거의 구현)

# class SNUMusicSystem

PM 2021: Project 2

- 구현해야 될 기능 : 10가지 서비스 제공
  - 5: Charge **SNU Money**  
= 은행 계좌로부터 **SNU Money** 충전
  - 6: Register a bank account to **SNU PAY**  
= **SNU PAY**에 은행 계좌 등록
  - 7: Register a card to **SNU PAY**  
= **SNU PAY**에 카드 등록
  - 8: View my bank account list  
= **SNU PAY**에 등록된 은행 계좌 목록 출력 (이미 거의 구현)
  - 9: View my card list  
= **SNU PAY**에 등록된 카드 목록 출력 (이미 거의 구현)

# 유의사항

PM 2021: Project 2

- **SNUMusicSystem**에서 앨범 및 곡 구매시 고객의 albumList로 앨범 또는 곡이 복사되어야 합니다!
- 앨범 수록곡 중 하나만 복사되어도 **전체 수록곡 목록이 같이 복사되어야 합니다!**
  - 물론 owned는 구매한 곡에만 true가 되어야 합니다.
- Album의 constructor는 수록곡이 몇 곡인지 확인하여 musicList의 공간을 확보해 놓아야 하지만 musicList array의 각 항목 (각각의 Music\* type 앨범 수록곡)까지 초기화하지는 않습니다.
- 잔고 부족 시 결제가 실패합니다. (처음 서비스 선택으로 복귀)

# Demo

PM 2021: Project 2

```
jongminkim@jongminkim-pc: ~  
File Edit View Search Terminal Help  
jongminkim@jongminkim-pc:~$ ./main  
Welcome to SNU Music System!  
  
SNU MUSIC  
  
Enter Your Name: Gildong Hong  
  
Choose Service  
0: Show entire album list  
1: Purchase an entire album  
2: Purchase a song  
3: View my song list  
4: View my SNU Money balance  
5: Charge SNU Money  
6: Register a bank account to SNU PAY  
7: Register a card to SNU PAY  
8: View my bank account list  
9: View my card list  
What service do you need?:
```



# Advanced C++ Functionality

PM 2021: Project 2

- `const`

```
const std::string& Client::GetName const {  
    return clientName;  
}
```

- 앞의 `const`는 `std::string&` 반환값을 수식변하지 않아야 하는 문자열에 대한 `reference`를 반환한다는 뜻
- 뒤의 `const`는 member function의 특성을 의미
- 이 member function으로는 Client 클래스의 member variable 값을 변경할 수 없다는 뜻

# Advanced C++ Functionality

PM 2021: Project 2

- delete

```
class BankAccount {  
    BankAccount(const BankAccount&) = delete;  
    BankAccount& operator=(const BankAccount&) = delete;  
}
```

- BankAccount와 같이 함부로 copy되면 안 되는 class에 대해서 delete 키워드를 이용해서 명시적으로 copy constructor와 assignment operator가 생성되지 않도록 방지할 수 있다
- 프로젝트에서는 BankAccount, Card, Client, SNUMusicSystem 클래스에 이렇게 구현되어 있음

# Tips

PM 2021: Project 2

- Comment가 꼼꼼히 적혀 있으니 열심히 참고합시다
- 시스템은 상식적인 선에서 구현하시면 됩니다!
- 이번 프로젝트를 통해 클래스 구현 및 이용 방법에 대해 숙지합시다
- 특히 skeleton으로 이미 주어진 부분이라도 꼼꼼하게 살펴보세요!
- Memory-safe한 클래스를 구현하는 것이 이번 Project 2의 최대 목표입니다.
- Happy Coding!

# Deadline

PM 2021: Project 2

~ May 10<sup>th</sup>, 2021 | 23:59 KST (UTC +9)

Include README.md  
in your GitHub Classroom repository  
for short description of your work.

# Q & A

# Dynamic allocation

① Add account

② Add card

) → client.cpp or destructor

③ class Album

Music \*\* musicList

= 배열의 크기 dynamic array

= Music \*[] array

) → music.cpp or destructor

num Music  
num Album ( [ -, -, - ], ← Album  
[ -, - ]  
[ -, -, -, - ] )

④ class Music System

= Album \*\* albumList

= Album \*[] dynamic array

→ SNUmusicSystem.cpp destructor

```

Album& Album::operator=(const Album& rhs)
// TODO
// make assignment operator (deep copy)
// be sure to make it memory-safe

// for (int i=0; i<numMusic; i++){
//     delete musicList[i];
// }
// delete []musicList;

albumName = rhs.albumName;
artistName = rhs.artistName;
numMusic = rhs.numMusic;
for (int i=0; i<numMusic; i++){
    musicList[i] = rhs.musicList[i];
}

// //Music** musicList = new Music*[numMusic];
// for (int i=0; i<numMusic; i++){
//     const std::string& _musicName = rhs.musicList[i]->GetName();
//     Album* _album = rhs.musicList[i]->GetAlbum();
//     bool _owned = rhs.musicList[i]->IsOwned();
//     musicList[i] = new Music(_musicName, _album, _owned);
return *this;

```

먼저 assignment operator는 일반적으로 다음과 같은 형태를 띌 것입니다.

```
if(this == &rhs) return;
```

```
// 1. delete some dynamic allocated area
```

```
// 2. dynamically allocate the area
```

```
// 3. deep copy
```

지금 1번은 주석 처리된 부분에 있고, 3번도 있지만 2번이 이루어지고 있지 않은 상황이겠습니다.

감사합니다

김종민 드림