042 단위 모듈(Unit Module) 구현

단위모듈은 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 기능을 모듈로 구현한 것

- 단위모듈로 구현되는 하나의 기능을 단위 기능이라고 부름.
  (모듈의 개수 = 기능의 개수)
- 독립적 컴파일이 가능하고, 처리/명령문, 데이터구조를 포함
  - 컴파일 : 컴퓨터가 이해할 수 있는 언어로 바꿔주는 과정

단위 모듈의 구현 순서

1. 단위 기능 명세서 작성
   복잡한 시스템을 단순하게 구현하는 추상화 →
   대형시스템을 분해하여 단위 기능별로 구분, 각 기능들을 계층적으로 구조화(구체화) →
   모듈의 독립적인 운용과 한 모듈 내의 정보가 다른 모듈에 영향을 주지 않도록 캡슐화(정보은닉)
2. 입/출력 기능 및 인터페이스 구현
   단위 모듈 간의 연동 또는 통신을 위한 입출력 데이터를 구현
   구현 시 사용자 인터페이스인 CLI, GUI와의 연동 고려
   입, 출력 기능 구현 시 네트워크나 외부 장치와의 입,출력은 무료로 공개되어 있는 **Open Source API**를 이용하면 간편하게 구현
   Open Source : 일정한 조건을 준수하면 누구나 사용, 수정, 재배포 허가
   ➢ IPC(Inter-Process Communication : 모듈 간 통신 방식을 구현하기 위해 사용되는 대표적인 프로그래밍 인터페이스 집합.
   -> IPC의 대표 메소드 **5**가지

Shared Memory	메모리를 공유하여 프로세스 간 통신
Socket	네트워크 소켓을 통해 네트워크를 경유하는 프로세스들 간의 통신
Semaphores	공유 자원에 대한 접근제어를 통해 프로세스 간 통신 수행

Pipes & named Pipes	Pipe라고 불리는 선입선출 형태로 구성된 메모리를 여러 프로세스가 공유하여 통신 수행
Message Queueing	메세지가 발생하면 이를 전달하는 형태, 프로세스 간 통신 수행

3. 알고리즘 구현

   입/출력 데이터를 바탕으로 단위 기능별 요구사항들을 구현 가능한 언어를 이용하여 모듈로 구현

디바이스 드라이버 모듈	하드웨어 주변 장치의 동작을 구현한 모듈
네트워크 모듈	네트워크 장비 및 데이터 통신을 위한 기능을 구현한 모듈
파일 모듈	컴퓨터 내부의 데이터 구조영역에 접근하는 방법을 구현한 모듈
메모리 모듈	파일을 프로세스의 가상메모리에 매핑/해제하는 방법, 프로세스 사이의 통신기능을 구현한 모듈
프로세스 모듈	하나의 프로세스 안에서 다른 프로세스를 생성하는 방법을 구현한 모듈

043 단위 모듈 테스트

4. 단위 모듈 테스트(단위 테스트 Unit Test)
   프로그램의 단위 기능을 구현하는 모듈이 정해진 기능을 정확히 수행하는지 검증, 시스템 수준의 오류는 잡아낼 수 없다.

- 화이트박스 테스트 / 블랙박스 테스트 기법 사용
  - 화이트박스 테스트 : 소프트웨어 혹은 제품의 내부 구조, 동작을 세밀하게 검사하는 테스트 방식
  - 블랙박스 테스트 : 소프트웨어의 내부 구조나 작동 원리를 모르는 상태에서 소프트웨어의 동작을 검사하는 방법
- 테스트 케이스(Test Case) : 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서, 명세 기반 테스트의 설계산출물에 해당됨.
    - 명세기반 테스트 : 사용자의 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 구현하고 있는지 확인하는 것. 테스트의 수행 증거로도 활용됨.
  - 단위 모듈 테스트하기 전 테스트에 필요한 입력데이터, 테스트조건, 예상결과 등을 모아 테스트케이스를 만듦.
  - 테스트케이스를 이용하지 않고 수행하는 직관적인 테스트는 특정 요소에 대한 검증이 누락되거나 불필요한 검증의 반복으로 인해 인력과 시간을 낭비할 수 있다.

식별자(Identifier)	항목 식별자, 일련번호
테스트 항목(Test Item)	테스트 대상(모듈 또는 기능)
입력 명세(Input Specification)	입력데이터 또는 테스트조건
출력 명세(Output Specification)	테스트케이스 수행 시 예상되는 출력 결과
환경설정(Environmental Needs)	필요한 하드웨어/소프트웨어의 환경
특수 절차 요구(Special Procedure Requirement)	테스트케이스 수행 시 특별히 요구되는 절차

의존성 기술(Inter-case Dependencies)	테스트케이스 간의 의존성

- 테스트 프로세스 : 테스트를 위해 수행하는 모든 작업들이 테스트의 목적과 조건을 달성할 수 있도록 도와주는 과정
- 테스트 프로세스 5단계
    1) 계획 및 제어 단계 : 테스트 목표를 달성하기 위한 계획 수립, 계획대로 진행되도록 제어
    2) 분석 및 설계 단계 : 테스트 목표를 구체화하여 테스트 시나리오와 테스트케이스를 작성하는 단계
        - 테스트 시나리오 : 테스트케이스를 적용하는 순서에 따라 여러개의 테스트케이스들을 묶은 집합, 테스트케이스들을 적용하는 구체적인 절차를 명세한 문서.
    3) 구현 및 실현 단계 : 효율적인 테스트 수행을 위해 테스트케이스들을 조합하여 테스트 프로시저에 명세하는 단계
        - 테스트 프로시저 : 테스트케이스의 실행 순서를 의미, 테스트 스크립트(Test Script)라고도 불림
    4) 평가 단계 : 테스트가 계획과 목표에 맞게 수행되었는지 평가 / 기록
    5) 완료 단계 : 이후의 테스트를 위한 참고 자료 및 테스트 수행에 대한 증거 자료로 활용하기 위해 수행과정과 산출물을 기록 및 저장.

044 개발 지원 도구

1. 통합 개발 환경(IDE; Integrated Development Environment)

통합 개발 환경은 개발에 필요한 환경, 즉 편집기(Editor), 컴파일러(Compiler), 디버거(Debugger) 등의 다양한 툴을 하나의 인터페이스에 통합하여 제공하는 것.

- 코드의 자동 생성 및 컴파일이 가능하고, 추가 기능을 위한 도구들을 다운로드하여 추가할 수 있음.
    - 컴파일(Compile) : 개발자가 작성한 고급언어로 된 프로그램을 컴퓨터가 이해할 수 있는 목적 프로그램으로 번역하여 컴퓨터에서 실행가능한 형태로 변환하는 작업
- 코드를 실행하거나 테스트할 때 오류가 발생한 부분을 시각화하므로 수정 용이
- 외부의 다양한 서비스와 연동하여 개발에 편의를 제공, 필요한 정보 공유 가능
- IDE를 지원하는 대표적인 도구

프로그램	운영체제
이클립스(Eclipse)	Windows, Linux, MacOS 등
비주얼 스튜디오(Visual Studio)	Windows
엑스코드(Xcode)	MacOS, iOS
안드로이드 스튜디오(Android Studio)	Windows, Linux, MacOS
IDEA	Windows, Linux, MacOS

2. 빌드도구

빌드는 소스코드 파일들 -> 제품 소프트웨어로 변환하는 과정 또는 결과물

- 이 과정에서 필요한 전처리(Preprocessing), 컴파일(Compile)등의 작업들을 수행하는 소프트웨어
    - 전처리(Preprocessing) : 컴파일에 앞서 코드에 삽입된 주석을 제거 or 매크로들을 처리하는 과정

- 대표적인 도구

Ant(Another Neat Tool)	- 아파치 소프트웨어 재단(Apache Software Foundation)에서 개발한 소프트웨어, 자바 프로젝트의 공식적인 빌드도구로 사용되고 있음 - XML 기반의 빌드 스크립트를 사용, 자유도&유연성이 높아 복잡한 빌드환경에 대처 가능 - 정해진 규칙이나 표준이 없어 개발자가 모든것을 정의, 스크립트의 재사용이 어려움
Maven	- Ant와 동일한 아파치 소프트웨어 재단에서 개발되었고, Ant의 대안. - 규칙이나 표준이 존재하여 예외사항만 기록하면 됨, 컴파일과 빌드 동시 수행 가능 - 의존성(Dependency)을 설정하여 라이브러리 관리
Gradle	- 기존의 Ant/Maven 보완하여 개발된 빌드도구 - 한스 도커(Hans Dockter) 외 6인의 개발자 공동 개발 - 안드로이드 스튜디오의 공식 빌드도구로 채택된 소프트웨어 - Maven과 동일하게 의존성 활용, 그루비(Groovy) 기반의 빌드 스크립트 사용

- XML : W3C(World Wide Web Consortium)가 채택한 인터넷 표준 언어, 인터넷 환경에 적합하도록 구성된 메타 언어

    -> 메타언어 : 프로그램 언어의 규칙을 기술하는데 사용하는 언어

- 라이브러리(Library) : 코드,API,클래스,값,자료형 등 다양한 자원들을 모아둠.

- 의존성(Dependency) : Maven/Gradle에서 라이브러리를 관리할 때 사용하는 명령어,

    빌드 스크립트 안에 사용하고자 하는 라이브러리를 <dependency> 예약어로

등록하면, 빌드 수행 시 인터넷상의 라이브러리 저장소에서 해당 라이브러리를 찾아 코드에 추가해줌.

- 그루비(Groovy) : 자바를 기반으로 여러 프로그래밍 언어들의 장점을 모아 만들어진 동적 객체지향 프로그래밍언어.

3. 기타 협업 도구

협업도구는 개발에 참여하는 사람들이 서로 다른 작업 환경에서 원활히 프로젝트를 수행하도록 도와주는 도구(Tool), 협업 소프트웨어, 그룹웨어(Groupware) 등으로도 불림.

- 협업도구에는 일정관리, 업무흐름 관리, 정보 공유, 커뮤니케이션 등의 업무보조 도구가 포함됨.
- 웹 기반, PC, 스마트폰 등 다양한 플랫폼에서 사용할 수 있도록 제공됨
- 협업도구에 익숙하지 않거나 이용할 의지가 없다면 협업도구가 오히려 협업의 방해요소가 될 수 있음.

- 협업도구의 종류

프로젝트 및 일정 관리	- 전체 프로젝트와 개별 업무들의 진행상태, 일정 등을 공유하는 기능 제공 - 구글 캘린더, 분더리스트(Wunderlist, 트렐로(Trello), 지라(Jira), 플로우(Flow) 등등
정보 공유 및 커뮤니케이션	- 주제별로 구성원들을 지목하여 방을 개설한 후 정보를 공유, 대화하는 것이 가능 - 파일관리가 간편, 의사소통이 자유로운 것이 특징 - 종류 : 슬랙(Slack), 잔디(Jandi), 태스크월드(Taskworld) 등등
디자인	디자이너가 설계한 UI나 이미지의 정보들을 코드화 하여 개발자에게 전달하는 기능 제공 - 종류 : 스케치(Sketch), 제플린(Zeplin) 등
기타	- 아이디어 공유에 사용되는 에버노트(Evernote)

	- API를 문서화하여 개발자들 간 협업을 도와주는 스웨거(Swagger) - 깃(Git)의 웹호스팅 서비스인 깃허브(GitHub)