3주차 예비보고서

전공: 경제학과 학년: 4학년 학번: 20180501 이름: 김연수

1. 문제 해결 방법, 단계별 수행내용, 자료구조.

이 문제는 Array 클래스와 RangeArray 클래스를 구현하고, 연산자 다중정의와 멤버 함수를 사용하여 배열의 원소값을 참조하고 대입하는 기능을 구현하는 것이 목적이다.

단계별 수행 내용:

Array 클래스를 구현한다.

사용자가 입력한 크기의 배열을 할당하는 생성자를 구현한다.

배열의 원소값을 참조하는 right value 용도의 int operator[]와 배열의 원소에 새로운 값을 대입할 수 있는 left value 용도의 int &operator[]을 각각 구현하고, 잘못된 인덱스를 가질 때 에러 메시지를 출력한다.

배열의 모든 원소에 대하여 출력하는 print() 멤버 함수를 구현한다.

RangeArray 클래스를 구현한다.

동적으로 정해진 배열 인덱스 범위를 나타내는 base와 end 변수를 선언한다.

입력받은 범위로부터 배열 크기를 계산하여 Array 클래스의 생성자를 통해 메모리를 할당받는 생성자를 구현한다.

Array 클래스를 상속받아, int operator[]와 int &operator[] 멤버 함수를 구현한다. 각 함수에서 배열의 범위에 속하지 않는 인덱스를 가질 때 에러 메시지를 출력한다.

baseValue()와 endValue() 멤버 함수를 추가하여, RangeArray 클래스의 배열 범위를 사용자가 확인할 수 있도록 한다.

1. opp

OPP란 Object-Oriented Programming의 약자다. 즉 객체 지향 프로그래밍을 말한다. 객체 지향 프로그래밍은 말 그대로 프로그램을 디자인하기 위해서, 객체를 사용하는 프로그래밍 패러다임이다.

전통적으로 사용되었던 절차적 프로그래밍과는 다르다. 절차적 프로그래밍은 알고리즘이 우선적이고 자료구조를 나중에 고려하지만, 객체 지향 프로그래밍은 자료구조가 우선이고, 그 후 알고리즘을 생각한다. 객체 지향은 재사용성을 최대화하고, 데이터 의존성을 줄이고, 디버깅 시간을 단축시킬 수 있다.

1. 객체

객체는 상태와 행동이라는 두 가지 구성요소를 가지고 있다. 상태는 객체가 보유하는 속성이나 특성(데이터, 속성)을 말한다. 행동은 객체가 할 수 있는 행동 또는 할 수 있는 반응의 양식(메소드, 오퍼레이션)을 말한다. 객체는 클래스의 인스턴스라고 볼 수 있다.. 객체를 만들면 클래스에 의해 정의된 템플릿이나 청사진의 유니크한 복사본을 만드는 것과 같다. 객체는 재사용 가능하고, 모듈화된 코드를 작성할 수 있도록 해준다는 장점이 있다. 각 객체는 커스터마이즈될 수 있고, 특정 속성 또는 지원하는 함수의 조합에 따라서 다른 방식으로 사용될 수도 있다.

1. 클래스

OPP에서 클래스는 객체를 만들기 위한 청사진 또는 템플릿이라고 할 수 있다. 클래스에서는 속성이나 변수 같은 attributes를 정의하고 객체가 가질 수 있는 함수들을 정의한다. 클래스를 이해하기 위해서 현실세계의 비유를 들 수 있다. 예를 들어서 특정한 차는 색, 모양, 문의 개수 등등 다양한 특성이 있을 수 있다. 또 가속, 브레이크 등의 행동을 할 수 있다. 이렇게 특정 기능과 속성을 가진 차를 객체라고 부르고, 이런 속성과 행동을 정의해놓은 것을 클래스라고 부를 수 있다.

1. 인스턴스

인스턴스는 클래스에서 생성된 객체와 같은 의미를 가진다고 볼 수 있다. 즉 클래스의 특정한 구현을 말한다. 고유의 속성과 함수를 가진다. 아래의 코드를 보자.

class Dog:

species = "canine"

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

dog1 = Dog("Fido", 3)

dog2 = Dog("Buddy", 5)

print("dog1 name is", dog1.name)

print("dog2 age is", dog2.age)

print("dog1 species is", dog1.species)

여기서 클래스 Dog의 인스턴스는 dog1과 dog2다. dog1의 이름은 “fido”이고 dog2의 이름은 “Buddy”다. 하지만 종은 “canine”으로 동일하다. 종은 instance의 attribute가 아니라 class의 attribute이기 때문이다.

1. opp를 쓰는 이유

요약하자면 OPP는 다음과 같은 장점이 있다.

1. 데이터 추상화
2. 캡슐화, 데이터 은닉
3. 상속성
4. 다형성
5. 재활용코드

데이터 추상화: OOP를 사용하면 복잡한 실제 개체를 나타낼 수 있는 추상 데이터 유형을 만들 수 있다. 즉, 개체의 필수 기능에 집중하고 사용과 관련이 없는 세부 정보는 무시할 수 있다. 이렇게 하면 코드가 더 간단하고 이해하기 쉬울 뿐만 아니라 더 강력하고 오류가 덜 발생한다..

캡슐화, 데이터 은닉: OOP는 캡슐화도 지원한다.. 즉, 외부 세계에서 개체의 내부 세부 정보를 숨길 수 있다. 이렇게 하면 개체 데이터에 대한 무단 액세스를 방지하고 코드를 보다 모듈화하고 유지 관리하기 쉽게 만들 수 있다.

상속: OOP는 상속도 지원하므로 기존 클래스를 기반으로 새 클래스를 만들 수 있다. 이렇게 하면 작성해야 하는 코드의 양을 줄이고 코드를 보다 모듈화하고 유지 관리하기 쉽게 만들 수 있다. 또한 보다 일반적인 클래스의 속성과 메서드를 상속하는 특수 클래스를 생성할 수 있으므로 코드가 더 유연하고 다양한 사용 사례에 적응할 수 있다.

다형성: OOP는 다형성을 지원한다. 즉, 동일한 인터페이스를 사용하여 다른 개체를 나타낼 수 있다. 이를 통해 코드를 더 모듈화하고 유지 관리하기 쉽게 만들 수 있을 뿐만 아니라 다양한 사용 사례에 더 유연하고 적응할 수 있다.

코드 재사용성: 재사용이 가능하고 여러 프로젝트에서 공유할 수 있는 코드를 만들 수 있다. 클래스와 개체에 기능을 캡슐화하면 처음부터 코드를 다시 작성할 필요 없이 다양한 사용 사례에 쉽게 적용할 수 있는 코드를 만들 수 있다. 이는 시간을 절약하고 개발 비용을 줄이는 데 도움이 될 뿐만 아니라 코드의 전반적인 품질을 향상시킬 수 있다.

1. 상속

상속은 한 클래스(자식 클래스)가 다른 클래스(부모 클래스)에서 속성과 동작을 파생시키는 두 클래스 간의 관계를 정의하는 방법이다.

상위 클래스는 기본 클래스 또는 super 클래스라고도 하며 하위 클래스는 파생 클래스 또는 하위 클래스라고도 한다. 자식 클래스는 부모 클래스의 모든 속성과 메서드를 상속하며 자신의 속성과 메서드를 추가하거나 부모 클래스의 속성과 메서드를 재정의할 수도 있다.

상속을 사용하면 코드를 재사용할 수 있다. 클래스를 한 번 정의한 다음 해당 클래스를 기반으로 파생 클래스를 생성하면 코드 중복을 방지하고 개발 프로세스를 단순화할 수 있다. 또한 기본 클래스에 대한 변경 사항이 모든 파생 클래스에 자동으로 전파되므로 코드를 더 모듈화하고 유지 관리하기 쉽게 만들 수 있다.

예를 들어 Car, Truck, Motorcycle 등 다양한 유형의 차량을 관리하는 소프트웨어 시스템을 구축한다고 가정해보자. 바퀴 수, 엔진 크기 및 연비와 같은 모든 유형의 차량에 공통적인 속성 및 메서드를 포함하는 Vehicle 클래스를 정의할 수 있다. 그런 다음 Car 클래스, Truck 클래스 및 Motorcycle 클래스와 같은 각 차량 유형에 대한 파생 클래스를 만들 수 있다.

이러한 각 클래스는 Vehicle 클래스의 속성과 메서드를 상속하지만 필요에 따라 자체 속성과 메서드를 추가할 수도 있다. 예를 들어 Car 클래스에는 문 수에 대한 속성이 있고 Truck 클래스에는 화물 용량에 대한 속성이 있을 수 있다. 상속을 사용하면 공통 속성과 동작을 공유하지만 필요에 따라 특수화할 수도 있는 클래스의 계층 구조를 만들 수 있다.

상속을 통해서 부모 클래스의 속성과 메서드를 재정의하는 보다 전문적인 클래스를 만들 수도 있다. 이는 특정 사용 사례에 대해 클래스의 동작을 사용자 지정해야 할 때 유용하다. 예를 들어 연비를 계산하는 메서드가 있는 Vehicle 클래스가 있지만 전기 자동차에 대해 다른 계산을 사용해야 한다고 가정해보자. Vehicle 클래스에서 상속하지만 연비 메서드를 새 구현으로 재정의하는 ElectricVehicle 클래스를 만들 수 있다.

요약하면 상속은 기존 클래스를 기반으로 삼아서 새로운 클래스를 만들 수 있는 객체 지향 프로그래밍의 강력한 기능이다. 상속은 코드 재사용을 가능하게 하고 개발 프로세스를 단순화할 뿐만 아니라 코드를 더 모듈화하고 유지 관리하기 쉽게 만든다.