REPORT

LAB 번호 : LAB1

과목 및 분반 : 자바프로그래밍 2분반

제출일 : 2025.03.26

학번 : 32203919 (컴퓨터공학과)

이름 : 장천명

DKU 단국대학교 DANKOOK UNIVERSITY

1번 문제

구현 방식의 설명

1. 첫 번째 분모와 분자, 두 번째 분모와 분자의 선언
2. 첫 번째 분수와 두 번째 분수 곱하기
3. 곱의 결과를 소수로 변환(format 필요)

주요 코드 설명

1. 먼저, 문제에서 주어진 숫자를 선언 및 초기화를 한다.

```
int a = 3; // 첫 번째 분수 분자
int b = 7; // 첫 번째 분수 분모
int c = 15; // 두 번째 분수 분자
int d = 2; // 두 번째 분수 분모
```

2. 첫 번째 분수와 두 번째 분수 곱한다.

```
int child = a * c; // 분자
int parent = b * d; // 분모
```

3. 곱의 결과를 int가 아닌 double로 설정한다. 그리고 출력 할 때, %.9f를 사용해서 소수 9번째까지 소수점이 출력되도록 format한다.

```
double result = (double) child/parent; // 소수점으로 출력해야 하기 때문에 double로 바꿈
System.out.printf(format:"결과: %.9f\n", result); // 포매팅하여 아홉번째까지 소수점 출력
```

실행 결과 화면

```
첫번째 문제
결과: 3.214285714
```

곱해진 결과의 분자를 분모로 나눴을 때, 소수9번째까지 출력됨을 알 수 있다.

2번 문제

구현 방식의 설명

1. 랜덤 정수를 생성(1~100)
2. 랜덤 정수가 3의 배수인지 확인 확인
3. 3의 배수를 분기로 다른 출력값

주요 코드 설명

1. 먼저 랜덤 생성 기능을 이용하기 위해 import 필요

```
import java.util.Random; // 랜덤 생성 기능 가져오기
```

2. 랜덤 정수 객체를 생성한 후, 랜덤 정수를 생성(100) + 1을 통해 범위를 1이상 100이하로 설정

```
Random random = new Random(); // 랜덤 정수 객체 생성
int ran = random.nextInt(bound:100) + 1; // 랜덤 정수를 0~100의 값으로 설정
```

3. 생성된 값이 3의 배수인지 아닌지, if문을 통해 확인하고 그에 따른 출력

```
if (ran % 3 == 0) {
    System.out.println("The number " + ran + " is a multiple of 3"); // 3일 때, 출력값
}
else {
    System.out.println("The number " + ran + " is not a multiple of 3"); // 3이 아닐 때, 출력 값
}
```

실행 결과 화면

```
두번째 문제
The number 42 is a multiple of 3
```

```
두번째 문제
The number 46 is not a multiple of 3
```

3의 배수인 42일 때, 3의 배수라는 출력됨을 알 수 있다.

3의 배수가 아닌 46일 때, 3의 배수가 아니라고 출력됨을 알 수 있다.

3번 문제

구현 방식의 설명

1. 랜덤 기수와 지수 생성
2. 빠른 지수 계산법 함수 선언
3. Static 메서드로 빠른 지수 계산법 함수 구현
4. 지수의 짝수홀수 여부에 따라, 서로 다른 계산법 적용

주요 코드 설명

1. 위에 문제에서 랜덤 정수 객체를 생성했기 때문에, 기수와 지수 바로 생성

```
int base = random.nextInbound:t(10) + 1;       // 1부터 10까지
int exponent = random.nextInbound:t(6) + 1;    // 1부터 6까지
```

2. 빠른 지수 계산법 fastPower 함수 불러오기

```
int result3 = fastPower(base, exponent); // 빠른 지수 계산 함수
System.out.println(base + "^" + exponent + " = " + result3);
```

3. 객체 생성 필요없는 static 메서드 fastpower 구현
4. 지수가 0이 되면 어떤 수든 1이므로 재귀 종료
5. Half를 통해, 지수를 반으로 나눈 값을 재귀 호출하여 계산
6. 지수의 짝홀 여부에 따라, 각 방식의 값 return

```
public static int fastPower(int base, int exponent) { // main 함수 아래에 선언된 정적 메서드

    if (exponent == 0) return 1; // 기저 조건 추가 (왜냐하면 지수를 계속 반으로 나누어 재귀호출을 하기 때문에)

    int half = fastPower(base, exponent / 2); // half는 base와 exponent의 절반

    if (exponent % 2 == 0) { // 지수가 짝수일 때,
        return half * half; // half * half로 x^(n/2))^2 빠르게 계산
    } else { // 지수가 홀수일 때,
        return base * half * half; // x*x^(n-1)로 빠르게 계산
```

실행 결과 설명

```
세번째 문제
base = 10, exponent = 5
10^5 = 100000
```

기수가 10, 지수가 5 일 때, fastPower 메서드를 통해 100000로 계산되었음을 알 수 있다 (O(logN))

```
세번째 문제
base = 2, exponent = 2
2^2 = 4
```

기수가 2, 지수가 2 일 때, fastPower 메서드를 통해 4로 계산되었음을 알 수 있다.

4번 문제

구현 방식의 설명

1. 사용자 입력 값 받기
2. 반복문을 통해서 입력 값의 곱셈표 출력

주요 코드 설명

1. 사용자 입력을 위해 Scanner import하기

```
import java.util.Scanner; // 사용자 입력 받아들이는 기능 가져오기
```

2. 사용자 입력 값의 객체 생성 및 정수 변수 선언

```
Scanner scanner = new Scanner(System.in); // 사용자 입력값 받는 객체 생성
int input_value; // 사용자 입력값을 받는 정수 변수 선언
```

3. 먼저 do를 통해 한 번 사용자 값을 받기
4. While을 통해, 사용자 입력 값이 조건에 충족하지 않을 때, 다시 사용자 입력 값 받도록 하기

```
do { // do-while 문을 통해서 범위값 내에 받도록 설정
    System.out.println(x:"0부터 20까지 숫자를 입력하시오.");
    input_value = scanner.nextInt(); // 사용자 입력값 받기
} while ((input_value > 20) || (input_value < 0)); // 20 초과 또는 0미만일 때, 사용자 입력값 다시 입력 받기
```

5. For 문을 통해 사용자 입력 값을 1부터 10까지 곱하여 출력

```
for (int i = 1; i <= 10; i++) { // 1부터 10까지
    System.out.println(input_value + " x " + i + " = " + input_value*i); // 구구단 출력
}
```

실행 결과 설명

```
네번째 문제
0부터 20까지 숫자를 입력하시오.
-1
0부터 20까지 숫자를 입력하시오.
21
0부터 20까지 숫자를 입력하시오.
5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

범위를 벗어난 사용자 값일 때, 다시 사용자 값을 입력 받는다. 정상적인 사용자 값이 들어갔을 때, 사용자 입력값을 1단부터 10단까지 출력한다.

5번 문제

구현 방식의 설명

1. Color enum 정의
2. 사용자 Color 입력 값 받기
3. Color enum 값을 배열로 저장
4. 배열에서 random으로 색 선택
5. Random 값과 사용자 Color 입력 값 비교

주요 코드 설명

1. Color enum을 먼저 정의해준다.

```
public class LAB1 {
    enum Color { // 5번 문제를 위한 enum
        Red(r:255, g:0, b:0),
        Green(r:0, g:255, b:0),
        Blue(r:0, g:0, b:255),
        Yellow(r:255, g:255, b:0),
        White(r:255, g:255, b:255),
        Black(r:0, g:0, b:0);

        private final int r, g, b; // r g b 저장할 변수(final로 고정) 선언

        Color(int r, int g, int b) { // enum 선언으로 각각의 r g b 값으 설정
            this.r = r;
            this.g = g;
            this.b = b;
        }
    }
```

2. Scanner를 통해, 사용자 입력 값을 받는다.(4번 문제에서 사용자 입력 값을 받았기에, 개행 문자 제거 작업)

```
scanner.nextLine(); //
System.out.print(s:"색상(Red, Green, Blue, Yellow, White, Black) 중 하나를 입력하시오: ");
String userInput = scanner.nextLine(); // 앞의 nextInt 때문에 개행 문자 제거
```

3. Enum 값을 배열로 저장한 다음에, 랜덤으로 생성한다.(여기서 enum 자체가 상수로 취급되기에 const 정의된 것)

```
Color[] colors = Color.values(); // Color enum 값을 배열로 저장
Color randomColor = colors[random.nextInt(colors.length)]; // random 객체를 통해 색상 중 무작위로 선택
```

4. if문을 통해, 사용자 입력 값과 random 값 비교

```
System.out.println("사용자 입력 색상: " + userInput); // 사용자 입력 값
System.out.println("랜덤 선택 색상: " + randomColor); // 랜덤 값

if (userInput.equalsIgnoreCase(randomColor.name())) { // 사용자 입력 값이랑 랜덤 값이 같으면
    System.out.println(x:"The user selected const color and the random enum color is the same.");
} else { // 사용자 입력 값이랑 랜덤 값이 다르면
    System.out.println(x:"The user selected const color and the random enum color is different");
```

실행 결과 설명

```
다섯번째 문제
색상(Red, Green, Blue, Yellow, White, Black) 중 하나를 입력하시오: Red
사용자 입력 색상: Red
랜덤 선택 색상: Black
The user selected const color and the random enum color is different
```

사용자가 입력한 Red와 Random 값이 다르면 different를 출력한다.

```
다섯번째 문제
색상(Red, Green, Blue, Yellow, White, Black) 중 하나를 입력하시오: Red
사용자 입력 색상: Red
랜덤 선택 색상: Red
The user selected const color and the random enum color is the same.
```

사용자가 입력한 Red와 Random 값이 같으면 same을 출력한다.

6번 문제

구현 방식의 설명

1. for 문으로 1월부터 12월 설정
2. switch문으로 case를 days를 3가지로 나눔
3. 매개변수에 따른 값 설정

주요 코드 설명

1. for문을 통해 1월부터 12월까지 설정

```
for (int month = 1; month <= 12; month++) { // 1월부터 12월까지
    int days = 31;  // int days을 31로 선언
```

2. switch문을 통해 case를 통해 days 28, 30, 31로 설정(break를 통해서 케이스에 해당하면 정지하도록 하기)

```
switch(month) {
    case 2: // 2월
        days = 28;
        break; // case 문은 break를 꼭해줘야함. 만약 여기에 break 가 없으면 밑의 코드도 실행되서 30이됨.
    case 4: case 6: case 9: case 11: // 30일로 끝나는 달
        days = 30;
        break;
    default: // 디폴트값을 31로 설정
        days = 31;
}

System.out.println(month + " month, " + days + " days"); // 정상적으로 나오는지 확인인
```

실행 결과 설명

```
여섯번째 문제, your code
1 month, 31 days
2 month, 28 days
3 month, 31 days
4 month, 30 days
5 month, 31 days
6 month, 30 days
7 month, 31 days
8 month, 31 days
9 month, 30 days
10 month, 31 days
11 month, 30 days
12 month, 31 days
```

Switch 문에 따라 month가 2일 때는 28일이 출력되었으며, month가 4, 6, 9, 11 일 때는 30일이 출력되며, 나머지는 31일이 출력된다.