♦ 변수 ⁄ 프로그램 실행 중에 데이터를 저장하기 위한 공간

ν 변수 값은 프로그램 수행 중 변경될 수 있음

♦ 변수 선언 ⁄ 변수의 데이터 타입에서 정한 크기의 메모리 할당  
기본 타입 : 8 개 ν boolean ν char ν byte ν short ν int ν long ν float ν double

레퍼런스 타입 : 1 개

네 가지 용도로 사용됨

ν 클래스(class)에 대한 레퍼런스

ν 인터페이스(interface)에 대한 레퍼런스

ν 배열(array)에 대한 레퍼런스

ν 열거(enum)에 대한 레퍼런스

case의 비교 값과 일치하면 해당 case의 실행문장 수행

ν break를 만나면 switch문을 벗어남

⁄ case의 비교 값과 일치하는 것이 없으면 default 문 실행

ν default문은 생략 가능

♦ 배열(array) ⁄ 인덱스와 인덱스에 대응하는 데이터들로 이루어진 자료 구조

ν 배열을 이용하면 한 번에 많은 메모리 공간의 선언 가능

⁄ 배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간

ν 원소 데이터들이 순차적으로 저장됨

ν 인덱스를 이용하여 원소 데이터 접근

ν 반복문을 이용하여 처리하기에 적합한 자료 구조

♦ 예외(Exception)

⁄ 실행 중 오동작이나 결과에 악영향을 미치는 예상치 못한 상황 발생

ν 자바에서는 실행 중 발생하는 에러를 예외로 처리

♦ 예외 발생 경우 ⁄ 정수를 0으로 나누는 경우 ⁄ 배열의 크기보다 큰 인덱스로 배열의 원소를 접근하는 경우⁄ 정수를 읽는 코드가 실행되고 있을 때 사용자가 문자를 입력한 경우



실세계 객체의 특징 ⁄ 객체마다 고유한 특성(state, property)와 행동(behavior)를 가짐⁄ 다른 객체들과 정보를 주고 받는 등, 상호작용하면서 살아감

캡슐화 : 객체를 캡슐로 싸서 내부를 볼 수 없게 하는 것

⁄ 객체의 가장 본질적인 특징 : 외부의 접근으로부터 객체 보호

클래스(class): 객체 모양을 선언한 틀(캡슐화하는 틀)

⁄ 객체: 생성된 실체(instance)

상속(inheritance) ⁄ 상위 개체의 속성이 하위 개체에 물려짐 ⁄ 하위 개체가 상위 개체의 속성을 모두 가지는 관계

다형성(polymorphism) ⁄ 같은 이름의 메소드가 클래스 혹은 객체에 따라 다르게 구현되는 것

⁄ 다형성 사례

ν 메소드 오버로딩 : 한 클래스 내에서 같은 이름이지만 다르게 작동하는여러 메소드

ν 메소드 오버라이딩 : 슈퍼 클래스의 메소드를 동일한 이름으로 서브 클래스마다 다르게 구현

절차 지향 프로그래밍 ⁄ 작업 순서를 표현하는 컴퓨터 명령 집합 ⁄ 함수들의 집합으로 프로그램 작성

♦ 객체 지향 프로그래밍 ⁄ 컴퓨터가 수행하는 작업을 객체들간의 상호 작용으로 표현⁄ 클래스 혹은 객체들의 집합으로 프로그램 작성

클래스 ⁄ 객체의 속성(state, property)과 행위(behavior) 선언

⁄ 객체의 설계도 혹은 틀, 클래스는 객체의 타입이다.

⁄ 속성 -> 멤버 변수(필드), 행동 -> 멤버 메소드(함수) 구현

객체 ⁄ 클래스의 틀로 찍어낸 실체 ν 프로그램 실행 중에 생성되는 실체 ν 메모리 공간을 갖는 구체적인 실체 ν 인스턴스(instance)라고도 부름

생성자 이름은 클래스 이름과 동일 ♦ 생성자는 여러 개 작성 가능(생성자 중복) 매개 변수의 개수와 타입이 달라야 함

♦ 생성자의 목적은 객체 생성 시 초기화 this ⁄ 객체 자신에 대한 레퍼런스

메소드 ⁄ 메소드는 C/C++의 함수와 동일 ⁄ 자바의 모든 메소드는 반드시 클래스 안에 있어야 함(캡슐화 원칙)

오버로딩(Overloading) ⁄ 한 클래스 내에서 두 개 이상의 이름이 같은 메소드 작성ν 메소드 이름이 동일하여야 함 ν 매개 변수의 개수가 서로 다르거나, 타입이 서로 달라야 함

♦ 자바의 접근 지정자 ⁄ 4가지: private, protected, public, 디폴트(접근지정자 생략)

public 멤버 ⁄ 패키지에 관계 없이 모든 클래스에게 접근 허용

♦ private 멤버 ⁄ 동일 클래스 내에만 접근 허용 ⁄ 상속 받은 서브 클래스에서 접근 불가 ♦ protected 멤버 ⁄ 같은 패키지 내의 다른 모든 클래스에게 접근 허용⁄ 상속 받은 서브 클래스는 다른 패키지에 있어도 접근 가능

♦ 디폴트(default) 멤버 ⁄ 같은 패키지 내의 다른 모든 클래스에게 접근 허용

final 클래스 - 더 이상 클래스 상속 불가능 final class FinalClass {

♦ final 메소드 - 더 이상 오버라이딩 불가능 protected final int finalMethod()

WORA(Write Once Run Anywhere)

⁄ 한번 작성되어 컴파일 된 자바 코드는 모든 플랫폼에서 자바 가상 머쉰(JVM)만 있으면 바로 실행

⁄ C/C++ 등 기존 언어가 가진 플랫폼 종속성 극복

ν OS, H/W에 상관없이 자바 프로그램이 동일하게 실행

⁄ 네트워크에 연결된 어느 클라이언트에서나 실행

ν 웹 브라우저, 분산 환경 지원

♦ WORA를 가능하게 하는 자바의 특징

⁄ 바이트 코드(byte code) ν 자바 소스코드를 컴파일 한 일종의 목적코드

ν JVM에 의해 해석되고 실행됨

⁄ JVM(Java Virtual Machine)

ν 자바 바이트 코드를 실행하는 자바 가상 기계(소프트웨어)