2017학년도 1학기

JAVA 프로그래밍 II

514770-1
2017년 봄학기
4/5/2017
박경신

Lab #3 (이미지 매니저 프로그램)

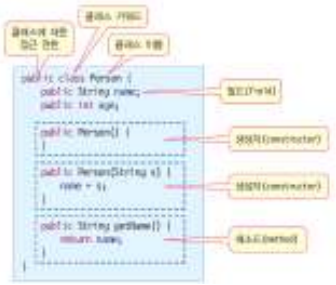
- 기존 요구사항 분석
  - Lab #2는 **"inputDir", "ext", "format"**에 따라 디렉토리 안에 원하는 이미지 파일 포맷을 변환시켜줌
    - **ImageConverter 클래스는 이미지 파일 포맷 변환(convert) 기능 제공**
    - **ImageConverterTest 클래스는 입력이미지파일명(inputImageFile), 출력이미지파일명(outputImageFile), 이미지포맷(format)을 입력받아서 변환하는 메인 프로그램**
  - **Lab #3는 사용자가 지정한 영상파일이나 디렉토리안에 모든 영상파일을 Photo 클래스 객체로 관리함**
  - **사용자가 지정한 폴더에는 원본 영상 포맷이 아닌 파일들이 있는 경우 에러처리**
  - **지정된 폴더 안에 서브 폴더가 있다면, 그 안에 있는 모든 영상도 동일하게 관리**
  - **사용자입력, "photolist.ini" 텍스트 파일로 입력받는 기능**
- Photo class, array, ArrayList, List<Photo>
- File, BufferedReader, static/non-static member

Lab #3_1 Class

- **Lab#3_1에서는 이미지 정보를 가진 Photo 클래스를 만든다. 그리고 Photo 클래스 객체의 배열 Photo[] photoList 을 생성하여 화면에 출력해본다 (for-loop 방식과 사용자 입력 방식)**
  - **Photo 클래스는 파일명, 확장자, 가로, 세로를 멤버로 한다.**
    - **private String fullPath; // 파일명**
    - **private String ext; // 확장자**
    - **private int width; // 가로**
    - **private int height; // 세로**
  - ***그리고 메소드는 다음을 포함한다.***
    - ***public Photo(String fullPath) // 형변환 생성자***
    - ***public Photo(Photo other) // 복사 생성자***
    - ***각 멤버필드에 대한 get/set 메소드 // 예: public String getFullPath() {return fullPath; } 등등***
    - ***Public String toString() // toString 메소드 오버라이드 (override)***
- **UserInput 클래스 String getImageFullPath()**는 사용자입력 파일명을 받음
- Java2-lab3_1 폴더에 저장 후 제출

클래스 구조

- 클래스 접근 지정자, public
  - 다른 클래스들에서 이 클래스를 사용하거나 접근할 수 있음을 선언
- 클래스(class)
  - Person이라는 이름의 클래스 선언
  - 클래스는 {로 시작하여 }로 닫으며 이곳에 모든 필드와 메소드 구현
- 필드(field)
  - 값을 저장할 멤버 변수 (혹은 필드)
  - 필드의 접근 지정자 public (다른 클래스의 메소드에서 호출할 수 있도록 공개한다는 의미)
- 메소드(method)
  - 메소드는 함수이며 객체의 행위를 구현
  - 메소드의 접근 지정자 public (다른 클래스의 메소드에서 호출할 수 있도록 공개한다는 의미)
- 생성자(constructor)
  - 클래스의 이름과 동일한 메소드
  - 클래스의 객체가 생성될 때만 호출되는 메소드



객체 생성

- 객체 생성
  - new 키워드를 이용하여 생성
    - new는 객체의 생성자 호출
- 객체 생성 과정
  1. 객체에 대한 레퍼런스 변수 선언
  2. 객체 생성

```
public static void main (String args[]) {
    Person aPerson; // 1. 레퍼런스 변수 aPerson 선언
    aPerson = new Person("김미남");  // 2. Person 객체 생성

    aPerson.age = 30;
    int i = aPerson.age;
    String s = aPerson.getName();
}
```

객체의 필드에 값 대입

객체의 필드에서 값 읽기

객체의 메소드 호출

static member와 non-static member

- non-static (instance) 멤버의 특성
  - 공간적 - 멤버들은 객체마다 독립적으로 별도 존재
  - 시간적 - 필드와 메소드는 객체 생성 후 비로소 사용 가능
  - 비공유의 특성 - 멤버들은 여러 객체에 의해 공유되지 않고 배타적
- static 멤버
  - 객체를 생성하지 않고 사용가능
  - 객체마다 생기는 것이 아님
  - 클래스당 하나만 생성됨
    - 일명 클래스 멤버라고도 부름
  - 특성
    - 공간적 특성 - static 멤버들은 클래스 당 하나만 생성.
    - 시간적 특성 - static 멤버들은 클래스가 로딩될 때 공간 할당.
    - 공유의 특성 - static 멤버들은 동일한 클래스의 모든 객체에 의해 공유

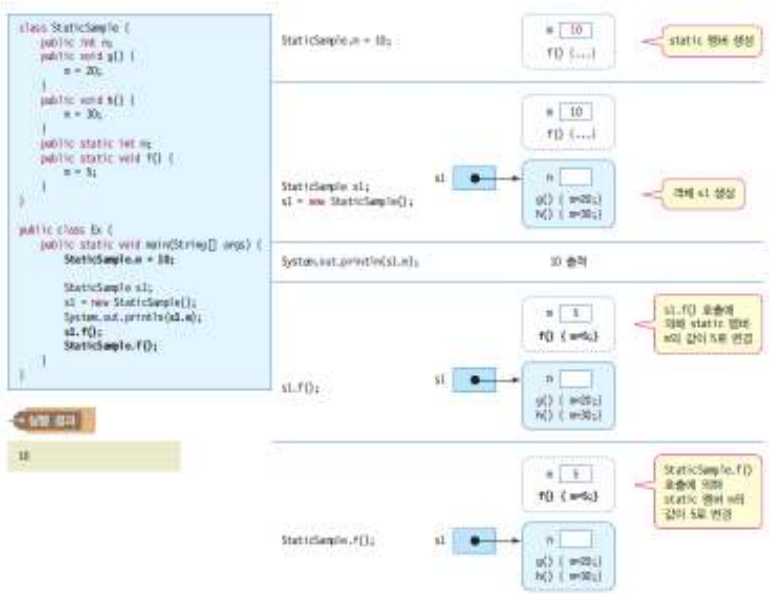
```
class StaticSample {
  int n; // instance 필드
  void g() {...} // instance 메소드

  static int m; // static 필드
  static void f() {...} // static 메소드
}
```

non-static 멤버와 static 멤버의 차이

	non-static 멤버	static 멤버

선언	class Sample {
	  int n;
	  void g() {...}
	}
		class Sample {
		  static int m;
		  static void g() {...}
		}

공간적 특성	멤버는 객체마다 별도 존재 • 인스턴스 멤버라고 부름	멤버는 클래스당 하나 생성 • 멤버는 객체 내부가 아닌 별도의 공간에 생성 • 클래스 멤버라고 부름
시간적 특성	객체 생성 시에 멤버 생성됨 • 객체가 생길 때 멤버도 생성 • 객체 생성 후 멤버 사용 가능 • 객체가 사라지면 멤버도 사라짐	클래스 로딩 시에 멤버 생성 • 객체가 생기기 전에 이미 생성 • 객체가 생기기 전에도 사용 가능 • 객체가 사라져도 멤버는 사라지지 않음 • 멤버는 프로그램이 종료될 때 사라짐
공유의 특성	공유되지 않음 • 멤버는 객체 내에 각각 공간 유지	동일한 클래스의 모든 객체들에 의해 공유됨



static의 활용

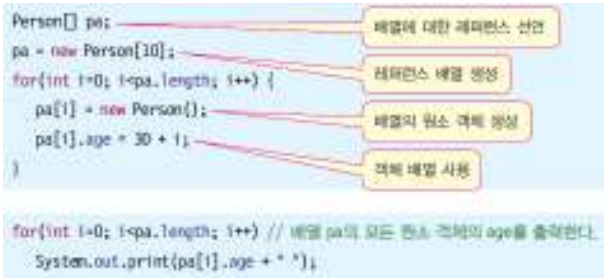
- 전역 변수와 전역 함수를 만들 때 활용
  - 자바의 캡슐화 원칙 지킴
    - 다른 클래스에서 공유하는 전역 변수나 전역 함수도 반드시 클래스 내부에 구현해야 함
- static 멤버를 가진 클래스 사례
  - JDK와 함께 배포되는 java.lang.Math 클래스
    - 모든 필드와 메소드가 public static으로 선언
    - 다른 모든 클래스에서 사용할 수 있음

```
public class Math {
   public static int abs(int a);
   public static double cos(double a);
   public static int max(int a, int b);
   public static double random();
   ...
}
```

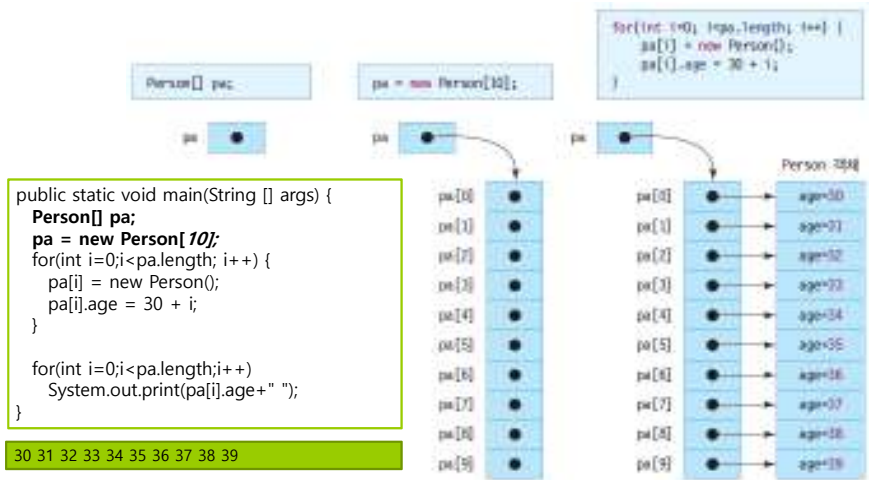
```
// 잘못된 사용법
//Math m = new Math();  // Math() 생성자는 private
//int n = m.abs(-5);

// 바른 사용법
int n = Math.abs(-5);
```

객체 배열

- 객체 배열 생성 및 사용



객체 배열 선언과 생성 사례



사용자 입력을 이용하여 Photo 객체 생성

```
public class UserInput {
   static Scanner scan = new Scanner(System.in);
   public static String getImageFullPath() {
      System.out.print("Please enter [image fullPath]: ");
      return scan.nextLine();
   }
}
```

```
Photo[] photoList = new Photo[5];
int index = 0;
while (index < 5) {
   try {
      photoList[index] = new Photo(UserInput.getImageFullPath());
      index++;
   } catch (IOException e) {
      System.out.println("Such image file is not found");
   }
}
```

Lab #3_2 FileI/O

- **Lab#2_3에서는 inputDir, imageFileExt, format를 "command.ini" 파일에서 읽어들임**
- **Lab#3_2에서는 Photo 영상파일리스트를 "photolist.ini" 파일에서 읽어들임**
- **FileInput 클래스에**

  Photo[] getPhotoListFromFile(String filename)는

  "filename" 텍스트 파일을 읽어서, Photo 배열을 생성하여 반환함
- Java2-lab3_2 폴더에 저장 후 제출

파일 입력을 이용하여 Photo 객체 생성

```
public class FileInput {
   public Photo[] getPhotoListFromFile(String filename) {
      Photo[] photoList = new Photo[10];
      String line = "";
      int index = 0;
      BufferedReader br = new BufferedReader(new FileReader(filename));
      while ((line = br.readLine()) != null) {
         Photo p = new Photo(line);
         if (p != null) photoList[index++] = p;
      }
      br.close();
      return photoList;
   }
}
```

```
Photo[] photoList = FileInput.getPhotoListFromFile ("photolist.ini");
```

Lab #3_3 ImageConverter/ImageResizer

- **Lab#3_2에서 void convertIfTo(Photo[] photoList, String ext, String format) 메소드 추가 작성 – photoList 배열에서 ext가 같은 모든 이미지를 format으로 변환함**
- **Lab#3_2에서 void resizeTo(Photo[] photoList, int width, int height) 메소드 추가 작성 – photoList 배열에서 모든 이미지를 지정한 크기 (width x height)로 resize함**
- Java2-lab3_3 폴더에 저장 후 제출

Image Resize

http://www.codejava.net/java-se/graphics/how-to-resize-images-in-java

```
static boolean resize(String inputImageFile, String outputImageFile, int width, int height)
throws IOException {
        // file I/O
        FileInputStream inputStream = new FileInputStream(inputImageFile);
        // reads input image from file
        BufferedImage inputImage = ImageIO.read(inputStream);
        // create output image
        BufferedImage outputImage = new BufferedImage(width, height, inputImage.getType());
        // scale the input image to output image
        Graphics2D g2d = outputImage.createGraphics();
        g2d.drawImage(inputImage, 0, 0, width, height, null);
        g2d.dispose();
        // extract extensions of output file
        String format = outputImageFile.substring(outputImageFile.lastIndexOf('.')+1);
        // write output image file
        boolean result = ImageIO.write(outputImageFile, format, new File(outputImageFile));
        return result;
}
```

Lab #3_4 Photo[]

- **Lab#3_4는 사용자가 지정한 특정 디렉토리 (inputDir) 안에 있는 모든 이미지 파일을 Photo 배열에 넣은 후 출력**
- **Lab#3_3에서 void addImagesInArray(String dirPath) 메소드 추가 작성**
- **Lab#3_3에서 void getNumImageFilesInDirectory(String dirPath) & void addImagesInArray2(String dirPath) 메소드 추가 작성**
- Java2-lab3_4 폴더에 저장 후 제출

Lab #3_5 List<Photo>, ArrayList

- **Lab#3_4에서 void addImagesInList(String dirPath, List<Photo> photoList) 메소드 추가 작성 – 디렉토리 안에 있는 모든 이미지를 photoList에 추가함**
- Java2-lab3_5 폴더에 저장 후 제출

Difference between Array and ArrayList

- Resizable
  - **Array is static in size** that is fixed length data structure, One can not change the length after creating the Array object.
  - **ArrayList is dynamic in size**. Each ArrayList object has instance variable capacity which indicates the size of the ArrayList. Its capacity grows automatically.
- Primitives
  - Array can contain both primitive data types (e.g. int, float, double) as well as objects.
  - **ArrayList can not contains primitive data types it can only contains objects.**
- Adding elements
  - In array we insert elements using the **assignment(=)** operator.
  - We can insert elements into the ArrayList using the **add()** method
- Length
  - Each array object has the **length** variable which returns the length of the array.
  - Length of the ArrayList is provided by the **size()** method.

Array vs ArrayList

```
int[] integerArray = new int[3];
integerArray[0] = 1;
integerArray[1] = 2;
integerArray[2] = 3;
for (int i : integerArray) System.out.println(i);
for (int j=0; j<integerArray.length; j++) System.out.println(integerArray[ j ]);
int k = 0;
while (k < integerArray.length) System.out.println(integerArray[k++]);
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(1); //cannot store primitive in ArrayList, instead autoboxing will convert int to Integer object
integerList.add(2); //cannot store primitive in ArrayList, instead autoboxing will convert int to Integer object
integerList.add(3); //cannot store primitive in ArrayList, instead autoboxing will convert int to Integer object
for (int m : integerList) System.out.println(m);
for (int n=0; n<integerList.size(); n++) System.out.println(integerList.get(n));
Iterator<Integer> itr = integerList.iterator();
while (itr.hasNext()) System.out.println(itr.next());
```

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 파일도 같이 있음.
- Lab03_1 ~ Lab03_5와 보고서를 전체적으로 묶어서 e-learning에 과제 제출