평가 시험

주의 사항

- 본 시험은 자바 언어에 대한 이해와 프로그래밍에 대한 지식을 확인하기 위한 시험입니다.
- 본 시험에서 제시된 문제 중 해결 가능하다고 판단되는 문제 중심으로 풀어 주시기 바랍니다.
  - 인수 조건/결과 조건으로 표기된 부분의 구현이 어려운 경우, 고려하지 않아도 됩니다.
- 본 시험 결과는 상담, 학습 방향 및 향후 실력 향상을 위한 기본 자료로 사용될 뿐 평가 보고서에는 반영되지 않습니다.
  - 따라서, 최대한 자기 실력으로 구현해 주시기 바랍니다.

제출 방법

- 학교와 이름으로 폴더를 만들어 저장합니다.
  - 대학교_이름
- 생성된 폴더를 압축합니다.
  - xx대학교_이름.zip
- 압축된 파일을 nhnexam@gmail.com으로 전송합니다.

간단한 계산 해보기

문제 1-1. 두 정수의 최대 공약수를 구하라.

두 다항식 또는 자연수 사이의 최대공약수를 구하는 알고리즘 중 하나인 유클리드 호제법을 이용해 구할 수 있다.

$$gcd(a, b) = \begin{cases} gcd(b, a\%b) & b \neq 0 \\ a & b = 0 \end{cases}$$

- 양의 두 정수 a와 b가 있다.
  - 음수는 양수로 변환합니다.
- 큰 수(a)를 작은 수(b)로 나눈다.
- 나머지가 0이면, b는 최대공약수이다.
- 나머지가 0이 아니면, b와 나머지의 최대공약수를 구한다.
- 계산은 다음과 같다.

```
192 % 72 = 2 ... 48
 72 % 48 = 1 ... 24
 48 % 24 = 2 ... 0

최대 공약수는 24
```

테스트 코드

```
public static void main(String[] args) {
    System.out.println("14와 49의 최대 공약수는 " + Exam1.gcd(14, 49));
    System.out.println("49와 14의 최대 공약수는 " + Exam1.gcd(49, 14));
    System.out.println("0과 5의 최대 공약수는 " + Exam1.gcd(0, 5));
    System.out.println("5와 0의 최대 공약수는 " + Exam1.gcd(5, 0));
}
```

결과는 다음과 같다.

```
14와 49의 최대 공약수는 7
49와 14의 최대 공약수는 7
0과 5의 최대 공약수는 5
5와 0의 최대 공약수는 5
```

문제 1-2. 피보나치수열에서 n번째 요소를 구하라.

피보나치수열의 정의는 아래와 같습니다.

$$f(n) = \begin{cases} f(n-1) + f(n-2) & n > 1 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

테스트 코드

```
public static void main(String[] args) {
    System.out.println("f(5) = " + Exam1.fibonacci(5));
    System.out.println("f(10) = " + Exam1.fibonacci(10));
}
```

결과는 다음과 같다.

```
f(5) = 5
f(10) = 55
```

진수 변환 함수 만들기

- 진수 변환 함수는 다음과 같이 정의된다.

```
public class Exam2 {
    public static int convert2to10(String value) {
```

```
        ...
    }


    public static int convert16to10(String value) {
        ...
    }


    public static String convertNtoM(String value, int sourceNotation, int targetNotation) {
        ...
    }
}
```

문제 2-1. 2진수를 받아서 10진수로 변환하라.

- 2진수를 문자열로 받는다.
- 10진수로 변환하여 출력한다.
- 16진수가 아닌 경우, 예외를 발생시킨다.

테스트 코드

```
public class Exam2_1 {
    public static void main(String[] args) {
        String [] binaries = {
            "0",
            "1",
            "10110",
            "10010110",
            "010110"
        };

        for(String binary : binaries) {
            try {
                System.out.println(binary + " -> " + Exam2.convert2to10(binary));
            } catch(ArithmeticException ignore) {
                System.out.println(binary + "은 2진수가 아닙니다.");
            }
        }
    }
}
```

결과는 다음과 같다.

```
0 -> 0
1 -> 1
```

```
10110 -> 22
10010110 -> 150
010110 is not binary number.
```

문제 2-2. 16진수를 받아서 10진수로 변환하라.

- 16진수를 문자열로 받는다.
- 10진수로 변환하여 출력한다.
- 16진수가 아닌 경우, 예외를 발생시킨다.

테스트 코드

```
public class Exam2_2 {
    public static void main(String[] args) {
        String [] hexdecimals = {
            "0",
            "1",
            "A",
            "1A",
            "0A1B",
            "AbCdEF"
        };

        for(String hexdecimal : hexdecimals) {
            try {
                System.out.println(hexdecimal + " -> "
                        + Exam2.convert16to10(hexdecimal));
            } catch(ArithmeticException ignore) {
                System.out.println(hexdecimal + "은 16진수가 아닙니다.");
            }
        }
    }
}
```

결과는 다음과 같다.

```
0 -> 0
1 -> 1
A -> 10
1A -> 26
16진수가 아닙니다.
AbCdEF -> 11259375
```

문제 2-3. 임의의 진수 문자열을 받아서 임의의 진수 문자열로 변환하라.

- 입력 문자열, 입력 진수, 반환 진수를 갖는 함수를 정의한다.

테스트 코드

```
public class Exam2_3 {
    public static void main(String[] args) {
        String [] hexdecimals = {
            "0",
            "1",
            "A",
            "1A",
            "0A1B",
            "AbCdEF"
        };

        int sourceNotation = 16;
        int targetNotation = 2;

        for (String hexdecimal : hexdecimals) {
            try {
                System.out.println(hexdecimal + " -> "
                        + Exam2.convertNtoM(hexdecimal, sourceNotation, targetNotation));
            } catch (ArithmeticException ignore) {
                System.out.println(sourceNotation + "진수가 아닙니다.");
            }
        }
    }
}
```

결과는 다음과 같다.

```
0 -> 0
1 -> 1
A -> 1010
1A -> 11010
16진수가 아닙니다.
AbCdEF -> 101010111100110111101111
```

클래스 만들기

문제 3-1. 유리수를 만들어 보자.

- 수학에서는 수를 자연수, 정수, 유리수, 실수 등으로 다양하게 구분하지만, Java에서의 표현은 정수, 실수로만

구분된다.

- 유리수는 실수에 포함이 되기는 하지만, 유리수의 정의는 다음과 같다.
  - 정수의 비로 표현이 가능한 수
  - 정수를 이용해 분자와 분모로 표현이 가능한 수. 단, 분모는 0이 될 수 없다.

유리수를 표현해 보자.

- 정수를 이용해 생성한다.
  - 여기서 정수는 int로 제한한다.
  - 유리수 조건이 만족하지 않을 경우 에러를 발생시킨다.
  - 분모와 분자는 기약분수로 표현되는 값을 갖는다.
- 클래스에 정의되어야 하는 함수는 테스트 코드를 참고한다.

테스트 코드

```
public class Exam3_1_1 {
    public static void main(String[] args) {
        RationalNumber rn1 = new RationalNumber(1, 2);
        RationalNumber rn2 = new RationalNumber(3, 2);
        try {
            RationalNumber rn3 = new RationalNumber(13, 0);
        } catch(ArithmeticException ignore) {
            System.out.println("분모 0");
        }
        RationalNumber rn3 = new RationalNumber(0, 13);

        System.out.println("rn1 : [" + rn1.getNumerator() + ", "
                + rn1.getDemominator() + "]");
        System.out.println("rn2 : [" + rn2.getNumerator() + ", "
                + rn2.getDemominator() + "]");
        System.out.println("rn3 : [" + rn3.getNumerator() + ", "
                + rn3.getDemominator() + "]");
    }
}
```

결과는 다음과 같다.

```
분모 0
rn1 : [1, 2]
rn2 : [3, 2]
rn3 : [0, 1]
```

4칙연산을 추가해 보자.

- 4칙연산을 지원한다.
- 문자열로 출력 시 가장 단순한 형태로 출력한다.

    - 분수 표현은 기약 분수를 기본으로 한다.
    - 정수 표현이 가능한 경우 정수로 한다.
- 클래스에 정의되어야 하는 함수는 테스트 코드를 참고한다.

테스트 코드

```
public class Exam3_1_2 {
    public static void main(String[] args) {
        RationalNumber rn1 = new RationalNumber(1, 2);
        RationalNumber rn2 = new RationalNumber(3, 2);
        RationalNumber rn3 = new RationalNumber(0, 123);
        RationalNumber rn4 = rn1.add(rn2);
        RationalNumber rn5 = rn1.subtract(rn2);
        RationalNumber rn6 = rn1.multiply(rn2);
        RationalNumber rn7 = rn1.divide(rn2);

        System.out.println("rn1 : " + rn1 + ", " + rn1.toDouble());
        System.out.println("rn2 : " + rn2 + ", " + rn2.toDouble());
        System.out.println("rn3 : " + rn3 + ", " + rn3.toDouble());
        System.out.println("rn4 : " + rn4 + ", " + rn4.toDouble());
        System.out.println("rn5 : " + rn5 + ", " + rn5.toDouble());
        System.out.println("rn6 : " + rn6 + ", " + rn6.toDouble());
        System.out.println("rn7 : " + rn7 + ", " + rn7.toDouble());
    }
}
```

실행 결과는 다음과 같다.

```
rn1 : [1,2], 0.5
rn2 : [3,2], 1.5
rn3 : 0, 0.0
rn4 : 2, 2.0
rn5 : -1, -1.0
rn6 : [3,4], 0.75
rn7 : [1,3], 0.3333333333333333
```

문제 3-2. 동물을 분류해 보자.

- 동물을 포유류와 조류로 분류한다.
- 포유류(Mammal)에는 박쥐(Batman), 호랑이(Tiger), 고래(Whale)가 있다.
- 조류(Bird)에는 매(Hawk), 타조(Ostrich), 펭귄(Penguin)이 있다.
- 동물 중 일부는 헤엄을 칠 수 있다.
- 동물 중 일부는 날 수 있다.
- 테스트 코드에 나와 있는 클래스 이름을 참고하여 작성하라.

동물의 이름을 출력해 보자.

```
import java.util.Arrays;

public class Exam3_2_1 {
   public static void main(String[] args) {
      Animal [] animals = { new Batman(), new Tiger(), new Whale(),
                  new Hawk(), new Ostrich(), new Penguin() };

      for(Animal animal : animals) {
         System.out.print(animal + " ");
      }
      System.out.println();
   }
}
```

결과는 다음과 같다.

```
batman tiger whale hawk Ostrich penguin
```

동물의 이름과 종을 출력해 보자.

```
import java.util.Arrays;

public class Exam3_2_2 {
   public static void main(String[] args) {
      Animal [] animals = { new Batman(), new Tiger(), new Whale(),
                  new Hawk(), new Ostrich(), new Penguin() };

      for(Animal animal : animals) {
         System.out.print(animal + "[" + animal.species() + "] ");
      }
      System.out.println();
   }
}
```

결과는 다음과 같다.

```
batman[mammal] tiger[mammal] whale[mammal] hawk[bird] Ostrich[bird] penguin[bird]
```

헤엄을 칠 수 있는 동물 종류를 출력해 보자.

```
import java.util.Arrays;
```

```
public class Exam3_2_3 {
    public static void main(String[] args) {
        Animal [] animals = { new Batman(), new Tiger(), new Whale(),
                              new Hawk(), new Ostrich(), new Penguin() };

        for(Animal animal : animals) {
            if (animal instanceof Swimmable) {
                System.out.print(animal + " ");
            }
        }
        System.out.println();
    }
}
```

결과는 다음과 같다.

```
batman tiger whale penguin
```

날 수 있는 동물 종류를 출력해 보자.

```
import java.util.Arrays;

public class Exam3_2_4 {
    public static void main(String[] args) {
        Animal [] animals = { new Batman(), new Tiger(), new Whale(),
            new Hawk(), new Ostrich(), new Penguin() };

        System.out.println(Arrays.toString(Arrays.stream(animals)
                .filter(Flyable.class::isInstance).toArray()));
    }
}
```

결과는 다음과 같다.

```
[batman, hawk]
```

문제의 해석 및 구현

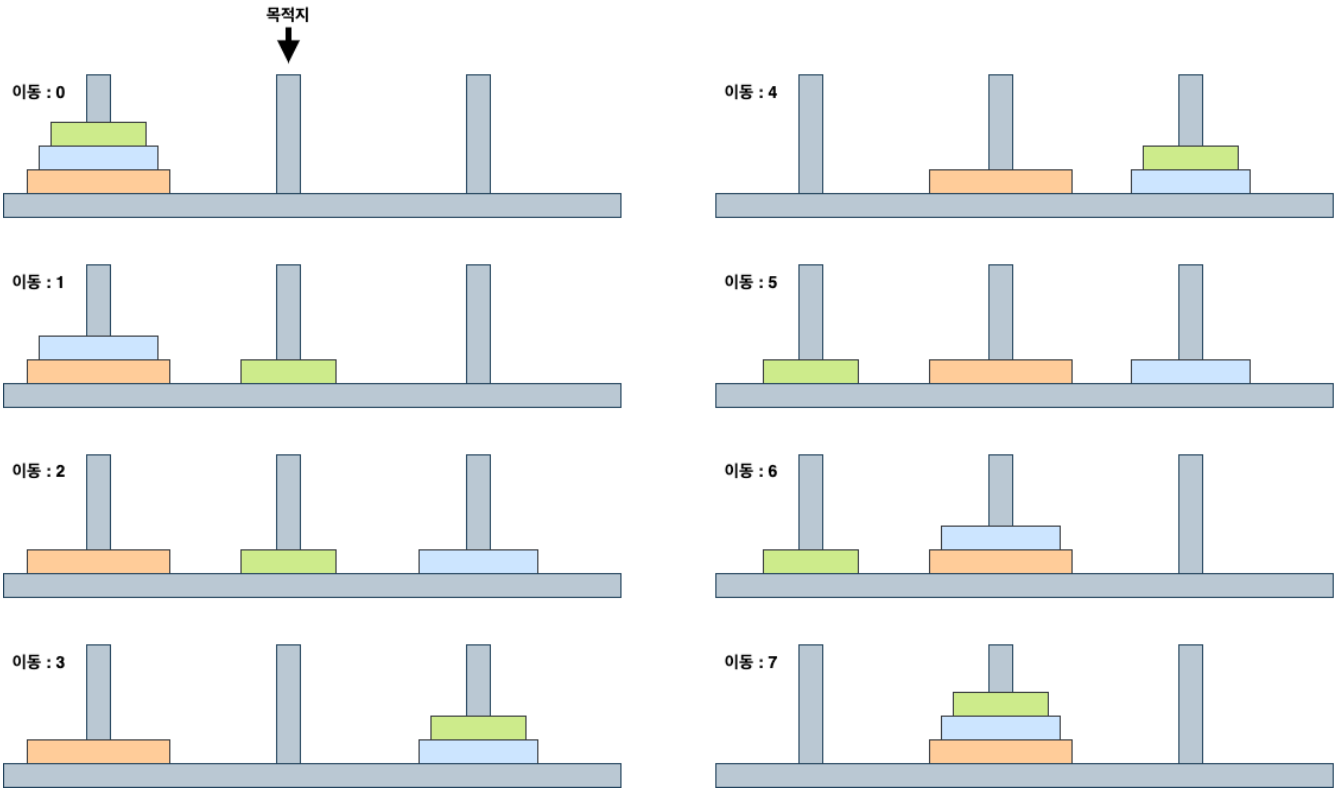
문제 4-1. 설명을 보고, 함수를 구현해 보자.

- 두 자연수 *n*과 *k*가 주어졌을 때, 1부터 *n*까지의 자연수를 원형으로 배치한다.
- 1부터 시작하여 **k**-1개의 수를 건너뛰고 다음 *k*번째 수를 제거한다.
- 숫자가 하나만 남을 때까지 반복한다.
- 예를 들어 **n**=7, **k**=3인 경우,

- 1, 2, 3, 4, 5, 6, 7 ⇒ 3
- 4, 5, 6, 7, 1, 2 ⇒ 6
- 7, 1, 2, 4, 5 ⇒ 2
- 4, 5, 7, 1 ⇒ 7
- 1, 4, 5 ⇒ 5
- 1, 4 ⇒ 1
- 4

문제 4-2. 설명을 보고, 함수를 구현해 보자.

- 3개의 막대가 있다.
- 하나의 막대에는 여러 개의 원판이 쌓여 있다.
- 원반은 위로 갈수록 점점 작아진다.
- 원반을 다른 막대로 옮기려 한다.
- 원반은 한 번에 한 개만 옮길 수 있다.
- 원반은 작은 원반 위로 옮길 수 없다.




```
import java.util.ArrayDeque;
import java.util.Arrays;
import java.util.Deque;

public class Exam4_2 {
    public static void move(Deque<Integer> from, Deque<Integer> to, int n, Deque<Integer>
temp) {
        System.out.println("From : " + toString(from)
                + ", To : " + toString(to)
```

```
                + ", Temp : " + toString(temp));
        // 코드 시작
        ...
        // 코드 끝
    }

    public static String toString(Deque<Integer> poll) {
        return  Arrays.toString(poll.toArray());
    }

    public static void main(String[] args) {
        Deque<Integer> pollA = new ArrayDeque<>();
        Deque<Integer> pollB = new ArrayDeque<>();
        Deque<Integer> pollC = new ArrayDeque<>();

        pollA.push(5);
        pollA.push(4);
        pollA.push(3);
        pollA.push(2);
        pollA.push(1);

        move(pollA, pollB, pollA.size(), pollC);
        System.out.println("From : " + toString(pollA)
                + ", To : " + toString(pollB)
                + ", Temp : " + toString(pollC));
    }
}
```