03-2. 메서드

Java 객체지향

메서드(Method)

- 메서드
  - 객체의 행동을 정의하는 procedure or function
  - 적용대상: 특정 객체(object) or 전체 클래스(class)
  - 구문:
    - `public return_type methodName(0+ parameters){..}`
  - 호출방식: "dot(.)" 연산자
    - `objectName.methodName(arguments)`

메서드(Method)

- 매개변수(*parameters*)
  - 메서드를 선언할 때 괄호 안에 표현된 Input 값을 나타내는 변수: **`(type1 name1, type2 name2, …)`**
  - 메서드 호출에서 들어가는 구체적인 값은 인자(Argument)라고 함
- 반환값 자료형(*return_type*)
  - 메서드는 0개 혹은 1개의 값을 Output으로 반환할 수 있음
  - 반환값이 없을 때: **`void`**
  - 반환값이 있을 때: **`int, boolean, Car, …`**
    - 반환되는 값은 메서드 선언에서 정의된 반환값의 유형과 일치해야 함

메서드(Method)

메서드 선언 (with parameters)	메서드 호출 (with arguments)
public void method1(){..}	obj.method1()
public int method2(boolean b){..}	obj.method2(true)
public int method3(int x,int y,Car t){..}	obj.method3(3,4,car)

▪ *선언 예*

```
class Car {
   …
   public void setSpeed(int s) {
          speed = s;
   }
   public int getSpeed() {
          return speed;
   }
}
```

▪ *호출 예*

```
public class CarTest {
   public static void main()
   {
          Car mcqueen = new Car();
          …
          mcqueen.setSpeed(300);
          System.out.println(mcqueen.name + “ “ +
                 mcqueen.getSpeed());
}
```

메서드(Method)

1. return 값과 매개변수(parameter)가 없는 메서드

```
class Car {
    String name;
    int speed;
     ….
     public void printCarInfo( ) {
          System.out.print("The Name is : " + name);
          System.out.println( "The Speed is" + speed);
     }
     .....
}
```

```
public class CarTest{
   public static void main()
    {
          Car mcqueen = new Car();
          …
          mcqueen.printCarInfo();
          …
```

메서드(Method)

2. return 값이 없고 매개변수가 있는 method

```
class Car {
   …
   public void setSpeed(int s) {
          speed = s;
   }
    …
```

```
public class CarTest {
   public static void main()
   {
          Car mcqueen = new Car();
          …
          mcqueen.setSpeed(300);
          …
```

메서드(Method)

3. return 값과 매개변수가 있는 method
   - return 문은 메서드의 실행을 중지하고 메서드를 호출한 곳으로 되돌아가게 하는 명령입니다.
   - 메서드를 호출한 곳으로 되돌아가면서 특정한 값을 전달할 수 있습니다. 이것을 '반환값'이라고 합니다.

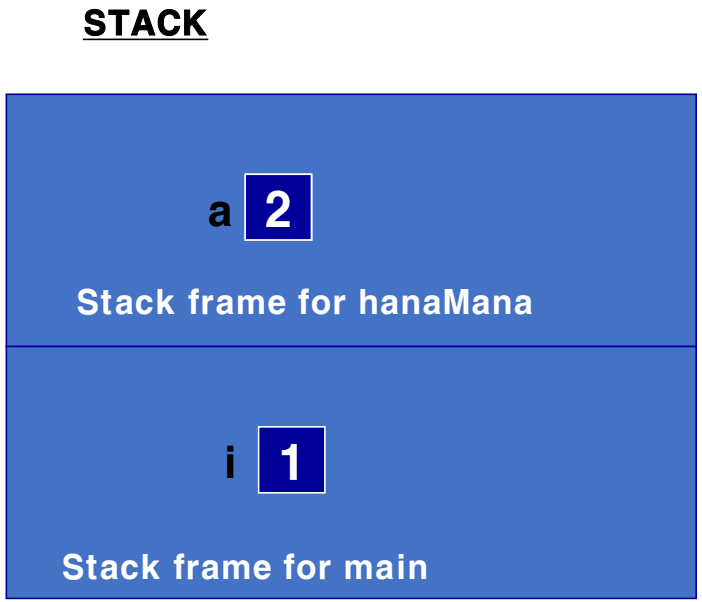
```
class intOp {
  …
 public int sum( int a, int b)
{
     return a+b;
}
  …
```

```
public class intOpTest {
   public static void main()
    {
          intOp obj1 = new intOp();
          …
          int sum = obj1.sum(3, 9);
          …
```

메서드(Method)

- 매개변수가 기본 자료형일 때
    - 호출되는 메쏘드의 값 자체가 복사되어짐
    - 해당 스택 프레임이 스택 메모리에 존재할 때만 살아 있음

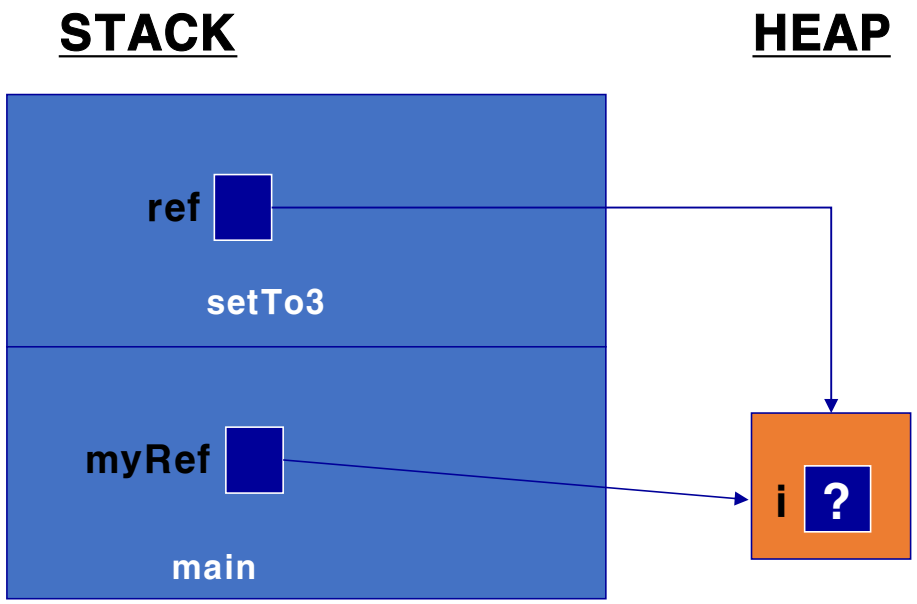
```
…

  static void hanaMana(int a) {
        a = 2;
   }
 public static void main(String[] args) {
  int i = 1;
  hanaMana(i);
…
```



메서드(Method)

- 매개변수가 객체를 참조하고 있는 변수일 때
  - 객체 자체가 아니라 객체의 참조 값을 넘겨줌
  - 참조하고 있는 객체의 주소 값 가체가 복사되고 객체는 공유됨
  - 객체 자체는 참조 변수가 하나라도 있으면 살아 있음

```
class MyClass {
     public int i;
     public MyClass() {
          i = 1;
     }
     static void setTo3(MyClass ref) {
          ref.i = 3;
     }
 public static void main(String[] args) {
  MyClass myRef = new MyClass();
  setTo3(myRef);
…
```



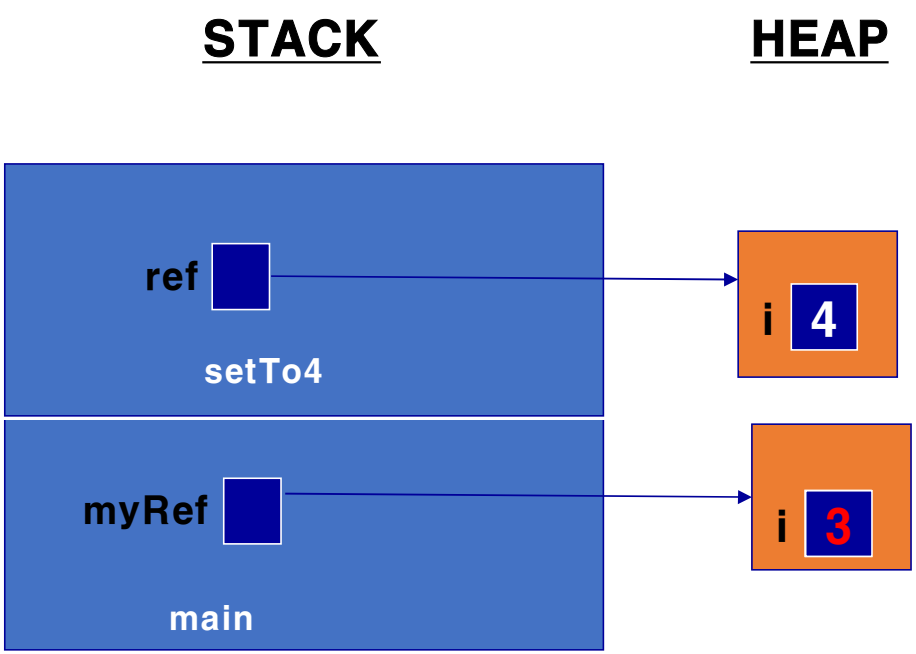
메서드(Method)

- 다음 메쏘드를 실행하여 MyClass 객체의 멤버 변수 i의 값을 3으로 만들고, 다시 4로 만들려고 합니다. 잘못된 부분을 찾으세요.

```
class MyClass {
    public int i;
    public MyClass() {
        i = 1;
    }
    static void setTo3(MyClass ref) {
        ref.i = 3;
    }
    static void setTo4(MyClass ref) {
        ref = new MyClass();
        ref.i = 4;
    }

public static void main(String[] args) {
 MyClass myRef = new MyClass();
 setTo3(myRef);
 setTo4(myRef);
…
```

메서드(Method)

```
class MyClass {
    public int i;
    public MyClass() {
        i = 1;
    }
    static void setTo3(MyClass ref) {
        ref.i = 3;
    }
    static void setTo4(MyClass ref) {
        ref = new MyClass();
        ref.i = 4;
    }

public static void main(String[] args) {
 MyClass myRef = new MyClass();
 setTo3(myRef);
 setTo4(myRef);
…
```



메서드(Method)

- 가변 길이 인자는 같은 타입의 데이터를 전달받는 메서드라면 꼭 오버로딩을 사용하지 않고도 인자의 개수를 다르게 호출할 수 있도록 선언하는 방법입니다.
- 가변 길이 인자 선언

Test01.java

```
01: package com.ruby.java.ch05;
02:
03: public class Test01 {
04:   static void test(int... v) {
05:     System.out.print(v.length + " : ");
06:     for(int x : v)
07:       System.out.print(x + " ");
08:     System.out.println();
09:   }
10:
11:   public static void main(String[] args) {
12:     test(1);
13:     test(1, 2);
14:     test(1, 2, 3);
15:   }
16: }
```

메서드(Method)

- 가변 길이 인자뿐만 아니라 다른 일반 인자도 함께 전달받는 메서드를 선언할 수 있습니다.

```
void test(String name, int... v) { }
```

- test( ) 메서드를 호출할 때 첫 번째 인자값은 name 매개변수에 저장되고, 두 번째 이후의 인자값(들)은 v 매개변수가 참조하는 배열 안에 저장됩니다

```
05:   static void test(String name, int... v) {
06:     System.out.print(name + " : ");
07:     for(int x : v)
08:       System.out.print(x + " ");
09:     System.out.println();
10:   }
11:
12:   public static void main(String[] args) {
13:     test("오정임", 98, 85, 88);
14:     test("김푸름", 90, 95, 92);
15:     test("김하늘", 80, 98, 95);
16:   }
17: }
```

메서드(Method) – Overloading

- 메서드 오버로딩 (Method Overloading)
  - 메쏘드 구문: ***public return_type methodName(0+ parameters){..}***
  - 객체에는 동일한 이름(methodName)을 가지는 여러 메서드가 존재할 수 있음
  - 동일한 이름을 가지는 메서드(method)의 호출은 매개변수(parameter) 리스트에 의해 결정됨
    - methodName (type1 name1, type2 name2, ...)
    - 메서드 호출시 인자(argument)*들의 수, 순서, 타입에 의해 다른 method가 호출됨
    - 동일한 이름의 메서드는 반환타입(return type)이 동일 해야 함

      Q) 아래에 public double sum(double x, double y) {..} 가 추가될 경우 나타나는 현상은?

메쏘드 선언 (Method Declaration)	메쏘드 호출(Method Call)
`public int sum(){..}`	`obj1.sum()`
`public int sum(int a, int b, int c){..}`	`obj1.sum(1, 2, 3)`
`public int sum(int a, double b){..}`	`obj1.sum(1, 2.5)`
`public int sum(double x, int y){..}`	`obj1.sum(3.14, 2)`

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare