

객체지향프로그래밍 3 주차 정리

프로그래밍

2.3

The image displays three screenshots of the Eclipse IDE, showing Java code and its execution results.

Top Left Screenshot: IntTest.java

```
1 public class IntTest
2 {
3     public static void main(String [] args)
4     {
5         int int1 = 7, int2 = 44;
6         int int3, int4, int5;
7
8         int3 = int2 * int1;
9         int4 = int2 / int1;
10        int5 = 44 / int1;
11
12        System.out.println("44 * 7 = " + int3);
13        System.out.println("44 / 7 = " + int4);
14        System.out.println("44 / 7 = " + int5);
15    }
16 }
17
18
```

Console Output:

```
<terminated> IntTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 6:32:1
44 * 7 = 308
44 / 7 = 6
44 / 7 = 6
```

Top Right Screenshot: FloatTest.java

```
1 public class FloatTest
2 {
3     public static void main(String[] args)
4     {
5         int int1 = 7, int2 = 44;
6         int int3, int4, int5;
7         float float1, float2, float3;
8
9         int3 = int2 * int1;
10        int4 = int2 / int1;
11        float1 = int2 / int1;
12        float2 = (float)int2 / int1;
13        float3 = 44 / (float)int1;
14
15        System.out.println("44 * 7 = " + int3);
16        System.out.println("44 / 7 = " + int4);
17        System.out.println("44 / 7 = " + float1);
18        System.out.println("44 / 7 = " + float2);
19        System.out.println("44 / 7 = " + float3);
20    }
21 }
22
23
```

Console Output:

```
<terminated> FloatTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 6:53
44 * 7 = 308
44 / 7 = 6
44 / 7 = 6.0
44 / 7 = 6.285714
44 / 7 = 6.285714
```

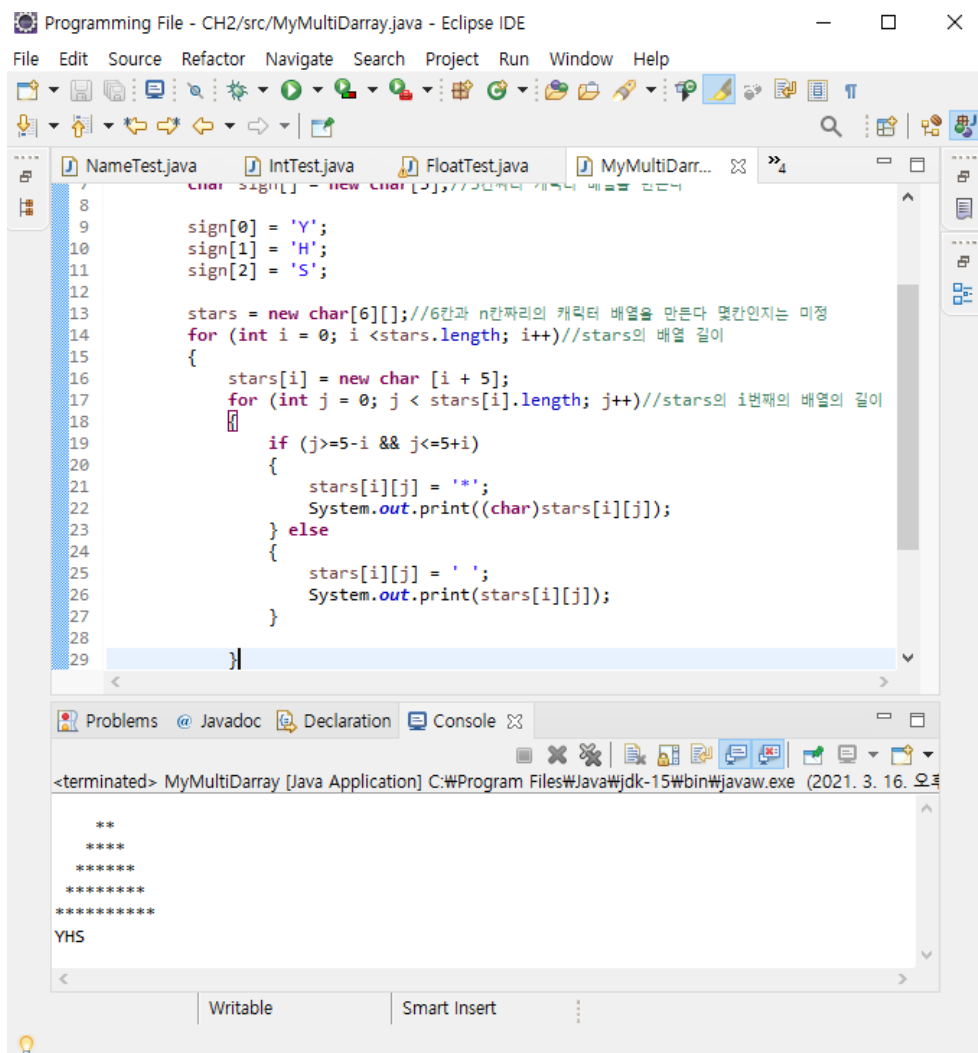
Bottom Screenshot: MultiDarray.java

```
1 public static void main(String args[])
2 {
3     char stars[] [];
4
5     char sign[] = new char[3]; //3칸짜리 캐릭터 배열을 만든다
6
7     sign[0] = 'Y';
8     sign[1] = 'H';
9     sign[2] = 'S';
10
11     stars = new char[6][]; //6칸과 n칸짜리의 캐릭터 배열을 만든다. 몇칸인지는 미정
12     for (int i = 0; i < stars.length; i++) //stars의 배열 길이
13     {
14         stars[i] = new char [i + 1];
15         for (int j = 0; j < stars[i].length; j++) //stars의 i번째의 배열의 길이
16         {
17             stars[i][j] = '*';
18             System.out.print(stars[i][j]);
19         }
20         System.out.println();
21     }
22     System.out.print((char)sign[0]);
23     System.out.print((char)sign[1]);
24 }
25
26
```

Console Output:

```
<terminated> MultiDarray [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 8
***
****
*****
YHS
```

HW1



```
8
9
10 sign[0] = 'Y';
11 sign[1] = 'H';
12 sign[2] = 'S';
13
14 stars = new char[6][6]; // 6칸과 n칸짜리의 캐릭터 배열을 만든다 몇칸인지는 미정
15 for (int i = 0; i < stars.length; i++) // stars의 배열 길이
16 {
17     stars[i] = new char [i + 5];
18     for (int j = 0; j < stars[i].length; j++) // stars의 i번째의 배열의 길이
19     {
20         if (j >= 5 - i && j <= 5 + i)
21         {
22             stars[i][j] = '*';
23             System.out.print((char)stars[i][j]);
24         } else
25         {
26             stars[i][j] = ' ';
27             System.out.print(stars[i][j]);
28         }
29     }
30 }
```

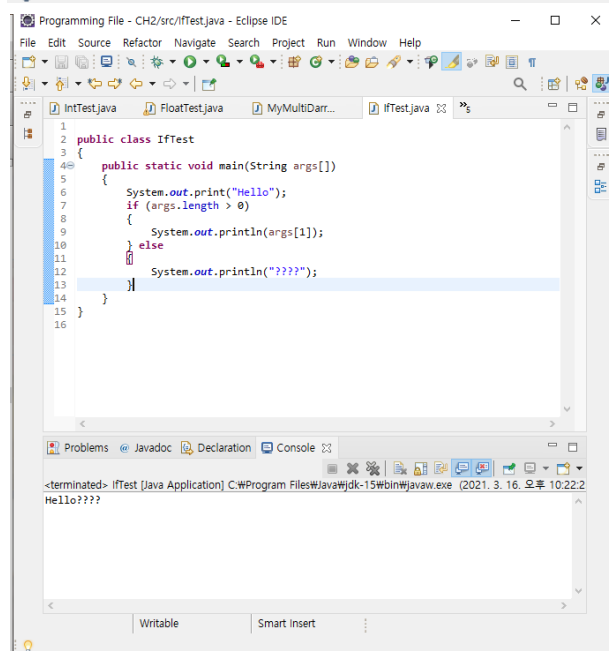
Problems Javadoc Declaration Console

<terminated> MyMultiDarry [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 10:22:22)

```

**
****
*****
*****
*****
*****
YHS
```

Writable Smart Insert



```
1
2 public class IfTest
3 {
4     public static void main(String args[])
5     {
6         System.out.print("Hello");
7         if (args.length > 0)
8         {
9             System.out.println(args[1]);
10        } else
11        {
12            System.out.println("????");
13        }
14    }
15 }
16 }
```

Problems Javadoc Declaration Console

<terminated> IfTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 10:22:22)

```
Hello???
```

Writable Smart Insert

```
C:\WINDOWS\system32>cd C:\Users\양현서\Desktop\백\체\Programming File\CH2\src
C:\Users\양현서\Desktop\백\체\Programming File\CH2\src>javac IfTest.java
C:\Users\양현서\Desktop\백\체\Programming File\CH2\src>java IfTest Yang Hyeon Seo
HelloHyeon
C:\Users\양현서\Desktop\백\체\Programming File\CH2\src>
```

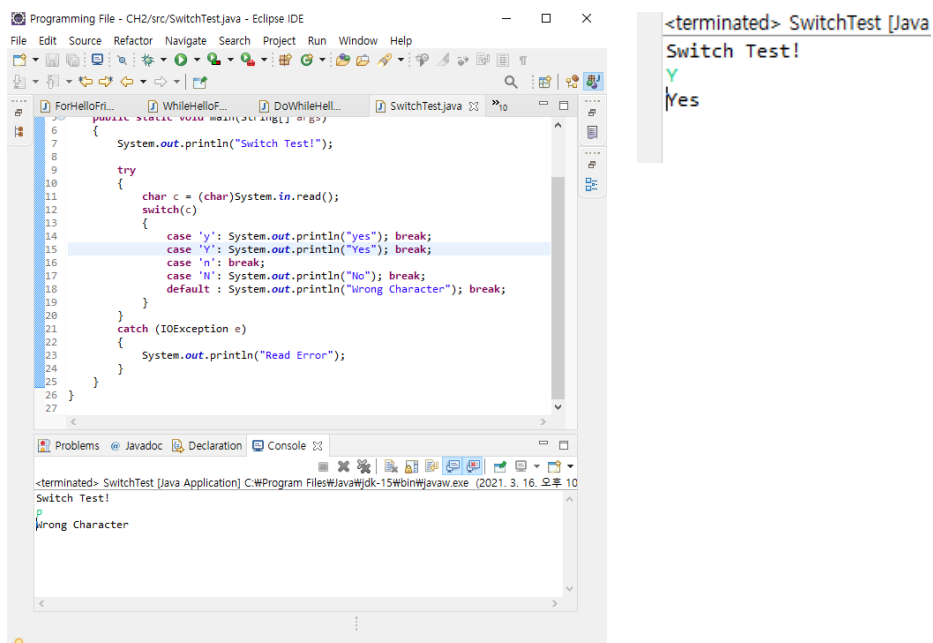
Lab 2-5-2

```
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>javac ForHelloNames.java
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>java ForHelloNames Yang Hyeon Seo
HelloYang Hyeon Seo
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>javac ForHelloFriends.java
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>java ForHelloFriends Yang Hyeon Seo
HelloYang Hyeon Seo
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>
```

Lab 2-5-3

```
helloYang Hyeon Seo
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>javac WhileHelloFriends.java
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>java WhileHelloFriends Yang Hyeon Seo
HelloYang Hyeon Seo
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>javac DoWhileHelloFriends.java
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>java DoWhileHelloFriends Yang Hyeon Seo
Hello Yang Hyeon Seo
C:\Users\양현 서\Desktop\각 체\Programming File\CH2\src>
```

2-5-4



The screenshot shows the Eclipse IDE with a Java file named `SwitchTest.java` open. The code is as follows:

```
6  {
7      System.out.println("Switch Test!");
8
9      try
10     {
11         char c = (char)System.in.read();
12         switch(c)
13         {
14             case 'y': System.out.println("yes"); break;
15             case 'Y': System.out.println("Yes"); break;
16             case 'n': break;
17             case 'N': System.out.println("No"); break;
18             default : System.out.println("Wrong Character"); break;
19         }
20     }
21     catch (IOException e)
22     {
23         System.out.println("Read Error");
24     }
25 }
26 }
27 }
```

The console output shows the program running and printing "Switch Test!". The user has entered 'y', and the program has printed "yes".

```
<terminated> SwitchTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 10:10)
Switch Test!
y
yes
```

2-6-1

Programming File - CH2/src/NoTry.java - Eclipse IDE

```

1 class NoTry
2 {
3     public static void main(String[] args)
4     {
5         String msg[] = {"Yang", "Hyeon", "Seo"};
6         int n = msg.length;
7
8         for (int i = 0; i < n; i++)
9         {
10             System.out.println(msg[i]);
11         }
12         System.out.println("Every thing is done");
13     }
14 }
15
16

```

Problems @ Javadoc Declaration Console

<terminated> NoTry [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 10:50:4)

Yang
Hyeon
Seo
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for array of length 3
at NoTry.main(NoTry.java:11)

2-6-2

Programming File - CH2/src/TryTest.java - Eclipse IDE

```

1 public class TryTest
2 {
3     public static void main(String[] args)
4     {
5         String msg[] = {"Yang", "Hyeon", "Seo"};
6         int n = msg.length;
7
8         for(int i = 0; i <= n ; i++)
9         {
10             try
11             {
12                 System.out.println(msg[i]);
13             } catch (ArrayIndexOutOfBoundsException ex)
14             {
15                 System.out.println("Oops, sorry. There is an exception.");
16             }
17         }
18         System.out.println("Every thing is done.");
19     }
20 }
21
22

```

Problems @ Javadoc Declaration Console

<terminated> TryTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 16. 오후 10:54:4)

Yang
Hyeon
Seo
Oops, sorry. There is an exception.
Every thing is done.

Programming File - CH2/src/CallbyTest.java - Eclipse IDE

```

1 public class CallbyTest
2 {
3     public static void main(String args[])
4     {
5         int a = 10;
6         Car mycar = new Car("Father", 500);
7
8         System.out.println("Call-by-value");
9         System.out.println("before calling: a = " + a);
10        Callby.value(a);
11        System.out.println("after calling: a = " + a);
12
13        System.out.println();
14        System.out.println("call-by-reference");
15        System.out.println("before calling: mycar.price = " + mycar.price + "\t mycar.owner = " + mycar.owner);
16        Callby.ref(mycar);
17        System.out.println("after calling: mycar.price = " + mycar.price + "\t mycar.owner = " + mycar.owner);
18    }
19 }
20
21

```

Problems @ Javadoc Declaration Console

<terminated> CallbyTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (2021. 3. 18. 오전 10:58:13 - 오전 10:58:14)

Call-by-value
before calling: a = 10
value() method: a = 60
after calling: a = 10

call-by-reference
before calling: mycar.price = 500 mycar.owner = Father
refer()method: car.price = 400 car.owner = Me
after calling: mycar.price = 400 mycar.owner = Me

Programming File - CH2/src/Callby.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

FirstTest.java StringBuffe... TryTest.java Car.java Callby.java CallbyTest.java »13

```
1
2 public class Callby
3 {
4     public static void value(int a)
5     {
6         a +=50;
7         System.out.println("value() method: a = "+a);
8     }
9     public static void ref(Car car)
10    {
11        car.price -= 100;
12        car.owner = "Me";
13        System.out.println("refer()method: car.price = " + car.price + "\t car.owner = " + car.owner);
14    }
15 }
16
```

Programming File - CH2/src/Car.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

FirstTest.java StringBuffe... TryTest.java Car.java Callby.java Ca

```
1
2 public class Car
3 {
4     String owner;
5     int price;
6
7     public Car(String o, int p)
8     {
9         owner = o;
10        price = p;
11    }
12 }
13
```

Data Type

Basic Data Type

Boolean

Numeric

Integer

byte

char

short

int

long

Floating

float

double

Reference Type

new 키워드 이용 시 Heap area 에 동적 메모리 할당

new

객체 생성 키워드

객체를 만들려면 항상 필요하다

new 키워드 사용 시 메모리가 할당된다

Heap 구조

메모리 정렬 알고리즘

클래스 객체나 배열의 공간을 만들 때 사용된다.

컴퓨터의 기억 장소에서 일부분이 프로그램들에 할당되었다가 회수되는 되풀이 영역

프로그램이 요구하는 블록의 크기나 요구 횟수, 순서가 일정한 규칙이 없다

기억장소를 동적으로 할당하고 돌려준다.

(출처 : IT 용어사전)

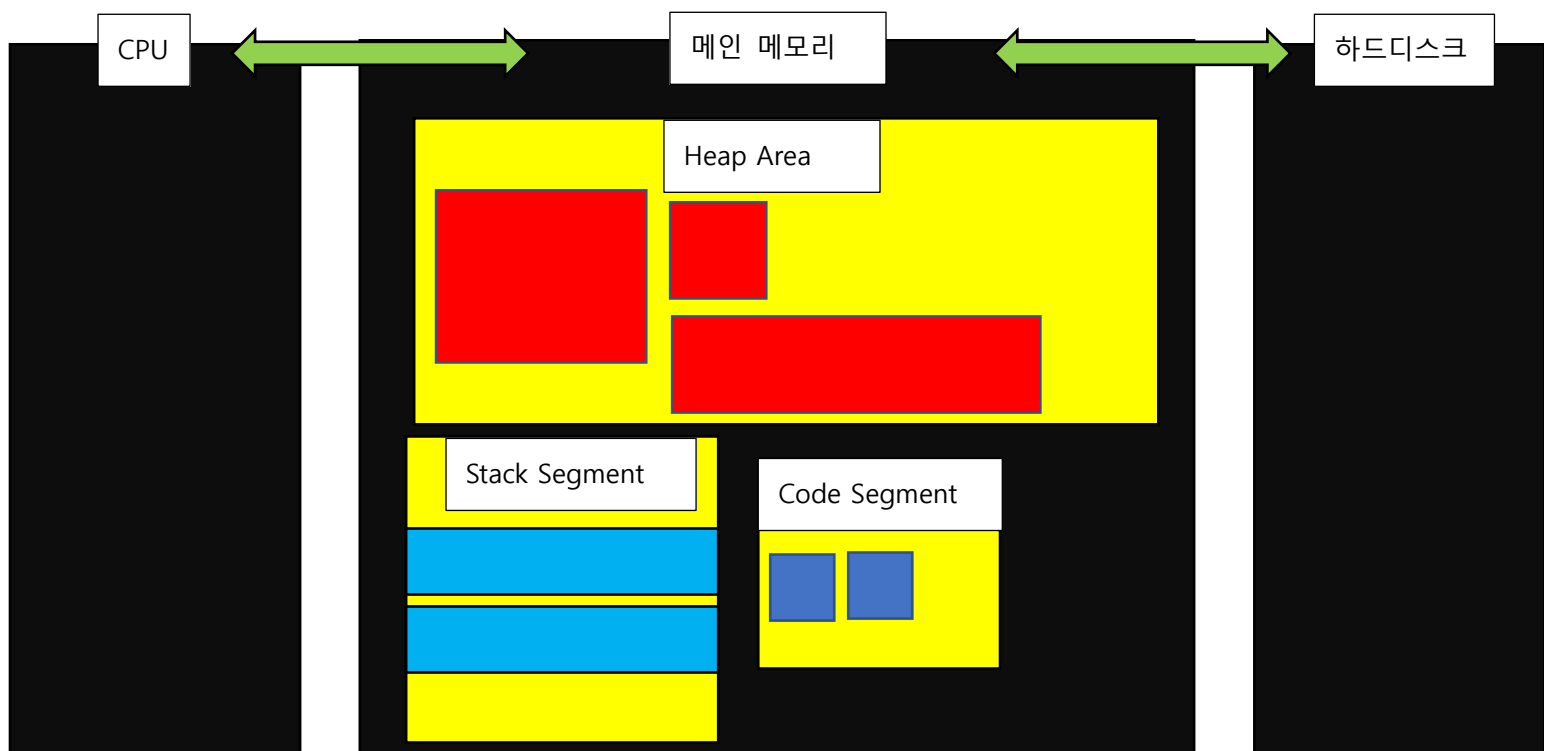
여러 개의 값들 중에서 최댓값이나 최솟값을 빠르게 찾아내도록 만들어진 자료구조(출처 : 블로그 개나의 취업일기)

Class

Interface

Array

컴퓨터 동작 구조



메인 메모리

Heap Area

메모리의 영역 중 동적으로 관리할 수 있는 영역

Heap 의 방법으로 메모리를 관리한다.

new 키워드를 사용하면 Array 와 Object instance 등에게 여기 메모리가 할당된다.

Segment

주 기억장치를 효율적으로 관리하기 위해 이를 일정한 크기의 논리적 단위로 나누어 관리한다. 이 논리적 단위를 세그먼트라고 한다.

서로 관련이 있는 데이터와 명령어를 하나의 세그먼트로 관리하는 것은 아니고, 데이터를 저장하는 데이터 세그먼트 영역과 명령어를 저장하는 코드 세그먼트 영역으로 구분해서 사용한다.

데이터 세그먼트 영역은 기억 장소의 할당 방법에 따라 동적할당에 의해 관리되는 스택 세그먼트와 힙 세그먼트, 그리고 정적 할당에 의해 관리되는 데이터 세그먼트로 구분된다.

(출처 : 티스토리 행복을 기록합니다)

Stack Segment

지역 변수가 들어가는 메모리의 한 장소

동시에 여러 함수에 호출되어도 서로의 변수 영역을 침범하지 않는다

함수마다 변수 할당을 하지 않아 메모리를 절약한다

재귀호출 가능(자신을 풀기 위해 자신을 호출하는 것)

(출처 : <https://jvinci.tistory.com/244>)

스택(stack)

마지막에 들어온 데이터가 먼저 나가는 자료 구조이다(LIFO)

자료를 넣는 것을 밀어 넣는다고 하여 push, 자료를 꺼내는 것을 pop 이라고 한다.

스택은 힙 영역 메모리에서 일반적인 데이터를 저장하는 스택과 스택 영역 메모리에서 프로그램의 각 분기점에 변수와 같은 정보를 저장하기 위한 스택이라는 두가지 의미로 사용될 수 있어 유의해야 한다

(출처 : 나무위키)

Code Segment

코드만 들어갈 수 있는 메모리 영역

CPU

중앙 처리 장치

프로그램을 수행할 수 있는 프로세서

하드디스크

컴퓨터의 주요 부품 중 하나로 보조기억장치

비 휘발성 데이터 저장소

(출처 : http://ingen.co.kr/xe/board_blog/2784)

아키텍처(architecture)

시스템 목적을 달성하기 위해 시스템의 상호작용 등의 시스템 디자인에 대한 제약 및 설계

어떤 구체적인 컴퓨터 시스템의 조직과 장기적인 개선 계획

시스템 구성 요소들의 정보 전달

실제로 설계 및 구현을 수행하는 사람에게 효율적인 기술을 전달하는 지식 전달 수단

기본 요구 사항

시스템 구성 및 동작원리를 나타낸다

시스템 구성 요소(부품)에 대해 설계 및 구현을 지원하는 수준으로 자세히 기술된다

구성 요소 간의 관계 및 시스템 외부 환경과의 관계가 묘사된다

요구 사양 및 시스템의 전체 수명주기를 고려한다

시스템 전체(하드웨어와 소프트웨어를 포괄한 것)에 대한 논리적인 기능 체계와 그것을 실현하기 위한 구성 방식으로 시스템의 전체적인 최적화를 목표하고 있다

최적화를 목표로 두고 시스템 구성과 동작원리 그리고 시스템의 구성환경 등을 설명 및 설계하는 설계도

(출처 : <https://tuhbm.github.io/2019/04/24/architecture/> , 위키백과 시스템 아키텍처)

객체지향프로그래밍의 실행상의 핵심

>>>클래스에 대한 instance 생성

Instance

Class 에 대한 instance object > instance 나 object 라고 부른다

어떤 집합에 대해서 그 집합의 개별적인 요소

OOP 에서 어떤 등급에 속하는 각 객체, 실제로 존재하는 것을 의미

객체를 생성(기억장치를 할당)함으로 그 등급의 인스턴스를 생성할 수 있다

논리식의 변수에 구체적인 값을 대입함으로 원래 식의 인스턴스를 만들기도 한다

용도 : 객체의 속성과 기능에 따라 다르다

다수의 속성(멤버 변수, 특성, 필드, 상태 등)과 기능(메소드, 함수, 행위 등)의 집합

객체지향프로그래밍(OOP)

실제 세계의 현상을 컴퓨터 상에 객체로 실현한 것(모델화)

이를 통해 컴퓨터를 자연스러운 형태로 사용하기 위함

공통의 성질을 가지는 객체를 같은 객체 등급으로 정의

>계층화

하위 등급은 상위 등급의 성질과 기능을 계승한다

이를 통해 복잡하고 거대한 소프트웨어를 사용하고 작성, 유지보수하기 쉬워졌다

객체 등급

OOP 에서 구조, 형태 등 공통의 속성을 가지는 객체의 집합

클래스

객체의 속성과 행위를 정의한 것

객체를 생성하기 위해 만든다

Class 클래스이름()

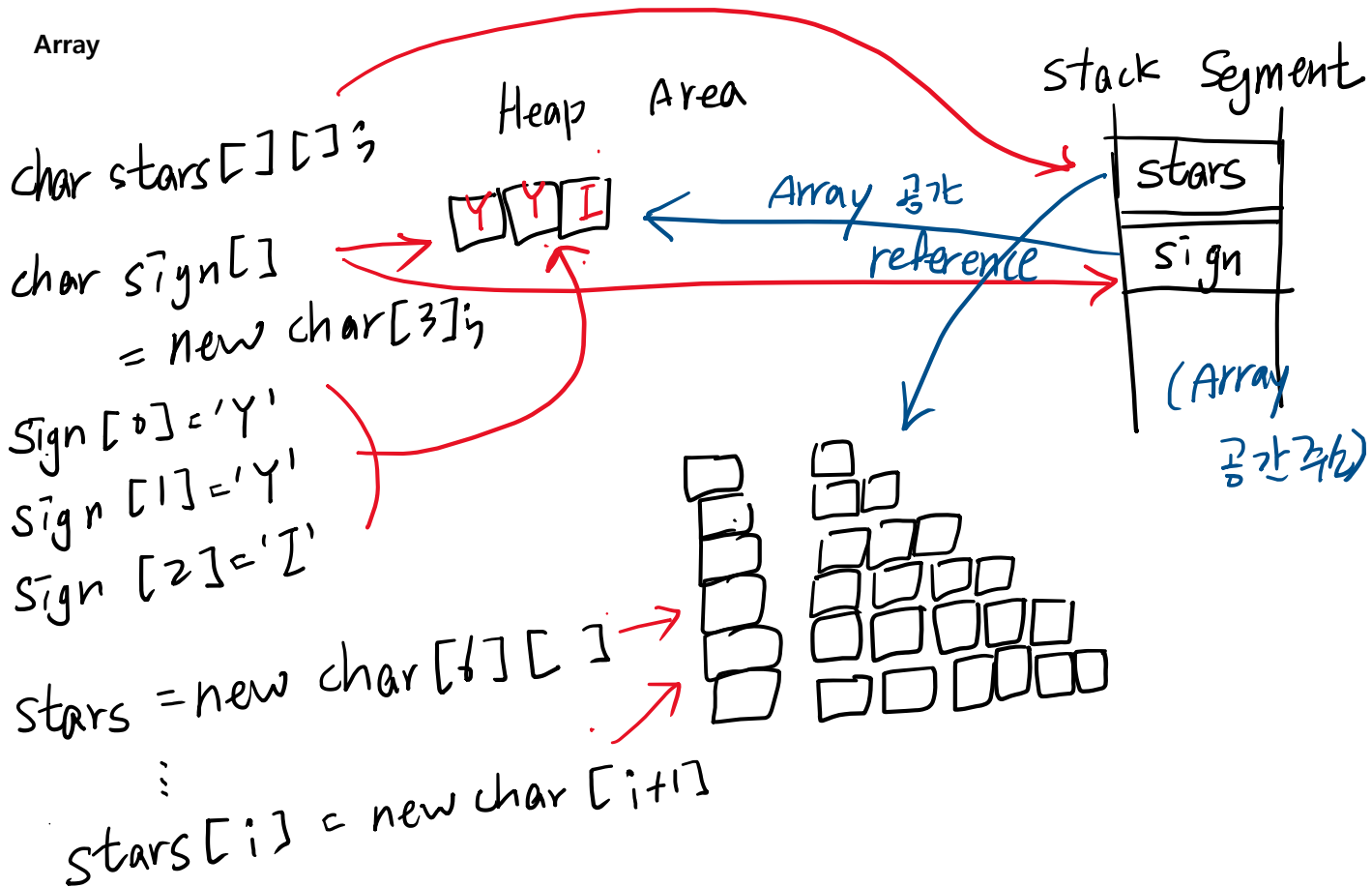
{

클래스 속성들;

}

Object-Oriented 실행 > Class 내에서 정의된 method 의 실행(변수를 참조) > Object 만들어야 함(Object 를 만들지 않으면 class 정의는 아무 의미가 없음)>>instance 생성이 중요(메모리 공간 확보, 실체화)

Array



객체 지향에서의 reference(참조)

참조는 객체의 실제 위치를 가리키는 포인터이다

복제는 파일을 복사하는 것이고 참조는 둘 이상이 같은 것을 가리키고, 참조는 변수가 특정 주소 안의 값을 가리키는 것을 말한다. 둘 이상의 변수가 하나의 주소를 참조했을 때 변수값을 변경하면 다른 하나도 동일하게 변화한다. 둘 이상의 변수가 하나의 주소를 가리키고 그 주소 안의 내용물을 바꿨을 때 변수들이 의미하는 내용물이 바뀐다.

(출처 : <https://webclub.tistory.com/115> , <https://opentutorials.org/course/2517/14152>)

If 문

If (조건문){ }

for 문

for (선언문; 조건문; 증감식) { }

while 문

while (조건문) { }

do-while 문

do { }**while**(조건문)

args[]

main 안에 존재

argument 의 내용, input 할 것, 실행시킬 때 들어가는 것 의미

cmd 창에서

```
java IfTest Yang Hyeon Seo
```

```
args[0] [1] [2]
```

예외처리 try-catch-finally

에러가 발생할 때 사용한다

예외 발생 시 에러 내용을 잘 읽어보면 어디서 어떤 에러가 나온 것인지 알 수 있다

```
try {
```

```
    1 버전
```

```
} catch (예외이름) {
```

```
    2 버전
```

```
} ... finally {
```

```
    n 버전
```

```
}
```

메소드 콜링 (함수 호출)

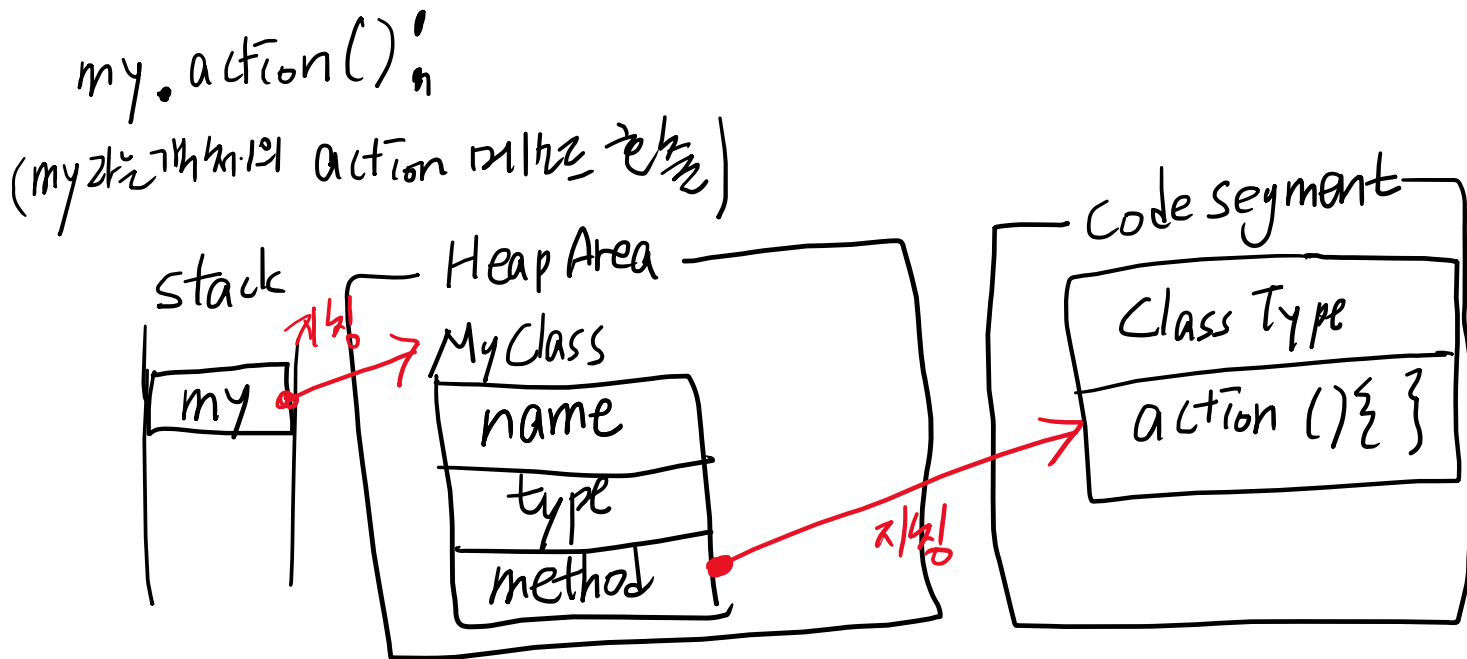
메소드

클래스 안에 위치한 함수

Code segment 에 정의 -(호출)-> 메소드를 지칭하는 공간이 만들어져야 함

객체 .안에 해당되는 메소드의 위치 존재

Method Invocation



`my.action();`이 실행될 때 stack 의 `my` 가 지칭하는 Heap Area 의 `MyClass` 안의 `method` 가 Code Segment 안에 있는 `action() { }`을 호출하여 `my.action();`이 실행되게 된다.

이를 method invocation 이라고 한다.

메소드 인보케이션(method invocation, 메소드 호출 문)은 객체의 메소드를 호출하는 것

메소드 호출 ≠ 메소드 실행

Remote Method Invocation(RMI, 자바 원격 함수 호출)

분산되어 존재하는 객체간, 컴퓨터간 메소드 호출을 가능하게 하는 기술

RMI 시스템을 사용하면 하나의 JVM(Java Virtual Machine)에서 실행 중인 객체가 다른 JVM에서 실행 중인 객체에서 메소드를 호출할 수 있다.

RMI는 전송 계층을 은폐한다

소켓상의 통신

소켓:

CPU 소켓 : 중앙 처리 장치의 소켓(CPU 슬롯)

인터넷 소켓 : IP 프로토콜의 통신접점

SATA : 사타는 하드 디스크나 광 장치 같은 대용량 저장 장치를 호스트에 연결하는데 사용하는 인터페이스 중 하나. 기존 병렬 방식의 에이티에이 규격과 호환된다

(출처:

위키백과 https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%9B%90%EA%B2%A9_%ED%95%A8%EC%88%98_%ED%98%B8%EC%B6%9C,https://ko.wikipedia.org/wiki/%EC%86%8C%EC%BC%93)

ECA rule(Event, Condition, Action)

이벤트 기반 아키텍처 및 활성 데이터베이스 시스템에서 활성 규칙의 구조를 참조하기 위한 바로가기

세부분으로 구성

1. 이벤트 부분: 규칙의 호출을 발동시키는 신호를 결정
2. 조건부분: if를 만족하거나 참일 때 action을 발생시키는 논리적인 테스트
3. 액션부분: 로컬 데이터를 업데이트 하거나 호출

Event가 발생하면 signature(▷함수 이름, 변수의 수, 변수의 타입 등)를 생성,

Condition에서 어떤 함수(메소드)를 실행할지 결정, signature에서 message passing(혹은 Method Invocation) 발생

Action에서 Condition에서 받아온 메소드를 실행

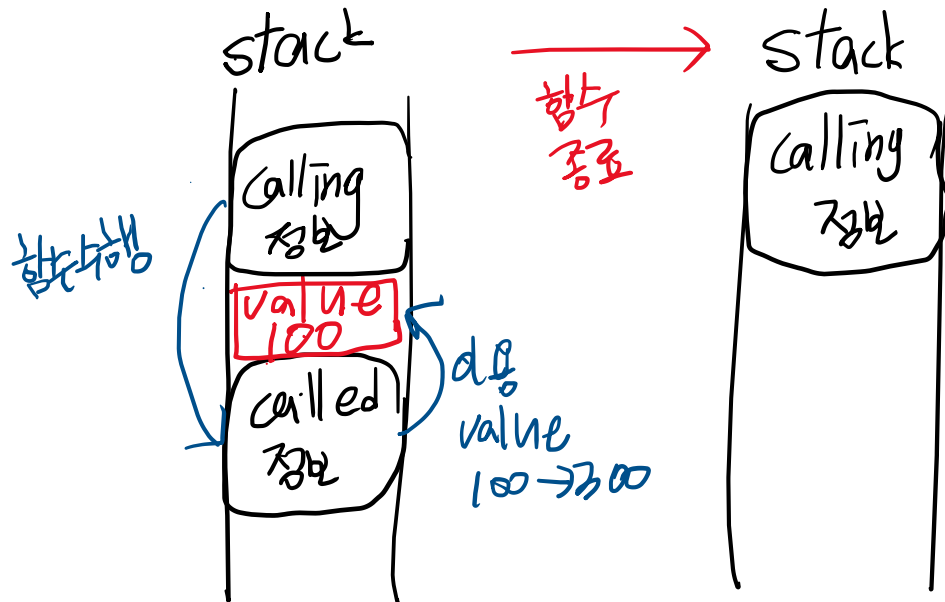
매개변수 전달(parameter passing) 방식

Call-by-value

함수 호출 시 전달되는 값: value

함수가 사용하는 변수값은 Stack 에 저장된다

함수가 종료된 이후 함수가 사용한 값이 사라진다



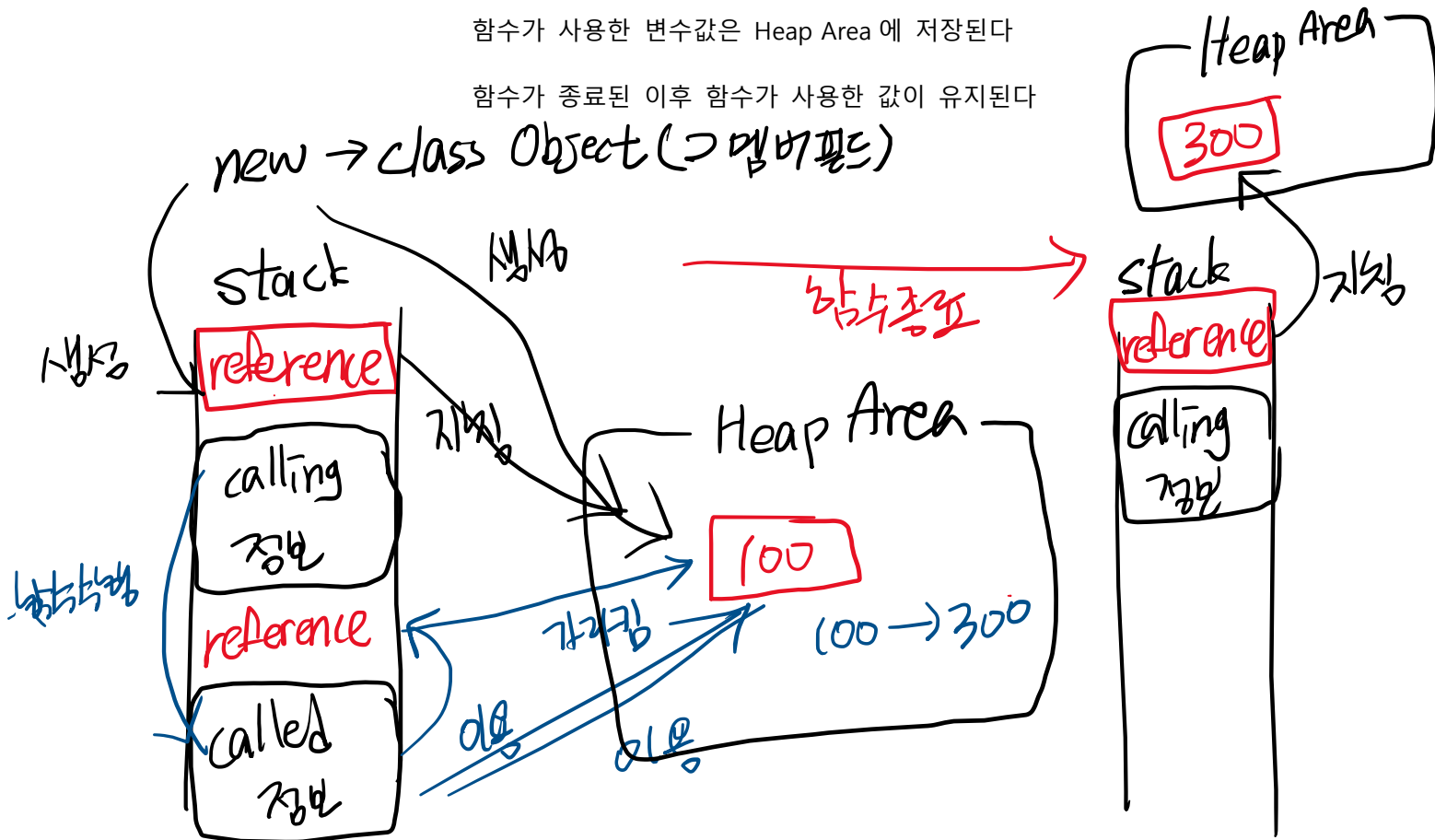
Call-by-reference

함수 호출 시 전달되는 값: reference(주소값)

함수가 사용한 변수값은 Heap Area 에 저장된다

함수가 종료된 이후 함수가 사용한 값이 유지된다

new → class Object (그 멤버 필드)



Activation Record

Procedure(절차)가 한 번 수행되기 위해 필요한 정보들은 기억장소의 연속 블록을 사용하여 관리하게 되는데 이러한 연속블록을 말한다

Procedure: 컴퓨터 프로그래밍에서 동일한 목적에 사용하는 일련의 명령문들을 모은 것. 즉, 프로그래밍에서 루틴이나 서브루틴, 함수와 같은 뜻

Activation Record 는 해당 함수가 끝날 때까지 유지된다. 함수가 끝나면 사라진다

Activation Record 는 나중에 만들어진 것이 먼저 없어져야 함 >> Stack

(출처 : https://www.youtube.com/watch?v=3eQh_W8YF_g)

지역변수(local Variable)

함수 내에서 선언하며 블록 내에서만 접근 가능한 변수

3 장(OOP)

객체지향프로그래밍?

Object 라고 불리는 소프트웨어 컴포넌트로부터 프로그램이 만들어진다

OOP 는 컴포넌트를 잘 쓰기 위해 사용한다

소프트웨어 컴포넌트

소프트웨어 컴포넌트는 시스템이나 어플리케이션의 부분이다

컴포넌트는 소프트웨어의 복잡함을 다룰 수 있는 작은 부분으로 쪼개는 것이다.

각 컴포넌트는 복잡성을 숨긴다

이는 소프트웨어 개발, 유지, 작동, 지원의 복잡성을 줄이고 코드를 다시 사용할 수 있게 한다(ex 클래스 재사용)

CBD(Component Base Design, 구성요소 기반 설계)

구성요소로 사용되도록 설계되고 프로그래밍 된 세그먼트인 구성요소로 프로그래밍하는 것

모듈식 특성>재사용 용이

구성요소 = 레고

>>원하는 목표를 위해 다양한 방법으로 구성하고 조립할 수 있음

특정 위치에 구속되지 않고 전체에서 사용할 수 있음

모든 요소가 미리 코딩 되어있어 처음부터 코딩할 필요가 없다

중복 작업을 제거할 수 있음

(출처 : <https://www.droptica.com/blog/component-based-design/>)

OOP 에서 component 는 Object Instance 를 의미한다

여러 개의 요소들이 나옴>>하나의 object 로 묶음 =>컴포넌트

Object(객체) ⊃ 데이터, 메소드

Object 는 데이터와 메소드들을 서비스에 제공한다

서비스 = function(기능, 행위, 할 일)

Component = Class, Object

OOP 는 Object Instance 에 기반한다

Object Instance

현실세계를 추상화한 것

데이터와 메소드를 포함

추상화

자세한 것은 무시하고 필수적이고 중요한 속성만을
골라서 단순화시키는 과정

프로그램 속의 복잡한 Data 에 추상화 개념을
적용하여 단순화

기존의 잘 정의된 개념들을 이용하여 표현

(출처 : 데이터구조 1 장 강의자료)

Java >>OOP 언어

class 를 이용해 object 정의

Instance 를 new 를 통해 생성

클래스

현실세계의 개념을 추상화하고 대표한다

클래스 객체는 프로그램을 정의한다

키워드 new 를 통해 Object instance 를 만들 수 있다

하나의 클래스로 여러 개의 객체를 만들 수 있다

오브젝트(객체, 인스턴스)

오브젝트는 메모리 공간이 할당된 인스턴스

new 의 실행 결과로 만들어진다

OOP 의 단계

1. 실제 세상을 추상화한다
2. 클래스를 정의한다
3. 클래스 인스턴스 오브젝트를 만든다
4. 클래스 오브젝트를 실행한다

Ex)

Class: Car

Car 클래스로 만들 수 있는 Object

My car

Dad's car

Mom's car