3주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. Array 클래스 구현**

Array 클래스를 구현하기 위한 단계별 수행은 다음과 같다.

protected

* int \*data : 1차원 배열에 메모리를 동적 할당하여 자료가 저장되는 변수이다.
* int len : 배열의 크기를 저장한다.

public

* Array(int size) / ~Array() : size를 받아 크기가 size인 배열을 생성한다. / 할당받은 메모리를 해제한다.
* int length() : 배열의 크기len의 값을 반환한다.
* int& operator[](int i) : i가 배열 범위 내 (0 <= i < len) 일 경우 값을 할당한다. left value 용도로 사용한다.
* int operator[](int i) : i가 배열 범위 내 (0 <= i < len) 일 경우 값을 반환한다. right value 용도로 사용한다.
* void print() : 배열의 원소를 모두 출력한다.

**2. RangeArray 구현**

Array 클래스의 구현과 크게 다르지는 않지만, 상속의 개념을 추가해줘야 한다.

* class RangeArray : public Array : RangeArray는 Array를 상속받는다. protected와 public에 접근이 가능하다.

protected

* int row, high : 배열의 시작과 끝 인덱스를 지시한다.

public

* RangeArray(int,int) / ~RangeArray() : high와 row를 받아 크기가 high-row+1 인 배열을 생성한다. / 메모리를 해제한다.
* Int baseValue() / endValuse : 배열의 시작과 끝을 반환한다. (low / high)
* int& operator[](int i) : i가 배열 범위 내 (low<= i <high) 일 경우 data[i-low]값을 할당한다.
* int operator[](int i) : i가 배열 범위 내 (low<= i <high) 일 경우 data[i-low]값을 반환한다.

**3. OOP**

Object Oriented Programming 으로 객체 지향 프로그래밍이라고 한다. 절차 지향적인 C언어와 달리 객체의 관점에서 프로그래밍 하는 것을 말한다. C언어를 예로 포인터를 사용해 call by reference처럼 동작하게 구현하는 것이 객체 지향프로그래밍과 비슷하다. 자료구조가 우선이 되며, 클래스를 통해 객체(인스턴스)를 선언하고, 캡슐화, 추상화, 다형성, 상속의 특징이 있다.

**4. 객체**

필드(Field) 또는 상태(State)와 행동(Behavior)으로 구성된다. 상태는 객체가 지니고 있는 변수나, 속성을 말하고, 행동은 객체가 지니고 있는 메소드, 함수를 말한다. 자신만의 속성을 가지고 있어 구분이 가능하다. 객체지향 프로그래밍에서 클래스의 인스턴스를 의미한다.

**5. 클래스**

클래스는 인스턴스를 만들기 위한 설계도와 비슷하다. 클래스에는 객체가 가지는 상태와 행동이 정의되어 있다.

**6. 인스턴스**

클래스를 통해서 만들어진 객체들을 말한다. 하나의 클래스에서 여러 인스턴스들을 정의 할 수 있다. 하지만, 하나의 클래스에서 만들어졌다고 같은 것은 아니다.

**7. OOP를 사용하는 이유**

OOP에는 캡슐화, 상속, 다향성, 추상화등의 특징이 있다. 이에 대형프로젝트를 하기에 응집력은 강하고, 결합력은 약하다. 프로그램의 개발과 유지보수가 간편하고 유연하다. 또한, 자주 사용하는 라이브러리를 따로 만들어 다른 코드에 재사용하기 쉽고 직관적인 코드 분석이 가능하다.

**8. 상속**

새롭게 만든 클래스와 기존 클래스의 상태와 행동을 합쳐 하나 의 인스턴스를 생성할 수 있다. 상속받는 클래스를 하위 클래스, 상속하는 기존 클래스를 상위 클래스라고 부른다. 하위 클래스는 상위 클래스의 속성을 가지고, 캡슐화를 유지하며 자신의 기존 속성도 가진다. (Overriding이 가능하다.) 객체 지향 프로그래밍에서 재사용과 유연성이 증가하게 되는 이유 중 하나이다.