■ 구조적 프로그래밍 설계 구조 및 절차

1.구조적 프로그래밍 설계 구조

(1) 순차구조
(2) 선택구조
(3) 반복구조

2. 구조적 프로그래밍 언어 개발 절차

(1) 요구사항 분석
(2) 구조적분석
(3) 구조적설계
(4) 구조적프로그래밍

(순차, 선택, 반복의 논리구조를 구성하여 프로그램 복잡성을 최소화하여 프로그래밍 하는 단계이다)

■구조적 프로그램 구성요소

- 데이터 흐름도 DFD (Data Flow Diagram)
- 자료사전 DD (Data dictionary)
- 상태전이도 STD(State Transition Diagram)
- 소단위 명세

■구조적 프로그래밍 언어 개요

1. 절차식 언어
2. 명령어 언어
3. 함수 중심 언어

c언어, 파스칼, 에이다

선택: if, if~else, switch , 반복: for, while

■ 기능적 모듈화를 고려한다

: 모듈과 컴포넌트를 이용하여 기능적으로 분리하여 효율적인 프로그래밍이 될 수 있도록 개발한다
(함수단위, 클래스 단위로 개발한다)

- 응집도 :모듈내부 요소들이 서로 연관되어 있는 정도를 나타낸다
  응집도가 강할수록 품질이 높고 약할수록 품질이 낮다

응집도		
기능적 응집도	높음, 강함	한 모듈을 구성하는 요소들이 단일 기능을 수행하기 위해 협력하는 응집도
순차적 응집도		모듈을 구성하는 한 요소의 출력이 다음 요소의 입력 데이터로 사용하는 경우의 응집도
교환적 응집도		동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성요소들이 모였을 경우의 응집도
절차적 응집도		모듈이 가지는 기능들이 순서에 의해서 처리되는 경우의 응집도
시간적 응집도		특정 시간에 처리되는 기능을 모아 하나의 모듈로 작성할 경우의 응집도
논리적 응집도		모듈의 구성요소들이 유사한 기능을 수행하는 경우의 응집도
우연적 응집도		연관성이 없는 요소로만 구성되어 있는 경우의 응집도

우연적 -〉 논리적 -〉시간적 -〉 절차적 -〉 교환적 -〉 순차적 -〉 기능적응집도(제일높다) (높은 순, 강하다) =〉 순서기억하기

– 결합도

:모듈 간에 상호 의존하는 정도, 두 모듈 사이의 연관 관계를 말하고 결합도가 약할수록 품질이 높고, 강할수록 품질이 낮다

(결합도는 약할수록 좋음)

결합도	
내용결합도 (강함)	두 모듈이 코드를 공유하는 경우 내부 기능이나 그 내부 자료를 직접 참조하거나 수정하는 경우의 결합도
공통 결합도	두 모듈이 전역변수를 공유하는 경우의 결합도
외부 결합도	소프트웨어 외부의 환경에 좌우되는 경우의 결합도
제어결합도	한 모듈의 데이터가 다른 모듈의 실행 순서를 결정하기 위해 전달되는 경우의 결합도
스탬프결합도	배열, 레코드, 구조체 등의 자료구조가 전달될 때의 결합도
자료결합도 (약함)	모듈 간의 인터페이스 자료 요소로만 구성될 때의 결합도

구조적 프로그래밍 애플리케이션을 작성한다

1.변수를 선언한다

(1) 전역변수 및 지역변수를 선언한다

(가) 전역변수

(나) 지역변수

(2) 기능적 구현을 한다

2. 함수를 적용하여 구현한다

(1) 함수의 선언을 한다

(2) 함수 구성을 한다.

(3) 함수 호출을 한다

■객체지향 프로그래밍 설계

객체지향 설계 원칙의 이해

1. 객체지향 속성

(1) 캡슐화 : 객체의 속성과 행위를 하나로 묶고 실제 구현 내용 일부를 외부에 감추어 은닉하는 기법

외부객체는 객체 내부의 구조를 알지 못하게 하고 내부 객체에서 제어하여 제공하는 필드와 매소드만 이용할 수 있다

(2) 추상화 :추상화는 어떤 실체로부터 공통적인 부분이나 관심있는 특성들만 하나로 모은 것을 의미한다. 어떤 하위클래스들에 존재하는 공통적인 매소드를 인터페이스로 정의하는 것도 이에 포함된다

현실사람으로부터 학생,또는 환자, 고객이라는 클래스를 도출함 =〉 이 과정을 추상화라고 함

현실세계의 개념이나 개체를 프로그래밍적으로 모델링하는 과정

=〉 실체로부터 공통적인 속성으 끌어내서 클래스로 만드는 과정을 의미한다

(홍길동 고객, 김길동 고객, 정길동 고객으로부터

고객으로서의 공통된 속성을 뽑아 Customer라는 클래스를 만드는 과정이다)

(3) 다형성

다형성은 같은 모양의 함수가 상황에 따라 다르게 동작하는 것을 의미한다. 다형성을 이용하는 기법은 오버로딩과 오버라이딩이 있다

(가) 오버로딩 : 함수 이름은 같으나 함수의 매개변수 숫자, 타입등을 다르게 해서 사용하는 기법이다

(나) 오버라이딩 : 상위클래스의 매소드를 하위 클래스에서 똑같은 이름으로 재정의 하는 것을 말한다

(4) 정보은닉

(5) 상속성 : 부모클래스에서 새로운 기능을 추가해 자식 클래스를 만들어사용하는 기법을 상속성이라고 한다

자식 클래스에서 따로 정의하지 않아도 부모 클래스에서 정의된 것들을 자동으로 상속받아 구현할 수 있다

■ 객체지향 설계 원칙 (SOLID)

(1) 단일책임원칙 : 모든 클래스는 각각 하난의 책임만 가져야 한다. 클래스는 그 책임을 완전히 캡슐화해야 함을 말한다

(2) 개방-폐쇄원칙: 확정에는 열려 있고 수정에는 닫혀 있어 기존의 코드를 변경하지 않고 기능을 추가할 수 있도록 설계가 되어야 한다는 원칙이다

(3) 리스코프 치환 원칙:

자식클래스는 자신의 부모 클래스로 대체할수 있다는 원칙이다

(4) 인터페이스 분리 원칙:

한 클래스는 자신이 사용하지 않는 인터페이스는 구현하지 말아야 한다는 원칙이다

하나의 평범한 인터페이스보다 여러 개의 구체적인 인터페이스가 좋다

(5) 의존역전 원칙

: 의존역전 원칙은 변화가 엇는 것에 의존해야 한다는 원칙이다

구체적인 클래스보다 인터페이스나 추상클래스와 관계를 맺어야 하는 원칙이다

유즈케이스다이어그램: 요구분석, 시스템 설계, 시스템 구현 등의 시스템 개발 과정에서 개발자 간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어이다

클래스 다이어그램

: 클래스 다이어그램은 UML의 구조 다이어그램으로서 클래스 내부 구성요소 및 클래스간의 관계를 도식화하여 시스템의 특정 모듈이나 일부 또는 전체구조를 나타낸다

속성(Attribute):클래스의 멤머변수

행동(Operation) :해당 클래스가 수행할 수 있는 operation을 가진다 (멤버 매서드), 기능을 말함

■객체지향 프로그래밍 언어 활용

객체지향 프로그래밍 언어 개요
: 프로그래밍에서 필요한 데이터를 추상화시켜서 상태와 행위를 가진 객체를 만들고 그 객체들 간의 유기적인 메시지를 통해 상호작용 로직을 구성하는 프로그래밍 방법이다

1. 객체지향 프로그램 언어 구성요소
객체 프로그래밍은 기본적으로 클래스 구조에서 시작한다. 클래스는 객체지향 개념에서 객체를 정의하는 틀로서 많은 객체지향언어의 기본구조이다
클래스의 기본 구성요소는 변수와 매서드이다
(1) 클래스 : 사용자 정의 데이터형 (사용자 정의 자료형이다)
- 클래스이름과 소스파일명은 동일하다
- 클래스를 실행하기 위해서는 main매서드를 사용한다

(2) 객체(Object)
: 클래스의 인스턴스로 자신 고유의 데이터를 가지며 클래스에서 정의한 행위를 수행할 수 있다.
- 클래스를 사용하려면 new 연산자를 통해 인스턴스를 생성한다
- 인스턴스에서 변수와 매소드 사용은 인스턴스명.변수명, 인스턴스명.매서드명과 같은 형식으로 사용한다

(3) 매서드: 객체를 사용하는 방법으로 특정 기능을 수행한다
(4) 속성 : 객체들이 가지고 있는 데이터 값을 단위별로 정의한 것

2. 접근 지정자(제어자)
(1) public :모든 접근 허용
(2) protected: 자신 및 상속받은 자식 클래스에서의 접근을 허용
(3) private : 자신 클래스의 내부의 매소드에서만 접근을 허용
(4) default : 같은 패키지에서는 모두 접근 가능함 , 다른 패키지에서는 접근 불가

접근제어자가 없으면 기본이 default 임

■객체지향 프로그래밍 언어 기본 문법

1. 데이터 타입
2. 변수와 메서드
변수: 데이터를 저장 하기 위한 메모리 공간에 대한 이름 (메모리 공간을 의미한다) ,
저장할 데이터의 크기를 알아야 필요한 공간을 확보할 수 있다.
(1) 변수선언방법

```
int num1=10;
String msg="Hello";
Member member = new Member();
```

■접급제어자 선언 방법

[접근제어자] 타입 변수명;

```
private int kor;
```

(가) 접근제어자 : 변수의 접근 범위를 지정한 것으로 public, private, protected, default 등을 이용할 수 있다

(나) 타입: 자료형으로 명시적으로 타입을 지정한 것으로 사용할 수 있는 데이터형을 선언하는 곳이다

(2) 변수유형

(가) 멤버변수 : 클래스부에 선언된 변수들로 객체의 속성에 해당하고 인스턴스변수와 클래스 변수 (static 변수) 로 구분된다

(나) 인스턴스 변수 : 클래스가 인스턴스될 때 초기화되는 변수로서 인스턴스를 통해서만 접근할 수 있다.

(다) 매개변수 : 매서드의 인자로 전달되는 값을 받기 위한 변수로서 매서드 내에서는 지역변수처럼 사용된다.

```
public int add( int su1, int su2) {
      return su1+ su2;
}
```

이 때 su1, su2가 매개변수로서 add함수내에서 사용될수 있는 지역변수이다

(라) 지역변수 : 매소드 내에서 선언된 변수로서 멤버변수와 동일한 이름을 가질 수 있으며 지역적ㅇ로 우선일 때 사용하게 된다.

(마) 클래스변수 :

static으로 선언된 변수로 인스턴스 생성없이 클래스 이름의 변수명으로 사용 가능하고 클래스이름으로 사용가능하다

MyCat.count

(3) 매서드

: 매서드는 특정 객체의 동작이나 행위를 정의한 것으로 클래스의 주요 구성요소이다

[접근제어자] 리턴타입 매서드명 (인자...) {

}

■ 응집도와 결합도

결합도는 낮추고 응집도는 높여야 한다 (좋은프로그램) :외부와의 관계는 낮추고 내부에서의 관계는 높인다

결합도
모듈(클래스) 간의 상호 의존정도를 나타내는 지표로써
결합도가 낮으면 모듈간의 상호 의존성이 줄어들어서 객체의 재사용및 유지보수가 유리한다

응집도
하나의 모듈 내부에 존재하는 구성요소들의 기능적 관련성으로
응집도가 높은 모듈은 하나의 책임에 집중하고 독립성이 높아져 재사용 및 유지보수가 용이하다

1. 단일책임원칙 (SRP : Single Responsibility Principle)
 어떠한 클래스를 변경해야 하는 이유는 한가지 뿐 이여야 한다
 (클래스가 자신의 책임만 할 수 있도록 구현한다), 응집력에 대한 내용임

2. 개방폐쇄원칙 OCP(open closed principle)
   JDBC 인터페이스를 구현한 오라클드라이버 , mysql드라이버 ,마리아디비드라이버

3. 리스코트 치환 원칙 LSP (Liskov Substitution principle)
   자식의 타입이 부모의 타입으로 치환될 수 있어야 함

4. 인터페이스분리원칙 ISP (Interface Segregation Principle)
   클라이언트는 자신이 사용하지 않는 매서드에 의존관계를 맺으면 안된다
   프로젝트 요구사항과 설계에 따라서 SRP단일책임원칙 /ISP 인터페이스분리원칙 을 선택한다.
   (너무 다양한 기능을 지원하는 인터페이스를 만들지 않는다)

5. 의존역전원칙 DIP (Dependency Inversion Principle)
   자신보다 변하기 쉬운것에 의존하지 말아야 한다
   (예를 들어 인터페이스에 대해 의존해야 한다)