JAVA

JAVA는 네트워크상에서 쓸 수 있도록 미국의 썬 마이크로 시스템즈가 개발한 객체 지향 프로그래밍 언어
JAVA의 특징
a. 자바가상머신(JVM)만 설치하면 컴퓨터의 운영체제(OS)에 상관없이 작동한다.(즉, 운영체제에 독립적)
b. 기본 자료형을 제외한 모든 요소들이 객체로 표현
c. 객체 지향 개념의 특징인 캡슐화, 상속, 다형성이 잘 적용된 언어
d. Garbage Collector를 통한 자동적인 메모리 관리
e. 멀티쓰레드(Multi-thread)를 지원
f. 다양한 Open 라이브러리들이 존재한다
g. OOP(객체 지향 언어) : 부품에 해당하는 객체들을 먼저 만들고, 이것들을 하나씩 조립해 전체 프로그램을 완성하는 개발 기법

? main메서드는 왜 static인가요?
static은 메모리 선언을 사용하지 않아도 사용할 수 있습니다. main메서드는 자바가상머신(JVM)에 의해 호출되는 것이므로 반드시 static으로 선언되어 미리 올라가 있어야 합니다. 만일 메모리에 있지 않다면 시작점인 main메서드를 호출하려고 할 때 메모리에 없기 때문에 실행되지 않습니다.

? 캡슐화란 무엇인가요
관련된 데이터와 메서드를 하나의 단위로 묶는 원리입니다. 그로 인해 캡슐내부와 외부를 구별하게 됩니다. 캡슐화를 하게 되면 클래스의 필드 값에 권한을 설정할 수 있습니다. 또한 사용자는 데이터가 클래스에 어떻게 저장되는지 알 수 없습니다. 그리고 클래스의 결합도가 낮아져 재사용이 용이하게 됩니다.

? 직렬화란 무엇인가요
메모리에 있는 객체를 보조기억장치에 저장할 수 있도록 바이트 형태로 변환하는 것을 말합니다. 객체가 생성되어 데이터가 적재되는 메모리는 순간적이기 때문에 영구적으로 보관하기 위해 직렬화를 사용합니다.

? 제너릭이란 무엇인가요
클래스를 선언할 때 타입을 결정하지 않고 객체를 생성할 때 유동적인 타입으로 재사용하기 위한 것을 말합니다.

? 리플렉션이란 무엇인가요
리플렉션은 컴파일러를 무시하고 런타임 상황에서 메모리에 올라간 클래스나 메서드등의 정의를 동적으로 찾아서 조작할 수 있는 일련의 행위를 말합니다. 즉 동적인 언어의 특징이라 말 할 수 있습니다. 프레임워크에서 유연성이 있는 동작을 위해 자주 사용하기도 합니다.

? 자바란 무엇인가요
자바란 객체지향 프로그래밍 언어로써 가장 중요한 특징은 운영체제에 독립적이란 것입니다. 자바로 작성된 프로그램은 운영체제의 종류에 관계없이 실행이 가능합니다. 그 이유는 자바를 실행하기 위한 가상 머신인 JVM이 있기 때문입니다. 다른 애플리케이션은 프로그램 실행 시 바로 OS로 가는 반면 자바 애플리케이션은 각 운영체제에 맞는 JVM을 거쳐 OS로 진행되기 때문에 프로그램 수정 없이 실행 가능합니다.

자바를 만든 사람은 ?
“제임스 고슬링”

변수란?
하나의 값을 저장할수 있는 메모리 공간

객체와 클래스의 차이점?
클래스(Class) : 현실 세계의 객체의 속성과 동작을 추려내 필드와 메서드로 정의한 것으로 "아직 메모리가 할당되지 않은 상태"
객체(Object) : 이 Class라는 설계도를 기반으로 실제 메모리가 잡힌 것을 의미하며 이런 객체를 조합해 전체 프로그램을 완성해 나가는 방식을 OOP(객체지향 프로그래밍)이라고 한다.

객체지향이란? 그 특징은?
현실세계의 객체를 필드와 메서드로 정의한 Class를 기반으로 실제 메모리가 잡혀 만들어진 부품과 같은 객체들을 조합해 전체 프로그램을 완성해 나가는 개발 기법
특징
캡슐화, 은닉화 : 외부 객체에서 구현방식은 알 수 없도록 숨기고 별도로 접근할 수 있는 getter/setter 메서드를 통해 접근하도록 하는 방식

상속 : 부모 Class를 자식이 접근할 수 있도록 물려 받는 방식
다형성 : 부모 클래스 타입으로 해당 부모를 상속받는 여러 자식 class를 대입할 수 있는 성질
등을 들 수 있다.

자바의 메모리 영역을 간단하게 설명하라
메서드 영역 : static 변수, 전역변수, 코드에서 사용되는 Class 정보 등이 올라간다. 코드에서 사용되는 class들을 로더로 읽어 클래스별로 런타임 필드데이터, 메서드 데이터 등을 분류해 저장한다.
스택(Stack) : 지역변수, 함수(메서드) 등이 할당되는 LIFO(Last In First Out) 방식의 메모리
힙(Heap) : new 연산자를 통한 동작할당된 객체들이 저장되며, 가비지 컬렉션에 의해 메모리가 관리되어 진다.

? JDK란 무엇인가요
자바 프로그램 개발도구로써 개발을 위한 클래스, 컴파일러, 실행 및 배포도구를 포함하여 개발을 위한 전반적인 환경을 제공하는 것입니다.

2. OOP(객체지향 프로그래밍)
OOP란 Object-Oriented Programming의 약어로써 객체지향 프로그래밍을 의미
데이터를 객체로 취급하여 프로그램에 반영한 것이며, 순차적으로 프로그램이 동작하는 기존의 것들과는 다르게 객체와 객체의 상호작용을 통해 프로그램이 동작하는 것을 말한다.
OOP 특징
a. 객체지향 프로그래밍은 코드의 재사용성이 높다.
b. 코드의 변경이 용이
c. 직관적인 코드분석
d. 개발속도 향상
e. 상속을 통한 장점 극대화

3. Object(객체)
Object(객체)는 OOP에서 데이터(변수)와 그 데이터에 관련되는 동작(함수). 즉 절차, 방법, 기능을 모두 포함한 개념
예)기차역에서 승차권을 발매하는 경우, 실체인 '손님'과 동작인 '승차권 주문'은 하나의 객체이며, 실체인 '역무원'과 동작인 '승차권 발매'도 하나의 객체이다. 같은 성질, 같은 구조와 형태를 가지는 객체는 등급으로 정의하고 등급에 속하는 객체는 그 등급의 인스턴스라고 한다.

다형성이란?
서로 다른 클래스로부터 만들어진 객체지만 같은 부모의 Class 타입으로 이들을 관리할 수 있는(=대입될 수 있는) 성질

? 오버라이딩이란 무엇인가요
부모 클래스에게 상속 받은 것들을 다시 자신의 클래스에서 새로이 재정의 하는 것을 말합니다. 재정의 한 것은 자신의 클래스 내부에서만 영향을 끼치며 부모클래스에서는 영향을 끼치지 않습니다. 할머니클래스, 부모클래스, 자식클래스의 구조라면 자식클래스는 할머니,부모 클래스의 것들을 모두 상속 받을 수 있으며, 할머니와 부모클래스의 같은 변수가 있다면 부모클래스를 물려 받게 됩니다.

? 오버로딩이란 무엇인가요
상속이 아닌 하나의 클래스 내에서 이름이 같은 여러개의 메서드를 정의하는 것입니다. 이름이 같기 때문에 호출 시에 구분 방법은 매개변수입니다. 매개변수의 수, 배치(순서), 타입 이 달라야 합니다.

? 상속이란 무엇인가요
기존 클래스의 기능을 유지하면서 추가적인 기능을 추가하여 클래스를 만들고 싶을 때 사용하는 방법은 상속입니다. 새로운 클래스를 생성할 때 상위 클래스를 지정함으로써, 상위 클래스의 모든 기능, 속성을 제공받고 자신의 클래스에는 부가적인 기능, 속성을 추가 할 수 있습니다. 상속은 코드를 간결화하며, 재사용성을 높일 수 있습니다.

? 자바의 데이터 타입에 대해 설명하시오 (Primitive type , Reference type)
기본형은 실제 값을 저장하는 공간을 말하며 종류는 8가지가 있습니다. 이외의 모든 타입을 참조형 이라고 하며 실제 값이 저장된 곳의 주소를 저장하는 공간을 의미합니다.

? 다형성이란 무엇인가요
하나의 클래스나 함수가 다양한 방식으로 동작이 가능한 것을 말합니다. 하나의 메시지가 전달되었을 때 수신자가 누구냐에 따라 각각 다른 기능을 수행합니다. 자바에서 오버로딩, 오버라이딩 등이 있습니다. 쉽게 예를 들면 저는 하나이지만 학교에 있으면 학생이고 집에서는 아들이라는 구성원입니다. 이렇게 생각하면 쉽다고 생각합니다.

4. Overloading vs Overriding (한번공부할때 확실히 차이를 알아둘 것, 평소에 알고 있어도 헷갈림)
Overloading(오버로딩)
- 같은 이름의 메소드를 여러개 정의하는 것
- 매개변수의 타입이 다르거나 개수가 달라야 한다.
* return type과 접근 제어자는 영향을 주지 않음.

Overriding(오버라이딩)
- 상속에서 나온 개념
- 상위 클래스(부모 클래스)의 메소드를 하위 클래스(자식 클래스)에서 재정의.

5. Servlet vs JSP
Servlet - 자바 언어로 웹 개발을 위해 만들어진 것으로, Container가 이해할 수 있게 구성된 순수 자바 코드로만 이루어진 것(Html in JAVA)
JSP(Java Server Page) - html기반에 JAVA코드를 블록화하여 삽입한 것(JAVA in Html)으로 Servlet을 좀 더 쉽게 접근할 수 있도록 만들어 진 것

6. JDBC
Java Data Base Connection의 약자로 JAVA 언어를 통해 데이터 베이스에 접근 할 수 있는 프로그래밍을 의미

Request 전송방식에는 어떤것들이 있는지 아시나요?
Get 방식 : URL의 쿼리문자열에 데이터를 같이 전달하는 방식으로 데이터 길이에 제한이 있고, 보안에 취약하다.
POST 방식 : 헤더에 데이터를 넣어 보내기 때문에 보안에 조금 더 유리하고 데이터 길이에 제한이 없다. 하지만, Get에 비해 다소 느리다.
DELETE 방식 : RESTFUL에서 삭제 기능을 할 때 주로 사용된다.
PUT/PUSH 방식 : RESTFUL에서 수정 작업을 할 때 주로 사용된다.

RESTFUL이란?
해당 URL만 보더라도 바로 어떤 작업을 하는지를 알 수 있도록 하나의 데이터는 하나의 URL을 갖도록 작업하는 방식

7. Get과 Post 방식
Get 방식
- 클라이언트에서 서버로 데이터를 전달할 때, 주소 뒤에 "이름"과 "값"이 결합된 스트링 형태로 전달
- 주소창에 쿼리 스트링이 그대로 보여지기 때문에 보안성이 떨어진다.
- 길이에 제한이 있다.(=전송 데이터의 한계가 있다.)
- Post방식보다 상대적으로 전송 속도가 빠르다.

Post 방식
- 일정 크기 이상의 데이터를 보내야 할 때 사용한다.
- 서버로 보내기 전에 인코딩하고, 전송 후 서버에서는 다시 디코딩 작업을 한다.
- 주소창에 전송하는 데이터의 정보가 노출되지 않아 Get방식에 비해 보안성이 높다.
- 속도가 Get방식보다 느리다.
- 쿼리스트링(문자열) 데이터 뿐만 아니라, 라디오 버튼, 텍스트 박스 같은 객체들의 값도 전송가능.

Get과 Post 차이점
- Get은 주로 웹 브라우저가 웹 서버에 데이터를 요청할 때 사용
- Post는 웹 브라우저가 웹 서버에 데이터를 전달하기 위해 사용.
- Get을 사용하면 웹 브라우저에서 웹 서버로 전달되는 데이터가 인코딩되어 URL에 붙는다.
- Post방식은 전달되는 데이터가 보이지 않는다.
- Get방식은 전달되는 데이터가 255개의 문자를 초과하면 문제가 발생할 수 있다.
- 웹서버에 많은 데이터를 전달하기 위해서는 Post 방식을 사용하는 것이 바람직하다.

캐시(cache) 와 세션(Session)의 공통점과 차이점은?
공통점 : 둘 다 사용자의 데이터를 저장한다.
차이점
- 캐시 : 캐시는 Client 컴퓨터에 저장했다 서버 요청시 네트워크를 타고 서버로 전달되기 때문에 보안에 취약하다.
- 세션 : 세션은 서버에 저장되고 브라우저 단위로 관리된다. 캐시에 비해 보안성이 좋다.

8. Session과 Cookie
Session과 Cookie 사용 이유
- 현재 우리가 인터넷에서 사용하고 있는 HTTP프로토콜은 연결 지향적인 성격을 버렸기 때문에 새로운 페이지를 요청할 때마다 새로운 접속이 이루어지며 이전 페이지와 현재 페이지 간의 관계가 지속되지 않는다. 이에 따라 HTTP프로토콜을 이용하게 되는 웹사이트에서는 웹페이지에 특정 방문자가 머무르고 있는 동안에 그 방문자의 상태를 지속시키기 위해 쿠키와 세션을 이용한다.

Session
- 특정 웹사이트에서 사용자가 머무르는 기간 또는 한 명의 사용자의 한번의 방문을 의미한다.
- Session에 관련된 데이터는 Server에 저장된다.

- 웹 브라우저의 캐시에 저장되어 브라우저가 닫히거나 서버에서 삭제시 사라진다.
- Cookie에 비해 보안성이 좋다.

Cookie
- 사용자 정보를 유지할 수 없다는 HTTP의 한계를 극복할 수 있는 방법
- 인터넷 웹 사이트의 방문 기록을 남겨 사용자와 웹 사이트 사이를 매개해 주는 정보이다.
- Cookie는 인터넷 사용자가 특정 웹서버에 접속할 때, 생성되는 개인 아이디와 비밀번호, 방문한 사이트의 정보를 담은 임시 파일로써, Server가 아닌 Client에 텍스트 파일로 저장되어 다음에 해당 웹서버를 찾을 경우 웹서버에서는 그가 누구인지 어떤 정보를 주로 찾았는지 등을 파악할 때 사용된다.
- Cookie는 Client PC에 저장되는 정보기 때문에, 다른 사용자에 의해서 임의로 변경이 가능하다.(정보 유출 가능, Session보다 보안성이 낮은 이유)

Q. 보안성이 낮은 Cookie 대신 Session을 사용하면 되는데 안하는 이유?
A. 모든 정보를 Session에 저장하면 Server의 메모리를 과도하게 사용하게 되어 Server에 무리가 감

? MVC패턴이란 무엇인가요
애플리케이션을 크게 model, view, controller 세 영역으로 구분하여 영역 간의 결합도를 최소화한 논리적인 패턴입니다. 가장 큰 특징이며 장점은 비즈니스로직과 프리젠테이션로직이 분리 된다는 것입니다. 그로 인해 디자이너와 개발자의 영역이 분리 됨 으로써 작업의 분업화를 할 수 있습니다. 또한 유지보수에도 용이합니다.

9. MVC 패턴
MVC란?
- 객체지향프로그래밍에서, MVC란 사용자 인터페이스를 성공적이며 효과적으로 데이터 모형에 관련 시키기 위한 방법론 또는 설계 방식중 하나이다. MVC방식은 자바, Smalltalk,
- MVC 패턴은 목적 코드의 재사용에 유용한 것은 물론, 사용자 인터페이스와 응용프로그램 개발에 소요되는 시간을 현저하게 줄여주는 형식이라고 많은 개발자들이 평가하고 있다.

MVC 구성요소
Model - 소프트웨어 응용과 그와 관련된 고급 클래스 내의 논리적 데이터 기반 구조를 표현. 이 목적 모형은 사용자 인터페이스에 관한 어떠한 정보도 가지고 있지 않다.(data 처리와 접근)
View - 사용자 인터페이스 내의 구성요소들을 표현(사용자에게 보여지는 화면)
Controller - Model과 View를 연결하고 있는 클래스를 대표, Model과 View 내의 클래스들 간 정보 교환하는데 사용.(Model과 View를 제어)

? 추상클래스란 무엇인가요
하나이상의 추상메서드를 포함한 클래스입니다. 추상클래스는 객체를 생성 할 수 없으며 멤버변수, 일반 메서드, 상수 등도 가질 수 있습니다.

? 인터페이스란 무엇인가요
클래스가 상속을 통해 구현하기에 한계가 있는 경우, 자바에서 불가능한 다중상속을 흉내내기 위한 도구로써 사용됩니다. 추상클래스보다 추상정도가 높으며 추상메서드와 상수만을 가질 수 있습니다. Implements를 통해 구현합니다.

추상 메서드란? 추상 클래스란?
추상메서드 : 메서드의 정의부만 있고 구현부는 있지 않은 메서드
추상 클래스 : 추상메서드를 적어도 하나 이상 가지고 있는 클래스로 자식클래스에서 오버라이딩(재정의)가 필요한 추상메서드를 가지고 있기 때문에 객체화 할 수 없다.

인터페이스(interface)란?
인터페이스는 모든 메서드가 구현부가 없는 추상메서드로 이루어진 클래스로, abstract 키워드를 붙이지 않아도 자동으로 모든 메서드는 추상메서드로 정의가 된다. 또한 변수도 자동으로 final static 키워드가 붙게 된다.
-> 왜 사용하는가 ?
팀작업시 개발코드 부분과 객체가 서로 통신하는 접점 역할을 지원하게 되는데, 이는 개발코드에선 객체의 내부 구조를 모르더라도 인터페이스의 메서드 명만 알고 있으면 되기 때문이다. 이를 통해 얻을 수 있는 장점은 해당 메서드를 통해 나오는 결과물을 알고 있기 때문에 다른 팀의 작업을 기다리고 있지 않아도 되며, 또한 해당 객체가 수정될 경우 개발코드 부분은 수정을 하지 않아도 된다. 또한, 부가적으로 객체를 파일에 쓰기 위해 Serializable 인터페이스를 구현하거나, Collections.sort()를 하기 위해서 Comparable 인터페이스를 상속하는 것, Cloneable 을 구현하는 것처럼 특정 작업을 하겠다라는 "Mark"역할을 해주기도 한다.

0. Interface, Abstract
Interface
- 일종의 추상 클래스

- 오직 추상메서드와 상수만을 멤버로 갖는다.
- Implements 키워드를 사용
- 상속의 관계가 없는 클래스간 서로 공통되는 로직을 구현하여 쓸 수 있도록한다.
- Extends는 하나의 클래스만 상속 가능하나 Interface는 다중 상속이 가능하다.

Abstract
- 추상메서드를 하나 이상 가진 클래스
- 자신의 생성자로 객체 생성 불가능
- 하위 클래스를 참조하여 상위 클래스의 객체를 생성
- 하위 클래스를 제어하기 위해 사용

Interface vs Abstract

공통점
- new 연산자로 인스턴스 생성 불가능.
- 프로토타입만 있는 메서드를 갖는다.
- 사용하기 위해서는 하위클래스에서 확장/구현 해야 한다.

차이점
- 사용하는 키워드가 다르다.
- Abstract는 일반 메서드를 사용할 수 있지만, Interface는 메서드 선언만 가능하다.

11. Call by Reference, Call by Value

Call by Reference - 매개 변수의 원래 주소에 값을 저장하는 방식. 클래스 객체를 인수로 전달한 경우
Call by Value - 인수로 기본 데이터형을 사용. 주어진 값을 복사하여 처리하는 방식. 메서드 내의 처리 결과는 메서드 밖의 변수에 영향을 미치지 않는다.

12. Static의 의미
- 클래스가 로딩될 때, 메모리 공간을 할당하는데 처음 설정된 메모리 공간이 변하지 않음을 의미
- 객체를 아무리 많이 만들어도 해당 변수는 하나만 존재(객체와 무관한 키워드)

컬렉션프레임워크?

Collection 인터페이스
List 인터페이스 : 배열과 유사하되, 추가할때마다 자동으로 Boundary를 늘려주는 구조로, 중복된 데이터를 허용하며, 순서가 존재한다.
 - ArrayList : 배열로 구현됐으며, 인접해 있기 때문에 데이터 조회에 매우 빠르다 하지만, 빈번한 삽입, 삭제시 새로 배열을 만들고 데이터를 옮겨야 하기 때문에 LinkedList에 비하여 속도가 느리다.
 - LinkedList : 링크 구조로 되어 있기 때문에 조회는 ArrayList에 비해 느리지만, 삽입 삭제시 링크를 끊고 새로 추가되는 데이터에 링크만 연결하면 되기 때문에 삽입, 삭제에 유리하다.
 - Vector : 구현 방식은 ArrayList와 유사하지만 Vector를 개선한 것이 ArrayList이다. 또한 Vector의 경우에는 ArrayList와 달리 Synchronized(동기화)가 걸려 있어 여러 쓰레드에서 동시에 접근할 수 없다.
Set 인터페이스 : 집합처럼 중복된 데이터를 허용하지 않으며, 순서가 없다. 또한, 객체 내부의 중복된 데이터를 배제하고 싶은 경우 Object 클래스의 equals 메서드와 hashCode 메서드의 재정의가 반드시 필요하다.
- HashSet
- TreeSet : 순서가 있는 HashSet으로 이진 트리 구조로 만들어 졌다. 순서에 맞게 정렬되어 저장되기 위해서 Comparable을 구현해야한다.

Map 인터페이스 : key와 value 쌍으로 데이터를 저장하며, key는 중복될 수 없고, value는 중복 저장이 가능하다.
- HashMap
- TreeMap
- Properties : key value 쌍으로 저장되지만 value의 타입이 String만 가능하다.
- Hashtable : HashMap과 구조는 같으며, 단지 Synchronized(동기화) 되어져 있다는 점이 다른점이다.

디자인 패턴? (아는데로 말하시오)

싱글톤(SingleTone Pattern) : 대표적으로 Calendar 객체나 dataSource 객체처럼 객체가 하나만 생성되어야 하는 경우 전체 코드에서 하나의 객체만 존재할 수 있도록 이미 생성된 객체가 있으면 그 객체를 사용하도록 하는 방식입니다.
팩토리 패턴(Factory pattern) : 객체간 의존성을 줄이기 위해 객체의 생성과 데이터 주입만 담당하는 Factory Class를 정의하고 개발 코드 부분에서는 생성된 객체를 가져다 사용함으로서 의존성을 줄이는 방식입니다.
옵저버 패턴(Observer Pattern) : 기후 정보처럼 RSS 수신시 하나의 객체가 변하면 다른 객체에 객체가 변했다는 사항을 알려주어야 할 경우에 주로 사용됩니다.

? 프레임워크란 무엇인가요

소프트웨어 제작을 편리하게 할 수 있도록 미리 뼈대를 이루는 클래스와 인터페이스를 제작하여 이것을 모아둔 것입니다. 프레임워크를 사용하게 되면 개발 생산성이 증가하며 품질이 향상되고 유지보수가 편리하다는 장점이 있습니다. 반면 익숙해지는데 시간이 오래 걸리며 유연성이 부족하게 됩니다.

13. Framework
- 특정 형태의 소프트웨어 문제를 해결하기 위해 상호 협력하는 클래스프레임과 인터페이스 프레임의 집합
- 특정한 틀을 만들어놓고 거기에 살을 붙여 놓음으로써 프로그램을 만들어 작업시간을 줄여주는 것이다.
- 프레임워크는 특정 개념들의 추상화를 제공하는 여러 클래스나 컴포넌트로 구성된다.
- 프레임워크는 이렇게 추상적인 개념들이 문제를 해결하기 위해 같이 작업하는 방법을 정의한다.
- 프레임워크 컴포넌트 들은 재사용이 가능하다.
- 프레임워크는 좀 더 높은 수준에서 패턴을 조작한다.

* 프레임워크가 중요한 이유는 객체지향 개발을 하게 되면서 개발자의 취향에 따라 다양한 프로그램이 나오게 되었다. 프로그램 개발에 투입되는 개발자도 점점 늘어남에 따라 전체 시스템의 통합성, 일관성이 부족하게 되었기 때문이다. 그래서 개발자의 자유를 제한하기 위해 프레임워크를 도입했다.

프레임워크가 가져야할 특징

a. 개발자들이 따라야할 가이드라인을 가진다.
b. 개발할 수 있는 범위가 정해져 있다.
c. 개발자를 위한 다양한 도구들이 지원된다.

프레임워크의 장/단점

장점 - 개발 시간을 줄일 수 있고 오류로부터 자유로울 수 있다.
단점 - 프레임워크에 너무 의존하면 개발 능력이 떨어져서 프레임워크 없이 개발하는 것이 불가능해지는 점이다.

14. Garbage Collection(가비지 컬렉션)
시스템에서 더이상 사용하지 않는 동적 할당된 메로리 블럭을 찾아 자동으로 다시 사용 가능한 자원으로 회수하는 것으로
시스템에서 가비지컬렉션을 수행하는 부분을 가비지 컬렉터라 부른다.

15. Primitive type과 Reference type
Primitive type - 변수에 값 자체를 저장
정수형 byte, short, int, long
실수형 float, double
문자형 char
논리형 boolean
* Primitive type은 Wrapper Class를 통해 객체로 변형할 수 있다.
예) int→Integer, char→Character(int와 char를 제외한 Primitive type의 다른 자료형들은 맨 앞 알파벳을 대문자로 바꿔주면 된다. float→Float)
Reference type - 메모리상에 객체가 있는 위치를 저장
종류 - Class, Interface, Array 등

16. Wrapper Class 사용 이유?
기본 data 타입은 객체가 아니어서 Object로 받는 다형성을 지원할 수가 없다. 하지만, 메서드에서 실재로 기본데이터 타입을 다형성으로 넘겨주어야 하는 경우가 빈번히 발생하는데 이때, 기본 데이터 타입을 객체로 변환시켜 전달하기 위해 사용되며 최근에는 AUTO Boxing, AUTO UnBoxing이 지원된다.
Primitive type으로 표현할 수 있는 간단한 데이터를 객체로 만들어야 할 경우가 있는데 그러한 기능을 지원하는 클래스

? 스프링이 뭔가요
자바언어를 기반으로 다양한 애플리케이션을 개발하기 위한 경량급 프레임워크입니다. 경량급이란 말은 스프링자체가 아주 가볍거나 작은 규모의 코드로 이뤄졌다는 것이 아니라 불필요하게 무겁지 않다라는 의미입니다. 그리고 개발 중에 테스트가 쉽다는 특징이 있습니다.

17. Spring Framework(스프링 프레임워크)
자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 애플리케이션 프레임워크(Framework)
자바 엔터프라이즈 개발을 편하게 해주는 오픈 소스 경량급 애플리케이션 프레임워크
자바 개발을 위한 프레임워크로 종속 객체를 생성해주고, 조립해주는 도구
자바로 된 프레임워크로 자바SE로 된 자바 객체(POJO)를 자바EE에 의존적이지 않게 연결해주는 역할

스프링 특징 간단히

- 크기와 부하의 측면에서 경량.
- 제어 역행(IoC)이라는 기술을 통해 애플리케이션의 느슨한 결합을 도모
- 관점지향 프로그래밍(AOP)을 위한 풍부한 지원
- 애플리케이션 객체의 생명 주기와 설정을 포함하고 관리한다는 점에서 일종의 컨테이너(Container)라고 할 수 있음
- 간단한 컴포넌트로 복잡한 애플리케이션을 구성하고 설정할 수 있음

스프링 특징 자세히

a. 경량 컨테이너로서 자바 객체를 직접 관리.
   각각의 객체 생성, 소멸과 같은 라이프 사이클을 관리하며 스프링으로부터 필요한 객체를 얻어올 수 있다.
b. 스프링은 POJO(Plain Old Java Object) 방식의 프레임워크.

일반적인 J2EE 프레임워크에 비해 구현을 위해 특정한 인터페이스를 구현하거나 상속을 받을 필요가 없어 기존에 존재하는 라이브러리 등을 지원하기에 용이하고 객체가 가볍다.
c. 스프링은 제어의 역행(IoC : Inversion of Control)을 지원.
컨트롤의 제어권이 사용자가 아니라 프레임워크에 있어서 필요에 따라 스프링에서 사용자의 코드를 호출한다.
d. 스프링은 의존성 주입(DI : Dependency Injection)을 지원
각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결시켜준다.
e. 스프링은 관점 지향 프로그래밍(AOP : Aspect-Oriented Programming)을 지원
따라서 트랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리할 수 있다.
f. 스프링은 영속성과 관련된 다양한 서비스를 지원
iBatis나 Hibernate 등 이미 완성도가 높은 데이터베이스 처리 라이브러리와 연결할 수 있는 인터페이스를 제공한다.
g. 스프링은 확장성이 높음.
스프링 프레임워크에 통합하기 위해 간단하게 기존 라이브러리를 감싸는 정도로 스프링에서 사용이 가능하기 때문에 수많은 라이브러리가
이미 스프링에서 지원되고 있고 스프링에서 사용되는 라이브러리를 별도로 분리하기도 용이하다.

스프링에서 DI란?

DI는 Dependency Injection(의존성 주입)의 약자로, 객체들 간의 의존성을 줄이기 위해 사용되는 Spring의 IOC 컨테이너의 구체적인 구현 방식입니다. DI는 기존처럼 개발코드 부분에서 객체를 생성하는 것이 아니라, 팩토리 패턴처럼 객체의 생성과, 데이터를 주입만 담당하는 Factory에 해당 하는 별도의 공간에서 객체를 생성하고 데이터간의 의존성을 주입해 개발코드에서는 이를 가져다 씀으로서 의존성을 줄이는 방식입니다. 이때, Factory 패턴의 Factory Class의 역할을 스프링의 환경설정 파일이 담당합니다.

스프링의 AOP란?

AOP는 Aspect Oriented Programming 관점 지향 프로그래밍의 약자로, 기존의 OOP(객체 지향 프로그래밍)에서 기능별로 class를 분리했음에도 불구하고, 여전히 로그, 트랜잭션, 자원해제, 성능테스트 메서드 처럼 공통적으로 반복되는 중복코드가 여전히 발생하는 단점을 해결하고자 나온 방식으로 이러한 공통 코드를 "횡단 관심사"라 표현하며 개발코드에서는 비지니스 로직에 집중하고 실행시에 비지니스 로직 앞, 뒤 등 원하는 지점에 해당 공통 관심사를 수행할 수 있게 함으로서 중복 코드를 줄일 수 있는 방식입니다.

? 쓰레드란 무엇인가요

자바 프로그램을 구성하는 명령문은 순서대로 하나씩 처리되는 것이 기본입니다. 이러한 실행흐름을 쓰레드라고 합니다. 둘 이상의 흐름을 갖도록 만들고 싶다면 멀티 쓰레드 프로그램을 사용하면 됩니다.
? 멀티쓰레드란? 장점은 무엇인가요
하나의 프로그램에서 둘 이상의 작업이 필요로 할 때 사용합니다. 자원을 효율적으로 사용가능하며 작업이 분리되어 코드가 간결해 집니다.
? 쓰레드 생성방법이 무엇인가요
첫 번째로는 쓰레드 클래스를 상속받거나 인터페이스를 구현 받는 방법이 있습니다. 두 번째로는 구현한 클래스 내부에서 인스턴스 생성 즉시 쓰레드를 생성시키는 방법이 있습니다.

18. Thread

Thread(쓰레드) - 프로세스내에서 동시에 실행되는 독립적인 실행 단위를 말함, 장점으로는 자원을 많이 사용하지 않고 구현이 쉬우며 범용성이 높다
Process(프로세스) - 운영체제에서 실행중인 하나의 프로그램(하나 이상의 쓰레드를 포함한다.)
Thread 장점
- 빠른 프로세스 생성
- 적은 메모리 사용
- 쉬운 정보 공유
Thread 단점
- 교착상태에 빠질 수 있다.
* 교착상태 - 다중프로그래밍 체제에서 하나 또는 그 이상의 프로세스가 수행 할 수 없는 어떤 특정시간을 기다리고 있는 상태.
Thread와 Process 차이
여러 분야에서 '과정' 또는 '처리'라는 뜻으로 사용되는 용어로 컴퓨터 분야에서는 '실행중인 프로그램'이라는 뜻으로 쓰인다.
이 프로세스 내에서 실행되는 각각의 일을 스레드라고 한다. 프로세스 내에서 실행되는 세부 작업 단위로 여러 개의 스레드가 하나의 프로세스를 이루게 되는 것이다.

프로세스(Process)와 쓰레드(Thread)의 차이?
프로세스 : OS가 메모리 등의 자원을 할당해준 실행중인 프로그램을 가리킨다. 이때, 각각의 프로세스는 서로 메모리 공간을 독자적으로 갖기 때문에 서로 메모리 공간을 공유하지 못한다. 따라서 공유하기 위해서는 IPC(InterProcess Communication)과 같은 방식이 필요하다.

쓰레드 : 쓰레드는 프로세스 내에서 프로세스의 자원을 가지고 실제로 일하는 "일꾼"과 같으며 각 쓰레드는 독자적인 Stack 메모리를 갖고 그 외의 자원(메모리)는 프로세스 내에서 공유하게 된다.

19. 접근제한자(public > protected > default > private)
public - 접근 제한이 없다.(같은 프로젝트 내에 어디서든 사용가능)
protected - 같은 패키지 내, 다른 패키지에서 상속받아 자손클래스에서 접근 가능/ 같은 class 내부 + 상속받은 자식에서는 부모 class에 접근이 가능하다.
default - 아무 것도 선언하지 않은 경우로 같은 패키지 내부에서만 접근이 가능하다.
private - 같은 클래스 내에서만 접근 가능

20. 소켓 통신(TCP/UDP)
TCP(Transmission Control Protocol)
- 연결형 서비스 제공
- 높은 신뢰성 보장
- 연결의 설정(3-way handshaking)
- 연결의 해제(4-way handshaking)
- 데이터 흐름 제어, 혼잡 제어
- 전이중, 점대점 서비스(양방향 송수신 서비스)

UDP(User Datagram Protocol)
- 비연결형 서비스 제공
- 신뢰성이 낮음
- 데이터의 전송 순서가 바뀔 수 있음
- 데이터 수신 여부 확인 안함(3-way handshaking과 같은 과정 X)
- TCP보다 전송속도가 빠름

21. Stack, Queue
STACK
- LIFO(Last In First Out)의 후입선출 구조
- push();를 이용한 데이터 입력, pop();을 이용한 데이터 출력
- 예) 시스템 스택 : 함수의 호출과 복귀 순서는 스택의 구조를 응용하여 관리
- 역순 문자열 만들기, 수식의 괄호 검사, 수식의 후위 표기법 변환

QUEUE
- FIFO(First In First Out)의 선입선출 구조
- enQueue();를 이용한 데이터 입력, deQueue();를 이용한 데이터 출력
- 예) 우선순위가 같은 작업 예약(인쇄 대기열), 선입선출이 필요한 대기열(티켓 카운터)

* Linear Queue(선형큐)는 메모리 재사용이 불가능 이러한 문제점을 보완하여 Circular Queue(원형 큐)가 나옴

22. Singleton Design Patter(싱글톤 디자인 패턴, 싱글톤 패턴)
- 클래스 인스턴스가 하나만 만들어지도록 하고, 그 인스턴스에 대한 전역 접근을 제공한다.

23. Database에서 Index란?
인덱스는 데이터베이스 분야에 있어서 테이블에 대한 동작의 속도를 높여주는 자료 구조를 일컫는다.
인덱스는 테이블 내의 1개의 컬럼, 혹은 여러 개의 컬럼을 이용하여 생성될 수 있다.
고속의 검색 동작뿐만 아니라 레코드 접근과 관련 효율적인 순서 매김 동작에 대한 기초를 제공한다.
인덱스를 저장하는 데 필요한 디스크 공간은 보통 테이블을 저장하는 데 필요한 디스크 공간보다 작다.

데이터베이스에서 테이블과 클러스터에 연관되어 독립적인 저장 공간을 보유하고 있는 객체(object)이다.
사용자는 데이터베이스에 저장된 자료를 더욱 빠르게 조회하기 위하여 인덱스를 생성하고 사용한다.

DB에서 자료를 검색하는 두 가지 방법
FTS(Full Table Scan) : 테이블을 처음 부터 끝까지 검색하는 방법
Index Scan : 인덱스를 검색하여 해당 자료의 테이블을 액세스 하는 방법.

SI란?
System Integration의 약자로 시스템 통합 사업으로 고객의 기존 전산시스템을 통합하거나 새로운 시스템을 구축하는 작업입니다.

DB

? ORM이란 무엇인가요
ORM이란 객체와 관계형 데이터베이스을 중간에서 매핑하는 것입니다. SQL만 잘 작성하더라도 충분히 개발 할 수 있지만 SQL를 직접 작성하는 번거로운 작업을 줄여주고, 직접 계산하고 연산하는 부분을 지양해야 하기 때문에 이용됩니다.

? 정규화란 무엇인가요
정규화란 테이블의 데이터들간의 종속성, 중복성 등으로 인해 예기치 못한 오류를 제거 하는 과정이라 할 수 있습니다. 정규화를 진행했을 때 장점은 DB의 일관성을 향상시킬수 있습니다. 또한 DB의 논리적 구조를 견고하게 만들 수 있습니다. 하지만 테이블의 숫자가 늘어나고 결국 join 연산의 비용이 증가 하는 단점을 가질 수 있습니다.

? 무결성 제약조건이란 무엇인가요
데이터의 정확성과 일관성을 보장하기 위해 테이블 생성 시에 각 칼럼에 대해 정의하는 규칙을 의미합니다. 그로 인해 프로그래밍 과정을 줄일 수 있고, 데이터 오류 발생 가능성을 줄여줍니다.

? ERD란 무엇인가요
ERD란 계략적으로 데이터 및 데이터들의 관계를 표현한 도식화된 그림입니다. 조직의 데이터를 이해하고, 이를 응용시스템에 이용하고자 ERD를 작성합니다.

? 컬럼에 인덱스를 생성하는게 좋은지, 생성하지 않는게 좋은지 기술하시오
상황에 따라 다르다고 할 수 있습니다. 인덱스를 생성하려는 컬럼이 분포도가 좋아 활용도가 향상되거나 수정이 빈번하지 않다면 인덱스를 생성하는 것이 좋습니다. 하지만 인덱스 컬럼이 비교되기 전에 변형이 일어난 경우나 부정형으로 조건을 기술한 경우 인덱스를 생성 하지 않는 것이 좋습니다. 지나친 인덱스 선언은 많은 오버헤드(어떤 처리를 하기 위해 들어가는 간접적인 처리시간, 메모리 등을 말한다)를 발생 시킬 수 있습니다.

? 저장 프로시저(stored procedure)란 무엇인가
저장 프로시저란 한 개 이상의 복잡한 SQL Query문들을 데이터베이스에 저장하고 필요한 경우 호출해서 사용하는 객체입니다. 저장 프로시저를 사용 하면 트래픽을 감소 시킬 수 있고, 보안이 강화되며, 코드의 재사용이 가능하며, 유지 관리에 용이합니다.

? DBMS란 무엇인가요
데이터를 적절하고 효율적인 관리의 필요성으로 인해 등장한 체계적으로 데이터를 관리하는 시스템입니다. DBMS를 사용함으로써 사용자 중심의 데이터 처리가 가능하며, 중복성 통제, 데이터의 일관성 유지 등의 장점을 가지고 있습니다. 종류로는 관계형, 객체 지향형, 객체관계형 데이터베이스가 있습니다.

? JOIN은 언제 사용합니까
저희는 데이터 무결성과 중복성, 종속성을 해결하기 위해 테이블을 정규화 시켰습니다. 그렇기 때문에 여러 테이블에서 정보를 필요로 합니다. 그 때 사용하는 것이 JOIN입니다. 당연히 JOIN을 사용하기 위해서는 테이블간의 관계가 설정되어 있어야 하며 이를 통해 테이블간의 행을 비교 하면서 필요로 하는 정보를 가져 올 수 있습니다.

? Primary key와 Foreign key를 비교하여 설명하시오
Primary key란 테이블에서 각 Row를 유일하게 구분하는 컬럼키입니다. Foreign key는 하나의 테이블에 있는 컬럼으로는 그 의미를 표현 할 수 없는 경우, 다른 테이블의 Primary key를 참조한 값입니다.

? JDBC란 무엇인가요
JDBC란 자바에서 데이터베이스와 관련된 작업을 처리할 수 있도록 도와주는 API입니다. JDBC의 특징을 말씀드리면 DBMS에 의존하지 않습니다. 그 이유는 자바애플리케이션으로부터 사용하는 JDBC드라이버매니저와 DBMS에 의존하는 JDBC드라이버를 분리했기 때문입니다. 또한 표준적인 SQL을 실행하는 메서드를 가지고 있기 때문입니다.

1. ERP (Enterprise Resource planning)

① 전산적 자원 관리 시스템

② 기업의 모든 자원을 전체적으로 관리하여 최적화된 기업 활동을 통합, 관리 시스템에 근거하여 스피드 경영과 투명 경영의 효과를 꾀하는 것

2. Framework

① 특정 형태의 소프트웨어 문제를 해결하기 위해, 상호 협력하는 클래스들과 인터페이스의 집합

② 장점 : 재사용성, 단순성, 역할구분, 확장성, 유지보수 용이

③ 라이브러리와 프레임워크의 차이

? 라이브러리는 어플리케이션에서 호출할 수 있는 함수와 루틴으로 구성되어 있음

? 프레임워크는 어플리케이션에서 특정 기능들을 제공하기 위해 확장할 수 있는 일반적이고 상호 협력적인 컴포넌트를 제공

3. DI와 AOP에 대해 설명하시오.

① DI : Dependency injection = 의존성 주입

Spring을 적용하여 applicationContext.xml에서 설정만 해주면 외부 설정파일(xml)에서 연관관계에 있는 객체를 주입해주기 때문에 의존객체를 찾기 위한 코드가 필요하지 않게 됨.

즉, 외부설정에서 객체를 찾아서 쓰기 때문에 코드 내에 의존관계를 맺는 코드 생성이 불필요.

② AOP : Aspect oriented programming = 약자 관심지향 프로그래밍

프로그래밍 할 때 특정한 관심사를 가진 코드 부분을 별도의 모듈로 분리함으로써 기존 객체지향의 강력한 기능(상속, 위임)만으로는 처치가 곤란했던 중복을 할 수 있게 됨.

4. Spring

① Java Enterprise Application 개발에 사용되는 Application Framework : 빠르고 효율적인 개발을 할 수 있도록 Application의 바탕이 되는 틀과 공통프로그래밍 모델, 기술 API를 제공.

② 특징

? 스프링은 종속객체주입이라는 기술을 통해 낮은 결합도를 유지할 수 있음.

? AOP를 이용하여 객체지향만이 아닌 관심지향 기법을 활용.

? EJP 기능을 대체할 수 있음. 트랜젝션 처리를 위한 일관된 방법을 제공.

? 레이어간 연결이 interface로 이어지기 때문에 interface 생성이 필요.

? 다양한 프레임워크와의 통합

③ MVC 모델에서 커버하는 부분 : Controller 지원.

④ 실행 순서 (실행 구조)

? web.xml의 등록된 DispatcherServiet를 통해서 요청에 대해 진입.

? DispatcherServiet은 Client로부터 들어온 URL을 HandlerMapping이라는 곳으로 전송 후, URL을 분석하여 알맞은 Controller 이름을 다시 DispatcherServiet로 보냄.

? HandlerMapping이라는 것을 통해서 실행될 Controller의 이름을 입력받은 DispatcherServiet은 전달받은 Controller를 실행함. 이렇게 실행된 Controller는 스프링에서 제공하는 ModelAndView 객체에 View Page에 전달할 객체와 View Page 이름 정보를 담고 DispatcherServiet로 보냄.

? ViewResolver를 통해 보여질 View Page를 탐색한 후, View Page를 보여 줌.

5. iBatis / myBatis

① Java에서 DataBase를 편하게 Handling 할 수 있데 해주는 Framework.

② 특징

? SQL문과 Java코드와의 분리만으로도 Java개발자는 Query문을 신경 쓰지 않아도 됨. SQL문이 변경되더라도 파라미터 값이 변경되지 않는다면 Java소스에서 수정부분이 없기 때문.

③ MVC 모델에서 커버하는 부분 : Model 지원

④ 실행 순서 (실행 구조)

? 자바코드 내에서 특정 쿼리문을 실행하기 원할 때, 파라미터와 필요한 조건을 넘기기 위한 객체 생성.

? SQLMaps를 실행하기 위해 쿼리의 객체와 이름을 넘겨 줌.

? 쿼리가 실행되었을 때 SQLMaps는 쿼리 결과를 받기위해 정의된 클래스의 인스턴스를 생성하게 됨.

? 인스턴스는 데이터베이스에 의해 반환된 Resultset으로부터 만들어짐.

6. MVC 패턴이란?

? Model, View, Control 의 역할을 확실하게 분리시켜 놓는 프로그래밍 기법.

7. WAS (Web Application Server)

① Server와 Client 사이에 있는 3-tier 방식으로서, Server가 처리하는 양이 많아지면서 Server에 생기는 부하를 해결하기 위해 개발. Client에서 요청이 들어오면 실제적인 처리는 WAS가 하고 Server는 단지 Client에 결과 값을 뿌려주는 역할만 하게 됨.

② 대표적인 제품 : 톰캣, 웹스피어, 웹로직, 제우스 등.

8. String, StringBuffer, StringBuilder의 차이점

① String 클래스 : 상수 문자열, 한번 생성한 후 변하지 않는 문자열 용도.
② StringBuffer 클래스 : 프로그램 내에서 계속 변하는 문자열 용도.
③ StringBuilder 클래스 : Java5에 추가된 클래스로 StringBuffer와 기능이 같다.

* 차이점 : StringBuffer는 동기화(synchronized)되지만 StringBuilder는 그렇지 않다.
즉, StringBuilder는 다중 thread에서는 안전하지 않으므로 동기화가 필요한 경우는
StringBuffer를 사용하는 것이 좋다.
StringBuilder에서 동기화 하려면 synchronized블록으로 감싸야 한다.

9. Java Servet

? 자바를사용하여 엡페이지 동적으로 생성하는 서버 측 프로그램.

10. JSP란?

① Java Servet Page : 자바를 기반으로 하는 스크립트 언어.
② 특징
? 자바를 기반으로 하는 스크립트 언어로서 자바의 장점을 사용할 수 있음.
? 자바를 기반으로 하고있으므로 플랫폼에 상관없이 사용할 수 있음.
? 표현언어, 표현식, 스크립틀릿 등의 다양한 스크립트 요소와 액션 태그 등을 제공함으로써 보다 쉽게 웹 어플리케이션을 개발할 수 있음.
? 서블릿/EJB 등의 엔터프라이즈 기술들과 잘 융합됨.

11. JSP를 구동하기 위해 서버를 구축하는 과정

① 자바 인스톨 ? JDK 인스톨
? JRE 인스톨 (JDK를 설치하면 설치가 되므로 따로 설치할 필요 없음)
② 환경변수 setting : 컴퓨터의 환경변수(Path)를 잡아야 함.
③ 미들웨어 : JSP를 웹으로 변환해 줄 수 있는 톰켓 설치.

12. XML이란?

① 사용자 정의 태그를 통해 텍스트 데이터의 구조화된 문서 저장과 application간의 문서교환을 위해 1986년 ISO에 제안된 마크업 언어.
SGML의 일부 기능과 인터넷에서 이용하기 쉬운 HTML의 장점을 도입하여 개발한 언어.

② 특징
사용자 정의 태그가 가능.
텍스트 데이터을 구조화하고 전송할 목적(텍스트 데이터의 재사용과 정보 검색 용이).
Application and WepApplication 모두 사용 가능.

13. 서블릿 (Servlet)

? 서버용 애플릿, 웹서버에서 실행되는 작은 자바 코드. JVM에서 실행되므로 플랫폼의 구애를 받지 않고, 웹서버와 충돌이 없고 메모리 관리가 철저함.
? 웹브라우저에서 실행되지 않고 GUI로 구성되지 않는다는 점이 애플릿과 다름.
? 웹서버에서 실행되는 서블릿 엔진과 서비스 요청 및 이에 대한 반응 형태로 사용.

14. Applet

? 자바 애플릿 프로그램은 애플릿 뷰어(JDK에 포함)나 www(웹)검색기(웹브라우저)에 의해서 실행되는 작고 간단한 프로그램.
? 일반적으로 애플릿 프로그램은 인터넷과 연동된 웹 서버로부터 사용자의 컴퓨터로 다운로드 되어 검색기 상에서 실행됨.
? 애플릿 프로그램은 실행명령에 의해 실행되지 않고 웹의 HTML 코드 내에서 호출되는 실행 형태를 가지고 있음.

15. 자바의 특징

? 플랫폼에 독립적인 객체지향 언어.
? 플랫폼에 독립적이라는 것은 리눅스 그리고 위도우 등과 같은 개발환경에 제약을 받지 않고, 어느 플랫폼에서나 코드의 호환성과 재사용이 자유롭다는 것을 의미한다.

16. 객체(object)

? 효율적으로 정보를관리하기위하여 사람들이 의미를 부여하고 분류하는 논리적인 단위.
프로그래밍에서는 클래스에 정의된 내용대로 메모리에 생성된 것을 말함.

17. 클래스

? 객체를 만드는 설계도 (객체를 생성하는 틀의 개념)

18. 객체와 인스턴스

? 클래스로부터 객체를 만드는 과정을 클래스의 '인스턴스화'라고 하며, 어떤 클래스로부터 만들어진 객체를 그 클래스의 '인스턴스'라고 함.

19. 객체지향 프로그래밍(Objecr-Oriented Programming : OOP)

컴퓨터 프로그래밍의 패러다임의 하나다.

객체지향 프로그래밍(OOP)은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러개의 독립된단위, 즉 "객체"들의 모임으로 파악하고자 하는 것. 각각의 객체는 메시지를 주고 받고, 테이타를 처리할 수 있음.

① 캡슐화(Encapsulation): 하나의 문제 해결을 위한 data와 method를 한 단위로 묶는 것으로서, 클래스 내부 정의에 대해 외부에서 볼 수 없도록 함이 특징(은닉화)

② 추상화(Abstraction) : 객체(Object)의 자세한 성질을 무시하고(숨기고) 그들의 일반적인 성질을 나타낸다는 것. 일반적으로 클래스는 클래스로 표현할 서브클래스(또는 객체)의 공통적인 성질과 행위를 일반화해 디자인되며 그로부터 생성된 객체는 자신의 고유한 성질을 갖게 됨.

③ 다형성(Polymorphism) : 다형성이란 같은 메시지에 대해 클래스에 따라 다른 행위를 하는 특성.
일반적으로 같은 이름을 갖는 method에 대해 인자(Argument) 개수와 Data Type에 따라 수행되는 행위가 달라짐을 의미. 다형성을 통해 사용자는 약속된 인터페이스를 따르는 서로 다른 객체들를 같은 방식으로 사용할 수 있게 됨.

④ 상속(inheritance) : 기존에 있던 클래스(즉, 기존의 클래스로부터 상속받은)를 바탕으로 다른 특성을 추가해 새로운 클래스를 만들 수 있음.

⑤ 인스턴스(Instance) : 인스턴스는 추상화 개념 또는 클래스 객체, 컴퓨터 프로세스 등과 같은 템플릿(무엇인가를 만들 때 안내 역할 하는 데 사용되는 형식, 꼴, 틀 또는 모형 등을 의미)이 실제로 구현된 것.

20. 자바언어와 기존의 언어의 다른 특징

? 포인터를 사용하지 않는다. (포인터는 존재, 연산을 허용하지 않음)
? 자동으로 쓰레기 수집(garbage collection) 기능을 수행한다.
? 엄격한 형 검사(strict type checking)를 수행하여 에러를 조기에 발견한다.
? 실행시간에 발생하는 에러를 처리한다.

21. 다형성

? 객체지향 개념에서의 다형성이란 '여러 가지 형태를 가질 수 있는 능력'을 의미하며 자바에서는 한 타입의 참조변수로 여러 타입의 객체를 창조할 수 있도록 함으로써 다형성을 프로그램적으로 구현.

22. Overloading , Overriding

① Overloading (method 중복정의)
- 기존의 method 인자를 이용하여 하나의 함수에 여러 가지 기능을 만드는 것.

② Overriding (method 재정의)
- 상위 클래스에 있는 method와 똑같은 method를하위 클래스에 다시 만들기.
  즉 하위 클래스에서 method를 재정의하는 것.
  주로 생성자 method를 정의할 때 많이 사용.

23. API : 자바 API (Application Programming Interface)

? 자바개발 환경에서 제공되는 거대한 패키지.
? 자바언어는 작고 단순한 대신 많은 기능들을 제공하는 라이브러리를 API로 가지고 있음.

24. InnerClass 를 쓰는 이유

? 내부 클래스를 사용하면 같은 패키지에 있는 다른 클래스한테 까지도 숨길 수 있기 때문에 outer class를 통하지 않고서는 접근할 수 없음. 보안성이 좋아짐.

25. OOL (Object Oriented Language)

? 객체지향 언어.

26. OOP (Object Oriented Programming)

?객체지향 프로그램.

27. import

? C/C++의 #include 선행처리문과 비슷. - 헤더파일의 선언을 위해 필요한 구문.

인터프리터에게 클래스를 띄우라는 지시자의 역할을 함.

28. 예외처리의 필요성과 목적

? 자바에서 프로그램 실행 중 예외가 발생하면 발생된 시점에서 프로그램이 바로 종료가 된다. 때에 따라서는 예외가 발생 했을 때 프로그램을 종료시키는 것이 바른 판단일 수도 있다. 하지만 가벼운 예외나 예상을 한 예외라면 프로그램 종료가 조금은 가혹(?)하다고 느껴진다.

? 그래서 '예외처리'라는 수단(mechanism)이 제안되었고 예외 처리를 통해 우선 프로그램의 비정상적인 종료를 막고 발생한 예외에 대한 처리로 정상적인 프로그램을 계속 진행할 수 있도록 하는 것이 예외처리의 필요성이라 할 수 있다.

? 예외의 발생으로 실행 중인 프로그램의 갑작스런 비정상 종료를 막고, 정상 적인 실행 상태를 유지 할 수 있도록 하는 것.

29. Abstract 클래스

? 추상 method를 하나 이상 가지는 클래스.
? new 객체를 생성할 수 없음.
? 상속받은 클래스는 추상 method를 구현해야만 인스턴스를 생성할 수 있음.
(추상클래스 끼리의 상속은 method 재정의가 필요없음. 사용할 때 일반 클래스에서 재정의)
? 추상 method란 함수의 리턴타입과 파라메터만 있고 정의 부분이 없는 함수를 말함.

30. 추상화 (abstraction)

? 구체적인 개념으로부터 공통된 부분들만 추려내어 일반화 할 수 있도록 하는 것을 의미. 일반적으로 사용할 수 있는 단계가 아닌 아직 미완성적 개념.

31. Interface

? 상수 또는 추상 method만 정의 할 수 있음.
? 구현된 method는 포함 할 수가 없음.
? 모든 변수는 static(정태적)이고 final임.
? 자식클래스들이 공통된 method를 포함하도록 하는 기능만 함.
? 다중 상속이 가능함.

32. 추상클래스와 인터페이스의 공통점

? 객체를 발생시킬 수 없음.
? 상속해서 하위 class를 통하여 객체를 발생시킴. (method를 재정의 해야 함.)

33. 전역변수, 지역변수, 정적변수를 스태과 힙과 관련해서 설명

① 전역 변수
- 프로그램에서 전반적으로 다 사용이 가능.
  함수 밖에서 정의해 놓으면 다름 함수에서도 사용할 수 있음.

② 지역 변수
- 함수 안에서만 사용이 가능한 변수.
  다른 함수에서는 사용을 못함.

③ 정적 변수
- 한 번 정의하면 그 값은 계속적으로 메모리에 남아 있음.
  프로그램이 종료될 때, 메모리에서 지워짐.
  함수 안에서 사용해도 함수가 끝날 때 지워지지 않음.

34. 스레드(Multi threaded)

? 하나의 프로그램이 동시에 하나 이상의 처리(process)를 수행하는 것을 의미한다.

35. 디버깅
? 버그(결함) 에러를 수정하는 것.

36. 미들웨어 (Middleware)
? 최근에 네트워크 상호 호환과 연동상태를 관리하는 분산 환경에 관심을 갖고 데이터베이스, 스프레드쉬트, 윈도우 OLE등 다양한 소프트웨어를 다루는 데 더 관심을 가짐.
? 이러한 공통의 인터페이스를 준수하는 컴포넌트 기반의 소프트웨어는 많은 장점을 제공함.
? 이러한 공통의 인터페이스를 프레임웍=미들웨어(Middleware)라고 함

37. Camel Case (변수이름 정하는 규칙)
① 클래스: 첫 문자는 대문자로 시작. 둘 이상 단어 묶어 구성시 새로 시작 단어 대문자.
② 메소드: 첫 문자는 소문자로 시작. 둘 이상 단어 묶어 구성시 새로 시작 단어 대문자.
③ 상 수: 모든 문자를 대문자로 구성. 둘 이상 단어 연결시는 _(언더바)를 사용하여 연결 함.

38. JVM의 구성

① 클래스영역 : 클래스 코드를 저장하는 영역.
② 자바스택 : method를 호출할 때 관련 정보를 저장하는 영역.
③ 힙(Heap) : new 라는 키워드를 통해 객체가 생성될 때 할당받는 영역.
④ 네이티브 method 스택

39. Collection에서 데이터를 저장 하는 3가지와 그 특징
① Set : 순서가 없고, 동일한 데이터 허용 안 됨.
② List : 배열과 같은 구조지만, 가변적 길이를 가지고 있음. (크기가 지정 되어있지 않음.)
③ Map : key 값과 value값의 형식으로 저장되면 key값은 절대 중복이 안 됨.

40. 스트림
① 데이터를 목적지로 입 · 출력하기 위한 방법. 스트림에 데이터를 쓸 수 있고 읽을 수 있음.
연결하고자 하는 Device에 따라 다양한 스트림이 존재.
② 특징
? 스트림은 FIFO 구조. 읽기, 쓰기가 동시에 되지 않음.
? 읽기, 쓰기가 필요하다면 읽는 스트림과 쓰는 스트림을 각각 하나씩 열어 사용해야함.
? 데이터가 처리되기 전까지 스트림에 사용되는 스레드는
데이터가 모두 전송되기 전까지 블락킹 상태에 빠짐.

41. 동기화
? 여러 명이 접근 하는 것을 방지하기 위함. 이때 여러 명이 접근 하는 것을 막기 위해 모든 객체에 '락'을 포함시키는데, '락'이란 공유 객체에 여러 스레드가 동시에 접근하지 못하도록 하기 위한 것으로 모든 객체가 힙 영역에 생성될 때 자동으로 만들어 짐.

42. 더블버퍼링을 사용하는 이유
? 이미지를 출력 시키는 경우 화면을 지웠다가 다시 그리게 되면 깜박거리게 됨. 이러한 문제점의 해결을 위해, 화면에 직접 이미지를 그리지 않고 메모리상에 미리 그려놓고 필요할 때 한번만 화면에 출력 시키는 방법으로 출력 시 깜박거림을 최소화 시켜줌.

43. AWT와 Swing의 차이점
? AWT : flatform에 종속. 중량 Component. Frame 자체에 분할. 범용 Component.
? Swing : flatform에 자유로움. 경량 Component. Frame 포함 멤버 분할. Local Component.

44. Statement 와 PreparedStatement의 차이
① Statement : 정적 쿼리 시 사용. 매번 파싱과정을 거쳐야 함.(부하가 생길 수 있음)
SQL문 전체를 명확히 알 수 가 있어서 디버깅이 쉬움.
② Prepared Statement : 동적 쿼리 시 사용.
한번 파싱하면 그 동일 SQL문장을 곧 바로 파싱 과정 없이 Execution 가능.
(반복적인 다량의 SQL 수행 시 성능 상 이득이 있음)
오류발생 시, 변수에 입력되는 값을 알 수 없어서 디버깅이 어려움.

45. 서블릿의 실행 과정
? 서버가 클라이언트의 연결 요청 받음. 테이너는 연결 요청 정보를 담고 있는 Request 객체와 연결 응답 정보를 담고 있는 Response 객체를 생성.
? 접수된 Url을 분석 후 해당 서블릿 객체를 생성하고, 사용자의 요청을 처리하기 위해 스레드를 생성 후 service() method에 인자 값을 담아 호출.
? service() method는 Request 객체를 참고하여 어떤 연결 요청 방식으로 들어왔는지 파악함.(GET 방식인지 POST 방식인지)
? 들어온 요청 방식에 따라 get 방식은 doGet() method를, post 방식은 doPost() method를 호출하여 처리 함.
? service() method의 인자 값으로 넘겨받은 response 객체를 이용하여 클라이언트에게 결과를 보여줌. 사용자 요청을 처리하기 위해 생성한 스레드를 소멸.

46. 서블릿에서 데이터를 처리 하는 방식은?
① GET : 서버에 있는 정보를 가져오기 위해 설계 됨. 240바이트까지 전달 할 수 있음.
POST 방식에 비해 속도가 빠름. 검색엔진에서 검색단어 전송에 많이 이용함.
URL노출로 보안성이 요구되는 경우엔 사용 할 수 없음.
② POST : 서버로 정보를 올리기 위해 설계 됨. URL에 파라미터가 표시 되지 않음.
내부적으로 데이터가 이동함. GET 방식에 비해 속도 느림. 데이터크기 제한 없음.

47. **JSP 에서 페이지 이동 방법의** 대해 설명 하시오.
① Forward 방식 ? Url이 바뀌지 않음. 요청객체와 응답객체가 유지됨.
? 속도가 빠르며 요청객체에 소속 되어 있음.
처리 구조
? 요청이 들어오면 Servlet이 받음.

? 요청에 알맞은 페이지를 찾음. 알맞은 페이지가 있다면 응답.
? 알맞은 페이지가 없다면, Forwarding 방식으로 알맞은 페이지로 넘기는데, 요청객체와 응답 객체를 포함해 넘김.
? Url이 바뀌지 않은 상태로 응답 페이지를 통해 응답.
② Redirect 방식 ? Url이 바뀜. 요청객체와 응답객체가 유지 되지 않음.
? 속도가 느리며, 응답객체에 소속 되어 있음.
처리 구조
? 요청이 들어오면 Servlet이 받음.
? 요청에 알맞은 페이지를 찾음. 알맞은 페이지가 있다면 응답.
? 알맞은 페이지가 없다면, 알맞은 페이지로 다시 요청하게끔 응답 보냄.
? 클라이언트는 응답을 받고, 다시 그 요청의 맞는 Url로 요청함.

48. 자바 빈즈
? 자바에서 사용하는 컴포넌트.(부품)
Bean은 자바에서 컴포넌트를 이용하기 위해 만들어 놓은 기술.
? 컴포넌트를 사용하면 좋은 이유는 필요 할 때마다 가져다가 사용 할 수 있다는 점인데, JSP에서 자바의 컴포넌트를 이용해서 프로그래밍 하는 것을 빈즈 프로래밍이라 함.

49. Connection Pool 개념
? 미리 생성해 놓은 커넥션을 할당하고 반납함으로써 커넥션 생성 시간을 줄인다는 개념.
? 미리 생성할 때 너무 조금 생성하면 오히려 타임 오버헤드가 늘어나고, 반대로 너무 많이 생성하면 메모리 오버헤드가 늘어남.

50. Model 1 방식 과 Model 2 방식의 특징과 차이점
① Model 1 : 디자인 코드와 자바 코드(비즈니스로직)를 구분하지 않고, 하나의 JSP 내에 기술해서 웹 프로그램을 제작 하는 방식.
특징
? 개발하기가 쉽고, 배우기가 쉬움.
? 디자인코드와 비즈니스로직의 구분이 명확하지 않아, 복잡도가 높음.
? 수정 시 디자이너와 개발자의 협업이 필요.
? 비즈니스로직의 재 사용성이 어려우며, 유지 보수가 힘듬.
② Model 2 : 웹 어플리케이션을 개발할 때, MVC패턴을 적용하여, 웹 어플리케이션의 개발이 가능하도록 구현한 것.
특징
? 초기 설계에 많은 시간이 소요.
? 디자인코드와 비즈니스로직이 분리되며, 비즈니스로직의 재사용성이 높아짐.
? 비즈니스로직 계층의 확장성이 용이하며, 유지보수가 편하다.

51. Singleton의 대해 설명 하시오.
? 발전된 형태의 전역 변수.
? 프로그램 상에서 두 번째 인스턴스를 만들 수 없게 하는 기능.
? 만들어진 클래스의 객체를 단 하나만 사용하며 어디서든 그 객체를 사용가능토록 만들어 줌.

52. Cookie 와 Session 의 차이
① Cookie : 클라이언트에 정보를 저장하기 때문에 서버에 부하가 없음.
사라지는 때(시간)를 지정 할 수가 있음.
② Session : 서버에 정보를 저장하기 때문에 서버에 부하가 많이 생김.
해당 클라이언트와 일정시간동안 작용이 없으면 서버메모리에서 해제 됨.
직접적으로 해제 시킬 수 있는 method가 있음.

53. JSP 와 자바스크립트의 차이
? JSP는 자바에서 파생된 서버사이드 스크립트로서 웹 페이지를 작성하는데 사용.
? 자바스크립트는 클라이언트사이드 스크립트이며, 브라우저의 객체를 제어하는데 쓰임.

54. 엔티티빈 (EntityBean)
? 데이터를 객체화하여 재사용이 가능한 컴포넌트를 말함.
? 데이터의 객체화란 개발자들이 데이터에 접근, 변경 방법이 보다 단순하고 쉬워진다는 의미.
? 객체화된 데이터를 비지니스 객체 또는 현실 세계의 객체라고하며, 이 객체는 영속성을 가진 관계형 데이터베이스에 저장이 되고, 데이터베이스에서 테이블의 한 레코드와 대응됨.
? 여기서 중요한 것은 빈 인스턴스와 데이터베이스의 데이터가 동기화 되어야 한다는 것. 즉, 빈 인스턴스에서 새로운 변화가 일어날 때마다 데이터베이스도 같이 갱신이 되어야 함.

55. EJB의 사용 목적
? “대규모이고 구조가 복잡한 분산 객체 환경”을 쉽게 구현하기 위해서 등장.
? 컴포넌트(클래스들이 모여서 생긴 하나의 기능)들의 재사용을 목적.

? EnterpriseBean은 이식성이 좋음.

56. DBMS (Database Management System)
? 데이터베이스 관리시스템 : 응용프로그램이 데이터에 대한 액세스가 가능하도록 DB를 관리하는 소프트웨어 기능 및 데이터가 저장되어 있는 장소.

57. RDBMS (Relational Database Management System)
? 관계형 데이터베이스 : 일반적으로 관계형 데이터 모델에 기반한 데이터베이스를 관계형 데이터 베이스라고 함.
? 특징
? 데이터를 테이블로 표현하고 테이블을 집합으로 받아들임.
? 데이터를 조작하는 데는 집합론에 기초한 수학적인 연산을 테이블에 대해 실행.

58. ERD(Entity Relationship Diagram)란?
? 개체관계도 : 말로 되어있는 요구분석사항을 그림으로 그려 그 관계를 도출하는 것.

59. Index
① 장점 : 가장 큰 장점은 데이터의 빠른 검색이고, 다음은 Row의 유일성의 유지.
② 단점 : 인덱스를 만들면 그 정보의 유지를 위한 디스크 공간이 필요하고, 인덱스가 걸려있는 테이블은 인덱스가 없을 때보다 데이터 추가나 변경할 때 많은 시간이 소요된다.
③ 인덱스의 종류
? Unique Clustered Index ? Nonunique Clustered Index
? Unique Nonclustered Index ? Nonunique Nonclustered Index
④ Clustered Index는
? 한 테이블에 단 한개만 존재 ? 범위(Range)를 주어 검색할 때 탁월한 기능 발휘
(예) 키순으로 서세요. 나이순으로 서세요. 이름순으로 서세요.
⑤ Nonclustered Index는
? 한 테이블에 여러개 존재 가능 ? 특정한 값으로 찾아갈 때(Seek) 탁월한 성능 발휘.
(예) 도서명, 저자명, 분야명

60. 트랜젝션, 롤백, 커밋
① 트랜잭션 : 일련의 작업단위. 특성으로 ACID(원자성, 일관성, 고립성, 영구성)가 있다.
② 〃 의 필요성 : 여러 작업이 한 자원에서 동시에 변경, 참조 필요시. / 작업공정이 긴 경우.
③ 〃 의 시작 ? 첫 DML(data조작어)구문이 실행될 때. (SELECT에선 제외)
? 사용자가 ASVEPOINT를 설정할 때.
④ 〃 의 종료 ? commit 수행 ? rollback 수행 ? 오라클 내부에서 자동 commit 수정
? DDL(data조작어), DCL(data제어어)이 실행될 때.
? 사용자가 sqlplus를 종료할 때, 시스템 크래쉬 발생.
⑤ 커밋과 롤백 전 ? 롤백으로 데이터 회복 가능 ? 현재 유저는 자신이 바꾼 데이터 확인 가능
? 다른 유제에게는 바뀐 데이터 내용이 반영되지 않음
? 영향받은 레코드는 잠금 상태
⑥ 커밋 후 ? 데이터는 영구적으로 변경 ? 복구 불가능 ? 세이브 포인트 삭제
? 모든 유저에게 변경 내용 반영
? 해당 레코드 잠금 해제, 다른 유저들이 다룰 수 있음
⑦ 롤백 후 ? 데이터의 변경 취소 ? 데이터 이전 상태 회복 ? 데이터에 대한 잠금 해제

61. DB정규화의 목적
? 자료정규화 작업의 가장 큰 목적은 자료저장의 중복성 배제이다.
? 정규화이론에서는 릴레이션 형태가 여러 단계로 구분되며, 가장 기본적인 정규화조건도 만족하지 못하는 릴레이션을 비정규형, 만족하는 릴레이션을 제1정규형이라고 부름.
? 조건이 점점 엄격해짐에 따라 제2, 제3, 제4, 제5정규형으로 구분됨.
높은 단계의 정규형으로 나아갈수록 데이타의 본질적 의미가 릴레이션 구조에 보다 정확히 반영되고, 데이타 중복을 줄이고, 데이타 변경시 발생하는 문제점을 방지하고, 궁극적으로 데이타 무결성(data integrity)을 제고할 수 있다고 가정함.

62. ACID (Atomic Consistent Isolated Durable)
? 원자성, 일관성, 독립성, 지속성의 약어.
? 데이터베이스 트랜젝션이 안전하게 수행된다는 것을 보장하기 위한 성질.

63. DDL (Data Definition Language)
테이블 및 객체의 구조 생성 그리고 삭제, 변경과 관련된 명령어.
? CRATE : 새로운 테이블 생성.
? DROP : 기존 테이블의 구조 및 모든 행을 삭제.
? ALTER : 기존 테이블 변경, 필드를 추가 또는 수정 할 수 있지만 삭제 불가.

? TRUNCATE : 테이블의 구조를 남기고 모든 데이터 삭제.
? RENAME : 오브젝트의 이름을 변경.
? COMMENT : 테이블이나 컬럼에 주석문.

64. DML (Data Manipulation Language)
데이터 조작과 관계된 명령어.
데이터의 검색 및 출력, 정렬과 조인에 관계 됨.
? SELECT : 데이터의 조회.
? INSERT : 데이터의 입력.
? UPDATE : 데이터의 수정.
? DELETE : 데이터의 삭제.
? WHERE : 데이터의 조건.
? COMMIT : 수행한 데이터 작업들의 영구 저장.
? ROLLBACK : 수행한 데이터 작업들의 원상복구.

65. DCL (Data Control Language)
계정의 권한과 관계된 명령어.
DB 사용자의 권한 정의.
? COMMIT : 데이터베이스 조작을 영구적으로 반영.
? ROLLBACK : 마지막 COMMIT 상태로 복구.
? GRANT : 권한 부여.
? REVOKE : 권한 취소.

66. Struts2
① MVC 패턴을 지원하고, 웹 어플리케이션을 개발하기 위한 프레임워크.
② 특징
? Struts1 + Webwork2의 아키텍처 적용.
? POJO (Plain Old Java Object) 기반 액션 사용.
? Zero configrarion (설정 없이) 지향.
? Ajax기능을 태그와 테마 기반으로 구성.
? 다양한 표현식 언어 사용 가능. (JSTL, OGNL)
③ MVC 모델에서 커버하는 부분 : Controller 지원.
④ 실행 순서 (실행 구조)
? web.xml의 등록된 FilterDispatcher를 통해서 요청에 대해 진입.
? FilterDispatcher에선 모든 요청을 처리하고, ActionPloxy 생성.
? ActionPloxy가 생성되면 FilterDispatcher가 실행권한을 이임.
? configrarion은 Struts.xml을 통해서 설정관리자를 초기화함.
? 초기화 후, Action invocation 실행 - 실행할 Intercepter가 있으면 실행하고, Action으로 전달.
? Action에서 결과값을 통해, 해당 Template로 이동.
? Intercepter자원 반납 후 응답.

67. C와 C++의 차이점
① C : 함수 기반. 절차함수 기능만 가지고 있음.
필요하면 함수 prototype을 만들어서 본문 추가.
② C++ : 클래스 기반. 컴파일러 단에서 “클래스”라는 객체지향적 방법을 지원.
클래스에서 생성자/소멸자 등을 사용하여 자원 관리를 자동화 할 수 있음.
연산자 재정의가 가능.

68. OSI 7 Layers (OSI 7 계층 구조)
- 7계층: 응 용 계층 (application layer)
- 6계층: 표 현 계층 (presentation layer)
- 5계층: 세 션 계층 (session layer)
- 4계층: 전 송 계층 (transport layer)
- 3계층: 네트워크 계층 (network layer)
- 2계층: 데이터링크 계층 (data link layer)
- 1계층: 물 리 계층 (physical layer)

① 개요
컴퓨터 통신 분야에서 동일 업체 또는 유사 업체기리의 다양한 표준들이 양산됨에 따라 표준을 정의하는 골격으로서의 국제적인 기준 모델이 필요하게 되었다. 따라서 네트워크 상호간의 접속을 목적으로 각종 규격 개발작업을 조정하기 위한 공통 기반을 제공하고, 각 사업자별로 개발된 기존 규격에 대하여는 이러한 참조 모델에 쉽게 적응, 연결토록 하는 편의성을 제공하고자 했다.

② 표준화를 위한 기본 골격

개방형 시스템간 상호접속모델(Open System interconnection model)은 각 계층에서 표준이 개발, 발전될 수 있도록 기본들을 제공, 프로토콜의 발전을 유도, 인도하는데 목적이 있다. 결론적으로 OSI 참조모델은 표준의 표준으로써 단지 표준화의 유도를 위한 기본 골격이다.

③ 계층 간 독립성
각 계층별로 독립시킨다는 것은 융통성(flexibility)을 부여함을 의미한다. 예를 들어 보다 좋은 성능을 가진 전기적 인터페이스가 개발되면 상위계층에 영향을 미치지 않고 성능을 향상시킨 채 교체가 가능하기 때문이다.

④ 계층 내 프로토콜
OSI가 각 계층별로 하나의 프로토콜만이 존재하는 그릇된 생각을 할 수 있는데 이는 잘못이며 사실은 각 층을 구현하기 위해 다양한 프로토콜이 존재한다. 그러나 같은 계층에 있는 여러 다른 프로토콜은 바로 하위 계층에서 제공하는 서비스를 공유하여야 한다.

⑤ 연결 지향성 및 비연결성
OSI는 초기에 연결 지향성 서비스 위주의 프로토콜들에만 관심을 가져왔으나 후에 비연결성에도 관심을 가지게 되었다.

69. Stack과 Queue - Stack과 Queue는 컴퓨터의 알고리즘에서 가장 흔하게 사용되는 자료 구조.
① Stack : 모든 작업이 리스트의 한 쪽 끝에서만 수행되는 선형 리스트의 한 형태. 리스트의 한 쪽 끝(TOP)에서 원소를 삽입하거나 제거하는데 사용. 그러므로 리스트에서 Stack에 마지막으로 입력된 원소가 제일 먼저 제거의 대상이 됨. 그래서 Stack을 LIFO (Last in First Out) 라고 함.
② Queue방식 : 삽입은 REAR에서 이루어지고, 제거는 FRONT에서 이루어짐. Q에 A, B, C, D, E 순으로 원소가 삽입되었다면, 제일 먼저 제거되는 원소는 A. Q는 FIFO (First in First Out) 라고 함.

70. **AJAX**란?
① 웹클라이언트 화면 제어를 보다 세밀하게 하기 위한 기술.
② 특징
? 필요한 데이터만 서버로 보내고 해당 정보만 받음.
? 서버로부터 정보를 받을 때는 xml or josn 형태로 받고, Client에서는 그 정보들을 DOM을 통해 화면을 재가공하여 보여주므로 서버와 교환되는 데이터의 양이 적어지므로 점유율을 낮출 수 있음.
? Client에서의 데이터 처리로 서버에 부담이 덜 가므로 다른 응답 요청에 신속히 응대.

71. WIPI란?
① 한국무선인터넷 표준화 포럼(KWIS)에서 만든 모바일 플랫폼 표준 규격으로 무선인터넷을 통해 다운로드된 응용프로그램을 이동통신 단말기에 탑재시켜 실행키 위한 환경을 제공.
② 특징
? C/C++/Java 등 복수 언어, 모두를 지원함으로써 다양한 컨텐츠 개발자들을 수용할 수 있도록 함.
? 동적 링킹 라이브러리(DLL: Dynamic Linking Library)를 지원함에 따라 플랫폼의 API를 동적으로 추가 혹은 갱신할 수 있음.

72. 프로그램 개발하는데 제일 중요한 것은?
* 정답이 없으므로 주관적으로 답할 것.
* [참고]
"프로젝트 진행과정"
요구사항 분석-설계(데이타베이스,클래스,화면)-개발-테스트-이관/교육-트로젝트 종료

----------실제 면접에서 나왔던 질문들 ----------

1. CVS나 SVN에 대해서 아는대로 설명해 보시오.

2. 64bit CPU와 32bit CPU의 OS적 관점에서의 차이를 설명해 보시오.

3. 프로세스와 쓰레드의 차이점에 대해서 설명해 보시오.(메모리 구조 포함)

4. '데드락' 이란 무엇이고 이를 해결하기 위한 방법을 설명해 보시오.

5. 변수 명명법이 중요한 이유에 대해서 설명하고 예를 들어 보시오.

6. 자바의 JVM의 역할에 대해서 설명해 보시오.

7. 자바의 특징에 대해서 말해 보시오.

8. Linux에서 톰캣 환경설정을 잡는 것에 대해 설명해 보시오.

9. WAS와 웹서버의 차이점은?

10. Jquery와 Ajax에 대해 아는가?

11. 비동기와 동기 방식의 차이점에 대해서 말해보시오.(네트워크 동기,비동기 아님)

12. 개발시에 중요하다 생각하는 요소를 3가지 기술해 보시오.

13. 스프링의 MVC에 대해서 설명하시오.

14. AOP란 무엇이고 왜 사용하는지

15. '에자일' 방법론에 대해서 아는가?

16. 스프링 환경설정 혼자 잡을 수 있는가? 대강 어떻게 해야하는지 설명해 보시오.

17. 웹서버 내부 구동 방식에 대해 설명할 수 있는가?

18. 스프링 DI란?

19. UML 그려본 적 있는가?

20. Node js나 Angular JS를 사용해 본 적이 있는가?

21. 캐시와 세션의 공통점과 차이점에 대해 말해보시오.

22. 디자인 패턴 아는 것들만 간략히 설명해 보시오.

24. 크롬이나 파이어폭스에서 개발도구를 사용해 디버깅을 해보았는가?

25. JDBC는 무엇인가?

26. 스프링을 사용하지 않고 MVC를 JSP에서 만들어 보았는가?

27. DB 옵티마이저에 대해 아는가?

28. 최신 기술 동향에 대해 아는데로 말해보시오?