REPORT

LAB 번호 : LAB2

과목 및 분반 : 자바프로그래밍 2분반

제출일 : 2025.04.02

학번 : 32203919 (컴퓨터공학과)

이름 : 장천명

DKU 단국대학교 DANKOOK UNIVERSITY

1번 문제

구현 방식의 설명

1. 두 개의 분수(a/b, c/d)에 대해 덧셈, 뺄셈, 곱셈, 나눗셈 결과를 분수 형태로 출력

주요 코드 설명

```
double add = (a * d + b * c) / (b * d);
```

실수 이므로 double로 정의

```
System.out.printf(format:"Fraction Addition: %.1f/%.1f + %.1f/%.1f = %.1f/%.1f\n", a, b, c, d, (a*d + b*c), (b*d)); //
System.out.printf(format:"Fraction Subtraction: %.1f/%.1f - %.1f/%.1f = %.1f/%.1f\n", a, b, c, d, (a*d - b*c), (b*d));
System.out.printf(format:"Fraction Multiplication: %.1f/%.1f * %.1f/%.1f = %.1f/%.1f\n", a, b, c, d, (a*c), (b*d));
System.out.printf(format:"Fraction Division: %.1f/%.1f / %.1f/%.1f = %.1f/%.1f\n", a, b, c, d, (a*d), (b*c));
```

소수 첫 번째 자리까지 출력이 나오도록 해야 하기에, %.1f 입력

실행 결과 화면

```
1번 문제
Fraction Addition: 3.0/7.0 + 2.0/15.0 = 59.0/105.0
Fraction Subtraction: 3.0/7.0 - 2.0/15.0 = 31.0/105.0
Fraction Multiplication: 3.0/7.0 * 2.0/15.0 = 6.0/105.0
Fraction Division: 3.0/7.0 / 2.0/15.0 = 45.0/14.0
```

2번 문제

구현 방식의 설명

1. 짝수인지를 isEven 메서드로 확인
2. 3의 배수인지 isMultiplesOfThree 메서드로 확인
3. 소수인지 isPrime 메서드로 확인

주요 코드 설명

```
return new Random().nextInt(b - a + 1) + a;
```

랜덤 숫자 생성(1~100)

```
public static void checkNumber(int num) { // checkNumber 메서드 구현
    System.out.println(num + (isEven(num) ? " is even number." : " is odd number.")); // 짝수인지 판별
    System.out.println(num + (isMultiplesOfThree(num) ? " is a multiple of 3." : " is NOT a multiple of 3.")); // 3의 배수인지 판별
    System.out.println(num + (isPrime(num) ? " is a prime number." : " is NOT a prime number.")); // 소수인지 판별별
```

삼항 연산자를 통해, true와 false에 따라서 다른 출력 값(even과 긍정문이 true, odd와 부정문이 false)

실행 결과 화면

```
2번 문제
87 is odd number.
87 is a multiple of 3.
87 is NOT a prime number.
```

3번 문제

구현 방식의 설명

1. Base^exponent를 재귀 호출을 통해 빠르게 지수 계산

주요 코드 설명

```
if (exponent == 0) return 1; // 기저 조건 설정
long half = powerCalculation(base, exponent / 2); // 절반 지수로 재귀
return exponent % 2 == 0 ? half * half : base * half * half; // 홀짝 구분
```

삼항 연산자를 통해, 참(짝)일 때는 half * half, 거짓(홀)일 때는 base * half * half 출력

실행 결과 화면

```
3번 문제
[Power Calculation using Fast Exponentiation] 3 ^ 3 = 27
```

4번 문제

구현 방식의 설명

1. 사용자가 2~20 사이 정수를 입력할 때까지 반복적으로 사용자 입력 값을 받기
2. 유효한 사용자 입력 값을 받으면 그 숫자의 구구단 출력

주요 코드 설명

```
do {
    System.out.print(msg); // msg 문자열 출력
    value = scanner.nextInt(); // 입력값 받음
} while (value < min || value > max); // min보다 작거나 max보다 크면 반복복
```

Do while 문을 통해, 먼저 입력 값을 한번 받는다. 그리고 정해진 최솟값보다 작거나, 최댓값보다 크면 또 입력 값을 받는다.

실행 결과 화면

```
4번 문제
Please enter a number between [2-20]: 2
Multiplication Table for 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
```

5번 문제

구현 방식의 설명

1. RGB 값을 입력
2. Color 열거형 중 일치하는 색을 반환

주요 코드 설명

```
for (Color c : values()) { // Color의 모든 열거 상수
    if (c.r == r && c.g == g && c.b == b) return c;
}
return null; // 일치하는 색상 없으면 null 값 반환
```

1. values() 로 인해, 모든 열거 상수가 배열
2. 그 열거형 배열을 하나씩 c에 담기
3. 그 c를 이용하여 r,g,b가 동일하면 c를 반환
4. 일치하는 색상이 없으면 null 반환

실행 결과 화면

```
5번 문제
getColor(255,0,0) --> RED
getColor(0,255,0) --> GREEN
getColor(0,0,255) --> BLUE
getColor(255,255,0) --> YELLOW
getColor(255,255,255) --> WHITE
getColor(0,0,0) --> BLACK
getColor(255,0,255) --> null
```

6번 문제

구현 방식의 설명

1. 소수점 아래 특정 자리에서 반올림된 소수 부분만 반환

주요 코드 설명

```
double floor = Math.floor(num); // 소수점 아래를 버린 정수
double fraction = num - floor; // 소수 부분만, 1번의 fraction과 충돌될 수 있지 않을까 싶은데 서로 다른
double rounded = Math.round(fraction * Math.pow(a:10, decimalPlace)) / Math.pow(a:10, decimalPlace);
return rounded; // 소수 부분 decimalPlace자리 반올림 한 것것 반환
```

1. Math.floor를 통해, 소수점 아래는 버리기
2. num – floor를 통해 소수 부분만 남기기
3. Math.pow를 통해 10의 decimalPlace제곱 계산
4. frachtion * (10의 decimalPlace제곱)을 통해, 소수점 아래 자리를 정수처럼 만듬
5. Math.round를 통해 소수점 첫 번째 자리에서 반올림
6. / Math.pow(10, decimalPlace)를 통해 원래 자리로 되돌려서 소수로 만듬

실행 결과 화면

```
6번 문제
3.14159265 Decimal5 value: 0.14159
5.983 Decimal2 value: 0.98
```

첫 번째 결과

Num 값이 3.14159265 라면, floor는 3.0이고 fraction은 0.14159265이다.

여기서 decimal값이 5이면, Math.round(0.14159265 * 100000) / 100000 이다

그러면 Math.round(14159.265) / 100000 이므로, 14159 / 10000 이다.

그래서 결과값이 0.14159가 나온다.

두 번째 결과

Num 값이 5.983 이라면, floor는 5.0이고 fraction은 0.983이다.

여기서 decimal값이 2이면, Math.round(0.983 * 100) / 100 이다

그러면 Math.round(98.3) / 100 이므로, 98 / 100 이다.

그래서 결과값이 0.98이 나온다.

7번 문제

구현 방식의 설명

1. 문자열의 모음 찾기
2. 문자열에서 모음을 발견하면 대문자로 바꾸기
3. 나머지 모음이 아닌 것들은 소문자로 바꾸기

주요 코드 설명

```
String vowels = "aeiouAEIOU";
```

모음을 vowels 문자열로 정의

```
sb.append(vowels.indexOf(c) >= 0 ? Character.toUpperCase(c) : Character.toLowerCase(c));
```

Indexof(c)가 vowels의 모음이라면 대문자로, 아니라면 소문자로 sb에 추가한다.

실행 결과 화면

```
7번 문제
AstrIngthAtlOOkslIkEthIs
hEllO wOrld, jAvA!
```

모음은 대문자로, 자음은 소문자로 출력되었다.

8번 문제

구현 방식의 설명

1. 시, 분, 초를 받아서 초로 변환
2. 총 초를 시, 분, 초로 변환

주요 코드 설명

```
int hours = total / 3600; // 시간은
int minutes = (total % 3600) / 60;
int seconds = total % 60; // 60으으
```

시간은 3600으로 나눈 몫, 분은 3600으로 나눈 나머지를 60으로 나눈 몫, 초는 3600으로 나눈 몫으로 60 나눈 나머지

실행 결과 화면

```
8번 문제
2 hours, 30 minutes, 15 seconds converted to total seconds is: 9015 sec
9015 seconds is: 2 hours, 30 minutes, 15 seconds
```

9번 문제

구현 방식의 설명

1. 입력받은 행과 열을 기준으로 좌석 생성
2. 좌석 출력
3. q 누를 때까지 반복 출력

주요 코드 설명

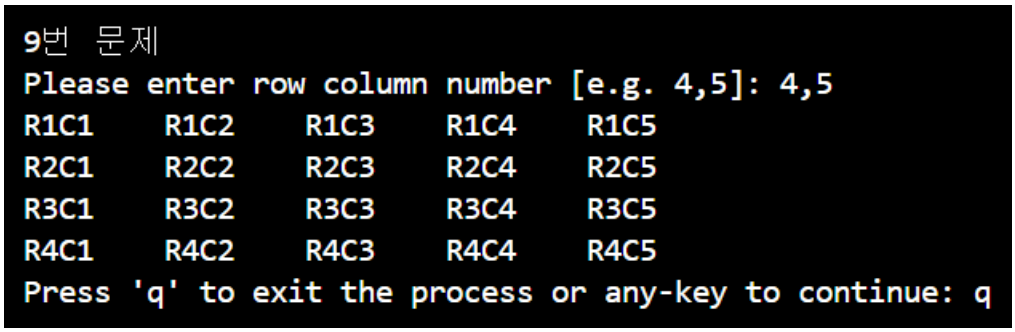
```
for (int r = 0; r < rows; r++) // 행
    for (int c = 0; c < cols; c++) // 열
        chart[r][c] = "R" + (r + 1) + "C" + (c + 1);
```

For 문 안에 for 문으로 둬서 2차원 배열의 좌석 생성

```
try { // 프로그램 멈추면 안되니깐 문제 생겨도 안내할
    String[] tokens = input.split(regex:","); // sp
    int rows = Integer.parseInt(tokens[0].trim());
    int cols = Integer.parseInt(tokens[1].trim());
```

Split(",")을 통해 ,을 기준으로 입력 값을 나눔. 첫 번째 행 값은 tokns[0], 두 번재 열 값은 tokens[1]로 정의

실행 결과 화면



입력 값으로 4,5 를 입력해서 행을 4, 열을 5로 정의. 4*5의 행렬이 행렬에 맞게 R(r+1)C(c+1)로 create되고 만들어진 seat가 print됨. q를 눌러서 종료할 수 있음

10번 문제

구현 방식의 설명

6번 이랑 동일

주요 코드 설명

6번 이랑 동일

실행 결과 화면

```
System.out.println(x:"\n10번 문제");
System.out.println("3.14159265 Decimal5 value: " + decimalValue(num:3.14159265, -1));
```

```
10번 문제
3.14159265 Decimal5 value: 0.0
```

dicmalValue 값에 음수를 집어넣으면 어떤 식으로 나오는지 궁금하여, 매개변수를 다르게 호출하여 테스트 하였다. decimal값이 음수로 줘도 에러 없이 처리되지만, 논리적으로 잘못된 값이라서 0으로 반환 값이 나왔다. 유효성 검사와 같은 코드의 필요성을 느낀다.