Java Programming | </> Algorithm & OOP | 실무 & 시험 요약

Java 핵심 실습문제 코드 & 개념 완전 정리

수학 알고리즘(약수·GCD·친화수·윤년)부터 핵심 수열(개미수열), 2차원 배열(마방진 I·II·III), 구현 게임(로또·야구 게임·달력·카드덱)까지 시험/실무 완벽 대비 가이드

한경닷컴 IT교육센터 강병진 강사 | 자바 실습과제 노트

01. 약수, 최대공약수(GCD), 최소공배수(LCM)

Basic Math

1. 약수 (Divisor)

핵심 원리

어떤 수 n을 나누어 떨어지게 하는 수 (나머지 `n % i == 0`)

```java
int n = 12;
for (int i = 1; i <= n; i++) {
    if (n % i == 0) {
        System.out.print(i + " ");
    }
}
// 출력: 1 2 3 4 6 12
```

2. 최대공약수 (GCD)

뺄셈 연산 방식

두 수가 같아질 때까지 반복하여 **[큰 수 - 작은 수]**를 수행

```java
int a = 10, b = 20;
while (a != b) {
    if (a > b) a -= b;
    else b -= a;
}
int gcd = a; // gcd = 10
```

3. 최소공배수 (LCM)

암기 공식

LCM = (두 수의 곱) / 최대공약수(GCD)

```java
int num1 = 10, num2 = 20;
// GCD를 사전 계산 후 활용
int lcm = (num1 * num2) / gcd;

System.out.println("LCM: " + lcm);
// 출력: 20
```

02. 친화수 (Amicable) & 완전수 (Perfect Number)

Number Theory

[공통 핵심] 진약수의 합 함수

학습 포인트

진약수란 **자기 자신을 제외한 약수**를 의미합니다. 반복문 범위를 num / 2까지 설정하면 최적화됩니다.

```java
// 진약수의 합(Proper Divisor Sum) 구하기
public static int getProperDivisorSum(int num) {
    int sum = 0;
    for (int i = 1; i <= num / 2; i++) {
        if (num % i == 0) {
            sum += i;
        }
    }
    return sum;
}
```

완전수 & 친화수 판별 로직

개념 비교

- **완전수(Perfect):** 진약수 합이 자기 자신과 같음 (6 = 1+2+3)
- **친화수(Amicable):** A의 진약수 합 = B이고, B의 진약수 합 = A (220 ↔ 284)

```java
// 1. 완전수 조건
boolean isPerfect = (getProperDivisorSum(n) == n);

// 2. 친화수 조건 (a = 220, b = 284)
int sumA = getProperDivisorSum(a);    // 284
int sumB = getProperDivisorSum(sumA); // 220

if (sumB == a && a != sumA) {
    System.out.println(a + "와 " + sumA + "는 친화수!");
}
```

03. 윤년(Leap Year) 판별 알고리즘

Condition Logic

윤년 판별 조건 및 함수

⚠ 암기 필수 윤년 조건식

연도가 4로 나누어 떨어지고 100으로 나누어 떨어지지 않거나, 또는 400으로 나누어 떨어질 때

```
public static boolean isLeapYear(int year) {
    // (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
    return (year % 4 == 0 && year % 100 != 0)
            || (year % 400 == 0);
}
```

2000년~현재 윤년 출력 예제

💡 실습 응용

반복문 내에서 판별 함수를 호출하여 true인 연도만 골라 출력합니다.

```
int startYear = 2000;
int currentYear = 2026;

for (int y = startYear; y ≤ currentYear; y++) {
    if (isLeapYear(y)) {
        System.out.println(y + "년: 윤년입니다.");
    }
}
// 2000년, 2004년, 2008년, 2012년 ... 윤년 출력
```

04. 개미 수열 (Look-and-say Sequence)

String Manipulation

</> 개미수열 생성 루프 코드

```
String str = "1";
for (int i = 1; i < n; i++) {
    StringBuilder next = new StringBuilder();
    char target = str.charAt(0);
    int count = 1;

    for (int j = 1; j < str.length(); j++) {
        if (str.charAt(j) == target) {
            count++;
        } else {
            next.append(target).append(count);
            target = str.charAt(j);
            count = 1;
        }
    }
    str = next.append(target).append(count).toString();
}
```

단계별 진행 상태 추적

- 1단계 1 (기본 시작)
- 2단계 11 (1이 1개)
- 3단계 12 (1이 2개)
- 4단계 1121 (1이 1개, 2가 1개)
- 5단계 122111 (1이 2개, 2가 1개, 1이 1개)

> **Key point**
> 문자 변경 시점과 마지막 문자 처리 append를 놓치지 않는 것이 핵심입니다.

05. 로또 번호 생성기 & LottoStore 객체 설계

OOP & Collections

1. Lotto 클래스 (중복 제거)

HashSet 활용

Set 자료구조는 중복을 허용하지 않으므로 6개 난수를 간단히 추출할 수 있습니다.

```
public class Lotto {
    private int[] numbers = new int[6];

    public Lotto() {
        Set<Integer> set = new HashSet<>();
        while (set.size() < 6) {
            set.add((int)(Math.random() * 45) + 1);
        }
        int idx = 0;
        for (int n : set) numbers[idx++] = n;
        Arrays.sort(numbers); // 오름차순 정렬
    }
}
```

2. LottoStore & 당첨 결과 판별

등수 매칭 기준

6개 일치: 1등 | 5개: 2등 | 4개: 3등 | 3개: 4등 | 2개: 5등

```
public class LottoStore {
    public Lotto[] buyLotto(int count) {
        Lotto[] lottos = new Lotto[count];
        for (int i = 0; i < count; i++) {
            lottos[i] = new Lotto();
        }
        return lottos;
    }
    // 구매한 로또 배열과 당첨 번호를 비교하여 등수 산출
}
```

06. 마방진 I - 홀수 마방진 알고리즘

2D Array

이동 규칙 (la Loubere 법칙)

- **시작 위치:** 첫 행의 중앙 `(0, n / 2) = 1`
- **기본 이동:** 대각선 위방향 `(x - 1, y - 1)`
- **행/열 범위를 벗어난 경우:**
  - `x < 0` 이면 `x = n - 1`
  - `y < 0` 이면 `y = n - 1`
- **이동할 칸에 이미 값이 있을 때:**
  - 이동 전 위치로 돌아와 `(x + 1, y)` 아래로 1칸 이동

홀수 마방진 채우기 루프

```
int[][] magic = new int[n][n];
int x = 0, y = n / 2;

for (int k = 1; k ≤ n * n; k++) {
    magic[x][y] = k;
    int nx = (x - 1 < 0) ? n - 1 : x - 1;
    int ny = (y - 1 < 0) ? n - 1 : y - 1;

    if (magic[nx][ny] ≠ 0) {
        x = x + 1; // 충돌 시 아래로 이동
    } else {
        x = nx; y = ny;
    }
}
```

07. 마방진 II – 4의 배수 짝수 마방진 (4n)

Pattern Matching

대칭 반전 메커니즘

채우기 방식

1부터 n^2까지 순서대로 배치한 후, 4x4 대각선 구역에 해당하는 셀은 값을 유지하고 나머지는 역순 `(n*n + 1 - val)`로 치환합니다.

구역 조건	처리 방식
4x4 대각선 라인	정방향 순서값 유지 `(val)`
대각선 외 라인	역방향 반전값 채우기 `(n*n+1-val)`

핵심 대칭 치환 코드

```
int[][] m = new int[n][n];

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        int val = i * n + j + 1;

        // 4×4 대각선 영역 조건 검사
        if ((i % 4 == j % 4) || (i % 4 + j % 4 == 3)) {
            m[i][j] = val;
        } else {
            m[i][j] = (n * n + 1) - val; // 역순 값
        }
    }
}
```

08. 마방진 III – 4의 배수가 아닌 짝수 (6, 10...)

LUX / 4-Quadrant

4분할 (A, B, C, D) 구조

영역	위치	오프셋 (n/2 홀수 마방진 기반)
A (좌상)	0 ~ n/2-1	base + 0
B (우하)	n/2 ~ n-1	base + (n/2)² * 1
C (우상)	0 ~ n/2-1	base + (n/2)² * 2
D (좌하)	n/2 ~ n-1	base + (n/2)² * 3

A/D 영역 열 교환 (Swap)

> **열 교환 이유**
> 가로·세로·대각선의 합을 동일하게 보정하기 위해 A영역과 D영역의 특정 열 (Column)을 교환합니다.

```
// A영역과 D영역 간 열 값 교환 (k = n/4)
for (int i = 0; i < n/2; i++) {
    for (int j = 0; j < k; j++) {
        int temp = grid[i][j];
        grid[i][j] = grid[i + n/2][j];
        grid[i + n/2][j] = temp;
    }
}
```

09. 숫자 야구 게임 (Number Baseball)

Game Logic

1. 컴퓨터 난수 생성 (중복 불가)

```
int[] com = new int[3];

for (int i = 0; i < 3; i++) {
    com[i] = (int)(Math.random() * 9) + 1;
    // 이전 숫자들과 중복 검사
    for (int j = 0; j < i; j++) {
        if (com[i] == com[j]) {
            i--; // 인덱스 되돌림
            break;
        }
    }
}
```

2. Strike & Ball 판별

판별 규칙

값과 위치 모두 일치: Strike | 값만 일치: Ball

```
int strike = 0, ball = 0;

for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (com[i] == user[j]) {
            if (i == j) strike++;
            else ball++;
        }
    }
}
// 3Strike 시 승리 및 게임 종료
```

10. 만년 달력 & 살아온 일수 계산

Date Algorithm

1. 경과일 기반 시작 요일 구하기

요일 산출식

서기 1년 1월 1일(월요일)부터 목표 년/월 1일까지 총 날짜 수를 7로 나눈 나머지가 공백수가 됩니다.

```
// 총 날짜 수 = 이전 연도 일수 + 이전 월 일수
int totalDays = (year - 1) * 365
              + (year - 1)/4 - (year - 1)/100 + (year - 1)/400;

for (int m = 1; m < month; m++) {
    totalDays += getMonthDays(year, m);
}
int startDay = (totalDays + 1) % 7; // 시작 요일
```

2. 살아온 일수 (Days Lived)

계산 로직

(오늘까지 총 경과일) - (생일까지 총 경과일) + 1

```
int birthTotal = getTotalDays(birthY, birthM, birthD);
int todayTotal = getTotalDays(nowY, nowM, nowD);

int daysLived = todayTotal - birthTotal + 1;
System.out.println("태어난 지 " + daysLived + "일째입니다.");
```

11. 트럼프 카드 & 카드덱 (Card & CardDeck)

OOP Design

Card 클래스 (개별 카드)

```
public class Card {
    private String shape;  // ♥, ♣, ♠, ◆
    private String number; // 2~10, J, Q, K, A

    public Card(String shape, String number) {
        this.shape = shape;
        this.number = number;
    }

    public String toString() {
        return shape + number;
    }
}
```

CardDeck 클래스 (52장 & 셔플)

> **Collections.shuffle()**
> ArrayList에 저장된 52장 카드를 자바 컬렉션 유틸리티로 손쉽게 섞을 수 있습니다.

```
public class CardDeck {
    private List<Card> deck = new ArrayList<>();

    public CardDeck() {
        String[] shapes = {"♥", "♣", "♠", "◆"};
        String[] nums = {"2","3","4","5","6","7","8","9","T","J","Q","K","A"};

        for (String s : shapes)
            for (String n : nums) deck.add(new Card(s, n));
    }
    public void shuffle() { Collections.shuffle(deck); }
}
```

12. 시험 & 코딩테스트 대비 핵심 복습 체크리스트

Final Checklist

1. 필수 수학식

- GCD: 벨셈 연산 루프 / 유클리드 호제법
- LCM: (a * b) / GCD
- 약수/진약수: 나머지 % == 0
- 윤년: (4배수 && !100배수) || 400배수

2. 2차원 배열 제어

- 홀수 마방진: 대각선 위 (x-1, y-1), 음수 처리 및 충돌 시 아래 이동
- 4n 마방진: 4x4 대각선 영역 반전 (n*n + 1 - val)
- 야구 게임: 2중 루프로 값/인덱스 일치 시 Strike 산출

3. OOP & 컬렉션

- HashSet: 로또 번호 중복 방지 자동 처리
- StringBuilder: 개미수열 문자열 결합 성능 향상
- Collections.shuffle(): 카드덱 섞기 유틸리티