일자 : 2019. 01. 09 20:00 ~ 22:00 (120 분)
참석자 : 김경민, 박지환, 임현철

1 1장 기본 프로그래밍 구조

1.1 첫 번째 프로그램

```
Package ch01.sec01;

// 첫 번째 자바 프로그램

public class Helloworld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

- ✓ 자바는 대부분 객체를 조작하여 일을 시킨다.
- ✓ 자바는 모든 코드를 클래스 안에 정의
- ✓ 각 객체는 특정 클래스 안에 속함.
- ✓ 그 객체를 클래스의 인스턴스라고 한다.
- ✓ Main 메서드 : 프로그램을 실행할 때 첫 번째로 호출하는 메서드
- ✓ Void : 리턴 값을 반환하지 않기 때문에 void로 선언
- ✓ 접근제어자 종류 : public, private, protected, default
- ✓ 패키지 : 관련있는 클래스를 모아놓은 집합
- ✓ 주석 : 한줄주석 // , 여러줄 주석 /* */
- ✓ System.out : 자바 프로그램에서 '표준 출력(standard output)'을 나타내는 객체
- ✓ 자바 컴파일 및 실행 > JDK 필요
- ✓ 자바 명령 실행 : 코드작성 > 바이트 코드로 컴파일해서 클래스 파일로 저장 > java 명령으로 가상 머신을 구동하고 클래스 파일을 로드해서 바이트 코드를 실행
- ✓ IDE(Integrated Development Environment/통합 개발 환경) : 이클립스, IntelliJ 등이 있음.
- ✓ 객체의 인스턴스 메서드를 호출하려면 점 표기법을 사용
  object.methodName(arguments)
- ✓ 자바는 객체 대부분을 생성(construct) 해야 한다.
- ✓ JShell(Java Shell) : JAVA9에 추가된 REPL(Read Evaluate Print Loop)

1.2 기본 타입

- ✓ 부호 있는 정수 타입 :

타입	저장 공간	범위(포함)
byte	1바이트	-128 ~ 127
short	2바이트	-32,768 ~ 32,767
Int	4바이트	-2,147,483,648 ~ 2,147,483,647
long	8바이트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

- ✓ 부동소수점 타입 : 소수점이 있는 숫자, float, double
- ✓ Char 타입 : UTF-16 문자 인코딩의 '코드 유닛'
- ✓ Boolean 타입 :
  값 : true, false, 숫자타입X

1.3 변수

✓ 변수 선언

```
int total = 0;                          // 변수 선언 후 초기화
int total = 0, count;      // 타입이 같은 변수 여러개를 한 문장에 선언
Random generator = new Random(); // 첫번째 Random은 generator 변수의 타입, 두번째 Random은 이 클래스이 객체를 생성하는 new 표현식의 일부
```

✓ 변수 이름 : 반드시 문자로 시작, 대 소문자 구분, 표기법은 여러가지가 존재

✓ 메서드 안에 변수는 반드시 초기화 후 사용

✓ final, 상수 : 한번 할당하면 변경할 수 없는 변수

1.4 산술 연산

연산자	결합 방향
[] . () (메서드 호출)	왼쪽
! ~ ++ -- +(단항) – (단항) () (캐스트) new	오른쪽
* / %(나머지)	왼쪽
+ -	왼쪽
<< >> >>> (산술 시프트)	왼쪽
<> <= >= instanceof	왼쪽
== !=	왼쪽
& (비트곱)	왼쪽
^ (베타적 비트합)	왼쪽
| (비트합)	왼쪽
&& (논리곱)	왼쪽
|| (논리합)	왼쪽
? : (조건부)	왼쪽
= += -= *= /= %/ <<= >>= >>>= &= ^= |₩	오른쪽

1.5 문자열

✓ 문자의 시퀀스(연속)

✓ 문자열을 연결할 때는 + 연산자 사용

```
String location = “Java”;
String greeting = “Hello “ + location;
```

✓ 문자열을 다른 값과 연결하면 연결되는 값이 문자열로 변환

✓ join 메서드 : 여러 문자열을 구분자로 결합할 때

✓ substring 메서드 : 문자열을 분리할 때

✓ equals 메서드 : 두 문자열이 같은지 검사할 때

✓ Integer.toString : 정수를 문자열로 변환할 때

✓ Integer.parseInt : 정수를 담고 있는 문자열을 숫자로 변환할 때

1.6 입력과 출력

```
Scanner in = new Scanner(system.in); // 문자열과 숫자를 읽기 위해 System.in에 연결된 Scanner를 생성

System.out.println("what is your name?");
String name = in.nextLine(); // nextLine 메서드는 입력을 안줄 읽는다.

String firstName = in.next(); // 공백으로 구분된 단어 한개를 호출

System.out.println("How old are you?");
```

```
int age - in.nextInt(); // 정수를 읽으려면 nextInt 메서드를 사용
```

✓ Snanner 패키지는 java.util 패캐지에 있어 이 클래스를 사용하려면 가장 위쪽에
import java.util.Scanner;
문장을 추가한다.

✓ println : 출력후 한줄 넘김,
print : 출력 후 줄넘김 X
printf : 숫자의 개수를 제한

1.7 제어 흐름

✓ if (분기)

```
if (count > 0) {
        double average = sum / count;
        system.out.println(average);
} else if (count == 0) { // else if 분기
        System.out.println(0);
} else { // else 분기
        System.out.println("Huh?");
```

✓ switch (분기)

```
switch (count) { // 일치하는 케이슬 확인해서 실행
        case 0:
                output = "None";
                break; // 없을 경우에 계속 실행
        case 1:
                output = "One";
                break;
        case 2:
        case 3:
        case 4:
        case 5:
                output = Integer.toString(count);
                break;
        default: // 일치하는 값이 없을 경우 실행
                output = "Many";
                break;
```

✓ while

```
while (sum < target) {
        int next = generator.nextInt(10);
        sum += next;
        count++;
}
```

✓ do while

```
int next;
do {
        next = generator.nextInt(10);
        count++;
} while (next != target);
```

✓ for

```
for (int i = 1; i <= 20; i++) {
        int next = generator.nextInt(10);
        sum += next;
}
```

✓ 루프를 중간에 빠져 나오려면 break 문을 사용

- ✓ 지역변수(local variable) : 메서드의 매개변수를 포함해 메서드 안에 선언한 변수로 변수의 유효범위(scope)는 변수 선언 지점에서 시작해 해당 선언을 감싹 블록{}의 끝까지 이어진다.

1.8 배열과 배열 리스트

- ✓ String 배열을 생성 후 초기화

```
String[] names;
names = new String[100];
// 초기화는 두줄(상단) 혹은 한줄(하단) 둘다 가능
String[] names = new String[100];

Int[] primes = {2, 3, 5, 7, 11, 13}; // 원하는 값은 {}를 이용하여 채울 수 있다. New 연산자 미사용
```

- ✓ New 연산자로 배열을 선언하면 배열을 기본 값으로 채운다.
  숫자타입(char 포함)의 배열은 0
  boolean은 false
  객체의 배열은 null 참조
- ✓ 배열리스트 : 길이 변경이 가능한 배열

```
ArrayList<String> friends; // 배열리스트 변수 선언
Friends = new ArrayList<>(); // 배열리스트 생성
//또는 new ArrayList<String>()
friends.add("peter");
friends.add("Paul"); // 배열리스트는 add 메서드로 요소를 끝에 추가할 수 있다.
Friends.remove(1); // 배열 삭제
Friends.add(0, "Paul"); // 인덱스 0 앞에 추가한다.
String first = friends.get(0); // 배열 리스트 요소에 접근하려면 get을 사용
friends.set(1, "Mary"); // 다른 값으로 변경하려면 set 메서드를 사용
friends.size() // 리스트의 현재 크기를 확인하려면 size 메서드를 사용
```

- ✓ 배열은 기본 타입을 타입 매개변수로 사용할 수 없어 **래퍼클래스**를 사용
  래퍼클래스 종류 : Byte, Short, Integer, Long, Float, Double, Character, Boolean

1.9 기능적 분해

- ✓ main 메서드가 너무 길어질 때는 프로그램을 여러 클래스로 분해하는 것이 더 좋다.
- ✓ 간단한 프로그램이라면 코드를 동일한 클래스 안에 여러 메서드로 나누어도 된다.
  이런 메서드는 main 메서드와 마찬가지로 반드시 static 제어자로 선언해야 한다.

1.10 핵심 내용 정리

- ✓ 자바는 모든 메서드를 클래스 안에 선언한다. 비정적(nonstatic) 메서드가 속한 클래스의 객체로 호출 한다.
- ✓ 정적 메서드는 객체로 호출하지 않는다. 프로그램 실행은 정적 메서드인 main에서 시작한다.
- ✓ 자바의 기본 타입은 총 여덟 가지(정수 타입 네가지, 부동소수점 타입 두가지, char, boolean)다.
- ✓ 자바의 연산자와 제어 구조는 C나 자바스크립트와 아주 비슷하다.
- ✓ Math 클래스는 일반적인 수학 함수를 제공한다.
- ✓ String 객체는 문자(더 엄밀히 말하면 UTF-16으로 인코드된 유니코드의 코드 포인트)의 연속이다.
- ✓ System.out 객체로 터미널에 결과를 표시할 수 있다. System.in에 Scanner를 연결하면 터미널에 입력한 것을 읽을 수 있다.
- ✓ 배열과 컬렉션은 타입이 같은 요소를 모으는 데 사용한다.

2 2장 객체 지향 프로그래밍

2.1 객체 사용

- ✓ 캡슐화 : 다른 사람이 구현한 객체의 메서드를 호출할 때 내부동작을 알 필요가 없음
- ✓ 클래스 : 같은 동작을 하는 객체를 생성하고 사용하는 매커니즘

2.2 클래스 구현

```
public class Employee {
    private String name; // 인스턴스 변수, 인스턴스 변수는 보통 private로 선언
    private double salary; // 인스턴스 변수

    public void raiseSalary(double byPercent) // 메서드 선언 시 메서드 이름(raiseSalary), 매개변수의 타입(double), 이름(byPercent),반환 타입(void) 을 지정하여야 함.
        double raise = salary * byPercent / 100;
        salary += raise;
    }
    public String getName() {
        return name; // 메서드에서 값을 돌려줄 때는 return 키워들 사용
    }
```

```
// 인스턴스 메서드 호출
Fred.raiseSalary(5);

Double raise = fred.salary * byPercent / 100;
Fred.salary += raise;
```

```
// this 참조 : 객체의 메서드를 호출할 때 해당 객체가 this로 설정
Public voud raiseSalary(double byPercent) {
    Double raise = this.salary * byPercent / 100;
    This.salary += raise;
}
```

2.3 객체 생성

```
// 생성자 구현
Public Employee(String name, double salary) { // 이름이 클래스와 같고 생성자는 반환 타입이 없다.
    this.name = name;
    this.salary = salary;
```

```
// 오버로딩 : 생성자는 두가지 버전으로 제공할 수 있다.
Public Employee(double salary) {
    this.name = “”;
    this.salary = salary;
}
```

```
// 호출되는 생성자는 인수에 따라 결정된다.
Employee james = new Employee(“James Bond”, 50000);
// Employee(String name, double salary) 호출
Employee anonymous = new Employee(40000);
// Employee(double salary) 호출
```

- ✓ 인스턴스 변수를 최종(final)으로 선언할 수 있다.

2.4 정적 변수와 정적 메서드

- ✓ 정적 변수 : 클래스당 하나만 존재하는 변수 (static 변수)
- ✓ 정적 상수 : 변하지 않는 정적 변수 (static final 변수)
- ✓ 정적 메서드 : 객체에 작동하지 않는 메서드

2.5 패키지

- ✓ 연관된 클래스들을 한 패키지 않에 넣음.
- ✓ 작업을 조직화 하고 다른 사람이 제공한 코드 라이브러리와 분리가 편리함.
- ✓ 클래스 이름의 유일성을 보장

```
java.util.regex // 패키지 이름은 점(.)으로 구분된 식별자 목록

package com.horstmann.corejava; // 클래스를 패키지 안에 넣으려면 클래스 소스 파일의 첫번째 문장으로 package문을 추가

public class Employee {
        ...
}


import java.util.Random; // 클래스 임포트
import java.util.* // 패키지에 들어있는 모든 클래스를 임포트
```

2.6 중첩 클래스

- ✓ 클래스를 다른 클래스 내부에 두는 방법(정적 중첩 클래스, 내부 클래스)
- ✓ 가시성을 제한하거나 일반적인 이름을 쓰면서도 정돈된 상태를 유지할 때 유용

2.7 문서화 주석

- ✓ javadoc : 소스 파일로 HTML 문서를 만듬
- ✓ javadoc 유틸리티는 다음 정보를 추출

  공개 클래스와 인터페이스

  공개 보호 생성자와 메서드

  공개 보호 변수

  패키지와 모듈

```
// 클래스 주석 클래스 선언 바로 앞에 붙여야 한다.

/**
 * <code>Invoice</code> 객체는 각 주문 항목에 해당하는
 * 품목을 나열한 청구서를 표현한다.
 * @author 프레드 프린스톤
 * @author 바니 러블
 * @version 1.1
 */
Public class Invouce {
    ...
}
```

```
// 메서드 주석

/**
 * 직원의 급여를 인상한다.
 * @param byPercent 급여 인상 백분율(dPfmf 들어 10은 10%를 의미)
 * @return 인상액
 */
Public double raiseSalary(double byPercent) {
    Double raise = salary * byPercent / 100;
    Salary += raise;
    Return raise;
}
```

```
// 변수 주석

/**
 * 연간 일 수(윤년 제외)
 */
Public static final int DAYS_PER_YEAR = 365;
```

```
// 일반 주석

@since version 1.7.1
// @deprecated 태그에는 해당 클래스, 메서드, 변수를 사용하지 말아야 한다는 주석을 추가한다.
@deprecated Use <code>setVisible(true)</code> instead
```

2.8 핵심 내용 정리

- ✓ 변경자 메서드는 객체 상태를 변경하지만, 접근자 메서드는 객체 상태를 변경하지 않는다.
- ✓ 자바에서 변수는 객체가 아니라 객체 참조를 저장한다.
- ✓ 클래스 선언 안에 인스턴스 변수와 메서드 구현을 선언한다.
- ✓ 인스턴스 메서드는 객체로 호출하며, 호출된 메서드에서는 이 객체를 this 참조로 접근할 수 있다.
- ✓ 생성자는 클래스와 이름이 같다. 클래스 안에 오버로드된 생성자를 여러 개 포함할 수 있다.
- ✓ 정적 변수는 어떤객체에도 속하지 않는다. 정적 메서드는 객체로 호출하지 않는다.
- ✓ 클래슨느 패키지로 조직화된다. 임포트 선언을 하면 프로그램 안에서 패키지 이름을 쓰지 않아도 된다.
- ✓ 클래스 안에 다른 클래스를 선언할 수 있다.
- ✓ 내부 클래스는 비정적 중첩 클래스다. 내부 클래스의 인스턴스는 자신을 생성한 바깥쪽 클래스의 객체를 참조한다.
- ✓ Javadoc 유틸리티는 소스 파일을 처리해 선언된 내용과 프로그래머가 작성한 주석으로 HTML 파일을 만든다.