중단원	소단원	번호	문제	키워드	답
		1	개발환경 구축 개념	개발환경 구축	개발 도구와 서버의 선정이 이루어져야 하고, 개발에 사용되는 도구들의 사용 편의성과 성능, 라이선스 등에 대한 내용도 파악해야 한다
		2	개발환경 구축 시 파악 사항	개발환경 구축	* 개발 도구와 서버의 선정 * 개발에 사용되는 도구들의 1. 사용 편의성 2. 성능 3. 라이선스 등에 대한 내용 파악
		3	개발 도구의 분류	개발환경 구축	* 구현 도구 ex) Eclipse, IntelliJ, Sprint Tool Suite, NetBeans, Visual Studio * 테스트 도구 ex) xUnit, PMD, Findbugs, Cppcheck, Sonar * 형상관리 도구 ex) CVS, Subversion, Git * 빌드 도구 ex) Ant, Maven, Gradle
		4	개발 도구의 분류	개발 도구의 분류	* 빌드 도구 - 작성한 코드의 빌드 및 배포를 수행하는 도구 - 각각의 구성요소와 모듈에 대한 의존성 관리를 지원 * 구현 도구 - 개발자의 코드 작성과 디버깅, 수정 등과 같은 작업을 지원하는 도구 - 프로그램을 개발할 때 가장 많이 사용되는 도구 * 테스트 도구 - 코드의 기능 검증과 전체의 품질을 높이기 위해 사용되는 도구 - 코드의 테스트. 테스트에 대한 계획, 수행 및 분석 등의 작업 가능 * 형상관리 도구 - 개발자들이 작성한 코드와 리소스 등 산출물에 대한 버전 관리를 위한 도구 - 프로젝트 진행 시 필수로 포함되는 도구
		5	개발 도구의 분류 중 구현 도구 개념 설명	개발 도구의 분류	개발자의 코드 작성과 디버깅, 수정 등과 같은 작업을 지원하는 도구이다
		6	개발 도구의 분류 중 형상관리 도구 개념 설명	개발 도구의 분류	1. 개발자들이 작성한 코드와 리소스 등 산출물에 대한 버전 관리를 위한 도구이다 2. 소프트웨어 프로젝트에서 나오는 결과물을 관리하는 소프트웨어이다
		7	개발환경 구성 요소	개발환경 구성 요소	1. 하드웨어 개발환경 - 서버 하드웨어 개발환경 - 클라이언트 하드웨어 개발환경 2. 소프트웨어 개발환경 3. 형상관리

01. 개발환경 구축	1. 개발환경 구축	8	서버 하드웨어 개발환경 종류	서버 하드웨어 개발환경	* 웹 서버 * 웹 애플리케이션 서버 * 데이터베이스 서버 * 파일 서버
		9	서버 하드웨어 개발환경 종류	서버 하드웨어 개발환경	* 웹 서버 - HTTP를 이용한 요청/응답을 처리 - 웹 상의 정적 콘텐츠(CSS, Javascript, Image)를 처리 - WEB-WAS-DB의 3계층 구조를 실무에서 활용 - 주요 제품으로 Apache 웹 서버, IIS 웹 서버, Google Web Server,Nginx 등 존재 * 웹 애플리케이션 서버(WAS) - 동적 콘텐츠(Servlet, JSP)를 처리/제공하기 위해 사용 - 주요 제품으로 Tomcat, Weblogic, Jeus, Resin 등 존재 * 데이터베이스 서버 - 데이터의 수집, 저장을 위한 용도로 사용 - 연계되는 주요 DBMS로 MySQL, Oracle, MS-SQL, DB2 등이 존재 * 파일 서버 - 파일 저장 하드웨어로 물리 저장장치를 활용한 사례 - 대용량 HDD, SSD 등의 장치가 존재
		10	웹 애플리케이션 서버(WAS) 개념	서버 하드웨어 개발환경	- 동적 콘텐츠(Servlet, JSP)를 처리/제공하기 위해 사용 - 주요 제품으로 Tomcat, Weblogic, Jeus, Resin 등 존재 - 사용자 요청 스레드를 처리하고, 데이터베이스에 접속하여 SQL 쿼리문에 대한 결과 값을 반환하는 역할을 수행하는 서버이다
		11	클라이언트 하드웨어 개발환경 종류	클라이언트 하드웨어 개발환경	* 클라이언트 프로그램 * 웹 브라우저 * 모바일 앱 * 모바일 웹
		12	클라이언트 하드웨어 개발환경 종류	클라이언트 하드웨어 개발환경	* 클라이언트 프로그램 - 설치를 통해 사용자와 커뮤니케이션하는 프로그램 - Visual Basic, C#, Delphi 등으로 개발되어 사용 - .exe(Windows), .dmg(Mac) 확장자를 가진 실행 파일 - 설치 통해 사용자에게 필요한 기능 제공 * 웹 브라우저 - 웹 서비스의 형태로 서버에서 웹 애플리케이션을 응답 시 브라우저를 통해 사용자와 커뮤니케이션 - 일반적인 형태의 웹 서비스가 해당 * 모바일 앱 - 모바일 디바이스에 설치되어 활용되는 애플리케이션 - 앱 스토어, 안드로이드 마켓 등을 통해 다운로드 가능 * 모바일 웹 - 웹 브라우저와 동일한 형태로 모바일상 웹 브라우저를 통해 서비스를 제공 - 모바일에 최적화되어 제공되는 웹사이트가 해당

		13	클라이언트 프로그램 개념	클라이언트 하드웨어 개발환경	* 클라이언트 프로그램 - 설치를 통해 사용자와 커뮤니케이션하는 프로그램 - Visual Basic, C#, Delphi 등으로 개발되어 사용 - .exe(Windows), .dmg(Mac) 확장자를 가진 실행 파일 - 설치 통해 사용자에게 필요한 기능 제공
		14	모바일 앱 개념	클라이언트 하드웨어 개발환경	* 모바일 앱 - 모바일 디바이스에 설치되어 활용되는 애플리케이션 - 앱 스토어, 안드로이드 마켓 등을 통해 다운로드 가능
		15	소프트웨어 개발환경	소프트웨어 개발환경	* 운영체제 * 미들웨어 * DBMS
		16	소프트웨어 개발환경	소프트웨어 개발환경	* 운영체제 - 서버의 하드웨어를 사용자 관점에서 편리하고 유용하게 사용하기 위한 소프트웨어 - 프로젝트의 성격에 따른 운영체제 사용 * 미들웨어 - 컴퓨터와 컴퓨터 간의 연결을 쉽고 안전하게 할 수 있도록 해주고 이에 대한 관리를 도와주는 소프트웨어 - 자바 기반 환경에서 JVM을 설치하여 컨테이너로의 기능을 주로 이용 * DBMS - 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고, 데이터베이스를 관리해주는 소프트웨어 - 데이터의 저장 및 활용을 위해 DBMS를 설치
		17	미들웨어 개념	소프트웨어 개발환경	* 미들웨어 - 컴퓨터와 컴퓨터 간의 연결을 쉽고 안전하게 할 수 있도록 해주고 이에 대한 관리를 도와주는 소프트웨어 - 자바 기반 환경에서 JVM을 설치하여 컨테이너로의 기능을 주로 이용
		18	형상관리 개념	형상 관리	소프트웨어 개발을 위한 전체 과정에서 발생하는 모든 항목의 변경 사항을 관리하기 위한 활동이다
		19	형상 관리 목적	형상 관리	* 프로젝트 생명주기 동안 제품의 무결성과 변경에 대한 추적성을 확보할 수 있음 * 프로젝트 변경이 발생 되었을 때 처리하는 메커니즘을 제공
		20	형상관리 절차	형상 관리	1. 형상 식별 2. 형상 통제 3. 형상 감사 4. 형상 기록
		21	형상관리 절차	형상 관리	1. 형상 식별 2. 형상 통제 3. 형상 감사 4. 형상 기록
		22	베이스라인 개념	형상 관리	개발 과정의 각 단계별 산출물을 검토, 평가, 조정, 처리 등의 변화를 통제하는 시점의 기준이며, 각 형상 항목들의 기술적 통제 시점이다
		1	모듈 개념	공통 모듈 구현	독립된 하나의 소프트웨어 또는 하드웨어 단위를 지칭하는 용어
		2	모듈화	공통 모듈 구현	소프트웨어의 성능을 향상시키거나 복잡한 시스템의 수정, 재사용, 유지 관리 등이 용이하도록 기능 단위의 모듈로 분해하는 설계 및 구현 기법
		3	높은 독립성 모듈은 수정 시에도 다른 모듈에 영향을 끼치지 않는다. 이때 모듈의 독립성을 높게 만드는 방법은?	공통 모듈 구현	독립성을 높이려면, 모듈의 결합도는 약하게(낮게), 응집도는 강하게(높게), 모듈의 크기는 작게 만들어야 한다

		4	모듈화 기법	공통 모듈 구현	* 루틴 (Routine) * 메인 루틴 (Main Routine) * 서브 루틴 (Sub Routine)
		5	모듈화 기법	공통 모듈 구현	* 루틴 (Routine) - 소프트웨어에서 특정 동작을 수행하는 일련의 코드로 기능을 가진 명령들의 모임 * 메인 루틴 (Main Routine) - 프로그램의 주요한 부분 - 전체의 개략적인 동작 절차를 표시하도록 만들어진 루틴 - 메인 루틴은 서브 루틴을 호출함 * 서브 루틴 (Sub Routine) - 메인 루틴에 의해 필요할 때마다 호출되는 루틴
		6	공통 모듈 구현 개념	공통 모듈 구현	* 소프트웨어 개발에 있어 기능을 분할하고 추상화하여 성능을 향상시키고 유지보수를 효과적으로 하기 위한 공통 컴포넌트 구현 기법 * 인터페이스 모듈, 데이터베이스 접근 모듈 등 필요한 공통 모듈을 구현한다 * 모듈 간의 결합도는 줄이고, 응집도는 높인 공통 모듈 구현을 권장하고 있다
		7	응집도 개념	소프트웨어 모듈 응집도	- 모듈의 독립성을 나타내는 개념 - 모듈 내부 구성요소 간 연관 정도 - 정보은닉 개념의 확장개념 - 하나의 모듈은 하나의 기능을 수행하는 것을 의미
		8	응집도 유형	소프트웨어 모듈 응집도	* 응집도 낮음 (Bad) - 우연적 응집도 (Coincidental Cohesion) - 논리적 응집도 (Logical Cohesion) - 시간적 응집도 (Temporal Cohesion) - 절차적 응집도 (Procedural Cohesion) - 통신적 응집도 (Communication Cohesion) - 순차적 응집도 (Sequential Cohesion) - 기능적 응집도 (Functional Cohesion) * 응집도 높음 (Good)
		9	응집도 유형 1	소프트웨어 모듈 응집도	* 우연적 응집도 (Coincidental Cohesion) - 모듈 내부의 각 구성요소들이 연관이 없을 경우 * 논리적 응집도 (Logical Cohesion) - 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우 * 시간적 응집도 (Temporal Cohesion) - 연관된 기능이라기보다는 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우 * 절차적 응집도 (Procedural Cohesion) - 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성요소들이 그 기능을 순차적으로 수행할 경우

	1. 공통 모듈 구현	10	응집도 유형 2	소프트웨어 모듈 응집도	* 통신적 응집도 (Communication Cohesion) - 동일한 입력과 출력을 사용하여 다른 기능을 수행하는 활동들이 모여 있을 경우 * 순차적 응집도 (Sequential Cohesion) - 모듈 내에서 한 활동으로부터 나온 출력값을 다른 활동에 사용할 경우 * 기능적 응집도 (Functional Cohesion) - 모듈 내부의 모든 기능이 단일한 목적을 위해 수행되는 경우
		11	결합도 유형	소프트웨어 모듈 결합도	* 결합도 낮음 (Good) - 자료 결합도 (Data Coupling) - 스탬프 결합도 (Stamp Coupling) - 제어 결합도 (Control Coupling) - 외부 결합도 (External Coupling) - 공통 결합도 (Common Coupling) - 내용 결합도 (Content Coupling) * 결합도 높음 (Bad)
		12	결합도 유형 1	소프트웨어 모듈 결합도	* 자료 결합도 (Data Coupling) - 모듈 간의 인터페이스로 전달되는 파라미터를 통해서만 모듈 간의 상호 작용이 일어나는 경우 * 스탬프 결합도 (Stamp Coupling) - 모듈 간의 인터페이스로 배열이나 객체, 구조 등이 전달되는 경우 * 제어 결합도 (Control Coupling) - 단순 처리할 대상인 값만 전달되는 게 아니라 어떻게 처리를 해야 한다는 제어 요소가 전달되는 경우
		13	결합도 유형 2	소프트웨어 모듈 결합도	* 외부 결합도 (External Coupling) - 두 개의 모듈이 외부에서 도입된 데이터 포맷, 통신 프로토콜, 또는 디바이스 인터페이스를 공유할 경우 * 공통 결합도 (Common Coupling) - 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 전역 변수를 갱신하는 식으로 상호작용하는 경우 * 내용 결합도 (Content Coupling) - 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우

02. 공통 모듈 구현		14	모델(Model), 컨트롤러(Controller), 뷰(View) 개념	MVC 패턴 역할	* 모델(Model) - 애플리케이션이 무엇을 할 것인지를 정의 - 내부 비즈니스 로직을 처리하기 위한 역할 * 뷰(View) - 화면에 무엇인가를 보여주기 위한 역할 - 모델, 컨트롤러가 보여주려고 하는 것들을 화면에 처리 * 컨트롤러(Controller) - 모델이 어떻게 처리할지를 알려주는 역할 - 뷰에 명령을 보내어 화면 요청 결과를 전달
		15	시스템 복잡도 최적하 하는 방법	팬인(Fan-In) 및 팬아웃(Fan-Out)	팬인은 높게, 팬아웃은 낮게 설계해야 한다
		16	팬인(Fan-In)	팬인(Fan-In) 및 팬아웃(Fan-Out)	* 개념 - 어떤 모듈을 제어(호출)하는 모듈의 수 * 모듈 숫자 계산 - 모듈 자신을 기준으로 모듈에 들어오면 팬인(in) * 고려사항 - 팬인이 높으면 재사용 측면에서 설계가 잘되었지만, 단일 장애점 발생 가능 - 팬인이 높으면 관리 비용 및 테스트 비용 증가
		17	팬아웃(Fan-Out)	팬인(Fan-In) 및 팬아웃(Fan-Out)	* 개념 - 어떤 모듈을 제어(호출)되는 모듈의 수 * 모듈 숫자 계산 - 모듈 자신을 기준으로 모듈에서 나가면 팬아웃(out) * 고려사항 - 팬아웃이 높을 경우는 불필요한 모듈 호출 여부 검토 필요 - 팬아웃이 높을 경우는 단순화 여부 검토 필요
		1	공통 모듈 테스트 개념	공통 모듈 테스트	* 공통 모듈 테스트를 위해 IDE(Integrated Development Environment) 도구를 활용하여 개별 공통 모듈에 대한 디버깅을 수행한다 * 공통 모듈 테스트는 화이트 박스 기법을 활용한다 * 대표적인 단위테스트 도구인 Junit을 활용하여 테스트 코드를 구현한다
		2	통합 개발환경 개념	공통 모듈 테스트	통합 개발환경인 IDE(Integrated Development Environment)는 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어이다 ex) 이클립스, Visual Studio 등
		3	디버깅 개념	공통 모듈 테스트	컴퓨터 프로그램의 논리적인 오류를 검출하여 제거하는 과정을 뜻한다
		4	공통 모듈 테스트 종류	공통 모듈 테스트	* 화이트 박스 * 메서드 기반 테스트 * 화면 기반 테스트 * 테스트 드라이버 / 테스트 스텁

	2. 공통 모듈 테스트	5	공통 모듈 테스트 종류	공통 모듈 테스트 종류	* 화이트 박스 - 응용 프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트 방식 - 소스 코드를 보면서 테스트 케이스를 다양하게 만들어 테스트를 수행 * 메서드 기반 테스트 - 공통 모듈의 외부에 공개된 메서드 기반의 테스트 - 메서드에 서로 다른 파라미터 값을 호출하면서 다양한 테스트를 수행 * 화면 기반 테스트 - 사용자용 화면이 있는 경우, 각각의 화면단위로 단위모듈을 개발 후에 화면에 직접 데이터를 입력하여 테스트를 수행 - 화면기반 테스트는 화면과 연계된 서비스, 비즈니스 컴포넌트 및 공통 컴포넌트를 한꺼번에 단위 테스트에 참여 - 사용자 시나리오에 기반한 공통 모듈 테스트를 할 수 있는 장점이 있음 * 테스트 드라이버 / 테스트 스텁 - 기능을 테스트할 수 있는 화면 또는 하위 모듈이 구현되지 않은 경우 테스트 드라이버, 테스트 스텁을 통해 테스트를 수행 - 테스트 드라이버 : 하위 모듈은 있지만 상위 모듈은 없는 경우 사용하는 기법 - 테스트 스텁 : 상위 모듈은 있지만 하위 모듈은 없는 경우 사용하는 기법
		6	화이트 박스 테스트 개념	공통 모듈 테스트 종류	* 응용 프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트 방식
		7	메서드 기반 테스트 개념	공통 모듈 테스트 종류	* 공통 모듈의 외부에 공개된 메서드 기반의 테스트 * 메서드에 서로 다른 파라미터 값을 호출하면서 다양한 테스트를 수행
		8	화면 기반 테스트 개념	공통 모듈 테스트 종류	* 사용자용 화면이 있는 경우, 각각의 화면단 위로 단위 모듈을 개발 후에 화면에 직접 데이터를 입력하여 테스트를 수행 * 화면기반 테스트는 화면과 연계된 서비스, 비즈니스 컴포넌트 및 공통 컴포넌트를 한꺼번에 단위 테스트에 참여 * 사용자 시나리오에 기반한 공통 모듈 테스트를 할 수 있는 장점이 있음
		9	Assert 메서드 개념	공통 모듈 테스트 구현	* assertEquals(a, b); * assertEquals(a, b, c); * assertSame(a, b); * assertTrue(a); * assertNotNull(a); * assertArrayEquals(a, b);

		10	Assert 메서드 개념	공통 모듈 테스트 구현	* assertEquals(a, b); - 객체 a와 b가 일치함을 확인 * assertEquals(a, b, c); - 객체 a와 b가 일치함을 확인 - a:예상값, b:결과값, c:오차값 * assertSame(a, b); - 객체 a와 b가 같은 객체임을 확인 * assertTrue(a); - 조건 a가 참인지 여부를 확인 * assertNotNull(a); - 객체 a가 null이 아님을 확인 * assertArrayEquals(a, b); - 배열 a와 b가 일치함을 확인
03. 서버 프로그램 구현	1. 서버 프로그램 구현	1	개발환경 구축 개념	서버 프로그램 구현	* 백엔드 * 프론트엔드
		2	백엔드 개념	서버 프로그램 구현	* 사용자와 만나지 않고 프론트엔드와 연동하여 핵심 로직을 처리하는 영역 * DB나 인터페이스 등을 통해 시스템 구성 실체에 접근
		3	프론트엔드 개념	서버 프로그램 구현	* 사용자(user)의 화면에 나타나는 웹 화면 영역 * 웹 페이지를 그리는 기술 (JSP, Javascript, CSS, HTML, Node.js, React.js, Angular.js 등) 활용
		4	서버 프로그램 구현 절차	서버 프로그램 구현	* 백엔드 (Back-end) 1. DTO/VO 구현 2. SQL문 구현 3. DAO 구현 4. Service 구현 5. Controller 구현 * 프론트엔드 (Front-end) 6. 화면 구현
		5	서버 프로그램 세부 구현 내용	서버 프로그램 세부 구현	VO, SQL문, DAO, Service, Controller 순서대로 구현
		6	SQL문 구현	서버 프로그램 세부 구현	1. 데이터베이스 테이블 정의 및 생성 2. MyBatis XML 구현
		7	ORM(Object Relation Mapping) 도구 중 하나인 MyBatis 개념	SQL문 구현	객체지향 언어인 자바의 관계형 DB 프로그래밍을 좀 더 쉽게 할 수 있도록 도와주는 개발 프레임워크이다

		8	DAO, DTO, VO 개념 차이	서버 프로그램 세부 구현	* DAO (Data Access Object) - 특정 타입의 데이터베이스에 추상 인터페이스를 제공하는 객체로 세부내용 노출 없이 데이터 조작 * DTO (Data Transfer Object) - 프로세스 사이에서 데이터를 전송하는 객체로 데이터 저장/회수 외에 다른 기능은 없다 * VO (Value Object) - 간단한 엔티티를 의미하는 작은 객체 가변 클래스인 DTO와 달리 고정 클래스를 가진다
04. 배치 프로그램 구현	1. 배치 프로그램	1	배치 프로그램 개념	배치 프로그램	* 사용자와의 상호작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 프로그램이다 * 개별적으로 어떤 요청이 있을 때마다 실시간으로 통신하는 것이 아닌 한꺼번에 일괄적으로 대량 건을 처리한다
		2	배치 프로그램 개념	배치 프로그램	* 배치 프로그램은 개별적으로 어떤 요청이 있을 때마다 실시간으로 통신하는 것이 아닌 한꺼번에 일괄적으로 대량 건을 처리한다 * 대량건의 데이터를 특정 시간에 일괄적으로 처리하는 것이 핵심이다
		3	배치 프로그램 유형	배치 프로그램	* 이벤트 배치 - 사전에 정의해 둔 조건 충족 시 자동으로 실행 * 온디맨드 배치 - 사용자의 명시적 요구가 있을 때마다 실행 * 정기 배치 - 정해진 시점에 정기적으로 실행
		4	배치 스케줄러 개념	배치 스케줄러	일괄 처리를 위해 주기적으로 발생하거나 반복적으로 발생하는 작업을 지원하는 도구
		5	배치 스케줄러 종류	배치 스케줄러	* 스프링 배치 - 스프링 프레임워크의 DI, AOP, 서비스 추상화 등 스프링 프레임워크의 3대 요소를 모두 사용할 수 있는 대용량 처리를 제공하는 스케줄러 * 쿼츠 스케줄러 - 스프링 프레임워크에 플러그인되어 수행하는 작어(Job)과 실행 스케줄을 정의하는 트리거를 분리하여 유연성을 제공하는 오픈 소스 기반 스케줄러
		6	Cron 표현식	배치 스케줄러	스케줄러를 실행시키기 위해 작업이 실행되는 시간 및 주기 등을 설정하게 되는데 이때 Cron 표현식을 통해 배치 수행시간을 설정한다
		7	Cron 표현식	배치 스케줄러	1. 초(Seconds) : 0~59, 특수문자 2. 분(Minutes) : 0~59, 특수문자 3. 시간(Hours) : 0~23, 특수문자 4. 일 (Days) : 1~31, 특수문자 5. 월 (Months) : 1~12, JAN~DEC, 특수문자 6. 요일 (Week) : 1~7, SUN-SAT, 특수문자 7. 연도(Year)(생략가능) : 1970~2099, 특수문자

		8	Cron 표현식	배치 스케줄러	* : 모든 수 ? : 해당 항목을 미사용 - : 기간 설정 , : 특정 기간 설정 / : 시작기간과 반복간격 설정 L : 마지막 기간에 동작 W : 가장 가까운 평일에 동작 # : 몇 번째 주, 요일 설정
		9	0 0 12 * * * 크론 표현식이 가리키는 의미 서술	Cron 표현식	스케줄러를 매일 12시에 실행한다
		10	0 0 7 ? * MON-SAT 크론 표현식이 가리키는 의미 서술	Cron 표현식	스케줄러를 월~토 7시 0분 0초에 실행한다
		11	0 /15 * * * * 크론 표현식이 가리키는 의미 서술	Cron 표현식	매 15분마다 스케줄러를 실행한다
		12	0 0 12 15W * * 크론 표현식이 가리키는 의미 서술	Cron 표현식	15일이 일요일이면 16일에, 토요일이면 14일에 스케줄러 실행
		13	0 /15 5-7 * * * 크론 표현식이 가리키는 의미 서술	Cron 표현식	스케줄러를 매일 5시부터 7시까지 10분 간격으로 실행
		14	배치 프로그램 설계 확인 사항	배치 프로그램 설계	* 배치 프로그램 관리대장 확인 * 배치 설계서 확인
		15	배치 설계서	배치 프로그램 설계	p. 4-30 참고 배치 프로그램 설계 단계에서 프로그램 관리 대장의 Id와 일치하는 배치 설계서를 확인한다. 배치 설계서를 통해 작업 내역을 참고하여 배치 프로그램을 구현한다
72 개					