# 자바 개념 정리

## 개념 흐름(순서도 요약)

**프로그램 시작**

→ `main(String[] args)` 진입

→ **클래스/객체 개념** (속성=필드, 행위=메서드) 이해

→ **배열**: 선언 → 생성(크기 확정) → 사용 (다차원·가변 포함)

→ **메서드**: 시그니처, 반환/매개변수, static/인스턴스, 오버로딩

→ **접근제한자**로 정보 은닉·캡슐화

→ **추상화** (인터페이스/추상클래스)로 설계 분리

→ **인스턴스화** (생성자, new)로 실제 객체 생성·사용

---

## 1) 클래스·객체·필드·메서드 (속성과 행위)

### 개념

- **클래스**: 객체를 만들기 위한 설계도.
- **객체(인스턴스)**: 클래스로부터 생성된 실체.
- **필드(Field)**: 객체의 상태/속성을 나타내는 변수.
  - 인스턴스 필드 → 객체마다 별도로 존재
  - 클래스(정적) 필드 → `static`으로 선언, 클래스에 하나만 존재 (모든 객체가 공유)
- **메서드(Method)**: 객체/클래스가 수행하는 행위(기능).
  - 인스턴스 메서드 → 객체 생성 후 사용 가능
  - 정적 메서드 → `static`으로 선언, 클래스 이름으로 바로 사용 가능

### 포인트

- "속성(필드) ↔ 행위(메서드)"는 객체지향의 기본 단위.
- 불변성이 필요하면 `final` 필드, setter 제거로 안전성을 높임.
- 정적은 공유, 인스턴스는 개별.

```
class Counter {
    private int count;          // 인스턴스 필드
    private static int total;   // 클래스(정적) 필드

    public void inc() { // 인스턴스 메서드
        count++;
        total++;
    }
    public int getCount() { return count; }
    public static int getTotal() { return total; } // 정적 메서드
}
```

---

# 2) main 메서드

## 개념

- 자바 프로그램의 시작점.
- 시그니처는 반드시 `public static void main(String[] args)`.

## 포인트

- `static`인 이유 → 프로그램 시작 시 객체가 아직 없으므로 클래스 차원에서 호출.
- `args`는 커맨드라인 인수 배열.

```
public class App {
    public static void main(String[] args) {
        System.out.println("시작!");
    }
}
```

---

# 3) 배열

## 개념

- **선언:** `int[] a;`

- **생성(크기 확정)**: `a = new int[5];`
- **사용**: `a[0] = 10;`

### 포인트

- 배열은 **정적 길이**: 생성 후 크기 변경 불가.
- 길이는 `a.length` (문자열은 `s.length()`, 메서드 형태).

```
int[] a;
a = new int[3];
a[0] = 10; a[1] = 20; a[2] = 30;
System.out.println(a.length); // 3
```

---

## 4) 다차원 배열과 가변 배열

### 개념

- `int[][]`는 **int[]를 원소로 갖는 배열**. 즉 `arr[i]`는 `int[]`, `arr[i][j]`가 `int`.
- 가변 배열(Jagged Array): 행마다 열의 길이가 다를 수 있음.

### 포인트

- `int[][]`에 직접 `int`를 넣을 수 없음. `arr[i] = new int[]{1,2};`처럼 배열을 넣어야 함.

```
int[][] grid = new int[2][3];
grid[0][1] = 7;

int[][] jagged = new int[3][]; // 행만 생성
jagged[0] = new int[1];
jagged[1] = new int[3];
jagged[2] = new int[2];
```

---

## 5) 배열에서 자주 나는 오류

- `ArrayIndexOutOfBoundsException`: 인덱스를 범위(0~length-1) 밖으로 접근.

- `NullPointerException`: 가변 배열에서 하위 배열을 아직 생성하지 않은 경우.

```
int[] a = new int[3];
System.out.println(a[3]); // ❌ 범위 초과

int[][] b = new int[2][];
System.out.println(b[0][0]); // ❌ b[0]이 null
```

---

# 6) 가변 길이 배열 vs 가변 인자(Varargs)

- **가변 배열(Jagged Array)**: 배열 구조에서 각 행 길이가 제각각.
- **가변 인자(Varargs)**: 메서드 문법 `int... nums` (내부적으로 `int[]`).

```
static int sum(int... nums) {
    int s = 0; for (int n : nums) s += n;
    return s;
}
int s1 = sum(1,2,3);
int s2 = sum(new int[]{4,5,6});
```

---

# 7) 메서드

## 개념

- 형식: `[접근제한자] [static] 리턴타입 메서드명(매개변수...) { ... }`
- 오버로딩: 같은 이름, 다른 매개변수 허용.
- static 메서드: 객체 없이 사용.
- 인스턴스 메서드: 객체 상태 변경/읽기.

## 포인트

- 자바는 모두 **call-by-value**.
- 원시값은 값 복사, 참조값은 주소 복사 → 내부 상태 변경은 반영되지만 참조 자체를 교체하면 영향 없음.

## 8) 접근제한자

- **public**: 어디서나 접근 가능
- **protected**: 같은 패키지 + 하위 클래스 접근 가능
- **default**: 같은 패키지 내부만
- **private**: 같은 클래스 내부만

### 포인트

- 캡슐화: 필드를 private으로 두고 getter/setter로 통제.

```
class Account {
    private int balance;
    public int getBalance() { return balance; }
    public void deposit(int amount) {
        if (amount <= 0) throw new IllegalArgumentException();
        balance += amount;
    }
}
```

## 9) 추상화

### 개념

- 복잡한 대상에서 핵심만 추출.
- 인터페이스: “무엇을 할 수 있는가”만 정의.
- 추상 클래스: 공통 구현 + 추상 메서드 혼합 가능.

```
interface PayService { void pay(int amount); }

class CardPay implements PayService {
    public void pay(int amount) { /* 카드 결제 */ }
}
class KakaoPay implements PayService {
    public void pay(int amount) { /* 카카오페이 결제 */ }
```

```
}
```

## 10) 인스턴스화와 생성자

- `new` 키워드로 객체 생성 → 힙(Heap)에 저장, 참조 변수는 스택에 위치.
- 아무도 참조하지 않으면 GC(가비지 컬렉터)가 제거.
- 생성자: 객체 초기화 담당, 클래스 이름과 동일, 리턴 타입 없음.

```
class Person {
    private final String name;
    private final int age;
    public Person(String name, int age) {
        this.name = name; this.age = age;
    }
}
Person p = new Person("Kim", 20); // 인스턴스화
```

## 객체와 클래스 개념 요약

- 객체는 세상의 사물/개념을 프로그램에 반영한 것.
- 클래스는 그 객체를 만들기 위한 설계도.
- 객체는 **속성(필드)**과 **행위(메서드)**를 가진다.

## 추상화 개념 요약

- 불필요한 세부 사항은 제거하고 꼭 필요한 속성과 행위만 모델링.
- 같은 객체라도 목적에 따라 추상화 결과가 달라질 수 있음.
    - 예: “고객”은 마트에서는 `구매내역`, 헬스클럽에서는 `회원권`, 비디오 대여점에서는 `대여내역`이 중요.