면접예상질문1_20230627

과정	java
회차	1회차
생성자	영선 김영선
작성자	코리아IT아카데미_김영선강사

※ 매끄럽게 답변할 수 있도록 연습 할 것

1. Java의 특징은 무엇인가요?

▼ 키워드

1. 운영체제 독립성 (Operating System Independence)
2. 객체지향 언어 (Object-Oriented Language)
3. 안전성과 보안성 (Safety and Security)
4. 멀티쓰레딩 지원 (Multithreading Support)
5. 풍부한 라이브러리 (Rich Library)
6. 자동 메모리 관리 (Automatic Memory Management)
7. 네트워크와 분산 환경 지원 (Network and Distributed Environment Support)

▼ 참고 답변

Java의 특징은 여러 가지가 있습니다.

첫번째로 Java는 운영체제에 독립적으로 실행될 수 있는 특징을 가지고 있어 다양한 플랫폼에서 동작할 수 있습니다.

두번째로 Java는 객체지향 언어로 개발되어 코드의 재사용성, 유지보수성, 확장성이 우수합니다.

세번째 특징으로는 안전성과 보안성도 강조되며, 타입 안전성과 예외 처리를 통해 안정적인 프로그램을 개발할 수 있습니다.

네번째로 Java는 멀티쓰레딩을 지원하여 동시에 여러 작업을 처리하고 성능을 향상시킬 수 있습니다.

다섯번째로 풍부한 라이브러리를 포함하여 다양한 기능을 제공하며, 가비지 컬렉션을 통한 자동 메모리 관리 기능을 갖추고 있습니다.

네트워크와 분산 환경에서의 개발을 지원하여 클라이언트-서버 애플리케이션 및 분산 시스템을 개발할 수 있습니다.

▼ 참고내용

1. 운영체제 독립성: Java는 운영체제에 독립적으로 실행될 수 있습니다. 개발한 코드를 한 번 작성하면 다양한 운영체제에서 동작할 수 있습니다.
2. 객체지향 언어: Java는 객체지향 프로그래밍(OOP)을 지원하는 언어입니다. 이는 코드의 재사용성, 유지보수성, 확장성을 높여 개발 효율성을 향상시킵니다.
3. 안전성과 보안성: Java는 타입 안전성을 강조하고 예외 처리를 강제하여 안정적인 프로그램을 개발할 수 있도록 합니다. 또한, 보안 기능을 내장하여 악성 코드의 실행을 제한합니다.
4. 멀티쓰레딩 지원: Java는 멀티쓰레드 프로그래밍을 쉽게 구현할 수 있습니다. 이를 통해 동시에 여러 작업을 처리하고 성능을 향상시킬 수 있습니다.
5. 풍부한 라이브러리: Java는 다양한 표준 라이브러리와 서드파티 라이브러리를 제공합니다. 이러한 라이브러리들은 다양한 기능을 갖추어 개발자의 생산성을 높여줍니다.
6. 자동 메모리 관리: Java는 가비지 컬렉션을 통해 자동적으로 메모리를 관리합니다. 개발자는 명시적인 메모리 할당과 해제에 대해 걱정하지 않아도 됩니다.
7. 네트워크와 분산 환경 지원: Java는 네트워크 프로그래밍과 분산 환경에서의 개발을 지원합니다. 이를 통해 클라이언트-서버 애플리케이션 및 분산 시스템을 개발할 수 있습니다.

2. 객체지향 프로그래밍(Object-Oriented Programming)에 대해 설명해주세요.

▼ 키워드

1. 클래스 (Class)
2. 객체 (Object)
3. 상속 (Inheritance)

4. 다형성 (Polymorphism)
5. 캡슐화 (Encapsulation)
6. 추상화 (Abstraction)
7. 메시지 전달 (Message Passing)

▼ 참고답변

객체지향 프로그래밍은 프로그래밍 패러다임으로, 코드를 객체들의 집합으로 구성합니다.

현실 세계의 개념을 모델링하고 코드의 재사용성, 유지보수성, 확장성을 향상시킵니다.
객체지향 프로그래밍에서는 클래스를 정의하여 객체를 생성하고 객체들 간에 상호작용하면서 애플리케이션을 개발합니다.
주요 개념으로는 클래스, 객체, 상속, 다형성, 캡슐화, 추상화, 메시지 전달 등이 있습니다.
이를 통해 코드의 구조화와 모듈화가 용이해지며, 대규모 프로젝트의 개발과 유지보수가 편리해집니다.

▼ 참고

상속 : 기존 클래스를 확장하고 재사용할 수 있습니다. 상속을 통해 부모 클래스의 속성과 동작을 자식 클래스가 상속받아 사용할 수 있으며, 코드의 중복을 피하고 유지보수성을 높일 수 있습니다.

다형성 : 객체지향 프로그래밍에서 중요한 개념입니다. 다형성을 통해 같은 타입이지만 다른 객체들이 동일한 메소드를 다른 방식으로 구현할 수 있습니다. 이는 코드의 유연성과 확장성을 높여줍니다.

캡슐화 : 객체의 내부 데이터와 동작을 외부에서 접근할 수 없도록 숨기고, 외부에서는 공개된 인터페이스를 통해 상호작용할 수 있도록 하는 개념입니다. 이를 통해 객체의 내부 구현에 대한 의존성을 낮출 수 있습니다.

추상화 : 객체들이 가진 공통된 특징을 추출하여 클래스로 정의하는 과정을 말합니다. 이를 통해 복잡한 현실 세계를 단순화하고 모델링할 수 있습니다.

객체 간의 상호작용은 메시지 전달을 통해 이루어집니다. 객체는 다른 객체에게 메시지를 보내고, 해당 객체는 메시지를 처리하여 원하는 동작을 수행합니다.

3. Java의 자료형(Data Types)에는 어떤 것들이 있나요?

▼ 키워드

1. 기본 자료형 (Primitive Types)
    - 정수형 (int, long)
    - 실수형 (float, double)
    - 문자형 (char)
    - 논리형 (boolean)
2. 참조 자료형 (Reference Types)
    - 클래스 (Class)
    - 인터페이스 (Interface)
    - 배열 (Array)

▼ 참고답변

Java의 자료형(Data Types)은 크게 기본 자료형(Primitive Data Types)과 참조 자료형(Reference Data Types)으로 나뉩니다.

기본 자료형(Primitive Data Types)에는 정수형(byte, short, int, long), 실수형(double, float), 문자형(char), 논리형(boolean) 이 있고 참조 자료형(Reference Data Types)에는 클래스(Class), 인터페이스(Interface), 배열(Array), 열거형(Enum), 기타 자료형 (String, Wrapper 클래스 등) 등이 있습니다.

기본 자료형은 JVM에 의해 직접적으로 관리되며, 메모리에 값을 저장합니다. 참조 자료형은 객체의 주소를 가리키는 참조 변수를 통해 관리됩니다.

기본 자료형은 값 자체를 저장하고, 연산을 수행할 수 있습니다. 참조 자료형은 객체를 생성하고, 객체의 메소드를 호출하거나 필드에 접근할 수 있습니다.

4. JVM(Java Virtual Machine)은 무엇이고 어떤 역할을 하나요?

▼ 키워드

1. 가상 머신 (Virtual Machine)
2. 바이트코드 (Bytecode)
3. 중개자 역할 (Mediator Role)

4. 메모리 관리 (Memory Management)
5. 가비지 컬렉션 (Garbage Collection)

▼ 참고답변

JVM은 Java Virtual Machine의 약자로, 자바 프로그램을 실행하기 위한 가상 머신(컴퓨터)입니다. JVM은 운영체제와 자바 프로그램 사이에서 중개자 역할을 수행합니다.

JVM은 자바 컴파일러에 의해 생성된 바이트코드를 이해하고 해석하여 기계어로 변환하고 실행합니다. 이를 통해 자바 프로그램은 운영체제에 종속되지 않고 여러 플랫폼에서 실행될 수 있습니다.

JVM은 또한 메모리 관리, 가비지 컬렉션, 스레드 관리, 예외 처리 등의 기능을 담당하여 자바 프로그램의 안정성과 이식성을 보장합니다. JVM은 자바 애플리케이션을 실행하는 데 필요한 실행환경을 제공하며, 자바의 특징 중 하나인 운영체제 독립성을 실현하는 데 중요한 역할을 합니다.

5. Java의 메모리 구조는 어떻게 되어있나요?

▼ 키워드

1. 스택 (Stack)
2. 힙 (Heap)
3. 메소드 영역 (Method Area 또는 클래스 영역)

▼ 참고내용

1. 메소드 영역(Method Area 또는 Class Area)
    - 메소드 영역은 JVM이 시작될 때 생성되며, 클래스와 관련된 정보를 저장하는 공간입니다.
    - 메소드 영역은 클래스 로더에 의해 로드된 클래스의 바이트코드, 상수 풀(Constant Pool), 메서드와 필드의 정보, 클래스와 인터페이스의 구조 등을 저장합니다.
    - 메소드 영역은 모든 스레드가 공유하는 영역이며, JVM이 종료될 때까지 유지됩니다.
2. 힙(Heap)
    - 힙은 동적으로 생성된 객체와 배열이 저장되는 공간입니다.

- 힙은 가비지 컬렉터에 의해 관리되며, 더 이상 사용되지 않는 객체는 자동으로 해제됩니다.
- 객체의 생성과 소멸은 힙 영역에서 이루어지며, 힙은 모든 스레드가 공유하는 영역입니다.

3. 스택(Stack):
   - 스택은 각 스레드마다 개별적으로 생성되며, 메서드 호출과 관련된 정보를 저장하는 공간입니다.
   - 스택은 메서드 호출 시마다 프레임(Frame)을 생성하여 호출된 메서드의 지역 변수, 매개변수, 리턴 값 등을 저장합니다.
   - 메서드 호출이 완료되면 해당 프레임은 스택에서 제거됩니다.
   - 스택은 호출된 메서드의 순서에 따라 동작하며, 스택은 각 스레드마다 독립적으로 존재합니다.
4. PC 레지스터(Program Counter Register)
   - 현재 실행 중인 스레드의 다음 명령어의 주소를 가리키는 역할을 합니다.
5. 네이티브 메소드 스택(Native Method Stack)
   - 네이티브 메소드 스택은 자바 언어가 아닌 다른 언어로 작성된 네이티브 메소드의 호출과 관련된 정보를 저장하는 공간입니다.

▼ 참고답변

Java의 메모리 구조에는 메소드 영역(Method Area 또는 Class Area), 힙(Heap), 스택(Stack), PC 레지스터Program Counter Register), 네이티브 메소드 스택(Native Method Stack) 등이 있습니다.

메소드영역은 JVM이 시작될 때 생성되며 클래스와 관련된 정보를 저장하는 공간으로 JVM이 종료될 때까지 유지되고, 힙영역은 동적으로 생성된 객체와 배열이 저장되는 공간으로 가비지 컬렉터에 의해 관리되며 더 이상 사용되지 않는 객체는 자동으로 해제됩니다. 스택영역은 메소드의 호출과 관련된 정보를 저장하는 공간으로 호출된 메소드의 지역변수, 매개변수, 리턴값등이 저장됩니다.
PC 레지스터는 현재 실행 중인 스레드의 다음 명령어의 주소를 가리키는 역할을 하고, 네이티브 메소드 스택은 자바 언어가 아닌 다른 언어로 작성된 네이티브 메소드의 호출과 관련된 정보를 저장하는 공간입니다.