2017학년도 1학기

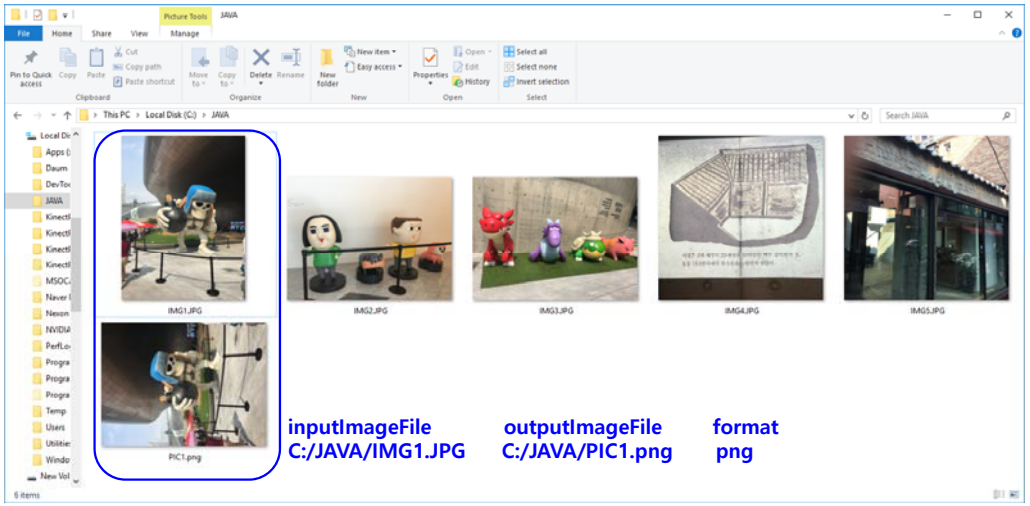
JAVA 프로그래밍 II

514770-1
2017년 봄학기
3/22/2017
박경신

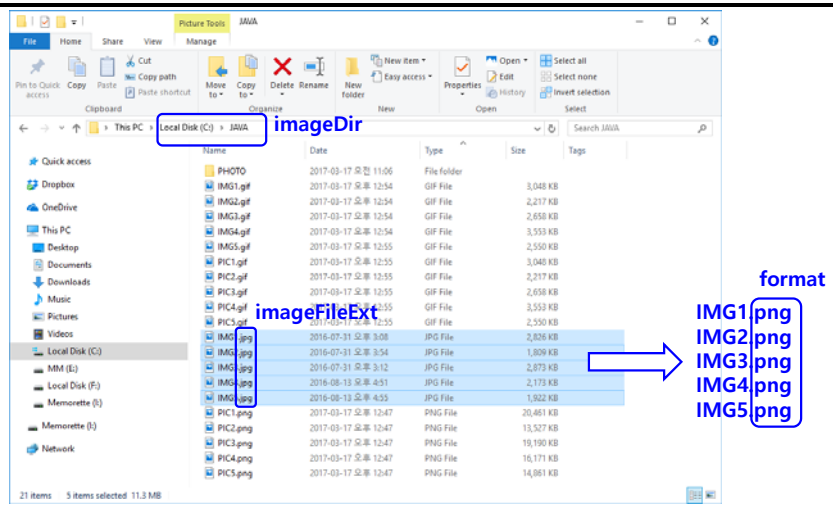
Lab #2 (디렉토리 안에 이미지 파일 포맷 변환 프로그램)

- 기존 요구사항 분석
  - Lab #1은 **"inputImageFile", "outputImageFile", "format"**에 따라 이미지 파일 포맷을 변환시켜줌 - 최소 jpg, png, gif 영상 포맷 지원
    - **ImageConverter 클래스는 이미지 파일 포맷 변환(convert) 기능 제공**
    - **ImageConverterTest 클래스는 입력이미지파일명(inputImageFile), 출력이미지파일명(outputImageFile), 이미지포맷(format)을 입력받아서 변환하는 메인 프로그램**
  - **사용자가 지정한 "특정 폴더" 안에 "특정 영상 포맷"을 "다른 영상 포맷"으로 변환**
  - **사용자가 지정한 폴더에는 원본 영상 포맷이 아닌 파일들이 있는 경우 변환 처리 안하고 그대로 둠**
  - **지정된 폴더 안에 서브 폴더가 있다면, 그 안에 있는 특정 영상 포맷도 동일하게 변환**
  - **"commands.ini" 텍스트 파일로 입력받는 기능**
- 기초 문법, array, for-each, parameter passing, file/directory, recursive call
- String, File, BufferedReader

Lab #1: Image File Convert



Lab #2: Directory Image File Convert



Code Block

- 여러 명령문을 논리적으로 결합해야 할 때 중괄호 ({ })를 사용하여 명령문 그룹을 만들어 표현 - 이러한 명령문 그룹을 코드 블록(code block)이라고 함
- 코드 블록 안에는 변수를 선언할 수 있고, 다른 코드 블록을 포함할 수도 있음

```
public class Example {
    public static void main(String[] args) {
        int outer;
        {
            int inner;
            outer = 1;
            inner = 2;
        }
        outer = 5;
        //inner = 10;       // 오류
    }
}
```

내부 코드블록

main() 메소드 코드블록

Example 클래스 코드블록

Lab #2_1 Array, For-each, Parameter Passing

- **Lab#1_5에서 inputImageFile, outputImagefile, format를 String[] commands로 입력 받아서 처리**

```
static String inputImageFile = "C:/Java/IMG1.jpg";
static String outputImageFile = "C:/Java/IMG1.png";
static String format = "png";
// array of strings
static String[] commands = new String[3];
commands[0] = "C:/JAVA/IMG1.jpg";
commands[1] = "C:/JAVA/IMG1.png";
commands[2] = "png";
```

- **Lab#1_5에서 boolean getCommands(String[] commands)에서는 commands로 사용자 입력을 받아옴**
- Java2-lab2_1 폴더에 저장 후 제출

Array

- 배열(Array)
  - 여러 개의 데이터를 같은 이름으로 활용할 수 있도록 해주는 자료 구조
    - 인덱스(Index, 순서 번호)와 인덱스에 대응하는 데이터들로 이루어진 자료 구조
    - 배열을 이용하면 한 번에 많은 메모리 공간 선언 가능
  - 배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간
    - 원소 데이터들이 순차적으로 저장됨
    - 인덱스를 이용하여 원소 데이터 접근
    - 반복문을 이용하여 처리하기에 적합한 자료 구조(주로 for 또는 for-each 반복문과 많이 사용됨)
  - 배열 인덱스
    - 0부터 시작
    - 인덱스는 배열의 시작 위치에서부터 데이터가 있는 상대 위치

Array

- 자바 배열의 필요성과 모양

Array

- 배열 선언과 배열 생성의 두 단계 필요
  - 배열 선언

```
int     intArray[];
char    charArray[];
```

또는

```
int[]   intArray;
char[]  charArray;
```

  - 배열 생성

```
intArray = new int[10];
charArray = new char[20];
```

또는

```
int intArray[] = new int[10];
char charArray[] = new char[20];
```

  - 선언과 초기화
    - 배열 생성과 값 초기화

```
// 총 10개의 정수 배열 생성 및 값 초기화
int intArray[] = {0,1,2,3,4,5,6,7,8,9};
```

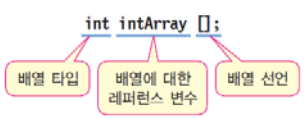
  - 잘못된 배열 선언

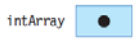
```
//int intArray[10]; // 컴파일 오류. 배열의 크기를 지정할 수 없음
```

Array

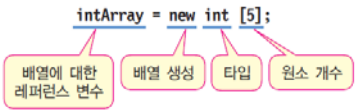
- 배열 선언과 생성

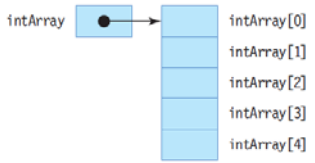
(1) 배열에 대한 레퍼런스 변수 intArray 선언





(2) 배열 생성





Array

- 배열을 초기화하면서 생성한 결과

```
int intArray[] = {4, 3, 2, 1, 0};
float floatArray[] = {0.01, 0.02, 0.03, 0.04};
```

Array

- 배열 참조
  - 생성된 1개의 배열을 다수의 레퍼런스가 참조 가능

```
int intArray[] = new int[5];
int myArray[] = intArray;
```


```
intArray[1] = 2;
myArray[1] = 6;
```

Array

- 배열 원소 접근
  - 반드시 배열 생성 후 접근

```
int intArray []; // 배열 선언
intArray[4] = 8; // 오류, intArray 배열의 메모리가 할당되지 않았음
```

  - 배열 변수명과 [] 사이에 원소의 인덱스를 적어 접근
    - 배열의 인덱스는 0부터 시작
    - 배열의 마지막 항목의 인덱스는 (배열 크기 – 1)

```
int[] intArray;
intArray = new int[10];

intArray[3]=6; // 배열에 값을 저장
int n = intArray[3]; // 배열로부터 값을 읽음
```

Array

- 배열 인덱스
  - 인덱스는 0부터 시작하며 마지막 인덱스는 (배열 크기 -1)
  - 인덱스는 정수 타입만 가능

```
int intArray [] = new int[5]; // 인덱스는 0~4까지 가능
int n = intArray[-2];  // 실행 오류. -2는 인덱스로 적합하지 않음
int m = intArray[5];  // 실행 오류. 5는 인덱스의 범위(0~4)를 넘었음
```

- 배열의 크기
  - 배열의 크기는 배열 레퍼런스 변수를 선언할 때 결정되지 않음
    - 배열의 크기는 배열 생성 시에 결정되며, 나중에 바꿀 수 없음
  - 배열의 크기는 배열의 length 필드에 저장

```
int size = intArray.length;
```

Array & For-each

- For-each 문
  - 배열이나 나열(enumeration)의 각 원소를 순차적으로 접근하는데 유용한 for 문

```
int[] num = { 1,2,3,4,5 };
int sum = 0;
// 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정
for (int k : num)
  sum += k;
System.out.println("합은 " + sum);
```

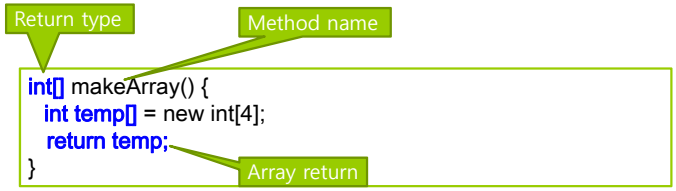
합은 15

```
String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" } ;
// 반복할 때마다 s는 names[0], names[1], ..., names[5] 로 설정
for (String s : names)
  System.out.print(s + " ");
```

사과 배 바나나 체리 딸기 포도

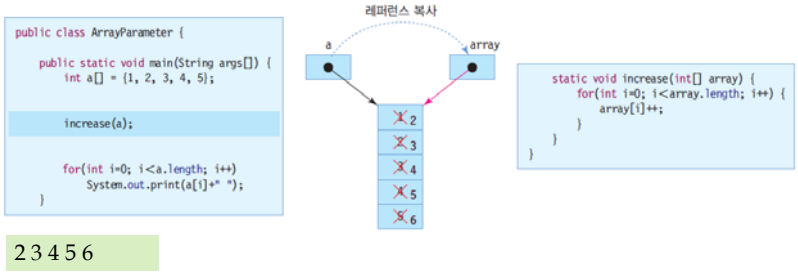
메소드에서 배열 리턴

- 메소드의 배열 리턴
  - 배열의 레퍼런스만 리턴
- 메소드의 리턴 타입
  - 메소드가 리턴하는 배열의 타입은 리턴 받는 배열 타입과 일치
  - 리턴 타입에 배열의 크기를 지정하지 않음



매개 변수에 배열이 전달되는 경우

- 매개변수에 배열이 전달되는 경우는 배열의 reference가 복사



Parameter Passing – Primitive Type

- 자바의 인자 전달 방식(Parameter Passing)
  - **값에 의한 호출(Pass-by-value)**
  - 기본 타입(Primitive Type: 예를 들어 int, double,)의 값을 전달하는 경우
    - 값이 복사되어 전달
    - 메소드의 매개 변수가 변경되어도 호출한 실제 인자 값은 변경되지 않음

Parameter Passing – Reference Type

- 자바의 인자 전달(Parameter Passing) 방식
  - 객체(class object) 혹은 배열(array)을 전달하는 경우
    - 객체나 배열의 레퍼런스만 전달**(Pass-by-reference)**
      - 객체 혹은 배열이 통째로 복사되어 전달되는 것이 아님
    - 메소드의 매개 변수와 호출한 실인자가 객체나 배열을 공유하게 됨

```
void assignArray(int[] values) {
    for (int i=0; i<values.length; i++) {
        values[i] = i; // values 배열에 0~9까지 값을 넣어줌
    }
}

int[] intArray = new int[10]; // create 10 integer array
assignArray(intArray);
for (int v : intArray) {
    System.out.println(v);
}
```

Lab #2_2 String, class

- **Lab#2_1에서 inputDir, imageFileExt, format 입력을 받아서 처리**
- **Lab#2_1에서 String[] commands는 imagedir, imageFileExt, format**

  String[] commands = {"C:/JAVA", "jpg", "png"};
  //commands[0]은 "특정 폴더(imageDir)"
  //commands[1]은 "특정 영상 포맷(imageFileExt)"
  //commands[2]는 "원하는 다른 영상 포맷(format)"
- **Lab#2_1에서 UserInput 클래스 작성**

  boolean getCommands(String[] commands)에서는 사용자로부터 "특정 폴더(imageDir)", "특정 영상 포맷(imageFileExt)", "원하는 다른 영상 포맷(format)"을 입력 받음
- Java2-lab2_2 폴더에 저장 후 제출

String

```
static String[] commands = {"C:/JAVA", "jpg", "png"};
// commands[0]은 imageDir
// commands[1]은 imageFileExt
// commands[2]는 format
UserInput.getCommands(commands);
String inputImageFile = commands[0] + "/IMG1." + commands[1]; // 기본 C:/JAVA/IMG1.jpg
String outputImageFile = commands[0] + "/PIC1." + commands[2]; // 기본 C:/JAVA/PIC1.png
String format = commands[2]; // 기본 png
// convertBy 메소드는 inputImageFile, outputImageFile, format을 입력으로 받아 이미지 변환
convertBy(inputImageFile, outputImageFile, format);
```

Lab #2_3 File/Directory, String substring/contains

- **Lab#2_2에서 사용자가 지정한 특정 디렉토리 (inputDir) 안에 있는 모든 특정 이미지 파일 (imageFileExt)를 다른 이미지 포맷 (format) 으로 모두 변환**
- **Lab#2_2에서 boolean convertImagesInDirectory(String dirPath, String ext, String format) 메소드 작성**
- **디렉토리에서 파일 목록을 읽고 원하는 확장자를 골라서 이미지 포맷 변환**
- Java2-lab2_3 폴더에 저장 후 제출

File 클래스

- File 클래스
  - 파일의 경로명을 다루는 클래스
    - java.io.File
    - 파일과 디렉터리 경로명의 추상적 표현
  - 파일 이름 변경, 삭제, 디렉터리 생성, 크기 등 파일 관리
    - File 객체는 파일 읽고 쓰기 기능 없음
  - 파일 입출력은 파일 입출력 스트림 이용

File 클래스 생성자와 주요 메소드

메소드	설명
File(File parent, String child)	parent 디렉터리에 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(String pathname)	pathname이 나타내는 File 객체 생성
File(String parent, String child)	parent 디렉터리에 child 이름의 디렉터리나 파일을 나타내는 File 객체 생성
File(URI uri)	file:URI를 추상 경로명으로 변환하여 File 객체 생성

메소드	설명
boolean mkdir()	새로운 디렉터리 생성
String[] list()	디렉터리 내의 파일과 디렉터리 이름의 문자열 배열 리턴
File[] listFiles()	디렉터리 내의 파일 이름의 File 배열 리턴
boolean renameTo(File dest)	dest가 지정하는 파일 이름 변경
boolean delete()	파일 또는 디렉터리 삭제
long length()	파일의 크기 리턴, 디렉터리나 장치 파일인 경우 0 리턴
String getPath()	파일 경로명 전체를 문자열로 변환하여 리턴
String getName()	파일 또는 디렉터리 이름을 문자열로 리턴
boolean isFile()	일반 파일이면 true 리턴
boolean isDirectory()	디렉터리이면 true 리턴
long lastModified()	파일이 마지막으로 변경된 시간 리턴
boolean exists()	파일 또는 디렉터리가 존재하면 true 리턴

파일 목록을 읽고 원하는 확장자 골라내기

- 파일 목록 뽑기
- 파일 목록의 배열에서 어떻게 원하는 파일만 뽑아낼까?
  - 폴더(디렉토리)인지 파일인지 확인
  - 파일이면 확장자가 원하는 파일인지 확인 (문자열 함수 활용)

```
import java.io.*;

File dir = new File(dirPath);
File[] files = dir.listFiles(); // 파일 목록 전체를 배열로 넘겨줌
for (File file : files) {
    if (file.isDirectory()) System.out.println("directory: " + file.getCanonicalPath());
    else if (file.isFile()) System.out.println("file: " + file.getName());
}
```

폴더(디렉토리) 또는 파일 확인

- String 메소드
  - String substring(int beginIndex)는 beginIndex부터 나머지 부분을 string으로 반환
  - String substring(int beginIndex, int endIndex)는 beginIndex부터 endIndex까지 부분을 strin으로 반환

```
String dir = "C:/JAVA";
String filename = "IMG1.jpg";
String fullPath = dir + "/" + filename;
String ext = filename.substring(filename.lastIndexOf('.') + 1);
String nameWithoutExt = filename.substring(0, filename.lastIndexOf('.'));
System.out.println("fullPath=" + fullPath); // fullPath=C:/JAVA/IMG1.jpg
System.out.println("ext=" + ext); // ext=jpg
System.out.println("nameWithoutExt=" + nameWithoutExt); // nameWithoutExt=IMG1
```

폴더(디렉토리) 또는 파일 확인

- String 메소드
  - boolean contains(CharSequence s)는 s를 가지고 있는지 여부를 true/false 반환
  - int lastIndexOf(String str)는 str을 가지고 있는 lastIndex 값을 반환

```
String dir = "C:/JAVA";
String filename = "IMG1.jpg";
String fullPath = dir + "/" + filename;
if (filename.contains("jpg")) {
   String format = "png";
   String newFilename = fullPath.substring(0, fullPath.lastIndexOf('.')+1) + format;
   System.out.println("newFilename=" + newFilename); // newFilename="C:/JAVA/IMG1.png"
}
```

Lab #2_4 FileI/O

- **Lab#2_3에서 inputDir, imageFileExt, forma를 "command.ini" 파일에서 읽어 들임**
- **FileInput 클래스 작성**

  boolean getCommandsFromFile(String filename, String[] commands)는 "filename" 텍스트 파일을 읽어서 "특정 폴더(imageDir)", "특정 영상 포맷(imageFileExt)", "원하는 다른 영상 포맷(format)"을 입력 받음
- Java2-lab2_4 폴더에 저장 후 제출

바이트 스트림 클래스

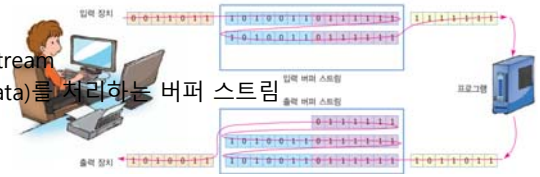
- 바이트 스트림
  - 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
- 바이트 스트림 클래스
  - java.io 패키지에 포함
  - InputStream/OutputStream
    - 추상 클래스
    - 바이트 스트림을 다루는 모든 클래스의 슈퍼 클래스
  - FileInputStream/FileOutputStream
    - 파일로부터 바이트 단위로 읽거나 저장하는 클래스
    - 바이너리 파일의 입출력 용도
  - DataInputStream/DataOutputStream
    - 자바의 기본 데이터 타입의 값(변수)을 바이너리 값 그대로 입출력
    - 문자열도 바이너리 형태로 입출력

문자 스트림 클래스

- 문자 스트림
  - 유니 코드로 된 문자를 입출력 하는 스트림
    - 문자 스트림은 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음 - 문자 스트림은 문자 데이터만 입출력 가능
- 문자 스트림을 다루는 클래스
  - Reader/Writer
    - java.io 패키지에 포함
    - 추상 클래스. 문자 스트림을 다루는 모든 클래스의 슈퍼 클래스
  - InputStreamReader/OutputStreamWriter
    - 바이트 스트림과 문자 스트림을 연결시켜주는 다리 역할
    - 지정된 문자집합 이용
    - InputStreamReader : 바이트를 읽어 문자로 인코딩
    - OutputStreamWriter : 문자를 바이트로 디코딩하여 출력
  - FileReader/FileWriter
    - 텍스트 파일에서 문자 데이터 입출력

버퍼 입출력 스트림 클래스

- 버퍼 스트림
  - 버퍼를 가진 스트림
  - 입출력 데이터를 일시적으로 저장하는 버퍼를 이용하여 입출력 효율 개선
- 버퍼 입출력의 목적
  - 입출력 시 운영체제의 API 호출 횟수를 줄여 입출력 성능 개선
    - 출력 시 여러 번 출력되는 데이터를 버퍼에 모아두고 한 번에 장치로 출력
    - 입력 시 입력 데이터를 버퍼에 모아두고 한번에 프로그램에게 전달
- 바이트 버퍼 스트림 클래스
  - BufferedInputStream와 BufferedOutputStream
  - 바이트 단위의 바이너리 데이터(Binary Data)를 처리하는 버퍼 스트림



- **문자 버퍼 스트림 클래스**
  - **BufferedReader와 BufferedWriter**
  - **유니코드의 문자 데이터(Text Data)만 처리하는 버퍼 스트림**

텍스트 파일 읽기

- BufferedReader 클래스를 사용한 텍스트 파일 read

```
import java.io.*;
public class BufferedReaderExample {
   public static void main(String args[])throws Exception{
        FileReader fr = new FileReader("C:/test.txt");
        BufferedReader br = new BufferedReader(fr);
        int i;
        while((i=br.read())!=-1){
        System.out.print((char)i);
        }
        br.close();
        fr.close();
   }
}
```

텍스트 파일 읽기

- BufferedReader 클래스를 사용하여 파일을 한 줄씩 읽어서 lines 배열에 저장

```
import java.io.*;
public class BufferedReaderExample2 {
   static String[] lines = new String[10]; // 10개의 라인 배열
   public static void main(String args[])throws Exception{
        BufferedReader br = new BufferedReader(new FileReader("C:/test.txt"));
        int i = 0;
        String line = "";
        while ((line=br.readLine())!= null){
           lines[i++] = line;
        }
        br.close();
        for (String l : lines) System.out.println(l); // lines 배열 출력
   }
}
```

Lab #2_5 Recursive Call

- **Lab#2_4에서 만약 사용자가 지정한 디렉토리 안에 서브 디렉토리가 존재한다면 그 안에 있는 이미지 파일이 존재한다면 이미지 파일 포맷 변환을 진행**
- **FileInput 클래스의 boolean getCommandsFromFile(String filename, String[] commands) 메소드는 TEXT 파일을 읽어서 commands로 넘겨줌**
- Java2-lab2_5 폴더에 저장 후 제출

재귀호출 (Recursive call)

- 함수에서 자기 자신을 다시 부르는 것을 재귀호출 (recursive call)이라고 함
- 함수 내부에서 사용되는 지역 변수의 값들은 자기 자신을 호출하기 전의 값들을 호출 후에도 그대로 보존함 (자기 자신을 다시 부르더라도 새로운 지역 변수들이 생성되는 것을 생각하면 됨)
- 재귀호출을 빠져나갈 수 있도록 검사하고 종료하는 부분이 반드시 존재해야 함 (재귀호출 탈출 조건)
- 재귀호출의 사용 예: 팩토리얼 (factorial), 최대공약수, 피보나치 수열, 등

재귀호출

- 팩토리얼을 구현하는 함수 작성
- n! = n x (n – 1) x (n – 2) x … x 1 = n x (n – 1)! (where n >= 1)
- 알고리즘
  - 만약 n이 1보다 작거나 같으면 값 1을 반환
  - 만약 n이 1보다 크다면 n * (n – 1)!을 반환

```
static int factorial(int n){
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
 }

factorial(1);  // 1
factorial(5);  // 5! = 5 * 4 * 3 * 2 * 1 = 120
```

재귀호출

- 최대공약수(Greatest Common Divisor)
- 알고리즘
  - b = 0이면 a 반환 // 나머지가 0이되면 연산을 중지
  - 아니면, gcd(b, a%b) 반환

```
static int gcd(int a, int b) {
    if (b == 0)
        return a;                // 재귀호출 탈출 조건
    else
        return gcd(b, a%b);
}
gcd(12, 4);                  // 4
gcd(12, 18);                 // 6
```

재귀호출

- 피보나치 수열 1 1 2 3 5 8 13 21 34 55 89
- 1항과 2항은 1
- 3항 이후부터의 n항은 (n – 1)항 + (n – 2)항
  - f(n) = f(n – 1) + f(n – 2)

```
static int fibonacci(int n) {
   if (n < 1)
      return n;              // 재귀호출 탈출 조건
   else
      return fibonacci(n – 1) + fibonacci(n – 2);
}
fibonacci(1)                 // 1
fibonacci(2)                 // 1
fibonacci(3)                 // 2
fibonacci(6)                 // 8
```

과제 제출

- 사용자가 지정한 폴더(C:/JAVA)에는 서브 폴더(C:/JAVA/PHOTO)가 존재함
- 폴더 안에는 jpg, png, gif 포맷의 파일들이 존재함 (그 외의 다른 종류의 파일은 없음)
- Lab02_1 ~ Lab02_5와 보고서를 전체적으로 묶어서 e-learning에 과제 제출