Object Oriented Programming (Week 11)

2023

KWANGWOON UNIVERSITY
DEPT. OF COMPUTER ENGINEERING

Contents

ASSIGNMENT 3-3. 1

Assignment 3-3. 1

(Class Inheritance with virtual) Suppose you are designing a price compare program. The program compares the discount price with the operator '<' in Sale class. Create class called Sale that has price (type double, private) and public functions are shown below. At first, the input data are price of the item. Function Bill is declared to be a virtual function. If a function is virtual, a new definition of the function is given in a derived class, then for any object of the derived class, that object will always use the definition of the virtual function that was given in the derived class. You should create the DiscountSale class which inherits from Sale class. It has a private value named DiscountPercent (type double) and public funcitons. The class also has a Bill function which calculates the discount price using the GetPrice function in Sale class. DiscountPercent is member variable in DiscountSale class. Do not add any variable or function in Sale. Prototype of Sale class and result as follows:

```
class Sale
{
   private:
      double Price;

   public:
      Sale();
      Sale(double ThePrice);
      ~Sale();

      double GetPrice();
      virtual double Bill();
      double Savings(Sale& Other);
      bool operator < (Sale& Other);
};
```

< Example >

```
==========================================
Price Compare Program
==========================================
Insert item1 price: $16
Insert item2 price: $12

----------------------------------------------------------
Insert discount percent: 25%
----------------------------------------------------------
Result:
Discount price of item2 is cheaper.
Saving discount price is $3.0
```

Assignment 3-3. 1

```
class DiscountSale: public Sale
{
private:
    double Discount;

public:
    DiscountSale();
    DiscountSale( double ThePrice, double theDiscount );
    ~DiscountSale ();
     double GetDiscount();
    void SetDiscount();
    double Bill();
    double Savings(DiscountSale& Other );
    bool operator < (const DiscountSale& first, const DiscountSale& second);
    //or bool operator < (const DiscountSale& second);

};
```

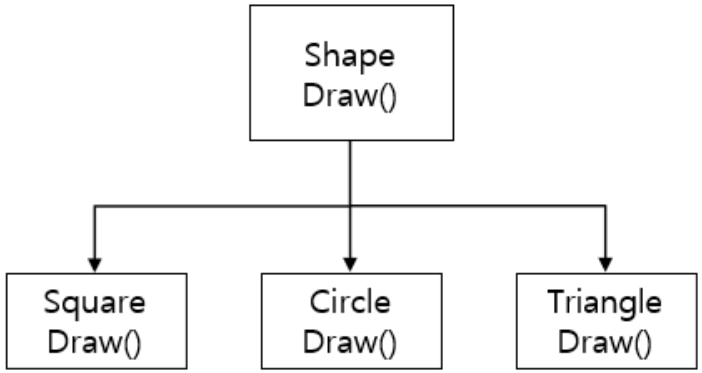
상속 접근 지정자	기반 클래스	파생 클래스로의 상속형태
public	public	public
	private	접근 불가
	protected	protected
private	public	private
	private	접근 불가
	protected	private
protected	public	protected
	private	접근 불가
	protected	protected

thrillfighter.tistory.com

Assignment 3-3. 1

- Inheritance
  - 부모 class의 멤버 변수, 함수 등을 자식 class에서도 사용할 수 있게 하는 것
  - 프로그램의 유연성 증가
- Inheritance Overriding
  - 부모 클래스에서 이미 정의 된 함수를 무시하고 자식 클래스에서 새롭게 정의하는 것

```
#include <iostream>

using namespace std;

class parent{
public:
    void over_riding() { cout<< "부모클래스" << endl; }
};
class child : public parent{
public:
    void over_riding() { cout<< "자식클래스" << endl; }
};
int main()
{
      child c;
      c.over_riding();
      return 0;
}
```



Assignment 3-3. 1

- Virtual feature
  - 부모 클래스의 포인터를 통해 자식 클래스의 함수 호출 가능
  - 순수 가상함수(정의가 없고 상속받은 클래스에서 정의)로 선언되면 해당 객체를 생성할 수 없음
    - virtual void over_riding() = 0;
  - C언어는 기본적으로 정적 바인딩(컴파일시에 결정), virtual 키워드를 통한 동적 바인딩(실행시간 중 결정)이 가능
- Virtual function

```
class shape{
public:
        virtual void over_riding() = 0; // 순수가상함수
        //virtual void over_riding() {cout<< "부모클래스의over_riding함수" << endl; } // 일반가상함수
};
class circle: public shape{
public:
         virtual void over_riding() { cout<< "자식클래스의over_riding함수" << endl; }
};
int main()
{
        shape *s;
        s = new circle;
        s->over_riding();
        return 0;
}
```

Virtual 사용 결과
: 자식 클래스의 over_riding 함수
단순 overriding 결과
: 부모 클래스의 over_riding 함수

Assignment 3-3. 1

```cpp
#include <iostream>


class A
{
public:
    A() :v(0) {
        std::cout << "Constructor A" << std::endl;
    }
    ~A() {
        std::cout << "Destructor A" << std::endl;
    }

    void incv() {
        this->v++;
    }
    void printv() {
        std::cout << "A " <<  v << std::endl;
    }

    int getV() {
        return v;
    }

private:
    int v;
};

class B : public A
{
public:

    B() :vv(0), A() {
        std::cout << "Constructor B" << std::endl;
    }
    ~B() {
        std::cout << "Destructor B" << std::endl;
    }

    void inc() {
        this->vv++;
    }
    void print() {
        std::cout << "B " << vv << std::endl;
    }

private:
    int vv;
};
```

```cpp
int main()
{
    B b;
    b.incv();
    b.incv();
    b.incv();
    b.printv();

    b.inc();
    b.print();


    return 0;
}
```

ASSIGNMENT 3-3. 2

Assignment 3-3. 2

(Class inheritance & Function overriding) Implement two classes, Professor and Student inheriting Person class. Person class is inherited by Professor and Student class. Professor class has extra member variables of a professor number and a major, and Student class has additional member variables of a student number, a major and a school year. Override Say function for inherited classes to display all member variables in the class object. Write a program that creates Professor and Student class objects and test your code properly. (Do not change the prototype of Say function)

```
Class Person
{
    protected:
        int age;
        char name[32];

    public:
        Person();
        ~Person();
        virtual void Say()=0;
};
```

Pure virtual function must be overwritten in an derived class

Assignment 3-3. 2

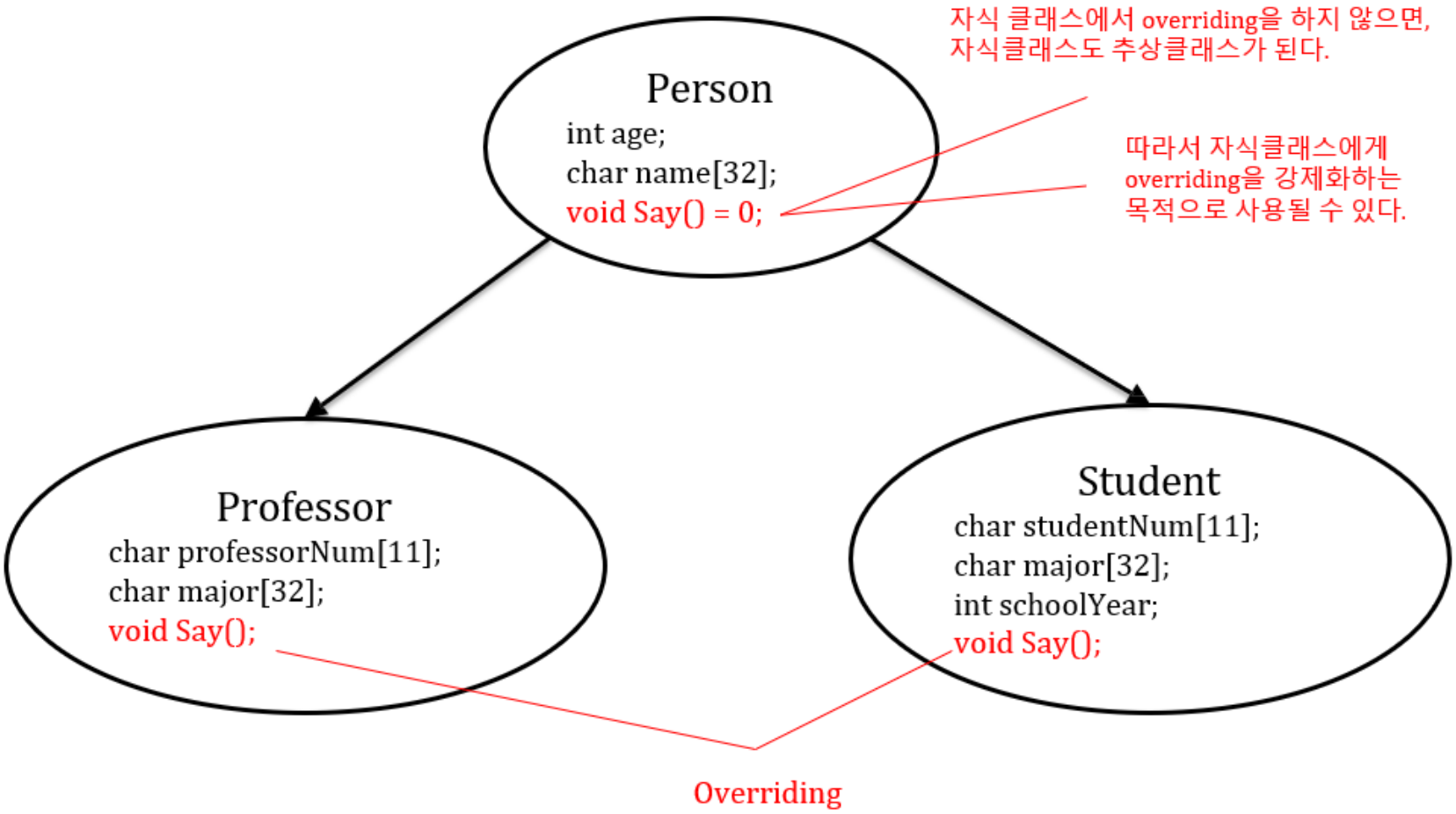
- Class prototype

```
class Person
{
 protected:
      int age;
      char name[32];

 public:
      Person();
      ~Person();

      virtual void Say() = 0;
};
```

순수 가상함수
- 순수 가상함수를 가지는 클래스는 추상클래스
- 추상클래스의 객체는 생성할 수 없다.
- 자식클래스에서의 overriding을 강제화

~~Person p1;~~

Assignment 3-3. 2



Assignment 3-3. 2

- Example of inheritance

```
class Student : public Person
{
 protected:
     char studentNum[11];
     char major[32];
     int schoolYear;

 public:
     Student();
     ~Student();
     int getAge();
     char* getName();
     char* getStudentNum();
     char* getMajor();
     int getSchoolYear();
     void setAge(int age);
     void setName(char* name);
     void setStudentNum(char* studentNum);
     void setMajor(char* major);
     void setSchoolYear(int year);
     void Say();
};
```

```
I'm a student of KW University.
My name is Hong Gill Dong.
I'm 21 years old and I'm a sophomore.
I'm majoring in Computer Engineering.
```

ASSIGNMENT 3-3. 3

Assignment 3-3. 3

(Operator overloading) Create a class called Matrix for performing arithmetic with 3x3 matrices. Write a program to test your class. Each element in the matrix should be represented using double precision. Provide a constructor that enables an object to be initialized when it is declared. The constructor should contain default values in case that no initializers are provided. Provide public member functions for each of the following:
a) Printing a matrix in the form of a 3x3 array, where each element is separated by a space and each row is separated by a newline.
b) Overloading the operators +=, -=, *, to simulate the corresponding matrix operations.

- Operator overloading
  - 기존에 c언어가 제공하는 연산자에 대하여 의미를 다시 부여하는 것

Operator overloading

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


class Dot {
      int x, y;
public:
      Dot( int_x=0, int _y=0) : x(_x), y(_y) { }
      Dot operator+(const Dot & p);
      Dot operator+(const int v);
      int operator<(const Dot & p);
};
```

Operator overloading

```
Dot Dot::operator+(const Dot & p)
{
      // Dot temp(x+p.x, y+p.y);
      //return temp;
      return Dot(x+p.x, y+p.y);
}
Dot Dot ::operator+(const int d)
{
      Dot temp(x+d, y+d);
      return temp;
}
int Dot ::operator<(const Dot & p)
{
      .....;
}
```

Operator overloading

```
int main(void)
{
      Dot d1(1, 2);
      Dot d2(2, 1);

      Dot d3 = d1 + d2;
      if(d3 < d1)
        cout<<"true!!₩n";
      else
        cout<<"false!!₩n";

      return 0;
}
```

ASSIGNMENT 3-3. 4

Assignment 3-3. 4

(Doubly Linked-list) Implement a student score management system with a doubly linked list. The system stores and manages student's average score of Math, English and Science. Each score of course are read from a user and calculates the average score in a main function. All the average scores are stored in each Score class and handled by a StudentScore-List class. Score class contains one double type variable for storing the average score and two pointers indicating next and previous Score node, respectively. Note that when implementing the "Insert()" function in a StudentScoreList class, the Score nodes should be linked into ascending order by average score. And the "PrintList()" function prints all the average scores in "Score" nodes as ascending or descending order according to the parameters, "IsAscending". (Ascending order if "is Ascending" is true, otherwise, descending order is used.)

```
class StudentScoreList
{
   private:
      Node* m_pHead;
      Node* m_pTail;

   public:
      StudnetScoreList()
      ~StudentScoreList();

      void Insert(Score* pScore);
      void PrintList(bool isAscending);
};
```

```
class Node
{
   private:
      Node* m_pNext;
      Node* m_pPrev;
      double m_Avg;

   public:
      Score();
      ~Score();

      void SetAvg(double avg);
      void SetNext(Score* pNext);
      void SetPrev(Score* pPrev);
      double GetAvr();
      Score* GetNext();
      Score* GetPrev();
};
```

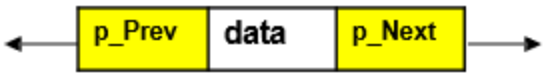
Assignment 3-3. 4

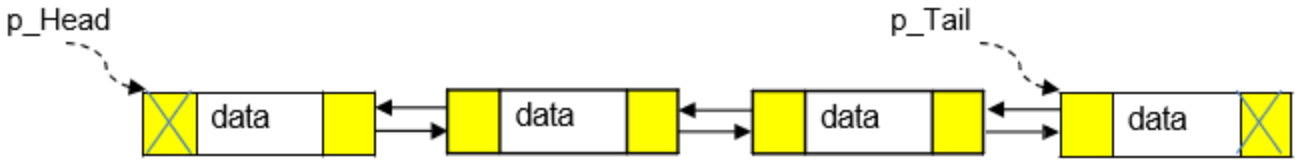
```
Class StudentScoreList
{
   private:
      Score* m_pHead;
      Score* m_pTail;

   public:
        StudentScoreList ( );
        ~ StudentScoreList ( );

        void Insert(Score* pScore );
        void PrintList(bool isAscending );
};
```

```
class Score
{
   private:
      Score*  m_pNext;
      Score*   m_pPrev;
      double m_Avg;

   public:
      Score ( );
      ~ Score ( );

        void SetAvg (double avg);
      void SetNext( Score* pNext );
      void SetPrev( Score* pPrev );
      double GetAvr( );
       Score* GetNext( );
       Score* GetPrev( );
};
```

Assignment 3-3. 4

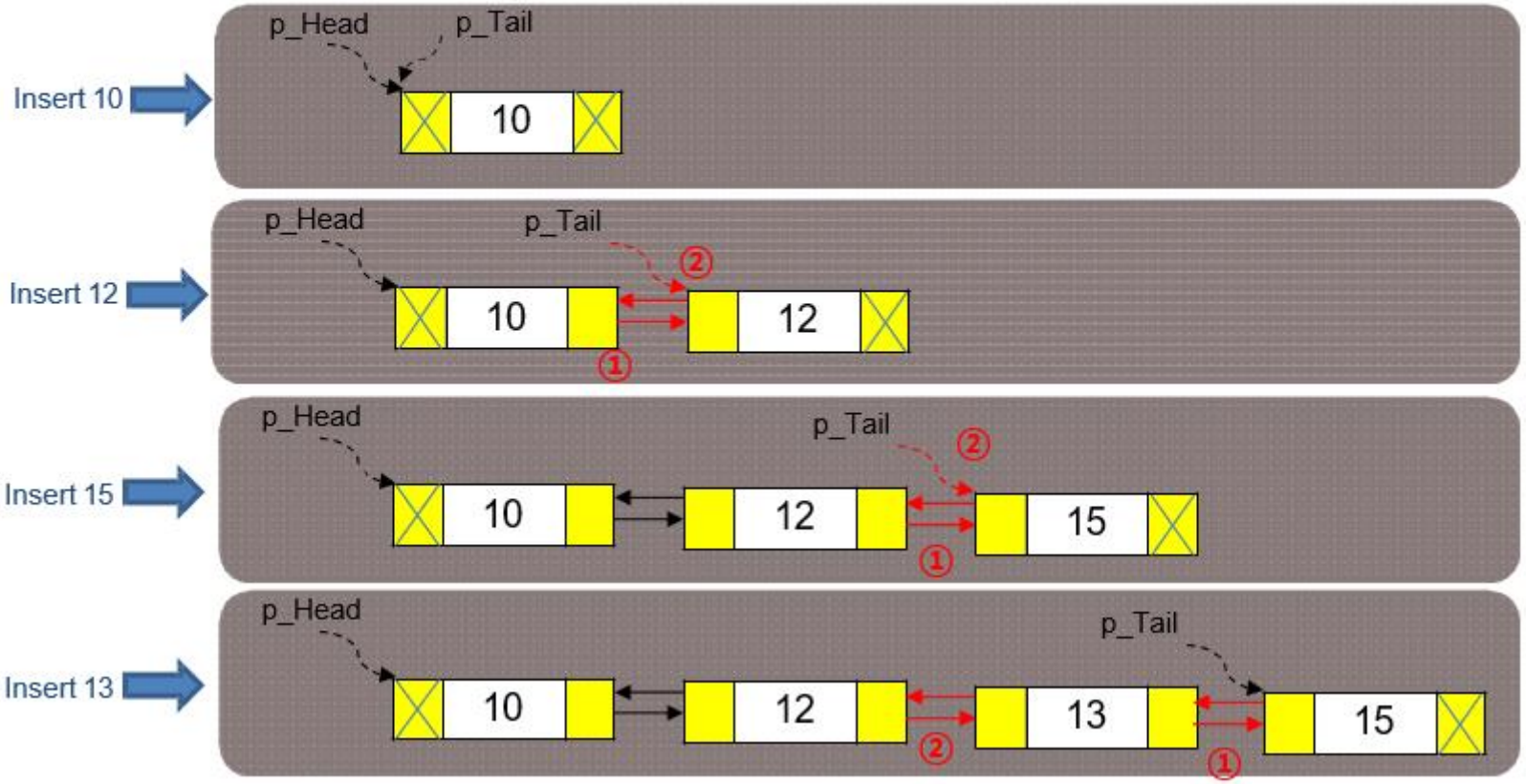
< Node >





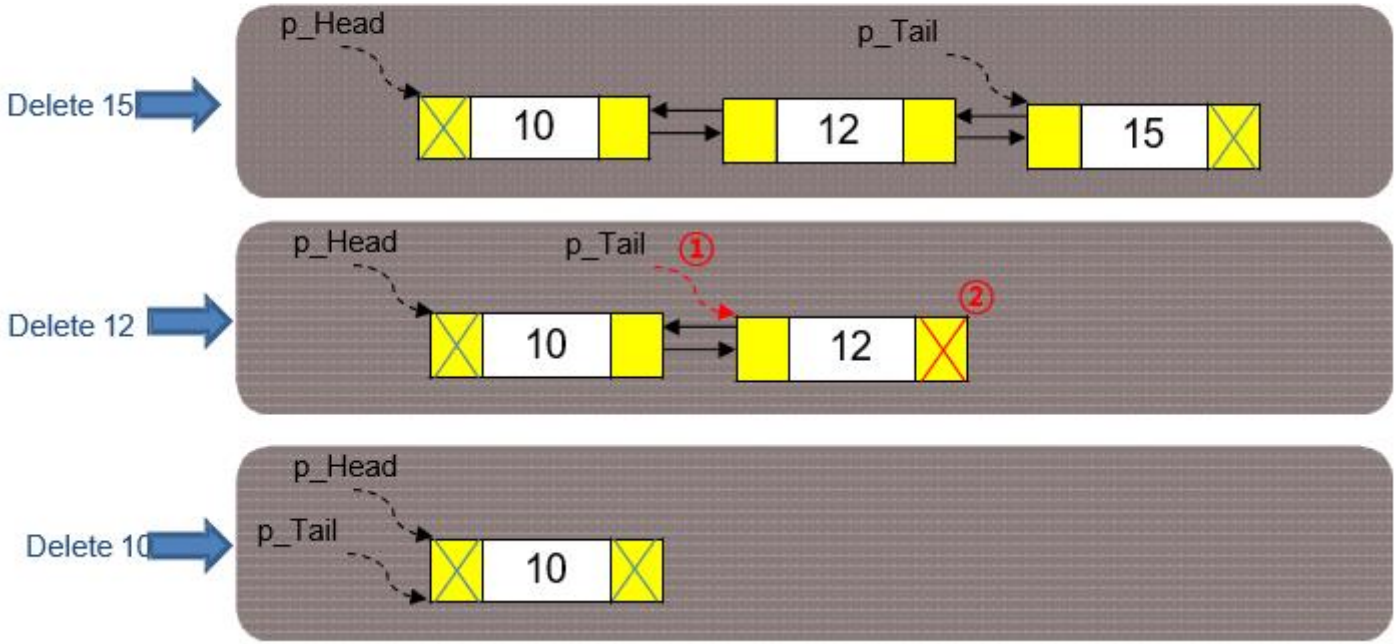
Assignment 3-3. 4

- Insert example



Assignment 3-3. 4

- Delete example



과제 제출 방법

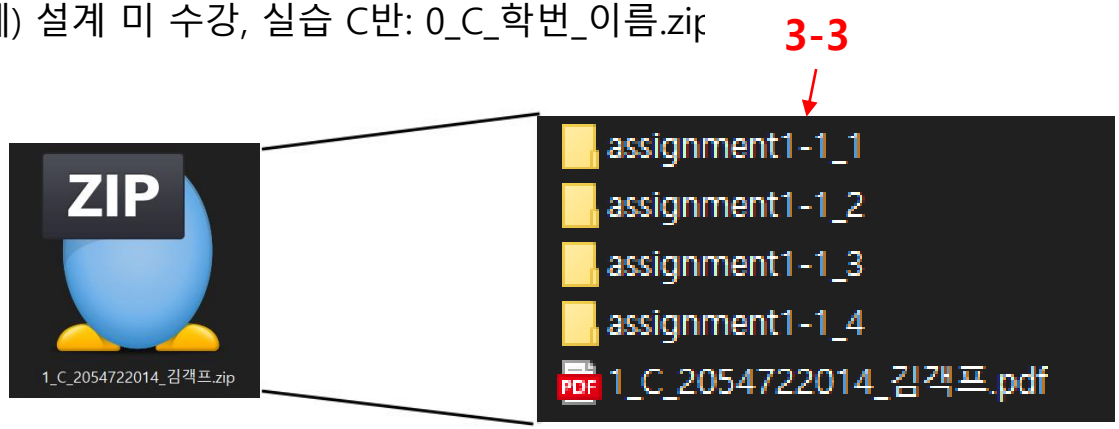
과제 제출 방법

- **FTP Upload (Klas 과제 제출 X)**
  - Address : ftp://223.194.8.1:1321
  - username : IPSL_OBJ
  - password : ipslobj_2023

- **Due date**
  - Soft copy: 마감일 5/26(금) 23:59:59까지 제출 (서버시간 기준)
  - Delay
    - 마감일 이후 +7일까지 제출 가능
    - 단, 1일 초과마다 과제 총점의 10%씩 감점

과제 제출 방법

- Soft copy
  - 과제(보고서, 소스 코드)를 압축한 파일 제출
    - 설계반_실습반_학번_이름.zip
      - 예) 설계1반 수강, 실습 A반: 1_A_학번_이름.zip
      - 예) 설계 수강, 실습 미수강: 2_0_학번_이름.zip
      - 예) 설계 미 수강, 실습 C반: 0_C_학번_이름.zip



  - 과제 수정하여 업로드 시 버전 명시
    - 설계반_실습반_학번_이름_verX.zip

과제 제출 방법

- Soft copy
  - 과제 보고서
    - 영문 또는 한글로 작성
    - **반드시 PDF**로 제출 (PDF 외 파일 형식으로 제출시 0점 처리)
    - 보고서 양식
      - 문제 및 설명(문제 capture 금지) / 결과 화면 / 고찰
      - 보고서 양식은 아래 경로에서 참고
        - https://www.ipsl.kw.ac.kr/post/1%EC%B0%A8-%EA%B3%BC%EC%A0%9C
    - **소스코드 제외**
    - 분량 제한 없음
    - **표절 적발 시 0점 처리**
  - 소스 코드
    - Visual Studio 2022 community 사용 필수
      - https://docs.microsoft.com/ko-kr/visualstudio/install/install-visual-studio?view=vs-2022
    - STL (Standard Template Library) 사용 금지 (vector, map, algorithm 등)
    - Debug 폴더를 제외한 모든 파일 제출
      - .sln 파일 포함(.cpp 만 제출하지 말것)
    - **각 문제마다 프로젝트 파일 생성 필수**
    - **주석 반드시 달기**
    - **소스코드 표절 적발 시 0점 처리**

END OF PRESENTATION

Q&A