4.6 자원관리

Saturday, November 11, 2023 11:47 AM

4장 설계 단계에서 다뤄야 할 문제들

- 인터페이스 : 어떤 서비스와 접근 권한을 제공할 것인가?
- 정보 은닉 : 어떤 정보를 드러내고, 어떤 정보를 숨길 것인가?
- 자원 관리 : 메모리나 기타 한정된 자원들을 누가 관리할 것인가?
- 에러처리 : 누가 에러를 감지하고, 누가 이 에러를 보고할 것인가? 보고는 어떻게 할 것인가? 에러를 감지했을 때 어떤 복구 방법을 시도할 것인가?

4.1 콤마 구분 값
4.2 프로토타입 라이브러리
4.3 다른 사림이 쓸 수 있는 라이브러리
4.4 C++구현
4.5 인터페이스 원칙

4.6 자원관리

자원관리가 필요한 이유는 무엇인가?

- 어떤 라이브러리(클래스나 패키지가 될 수도 있음)를 위한 인터페이스를 설계할 때 가장 어려운 점은 라이브러리를 소유하거나 그 라이브러리를 사용하는 호출자와 공유하는, 자원을 관리하는 문제이며, 이와 관련된 가장 분명한 자원은 메모리이다.
- 누가 공유 정보를 책임질지 결정해야 한다.
    - csvgetline은 원본 데이터를 리턴할까? 아니면 복사본을 만들까? (메모리를 해제하는 것도 사용자의 책임!)
    - 누가 입력 파일을 열고 닫을까? (파일을 여는 곳이 있으면 반드시 닫아야 하는 법!)
    - 4.6장에서는 csvgetline이 미리 열린 파일에 대한 FILE포인터를 인자로 삼아 호출되고, 작업이 끝났을 때 호출자가 이 파일 포인터를 닫는다고 가정

자원관리는 어떤 방식으로 할 수 있는가?

- 자원 관리는 초기화, 상태유지, 공유와 복사, 자원 해체(자원 해제)의 문제로 분류 가능
    - 초기화
        - CSV(comma-separated values) 프로토 타입의 경우
            - 포인터, 카운터 같은 변수들의 초기값을 설정할 때 정적 초기화를 이용
            - 루틴를 초기 상태로 되돌려 재 시작할 수 없기 때문에 한계가 있음
            - 이를 해결할 대안은 내부 값을 정확한 초기값으로 되돌리는 초기화 함수 제공
            - 이렇게 하면 재시작도 가능하지만 사용자가 명시적으로 초기화 함수를 호출해야만 하기 때문에 두번째 버전의 reset 함수를 공용으로 만들 것
        - C++나 자바의 경우

        - 생성자를 이용해 클래스의 데이터 멤버를 초기화
        - 생성자를 제대로 정의한다면 모든 데이터 멤버를 반드시 초기화하기 때문에, 초기화 되지 않은 클래스 객체를 만들 방법이 없어짐
        - 생성자를 여러 개 만들면 다양한 초기화 방법을 지원할 수 있음
- 상태유지, 공유와 복사
    - C버전의 csvgetline 프로그램은 입력 문자열(줄과 필드)을 가리키는 포인터를 리턴하기 때문에 호출자는 입력 문자열에 직접 접근이 가능
        - 이럴 경우에는 사본을 만들어 놓아야 함. C++버전에서는 사본이기 때문에 안전
    - 자바는 다른 개체를 참조할 때는 모두 참조 값을 이용
        - 이럴 경우에 참조 값을 복사본이라 착각할 수 있고 이것은 버그를 만드는 원인이 되기도 하기 때문에 clone 메서드를 만들어 두어 필요할 때 복사본으로 이용.
    - 최종정리 (finalization), 소멸(destruction) : 초기화 또는 생성과 반대되는 개념
        - 어떤 것을 사용할 필요가 없을 때 자원을 정리하여 시스템에 반환
        - 메모리를 반환하지 않으면 메모리 자원이 고갈되는 문제가 발생
        - 데이터를 버퍼에 저장했으면, 나중에 버퍼를 비워야 함
        - 표준 C라이브러리 함수는 정상 종료될 경우 자동으로 버퍼를 비우게 설정, 정상종료가 아닐 경우 버퍼를 비우도록 프로그래밍 해야 함
- 자원 해체 (자원을 결자해지 하라)
    - 의미.1: 결자해지 : 일을 맺은 자가 그것을 풀고, 일을 시작한 자가 마땅히 책임을 져야 한다.
    - 의미.2 : Free a resource in the same layer that allocated : 동일한 레이어에 할당된 리소스를 해제하라.
    - 자원 할당과 해제를 제어하는 방법 중 하나는 같은 라이브러리,패키지, 인터페이스를 써서 그 자원을 할당한 곳에서 해제까지 책임지는 것
        - CSV 라이브러리 : 라이브러리 호출자에서 파일을 닫기
        - C++ : 생성자와 소멸자를 이용
        - JAVA : 내장형 가비지 컬렉션(garbage collection) 제공
            - 메모리 관리가 수월
            - 새 객체를 메모리에 할당하면 프로그램에서 명시적으로 메모리를 해제할 방법은 없지만, 런타임 시스템이 계속 감시하면서 주기적으로 안 쓰이는 것들을 해제해 가용 메모리 풀에 반환
            - 사용 전략
                - 참조 횟수가 0이 되었을 때 해제
                - 메모리 풀에서 출발하여 참조하는 객체들을 추적하며 주기적으로 검사
            - 자동 가비지 컬렉션이 있다고 해도 설계할 때 메모리 관리를 해야 함, 언제 정확히 일어날 수 있는 지 예측할 수 없기 때문에
    - 문제가 생기는 것을 막기위해서는?
        - 재진입 가능한(reentrant) 코드를 작성
        - 재진입 가능이라는 말은 '코드가 몇 개씩 동시다발적으로 실행되더라도 제대로 동작해야 한다는 의미

- 좋은 멀티쓰레드 설계를 위한 열쇠는 컴포넌트들을 분리해서 반드시 잘 정의된 인터페이스를 통해 뭔가를 공유할 수 있도록 하는 것 - 부주의하게 공유된 라이브러리는 전체 모델을 파괴할 수도 있음
- 멀티 쓰레드는 상당히 큰 주제이고 프로그래밍 복잡도 증가로 다른 장에서 다룰 예정