java

– 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.

– 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.

– 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.

– JDK란 무엇인가요

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

– OOP란? 객체 지향 언어란?

객체 지향 프로그래밍를 의미합니다. 가장 중요한 특징은 캡슐화, 상속, 다형성입니다. 객체 지향 프로그래밍을 하게 되면 클래스라는 개념을 도입함으로써 코드의 재사용을 할 수 있습니다.

– MVC패턴이란 무엇인가요

애플리케이션은 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.

– DAO란 무엇입니까

데이터 접근을 목적으로 하는 객체입니다.

– 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.

– 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.

– 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.

– AOP란 무엇인가요

Aspect Oriented Programing의 약자로 다양한 곳에서 자주 사용되는 공통관심 요소를 단일 기능으로 뽑아내서 코드의 중복을 줄이고 관리의 효율성을 높이기 위해 사용합니다.

– Spring DI란 무엇인가요

Dependency Injection의 약자로 의존성을 주입하는 것을 말합니다. 설정 파일을 통해 객체간의 의존관계를 설정함으로서 객체는 의존하고 있는 객체를 생성하거나 검색 할 필요 없습니다. 그로 인해 코드관리가 편리해집니다.

– 디자인패턴이란 무엇인가요

설계시 반복적으로 나타나지는 설계 방법 또는 구조들을 말합니다. 디자인패턴을 이용하게 되면 코딩이 명확하고 단순하며 모듈을 세분화 시킬 수 있고, 재사용성이 높으며 유지보수가 쉬워집니다. 디자인패턴은 대표적으로 GoF 23가지 패턴이 있습니다. 패턴들의 대부분의 공통점은 추상클래스나 인터페이스로부터 상속받은 클래스를 사용하며, 이는 객체지향의 다형성을 이용한 방법입니다.

– 메모리 상수풀 영역 이란

힙영역(프로그래머가 관리하는 메모리 영역)에 생성되어 자바 프로세스 종료까지 계속 유지되는 메모리영역입니다. 기본적으로 JVM에서 관리하며 프로그래머가 작성한 상수에 대해 최우선적으로 찾아보고 없으면 상수풀에 추가한 이후 그 주소값을 리턴합니다. 그로 인해 메모리 절약 효가가 있습니다.

– 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.

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 설정 되어 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. Java

Java는 네트워크상에서 쓸 수 있도록 미국의 Sun마이크로시스템가 개발한 객체지향 프로그래밍 언어

특징

a. JVM만 설치하면 컴퓨터의 운영체제에 상관없이 작동한다.( 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 메모리 관리기능

e. 멀티쓰레드(Multi-thread)를 지원한다.

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 코드 재사용성이 높다.

b. 코드의 변경이 용이

c. 직관적인 코드 분석

d. 개발속도 향상

e. 상속을 통한 장점 극대화

3. Object

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.

4. Overloading vs Overriding

Overloading(오버로딩)

- 같은 이름의 메소드를 여러 개 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. JDBC

- Java Data Base Connection의 약자로 JAVA 언어를 통해 데이터베이스에 접근할 수 있는 프로그래밍

6. Interface, Abstract

Interface

- 일종의 추상 클래스

- 오직 추상메서드와 상수만을 멤버로 갖는다.

- Implements 키워드를 사용

- 상속의 관계가 없는 클래스간 서로 공동되는 로직을 구현하여 쓸 수 있도록 한다.

- Extends는 하나의 클래스만 상속 가능하나 Interface는 다중 상속이 가능하다.

Abstract

- 추상메소드를 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/구현해야 한다.

차이점

- 사용하는 키워드가 다르다.

- Abstract는 일반 메서드를 사용할 수 있지만, Interface는 메서드 선언만 가능하다

7. Call by Reference, Call by Value

Call by References – 매개변수의 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 않늗다.

8. 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)

9. Garbage Collection(가비지 컬렉션)

- 시스템에서 더 이상 사용하지 않는 동적 할당된 메모리 블록을 찾아 자동으로 다시 사용 가능한 자원으로 회수 하는 것. 시스템에서 가비지 컬렉션을 수행하는 부분을 가비지 컬렉터라 부른다. 고로 자바에서는 메모리 문제를 신경쓰지 않아도 된다!

10. Primitive type과 Reference type

Primitive type – 변수에 값 자체를 저장(정수형, 실수형, 문자형, 논리형) – Wrapper Class를 통해 객체로 변환 가능

Reference type – 메모리상에 객체가 있는 위치를 저장

종류 – Class, Interface, Array 등

11. Wrapper Class

Primitive type으로 표현할 수 있는 간단한 데이터를 객체로 만들어야 할 경우가 있는데 그러한 기능을 지원

12. Thread

Thraed(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 : 다중 프로그래밍 체제에서 하나 또는 그 이상의 프로세스가 수행할 수 없는 어떤 특정 시간을 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.

13. 접근 제한자(public > protected > default > private)

Public – 접근 제한이 없다.(같은 프로젝트 내에 어디서든 사용 가능)

Protected – 같은 패키지 내, 다른 패키지에서 상속 받아 자손 클래스에서 접근 가능

Default – 같은 패키지 내에서만 접근 가능

Private – 같은 클래스 내에서만 접근 가능

14. 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), 선입 선출이 필요한 대기열(티켓 카운터)

15. Singleton Design Pattern(싱글톤 디자인 패턴, 싱글톤 패턴)

- 클래스 인스턴스가 하나만 만들어지도록 하고, 그 인스턴스에 대한 전역 접근을 제공한다.

16. String과 StringBuffer의 차이

자바의 기본 데이터 타입인 int, float, char 등과 다르게 String은 데이터 타입이 아닌 클래스 객체입니다. 또한 String은 불변의 객체입니다. 한번 String name =”길동”; 라고 선언하게 되면 먼저 String 객체타입인 name이라는 인스턴스를 만들고 메모리에 “길동”을 올려버립니다. 그리고 name이 “길동”을 참조하는 레퍼런스가 되는 것입니다.

중요한점은 이제 이 name에 새로운 내용을 추가하겠습니다. name += “홍”; 와 같은 소스를 적용하게 되면 name인스턴스가 가리키는 값은 “길동홍”가 됩니다. 우리는 이런식으로 흔히 코딩을 했을 텐데 아까 위에서 String이 불변 객체라고 설명을 했듯이 한번 선언된 내용에 추가적으로 바뀌지 않습니다. String은 final char배열 형태이기 때문에 내용의 추가와 삭제가 되지 않거든요. 그럼 도대체 어떻게 name변수의 값이 바뀌는 건지 답은 내부에 있습니다.

name = name + “홍”; 구문이 실행될 때 실제로는 스트링 버퍼를 새로 생성해서 name이 가리키는 “길동”을 만들어주고 스트링 버퍼의 append 함수를 이용하여 “홍” 를 붙여줍니다. 그렇게 완성된 스트링 버퍼값을 메모리에 올리고 name은 다시 이 값을 참조하게 됩니다. 스트링 버퍼는 char타입의 배열로 되어 있어서 한글자 한글자를 append할 수 있다는 것을 아셔야합니다. 그럼 그와중에 생겨난 메모리 안의 “길동” 이라는 값과 “홍”라는 값은 가비지 컬렉터가 가지고 있다가 필요없어서 버립니다. 그 짧은 순간에 이런 일처리가 진행이 되어서 스트링 버퍼를 사용하는 것이 스트링 객체를 사용하는 것보다 빠르다는게 제 생각입니다.

17. 자바의 메모리 영역

- 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 동적 할당된 객체들이 저장되며, Garbage 컬렉션에 의해 메모리가

관리되어 진다.

18. DAO와 DTO

- DAO : Data Access Object의 약자로 간단히 데이터베이스의 데이터에 접근을 위한 객체이다. 데이터베이스에 접근을 하기위한 로직과 비즈니스 로직을 분리하기 위해서 사용한다. DB를 사용해 데이터를 조회하거나 조작하는 기능을 전담하도록 만든 오브젝트를 말한다.

- DTO : Data Transfer Object의 약자로 VO(Value Object)로 바꾸어 말할 수 있는데 계층간 데이터 교환을 위한 JavaBean을 말합니다. 여기서 말하는 계층이란 Controller, View, Business Layer, Persistent Layer를 말하며 각 계층간 데이터 교환을 위한 객체를 DTO 또는 VO라고 부릅니다. 그런데 VO는 동일한 개념이지만 read only 속성을 가집니다.

19. 변수 명명법

- 헝가리언 표기법 : 자료형을 식별자에 같이 포함

ex) inum; int int\_num; 인터페이스명.

- 파스칼 표기법 : 식별자가 한 단어나 혹은 여러 단어로 조합(언더바X), 각단어의 첫문자만 대문자로

ex) KorScore

- 캐멀 표기법 : 모든 단어를 공백없이 조합(언더바X) – 첫단어의 첫문자는 소문자로

ex) korScore

- 스네이크 표기법 : ex)eng\_score

1. OOP 란?

- 객체지향프로그램 (Object - oriented Programming)

- 데이터를 객체로 취급하여 프로그램에 반영

- 순차적으로 프로그램이 동작하는 기준과 다르게 객체의 상호작용을 통해 프로그램 작동

- 캡슐화, 다형성, 상속

2. 객체 란?

- Object(객체)는 OOP에서 데이터(변수)와 그 데이터에 관련되는 동작(함수). 즉 절차, 방법, 기능을 모두 포함한 개념

- 클래스는 객체를 만들기 위한 설계도, 객체를 감싸는 큰 틀

3. 추상화, 캡슐화, 상속성, 다형성에 대해 설명

- 상속성 : 부모클래스 상속가능

- 추상화 : Abstract,인터페이스 - 아직 두 개의 큰 차이점, 필요성을 못 느껴봄..

- 캡슐화 : 비슷한 속성, 메서드 하나의 클래스로 만든다. private, public, default, protected

- 다형성 : 오버로딩(같은 함수명, 다른 인자), 오버라이딩 (상속받은 함수를 자식클래스에서 다시 정의)

4. 추상클래스와 인터페이스의 차이점은?

- 추상클래스 : 한 개 이상의 추상메서드를 가지는 클래스

- 추상메서드 : 구현은 하지않고 선언만 한 메서드, 상속받은 자식에서 재정의해서 사용

추상메서드를 재정의 하지 않을 경우 컴파일러는 에러를 발생

- 인터페이스 : 추상클래스와 비슷한 기능을 가지나, 추상메서드와 상수로만 이루어짐(로직 설계 불가능)

다중 상속 가능

- 공통점 : 추상 클래스, 인터페이스 자체로 객체로 생성 불가능.

상속받은 자식이 구현하도록 함.

- 차이점

구현 객체의 같은 동작을 보장하기 위함 추상클래스를 상속받아 기능을 이용하고, 확장시키기 위함

다중 상속 가능 다중 상속 불가능

추상메서드만 가능 일반메서드+추상메서드 가능

상수+추상메서드 형태 일반변수(가능)+일반메서드(가능)+추상메서드 형태

생성자와 일반변수를 가질 수 없음 생성자와 일반변수 모두 가질 수 있음

implments extends

메서드 선언만 가능 메서드의 부분적인 구현이 가능

5. 오버로딩이란?

- 같은 이름의 메소드를 여러 개 정의하는 것

- 매개변수의 타입이 다르거나 개수가 달라야 한다.

\* return type과 접근 제어자는 영향을 주지 않음.

6. 오버라이딩이란?

- 상속에서 나온 개념

- 상위 클래스(부모 클래스)의 메소드를 하위 클래스(자식 클래스)에서 재정의

- 오버라이딩을 할 때 개발자의 실수를 방지하기 위해 메서드 위에 @Override 를 관례적으로 적는다.

7. 오버로딩과 오버라이딩의 차이점은?

- 위에 정리

8. Array List란?

- 배열인데 메모리 위치가 쭉 이어짐

9. Linked List란?

- 배열인데 메모리위치가 뒤죽박죽

10. 어레이 리스트와 링크드 리스트의 차이점은?

- ArrayList : 메모리 밀도가 1, 삭제 시 기능 고려 , 검색 빠름

- LinkedList : 메모리 밀도 분포, 삭제 시 빠름, 검색 시 기능 고려

11. 스트링, 스트링 버퍼, 스트링 빌더의 특징은?

- String : String은 immutable(불변) , new 연산을 통해 생성되면 그 인스턴스의 메모리 공간은 절대 변하지 않는다.

+,concat 시 메모리의 내용이 변하는게 아니라 새로운 String 인스턴스 생성

문자열 연산이 적고 조회가 많을 때 멀티쓰레드 환경에서 사용하면 좋음.

- StringBuffer, StringBuilder 공통점 :

mutable(변경가능)하다.

즉 문자열 연산에 있어서 클래스를 한 번만 만들고(new),

연산이 필요할 때 크기를 변경시켜서 문자열을 변경한다.

그러므로 문자열 연산이 자주 있을 때 사용하면 성능이 좋다!

- StringBuffer, StringBuilder 차이점 :

tringBuffer는 멀티스레드 환경에서 synchronized키워드가 가능하므로 동기화가 가능하다.

즉, thread-safe 하다.

문자열 연산이 많을 때 멀티스레드 환경에서는 StringBuffer,

싱글 스레드 또는 쓰레드를 신경 쓰지 않아도 되는 환경에서는 StringBuilder를 사용하는 것이 적절하다.

- Synchronized 란 :

동기화란 하나의 자원(데이터)에 대해서 여러 스레드가 사용하려고 할 때 한 시점에서 하나의 스레드만 사용할 수 있도록 하는 것이다.

12. GC에 대해 설명해보라.

- 불필요한 메모리를 정리해주는 프로그램

OS에게 메모리를 요구할 때 실행된다.

C에는 이게 없어서 되게 힘들다 한다.

13. primitive type과 reference type 설명

- Primitive Type (기본형) : int, char, long, double.... 같은 자료형

- Reference Type (참조형) : Class, Interface, Array.. 등등

14. DI 란?

- 의존성주입, 객체 내부에서 인스턴스를 생성하는 것이 아니라 외부 컨테이너에서 미리 생성 후 객체로 주입

15. json 이란?

- JSON은 경량(Lightweight)의 DATA-교환 형식

- Javascript에서 객체를 만들 때 사용하는 표현식을 의미한다.

- JSON 표현식은 사람과 기계 모두 이해하기 쉬우며 용량이 적어서, 최근에는 JSON이 XML을 대체해서 데이터 전송 등에 많이 사용한다.

- 특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서 JSON 포맷의 데이터를 핸들링할 수 있는 라이브러리를 제공한다.

16. xml 이란?

- XML은 HTML과 매우 비슷한 문자 기반의 마크업 언어(text-based markup language)

17. RDBMS 란?

- 관계형 데이터베이스 관리 시스템.

- 테이블-칼럼 형태의 데이터 저장 방식

- 테이블과 테이블 간의 연관관계(주로 외래 키 형태)를 이용해 필요한 정보를 구하는 방식

18. MVC 란?

- model-view-controller의 약자로 소프트웨어 공학에서 애플리케이션을 구성하는 패턴

19. 싱글톤 패턴이란?

- 애플리케이션이 시작될 때 어떤 클래스가 최초 한 번만 메모리를 할당하고(Static) 그 메모리에 인스턴스를 만들어 사용하는 디자인 패턴.

- 고정된 메모리 영역을 얻으면서 한 번의 new로 인스턴스를 사용하기 때문에 메모리 낭비를 방지

20. 팩토리 패턴이란?

- 객체를 생성하기 위한 인터페이스를 정의하는데, 어떤 클래스의 인스턴스를 만들지는 서브클래스에서 결정

- 팩토리 메서드 패턴을 이용하면 클래스의 인스턴스를 만드는 일을 서브클래스에게 맡기게 된다.

21. HashCode 설명

- 자바에서 동일한 이름의 객체가 여러 개 생성될 수 있는데 이 객체들은 이름만 같을 뿐 서로 다른 상태를 가진 가진 객체들이 되는 것이다.

이런 객체를 구별하기 위해, 고유한 정수값으로 출력시켜주는 메소드가 바로 hashCode()다

22. Call by Reference, Call by Value 의 차이점은?

- Call by reference : 참조에 대한 호출

- Call by Value : 값에 대한 호출

23. 링크드리스트와 배열의 차이?

- 10번 참조

24. Git, Github사용 여부

- 사용 O

25. 오픈소스 사용여부

- 사용 O

26. 프로젝트는 어떤 걸 해봤나?

- 포트폴리오 참조

27. DB는 어떤걸 써봤나?

- MYSQL

28. SQL은 어느 정도 할 줄 아나?

- 적당히...

29. DDL, DML, DCL에 대해 설명해보세요.

- DDL : Data Define Language 의 약자로, 스키마/도메인/테이블/뷰/인덱스를 정의/변경/제거할 때 사용하는 언어이다.

테이블을 생성하고, 테이블 내용을 변경하고, 테이블을 없애버리는 것.

흔히 CREATE, ALTER, DROP 을 떠올리면 된다.

- DML : Data Manipulation Language 의 약자로, Query(질의)를 통해서 저장된 데이터를 실질적으로 관리하는 데 사용한다.

테이블 안의 데이터 하나하나를 추가하고 삭제하고 수정하는 것.

흔히 INSERT, DELETE, UPDATE 를 떠올리면 된다.

- DCL : Data Control Language 의 약자로, 보안/무결성/회복/병행 제어 등을 정의하는데 사용한다. 데이터 관리 목적.

흔히 COMMIT, ROLLBACK, GRANT, REVOKE 를 떠올리면 된다.

30. 자바스크립트와 제이쿼리의 차이점은?

- javascript를 더 사용하기 쉽게 만든 것이 Jquery.

31. 게시판 만들 줄 아나?

- 넵

32. 그럼 게시판에서 페이징의 로직 순서에 대해 설명해보라.

- 따로 정리

- 또는 게시판 기능의 간단한 로직 설명해보라.

- CURE

33. SPA의 장점 및 단점은?

- Single Page Application

- 초기 구동 속도 느림

- SEO 검색엔진 최적화 문제

**전산 기초**

**개발상식**

**객체 지향 프로그래밍이란 무엇인가?**

절차지향은 오로지 컴퓨터 관점에서의 프로그래밍 패러다임이라면 객체지향은 인간이 구분할 수 있는 요소를 객체로 표현한 인간 중심적 프로그래밍 패러다임이다. 객체는 기억장소와 이 기억장소의 값을 변경할 수 있는 연산의 집합 클래스가 정의되면 객체를 선언할 수 있다. 많은 객체 지향 언어에서 객체는 클래스로 그룹화 된다. 생성된 클래스로 객체의 특정 예를 생성하는데 이를 인스턴스라고 부른다. 객체 간의 정보 교환이 모두 메시지 교환을 통해 일어난다.

**함수형 프로그래밍이란?**

객체 지향 언어의 유일한 단점은 객체가 상태를 가지고 있다는 것이다. 함수형 프로그래밍에선 모든 오퍼레이션이 새로운 오퍼레이션을 가진다. 인풋을 변경시키지 않고 변경한 아웃풋만 제공한다. 대표적인 함수형 언어 Rust-Lang에서는 기본적으로 변수가 immutable로 선언되고 변수의 변경이 필요한 경우 mutable로 따로 선언해야 한다. 함수형 프로그래밍이 언제나 높은 효율을 보이는 것은 아니다. 상황에 따라 다르며 특히 맵리듀스를 해야하는 상황이라면 효율적이다.

**RESTful API란 무엇인가?**

REST는 자원(Resource), 행위(Verb), 표현(Representations)으로 구성된 웹의 장점과 HTTP의 우수성을 적극 활용할 수 있는 아키텍처이다. HTTP URI를 통해서 자원을 명시하고 HTTP Method(POST, GET, PUT, DELETE)를 통해서 해당 자원의 CURD 연산을 적용하는 것을 의미한다. 코드의 재사용성을 높일 수 있으며 프론트엔드와 백엔드의 완전한 분업이 가능하다.

**TDD란 무엇이며 어떠한 장점이 있는가?**

테스트 주도 개발(Test Driven Development), 테스트를 먼저 만들고 테스트를 통과하기 위한 코드를 작성하는 것을 의미하며 모듈화가 자연스럽게 잘 이루어지면서 개발이 진행된다. 테스트 커버리지가 높아져 리팩토링과 유지보수가 쉬워진다.

**MVC 패턴이란 무엇인가?**

모델, 뷰, 컨트롤러가 분리된 형태의 아키텍처. 세가지가 결합된 형태에서는 어플리케이션의 확장이 어렵다. 모델은 데이터 처리, 뷰는 사용자 인터페이스 처리, 컨트롤러는 비즈니스 로직을 처리하는 등 각각의 요소가 하나의 역할만 담당한다.

**Git과 GitHub에 대해서**

버전 관리를 위한 도구

**Docker와 VM의 차이점**

Dcoker는 하드웨어를 가상화하는 계층(Hyper-V)이 없으며, 호스트의 자원을 직접 이용하기 때문에 메모리 접근, 파일시스템, 네트워크 속도가 가상머신에 비해 월등히 빠르다.

**자료구조**

**순차 자료구조 vs 연결 자료구조**

순차 자료구조는 메모리 상에서 일렬로 나열된 데이터형이며 연결 자료구조는 메모리 상에서는 분산되어 있지만 하나의 노드가 다음으로 이어지는 포인터를 가지고 있어 연속적으로 접근이 가능한 데이터형이다.

* 데이터 삽입 : 순차 자료구조의 마지막에 데이터를 삽입하는 경우는 빠르지만 처음과 중간에 삽입하는 경우에는 자리교환으로 인한 오버헤드가 발생하여 느리다. 연결 자료구조는 데이터를 어디에 삽입하던 해당 위치까지 엑세스하는 시간만 소요되지만 자료 추가시 링크만 교체하면 되므로 빠르다.
* 데이터 읽기 : 연결 자료구조는 위치를 알던 모르던 관계없이 헤더부터 찾으려는 위치까지 탐색해야 하므로 느리다. 순차 자료구조는 탐색하려는 위치를 알고 있다면 즉시 엑세스 할 수 있으므로 빠르며 탐색하려는 위치를 모른다고 하더라도 메모리 상에서 근접한 데이터의 접근이 더 빠르므로 연결 자료구조보다 빠르다.

**스택과 큐**

스택은 가장 먼저들어온 개체가 가장 마지막에 나가는(Last in first out, LIFO) 방식을 사용하는 자료구조이며 대부분은 펜케이크에 비유하곤 한다. 아래에서 위로 데이터를 쌓고 위에서 아래로 데이터를 지운다. 큐는 가장 먼저 들어온 개체가 가장 먼저 나가는(First in first out, FIFO) 방식을 사용하는 자료구조이며 대부분은 택시정거장에 비유하곤 한다.

**트리**

트리는 계층구조를 표현하기 위해서 사용하는 자료구조이다. 트리의 크기를 제한하면 트리의 연산이 단순해지고 명확해지는데 차수를 2개 이하로 정의한 것이 이진 트리이다. 이진 트리의 종류로는 스레드 이진 트리, 이진 탐색 트리, AVL 트리 등이 있다.

**힙(Heap)**

힙는 이진트리의 한 종류로 나열한 두 가지 조건이 성립하는 이진 트리를 의미한다. 우선 완전 이진 트리여야 한다. 또한 부모 노드와 자식 노드간에 크기 관계가 성립해야 한다. 루트 노드가 가장 크고 자식 노드가 부모 노드보다 작으면 최대 힙, 반대의 경우는 최소 힙이다. 각각 최댓값과 최솟값에 접근하기 위해 사용하며 성능이 매우 좋다.

**해쉬**

해쉬는 임의의 크기를 가진 데이터를 고정된 크기의 값으로 변환시키는 것을 말한다. 해쉬를 이용하여 임의의 데이터를 숫자로 변경하는 해쉬 함수를 정의하면 배열의 인덱스를 원하는 데이터 값으로 저장하거나 찾을 수 있다. 기존에는 탐색을 위한 시간이 소모됨에 반해 해쉬를 이용하면 즉시 데이터에 엑세스 할 수 있다.

**그래프**

정점과 에지로 이루어진 형태의 자료구조다. 에지의 방향성의 존재 유무에 따라서 유향 그래프와 무향 그래프로 분리되며 에지가 가중치를 가지고 있지고 있다면 가중치 그래프라고 부른다. 그래프는 행렬과 연결리스트를 활용하여 구현할 수 있는데 행렬의 경우 정점의 존재 여부와 상관없이 항상 n^2의 공간 복잡도를 가진다.

**트리와 그래프의 차이점**

구현이나 동작의 형태로는 트리와 그래프는 동일하다고 생각된다. 트리와 그래프의 유일한 차이점은 그래프에는 루트 노드가 존재하지 않는 다는 것. 트리에서는 루트 노드로 되돌아오는 에지가 존재하지 않다는 점이다.

**네트워크**

**GET, POST 방식의 차이점**

* GET : 데이터가 Header의 URI에 담겨서 전송되므로 데이터 크기가 제한적이며 데이터가 사용자에게 그대로 노출되어 보안이 필요한 경우 적절하지 않다. GET 요청은 캐싱이 가능하므로 데이터가 사용자에게 노출되어도 상관없다면 적극적으로 활용하는게 좋다.
* POST : 데이터의 크기가 GET 방식보다 현저하게 크며 데이터가 일반 사용자에게 노출되지 않으므로 안정적으로 데이터를 전송할 수 있다. POST는 대부분 데이터의 변경을 위해서 사용한다.

**TCP 3-way-handshake**

* Client는 Server에 접속 요청 메세지(SYN)을 전송하고 SYN\_SEND 상태가 된다.
* Server는 SYN 요청을 받고 Client에 요청을 수락(SYN+ACK)하고 SYN\_RECEIVED 상태가 된다.
* Client는 Server에게 수락 확인(ACK)를 보내고 Server는 ESTABLISHED 상태가 된다.

**TCP와 UDP의 차이점**

* TCP : 신뢰성과 순차적인 전달이 필요한 경우 사용한다. TCP 서비스는 송신자와 수신자 모두가 소켓이라고 부르는 종단점을 생성함으로써 이루어진다. TCP는 멀티캐스팅이나 브로드캐스팅을 지원하지 않는다.
* UDP : 비연결형 프로토콜이며 손상된 세그먼트의 수신에 대한 재전송을 하지 않는다. UDP를 사용하는 것에는 DNS가 있다. 사전에 설정이 필요하지 않으며 그 후에 해제가 필요하지 않다.

**HTTP와 HTTPS의 차이점**

HTTP의 문제점은 평문 통신이기에 도청이 가능하다는 것과 통신 상대를 확인하지 않기 때문에 위장이 가능하다는 것과 완정성을 증명할 수 없기 때문에 변조가 가능하다는 것이다. TCP/IP 구조의 통신은 패킷을 수접하는 것만으로도 도청할 수 있다. 이를 해결하는 방법으로 통신 자체에 SSL을 적용하거나 콘텐츠 자체를 암호화하고 복호화하는 처리를 진행할 수 있다.

**DNS round robin 방식**

DNS를 이용해서 하나의 서비스에 여러 대의 서버를 분산 시키는 방법이다. 동일한 이름으로 여러 레코드를 등록 시키면 질의 할 때마다 다른 결과를 반환하며, 이 동작을 이용함으로써 여러 대의 서버에 처리를 분산 시킬 수 가 있다. 단점은 아래와 같다.

* 서버의 수 만큼 공인 IP 주소가 필요하다.
* DNS 질의 결과 캐싱으로 인해 균등하게 분산되지 않는다.
* 서버가 다운되어도 확인이 어렵다,

**웹 통신의 흐름**

**운영체제**

**프로세스와 스레드의 차이**

* 프로세스 : CPU가 처리할 수 있는 작업의 단위 (자세한 건 PCB 참고)
* 스레드 : 프로세스 안에서 공유 자원 (공유 자원 관리법은 Mutex, Semaphore 참고) 을 가지고 작업을 하는 흐름의 단위 둘의 가장 큰 차이점은 ‘공유 자원을 가지고 있는가’ 로 표현될 수 있다. 기본적으로 프로세스는 프로세스의 메모리를 가지고 있고, 스레드는 프로세스 안에서 공유 메모리를 지니고 있다.

**스케줄러의 종류**

스케줄러는 적용 시점에 따라 선점, 비 선점 2가지로 구분할 수 있다

* 비선점형 : 상당히 Greedy한 스케줄링으로 프로세스가 CPU에 할당되면 그 프로세스가 종료하거나 인터럽트가 발생하기 전까지 계속 실행하도록 보장한다. 따라서 문맥 교환이 거의 일어나지 않아 오버헤드가 상당히 적고 프로세스의 처리 시간을 미리 예측이 가능하다. 하지만 Starving 현상이 일어날 위험성이 크다.
* 선점형 : 어떤 프로세스가 CPU에 할당되어 있어도 인터럽트를 발생시키면 사용중인 프로세스를 중지할 수 있다. 즉 모든 프로세스에게 동일한 사용 시간을 보장할 수 있어 Starving 현상이 일어나지 않지만 문맥 교환이 많이 일어나 오버헤드가 커질 수 있다.

**CPU 스케줄러**

**동기와 비동기의 차이**

동기와 비동기의 가장 큰 차이점은 return 값을 가져올 때의 시점이다 동기는 System Call이 끝날 때 까지 기다렸다가 return 값을 가져 오고 비동기는 System Call이 완료 되지 않아도 나중에 끝나면 return 값을 가져온다.

**멀티스레드**

**프로세스 동기화**

**메모리 관리 전략**

**가상 메모리**

**캐시의 지역성**

**데이터베이스**

**사용하는 이유**

**인덱스**

**정규화**

**트렌잭션**

**Statement vs PrepareStatement**

**NoSQL**

**프로그래밍 언어**

**Java**

**JVM, GC의 원리**

GC는 JVM의 Heap Memory 내에 존재하는 Garbage를 찾아 내 처리하여 Heap Memory를 회수하는 작업이다. 어떤 객체가 Garbage인지 판별은 Reachability라는 개념을 사용한다. Reachability 가 Unreachable 이 되면 GC 가 일어나게 되는 것이다.

**콜렉션**

데이터의 집합으로 어떤 데이터를 담을 수 있는 자료구조를 가진 클래스이다. 크게 Collection과 Map으로 나누어 진다. 컬렉션은 Value 값만 있는 자료구조인 List(LinkedList, Vector, ArrayList 등등)로 이루어져 있고 Map은 Key 와 Value가 있는 자료구조이다.

**어노테이션**

자바 소스 코드에 주석처럼 추가하여 사용할 수 있는 메타데이터의 일종이다.

**제내릭**

제네릭은 다양한 타입의 객체들을 다루는 메서드나 컬렉션에 컴파일 시 타입 체크를 해주는 기능이다. 클래스 내부에서 사용할 데이터 타입을 나중에 인스턴스를 생성할 때 확정하는 것을 제네릭이라 한다. 객체의 타입을 컴파일 시에 체크하기 때문에 객체의 타입 안정성을 높이고 형변환의 번거로움이 줄어든다.

**파이널**

로컬 원시 변수의 경우 immutable하여 상수에 사용 가능 객체 변수에 final를 하게 되면 레퍼런스 변경이 불가능하다. 하지만 그 객체 자체는 mutable하다. Method에 final 선언 시 override가 불가능 하다. Class에 final 선언 시 상속이 불가능하다.

**오버라이딩 오버로딩**

오버로딩 : 같은 이름의 Method를 여러 개 가지면서 Return 타입과 매개 변수가 다른 클래스를 정의 오버라이딩 : 상속 시 부모 Class의 Method를 상속 받은 Class에서 재정의(부모 Class의 Method와 같은 이름) 해서 사용한다

**Access Modifier**

접근 지정자로 OOP의 정보 은닉, 캡슐화를 구현해 준다.

**Wrapper class**

Java의 객체는 기본적으로 기본형, 참조형 둘로 나뉘어 진다. Wrapper class는 기본형 객체를 참조형 객체로 구현해 놓은 것이다. Java 1.5부터는 기본형과 Wrapper class 간에 Boxing, Unboxing으로 상호 변환이 쉽게 가능하다. Wrapper class를 사용하는 이유는 매개변수로 객체가 요구될 때. 기본형 값이 아닌 객체로 저장해야 할 때. 객체간 비교가 필요할 때. 등

**Multi-Thread 환경에서의 개발**

**Javascript**

**자바스크립트 이벤트 루프**

**Hoisting**

**Closure**

**this에 대해서**

**Promise**

**Python**

**Generator**

**클래스를 상속했을 때 메서드 실행 방식**

인스턴스의 메서드를 실행한다고 가정할 때 \_\_getattribute\_\_()로 bound 된 method 를 가져온 후 메서드를 실행한다. 상속할때 왼쪽에 가까운 순서대로 우선순위가 높아진다.

**GIL과 그로인한 성능 문제**

GIL은 여러 스레드가 동시에 실행되는 걸 막는다. 덕분에 구현이 간단하고 레퍼런스 카운팅 오버헤드가 적다. GIL 때문에 성능 문제가 대두되는 경우는 압축, 정렬, 인코딩 등 수행시간에 CPU 의 영향이 큰 작업(CPU bound)을 멀티 스레드로 수행하도록 한 경우다. 이 땐 GIL 때문에 멀티 스레드로 작업을 수행해도 싱글 스레드일 때와 별반 차이가 나지 않는다. 이를 해결하기 위해선 멀티 스레드는 파일, 네트워크 IO 같은 IO bound 프로그램에 사용하고 멀티 프로세스를 활용해야한다.

**GC 작동 방식**

레퍼런스 카운팅과 순환 참조 감시

**Celery**

**PyPy가 CPython 보다 빠른 이유**

CPython은 일반적인 인터프리터임에 반해 PyPy는 JIT(Just In Time)이 겸비된 인터프리터이기 때문이다. JIT는 인터프리터의 단점을 보완하기 위해 기계어 코드를 생성하면서 그 코드를 캐싱하여, 같은 함수가 여러 번 불릴 때 매번 기계어 코드를 생성하는 것을 방지한다.

**메모리 누수가 발생할 수 있는 경우**

메서드 기본 인자 값으로 mutable 객체를 사용하고 있는 경우, 클래스 내 \_\_del\_\_ 메서드를 재정의 하는 경우

**Duck Typing**

Duck typing이란 특히 동적 타입을 가지는 프로그래밍 언어에서 많이 사용되는 개념으로, 객체의 실제 타입보다는 객체의 변수와 메소드가 그 객체의 적합성을 결정하는 것을 의미한다. 하나의 외부 메서드를 통해서 서로 다른 객체의 인자의 같은 이름의 메서드를 호출할 수 있다.

**분야별**

**프론트엔드**

**브라우저의 작동 원리**

**Document Object Model**

**CORS**

**크로스 브라우징**

**웹 성능 문제점**

**서버 사이드 렌더링 vs 클라이언트 사이드 렌더링**

**CSS Methodology**

**normalize.css vs react.css**