14주차 2차시 람보우, 요든 객체지향설계

【학습목표】

1. 람바우의 객체설계과정 8단계로 구분하여 설명할 수 있다.
2. 코드와 요든의 객체지향 설계 모델을 4가지 요소로 구분하여 설명할 수 있다.

학습내용1 : 람바우의 객체지향 설계

1. 람바우의 객체지향 설계

1) 분석단계의 산출물인 모델(객체모델, 동적모델, 기능모델)을 통합하면서, 누락 · 추가적으로 요구하는 정보를 보완하여 구현 가능하도록 구체화하는 작업이 「객체설계(object design)」임

2) 람바우는 객체설계과정을 다음과 같이 8 단계로 구분하고 있음.
- 분석모델 통합
- 알고리즘 설계
- 설계내용 최적화
- 제어 전략
- 상속성 조정
- 객체 사이의 관계정의
- 객체 표현
- 패키징

2. 분석모델 통합

1) 분석 결과의 산출물(products)인 모델(model)만 보면 여러 곳에 연산(operation)이 산 재하여 어느 객체가 어떤 연산을 하는지 정확하게 정의할 수 없음.
- 그래서 각 객체가 수행할 연산이 구체적으로 어떤 것인지 정의할 필요가 있음.

2) 연산을 정의하기 위해서는 우선 분석모델에서 어떤 항목들이 연산으로 되는지 파악 하는 것이 선결과제임.

3) 분서과정에서 작성된 모델에서 연산이 될 수 있는 내용은 각각 다음과 같음.
- 객체모델
- 동적모델
- 기능모델

① 객체모델 : 활동(activity) 중에 객체의 속성 값을 수록하거나, 읽는 활동이 연산(operation)으로 됨.
② 동적모델 : 상태 다이어그램의 객체활동, 객체의 순간적 행위, 사건 수신 객체에서 발생하는 활동 등이 연산으로 됨.
③ 기능모델 : 자료 흐름도의 모든 프로세스(process)가 연산으로 됨.

4) 객체의 연산은 세 가지 모델에 서로 다른 형태로 존재하기 때문에 각 모델의 내 용을 종합적으로 분석하여「연산후보」를 찾아내야 함.

5) 연산을 정의하기 위해서는 우선 분석모델에서 어떤 항목들이 연산으로 되는지 파 악 하는 것이 선결과제임.

3. 알고리즘 설계

1) 객체에 대한 연산이 확정된 후에는 그에 적용할 기술적인 면을 고려하여, 수행할 처리절차를 구체화하는 「알고리즘 설계」 가 이루어져야 함.

2) 기능모델로 작성한 자료흐름도는 객체 수준의 상태 다이어그램에서 나타내고 있는 「상태」, 「활동」, 「행위」 등이 수행해야 할 절차에 대하여 자세히 나타내고 있으므로 「알고리즘 설계」 는 비교적 용이함.

3) 다음의 경우는 사용할 알고리즘에 따라서 전체적으로 성능과 밀접한 관계가 있으므로 여러 가지 대안을 비교 검토하여 최상의 방안을 택해야 함.
- 수행과정을 생략하고 선언적으로 정의한 기능
- 복잡한 수학적 계산이 요구되는 기능

4) 위의 기능들에 대한 결정에 고려할 조건들을 요약하면 다음과 같음.
- 연산시간의 최소화
- 이해·구현의 용이성
- 기능의 확장·보완에 대한 유연성

4. 설계내용의 최적화

1) 설계과정에서 분석모델에 내포되어 있는 불합리한 내용을 개선하여 시스템의 성능이 최대로 되도록 「설계 내용의 최적화」가 되어야 함

2) 설계 내용의 최적화를 위하여 고려 가능한 방안은 다음과 같음.
- 객체 사이의 관련성을 중복시켜서 효율성·편의성을 높여야 함.
- 수행순서를 변경시켜서 수행 횟수를 줄여야 함.
- 중복연산을 배제하기 위하여 유도된 속성 값을 보관시킴.

5. 제어 전략

1) 분석과정에서 작성한 상태 다이어그램을 구체적인 소프트웨어로 구현하는 방안을 결 정하는 것이 「소프트웨어 제어의 구현 전략」 임.

2) 소프트웨어 제어의 구현 전략은 다음과 같이 3 가지가 있음.
- 절차중심 기법 전통적 설계방식으로 프로그램이 제어 기능을 가짐.
- 이 방식에서는「상태 다이어그램」을 「프로그램 코드」로 변화시키는 요령은 다음과 같음.

① 주된 제어경로를 식별하여 따라 가야 한다.
② 주된 제어경로를 벗어났다가 다시 합쳐지는 분기조건의 경우를 식별해야 한다.
③ 주된 제어경로와 만나는 역경로(back paths)는 반복문이 되므로 식별해야 한다.

위의 3 조건에 해당하지 않는 경우에는 예외적인 사항들이기 때문에 에러 (error)로 처리하면 됨.

* 사건중심 기법 여기에는 「상태엔진(state engine)」이 필요.
- 보통 각 객체의 「인스탄스」 들은 「상태변수(state variable)」 를 가짐.
- 그런데 상태변수들은 시스템 전체를 제어하는 상태 엔진으로 부터 그 다음의 「상태」 나 「활동」 을 지시 받도록 되어 있음.

* 동시성 중심기법 이는 멀티타스킹(multi tasking) 방식으로 제어함.
- 보통 각 객체 는 독립된 「타스크」 로 활동하는데 이 경우 타스크들의 제어는 주로 OS에 의존함.

* 그러나 프로그래밍 언어(C++, Ada, Concurrent C, Pascal)의 특성을 이용하여 구현 할 수도 있음.

6. 상속성 조정

1) 객체를 설계하는 과정에서 이미 정의된 객체 클래스와 연산을 조정하여 상속성을 향상시켜야 하는 경우가 있음.
이 경우에 대응 방안들은 다음과 같다.
- 여러 객체 클래스에 독립적으로 분산된 유사한 기능의 연산들에 대하여 「객체·연산」 들을 재구성하여 「단일연산으로 구현」하는 방안을 모색해야 함.

* 설계단계에서「새로운 클래스」사이에 「공통적인 행위」 새로운 「클래스」 나 「기능」 의 추가가 불가피한 경우가 발생하면 일반화(generalization)와 같은 방안으로 해결할 수 있는 대책을 모색해야 함.
- 클래스 사이에 상속되는 과정에서 상호 완전한 상속이 되지 않을 경우에는 그 중에서 하나의 객체는 나머지 다른 객체의 「속성」 혹은 「보조객체」 로 조정해야 함.

7. 객체 사이의 관계 정의

1) 객체모델에 나타난 객체들 사이에 「관련(association)」을 소프트웨어에서 어떤 방법으로 구현할 것인가를 결정하는 것을 「객체 사이 관계정의」 라고 함.

2) 독립적인 객체의 연결은 「포인터(pointer)」 에 의해서 이루어짐.
- 그러므로 설계과정에서 객체모델에 나타난 「객체 사이의 관계」 는 「포인터」 를 이용하는 방법으로 표현해야 함.

3) 「포인터」를 이용한 「객체 사이의 관련」을 구현하는 전략은 다음과 같은 3 가 지가 있음.
① 두 개의 객체 사이를 양방향으로 탐색하기 위하여 관련되는 두 개의 객체 사 이에 「포인터」 를 추가하는 방식임.
② 관련되는 두 개의 객체를 연결시키기 위하여 객체의 연결조합 내용을 가지고 만 든 「관련객체」 를 추가해서 구현함.
③ 두 관련 객체 사이에 상속이 모호할 경우에는 그 중에서 하나의 객체를 상대객체의 속성으로 취급하여 포인터를 한 개만 사용해서 관련을 구현하는 방식임.

8. 객체 표현

1) 대체적으로 객체의 구현은 간단하지만 경우에 따라서는 객체의 표현방법, 데이터 타입의 선정 등은 다양할 수 있음.

2) 예를 들면 「학번」, 「군번」, 「주민등록번호」 등과 같은 클래스의 속성들은 「integer」 혹은 「string」 으로 될 수 있다는 사실을 의미함.

9. 패키징

1) 설계를 마친 클래스들은 기능이나 특성 등을 고려하여 구현의 용이성, 이해의 용이성, 유지보수의 편리성 등을 고려하여 패키징(packaging) 해야 함.

2) 분석단계에서 작성한 내용에 변화가 없는 것은 패키징이 간단함.

3) 설계과정에서 새로운 클래스가 추가로 정의되었거나, 분석과정에서 정의한 클래스에 대하여 분할이 많은 경우에는 이들을 포함하는 패키징이 이루어져야 함.
- 모듈 사이에 인터페이스는 최소화시켜야 함.
- 객체를 구성하는 요소 사이에 응집도(cohesion)를 최대화하도록 배려해야 함.
- 정보은익(information hiding)을 최대화시켜서 가능한 블랙박스(black box) 개념을 유지시킴.
- 관련이 강한 클래스는 동일한 모듈로 함.
- 정보교환이 빈번한 클래스는 동일한 모듈로 함.
- 조립구조 혹은 상속으로 관련된 클래스들은 동일한 모듈로 함.

학습내용2 : 코드와 요든의 객체지향 설계

1. 코드와 요든의 객체지향설계

1) 객체지향분석 모델이란.
- 구체화, 수정보완, 추가함으로써 객체지향설계 모델을 작성함.

2) 코드와 요든은 객체지향 설계 모델을 다음과 같이 4 가지 요소로 구분함.
- 그러므로 이 4 가지 분야에 대한 구체적 설계작업이 이루어져야 함.
- 문제영역(problem domain component)
- 인간 상호작용(human interaction component)
- 타스크 관리(task management component)

- 데이터 관리(data management component)

3) 코드와 요든이 객체지향기법에서 채택하는 「표기법(notation)」은 다음과 같음.
- 클래스와 객체의 표기
- 분류구조와 전체-부분 구조 표기
- 인스탄스 연결 표기
- 메시지 연결 등에 대해서는 각자 참고도서 참조요.

【학습정리】

1. 람바우의 객체지향 설계를 알아본다.

2. 코드와 요든의 객체지향 설계에 대하여 알아본다.

3. 객체지향분석 모델이란 무엇인지 파악한다.