5장. 데이터타입
(참조타입)

5.1.1. 데이터 타입 분류

- JVM이 사용하는 메모리 영역은 Method 영역, Stack영역, Heap 영역으로 구분됨.
- 기본타입은 메모리에 데이터를 저장하는 방식이고, 참조타입은 메모리에 번지를 참조하는 타입임

데이터 타입

구분	구분	예 시
기본타입	정수타입	byte, char, short, int, long
	실수타입	float, long
	논리타입	boolean
참조타입	배열타입	Int [] human = {10,20,30}; // human 배열변수에 10,20,30의 값을 나누어 저장
	열거타입	Public enum Week ={Sun, Mon, Tue, Wed, Thu, Fri, Sat}; // Week란 열거형변수에 요일 저장
	클래스	Class Human { 멤버변수; 메소드; ...} // Human 클래스안에 멤버변수 및 메소드 정의
	인터페이스	Interface Human {메소드; ...} // Human 인터페이스 안에 메소드의 형태 정의

프로그램 문법 – 1 : 기본타입

기본타입은 변수에 직접 값이 대입되는 형태임
메모리의 Stack 영역만 사용됨.

Stack 영역

변수	값
Int human	100
char kor	'A'

Heap 영역

프로그램 문법 – 2 : 참조타입

참조타입은 하나의 변수에 여러 개의 값이 입력되므로
메모리의 Heap 영역에 데이터를 생성하고,
Stack에서는 메모리의 번지수만 참조함.

Stack 영역

변수	번지참조
Int [] human	100번지
Enum Week	500번지

Heap 영역

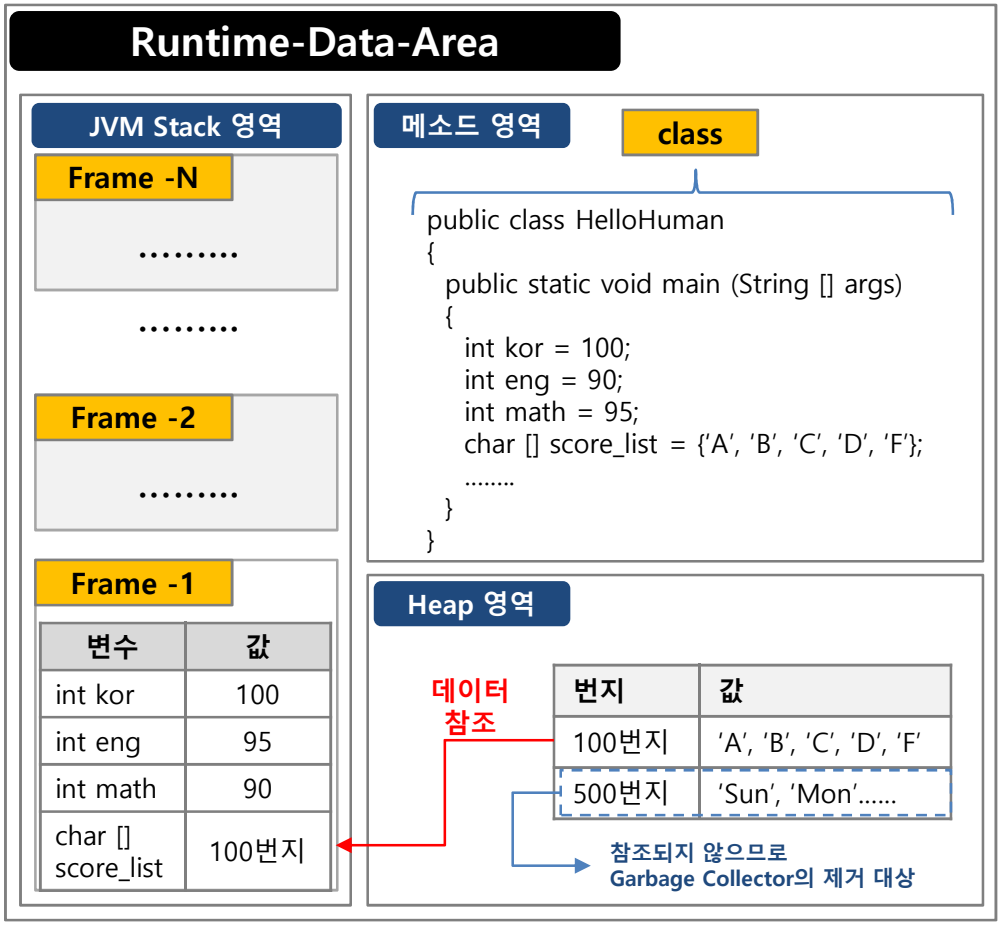
번지	값
100번지	10, 20, 30
500번지	'Sun', 'Mon'.....

데이터 참조

5.1.2. JVM 메모리 사용영역

- JVM 실행시 운영체제에서 Runtime-Data-Area를 할당 받아 운영됨.
- Runtime-Data-Area는 메소드영역, Heap 영역, JVM Stack 영역으로 구분됨.

메모리 사용영역



❖ JVM Stack 영역

- 프로그램별(정확하게는 쓰레드임)로 생성
- 프로그램에서 메소드를 호출할 때마다 Frame 호출
- 메소드가 종료되면 Frame 자동 종료

❖ 메소드 영역

- JVM을 시작할 때 생성됨.
- 로딩된 바이트 코드 (컴파일된 코드)를 분석 및 저장
- 모든 프로그램이 공유함.

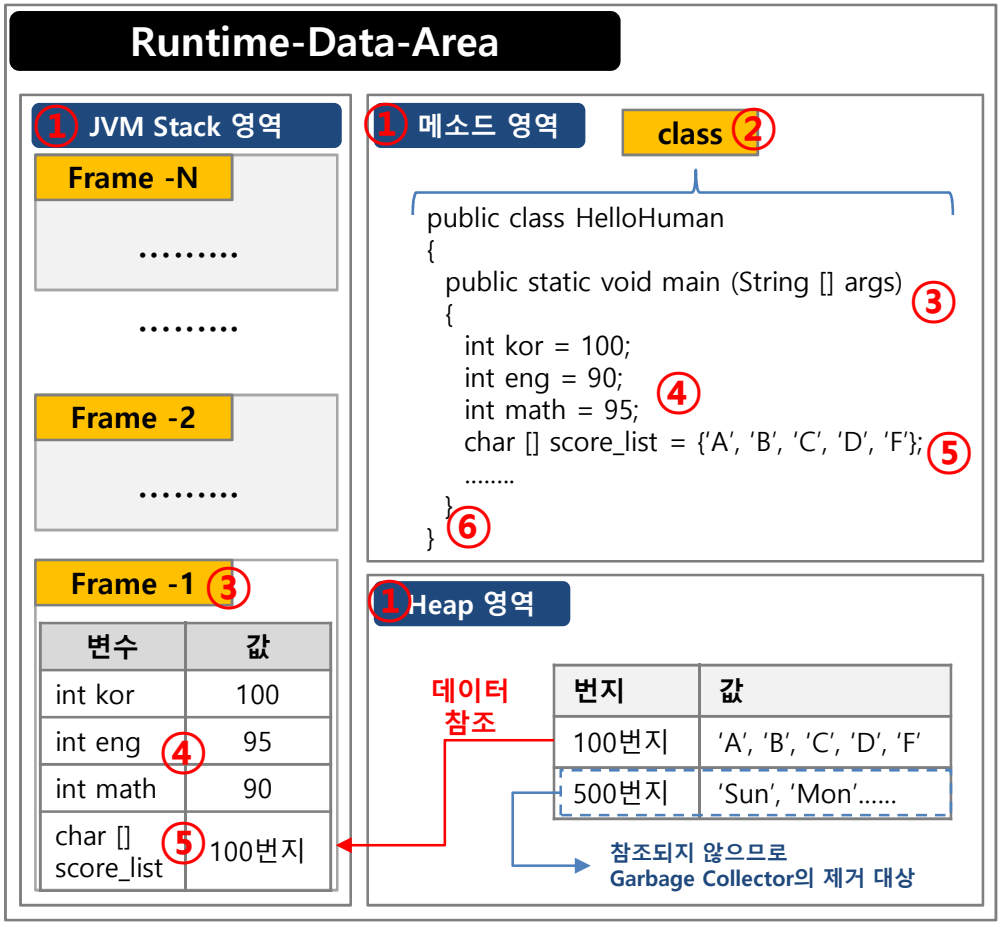
❖ Heap 영역

- JVM이 시작되면 생성됨.
- 객체 / 배열 저장됨.
- 사용되지 않는 객체는 Garbage Collector가 자동 제거함.

5.1.3. JVM 메모리 운영방식

- JVM 실행시 운영체제에서 Runtime-Data-Area를 할당 받아 운영됨.
- Runtime-Data-Area는 메소드영역, Heap 영역, JVM Stack 영역으로 구분됨.

메모리 사용영역



① java HelloHuman 실행
- JVM이 실행되며, 메모리로부터 JVM Stack 영역 / 메소드 영역 / Heap 영역을 할당받음

② 클래스를 메모리에 로딩
- HelloHuman.class 파일이 메소드 영역에 로딩됨.
- 프로그램이 실행됨.

③ public static void main 메서드 실행
- 이 때 Frame-1이 생성됨.

④ kor, eng, math 기본타입 변수 생성 및 값 저장
- 기본타입 변수이므로 JVM Stack 영역 내의 Frame-1 에 변수 및 값이 저장됨.

⑤ char 타입 배열변수 생성 및 값 저장
- 배열은 참조타입 변수이므로 Heap 영역에 참조데이터를 생성하고,
- JVM Stack 영역의 Frame-1의 영역에 데이터를 참조하는 배열변수를 생성함.

⑥ class의 블록 마지막에 오면 할당받은 Runtime Data Area도 메모리에서 제거됨

5.2. 참조타입 데이터 (String)

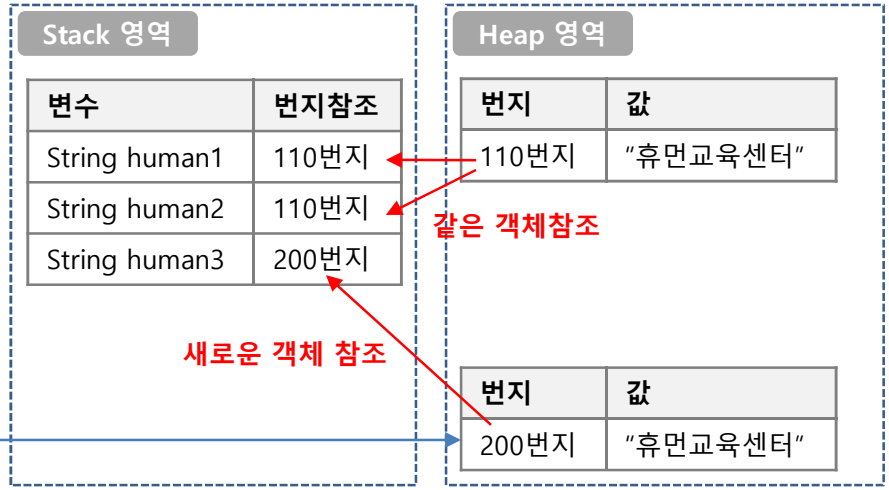
- String은 문자열을 다루는 클래스임
- 문자 단위를 Heap 영역에 두고 String 변수는 JVM Stack 영역에서 관리함.

String 클래스 사용 문법

사용 사례

```
String human1 ;
human1 = "휴먼교육센터";
String human2 = "휴먼교육센터";
String human3 = new String ("휴먼교육센터");
```

새로운 객체 생성



주의 사항

```
boolean result1 = (human1 == human2);        // 동일 객체를 참조하고 있기 때문에 true 임.
boolean result2 = (human1 == human3);        // human3은 새로운 객체이므로 false임.

boolean result3 = human1.equals(human3);     // 참조하고 있는 값을 비교하는 것이므로 true 반환함.
```

5.3.1 참조타입 데이터 (일차원 배열)

- 배열타입은 여러 개의 문자를 한 개의 변수에 담을 수 있음.
- 한 개의 변수에 index로 구분하여 여러 개의 변수를 사용가능 함.

일차원 배열 문법

사용 사례

한 반에 30명의 학생이 있고, 30명의 총점을 구하는 프로그램을 작성할 때 배열을 사용하지 않는다면??

배열 미사용

```
int score1 = 100; .... int score = 80;
int total = 0;
total = score1 + score2 + score3 + ....+ score30;
System.out.println ("학생의 총점 = ", total);
```

코드가 간결해지고 확장성 및 유지보수 용이함.

배열 사용

```
int [] score = {100, 90, 80, ..... , 80};
int total = 0;
for (int index = 0 ; index<30; index++) {
  total = total + score[index];
}
System.out.println ("학생의 총점 = ", total);
```

문법

```
int [] human ;              // JVM Stack 영역에 배열변수 생성
human = {100, 200, 300} ;
        // Heap 영역에 {100,200,300} 데이터 생성 및
        // JVM Stack 영역에 존재하는 human의 배열변수에 번지수 저장
```

주의사항

```
int [] human ;
human = {100, 200, 300};
```

정의한 int형의 데이터만 가능함.

```
human = {100, 200, "휴먼교육센터"};
```

String 타입이라 컴파일 에러 발생

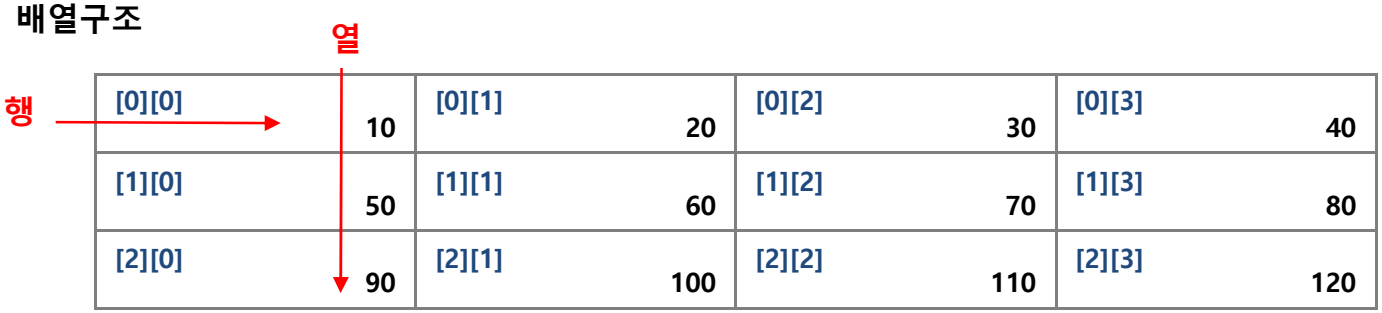
5.3.2 참조타입 데이터 (다차원 배열)

- 다차원배열은 배열의 차원이 2개 이상임을 의미함.
- 다차원 배열은 Heap 영역에 1차원 배열의 형태로 메모리를 관리하듯이 처리함.

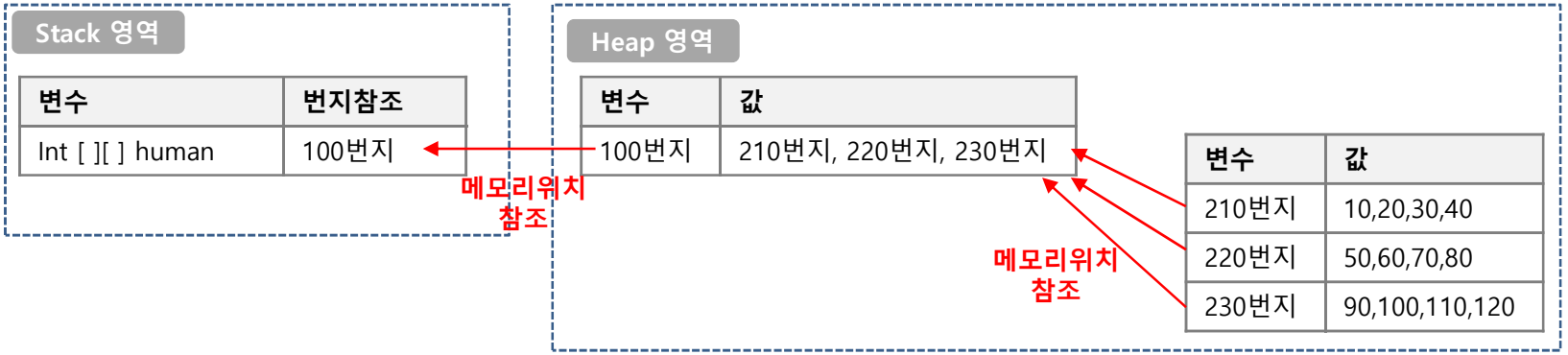
다차원 배열

구조

예시) 2차원 배열구조



메모리 운영



5.3.3 배열의 선언 및 초기화

- 배열은 “선언 + 초기화” 과정이 필요함
- 다차원 배열은 Heap 영역에 1차원 배열의 형태로 메모리를 관리하듯이 처리함.

배열의 초기화 방법

선언 방법

방법 1

방법 2

생성 방법

```
int[] value;                // int타입의 value라는 변수의 배열 선언
value = new int[5];         // int타입의 길이가 5인 value라는 변수의 배열 생성
// ↓↓↓
int[] value = new int[5];   // 배열 선언과 생성을 한 줄에 하기
```

초기화

초기값 미지정시

```
Int [ ] array = new int[2];          // 초기값 0
String [ ] arrStr = new String[5];   // 초기값 null
```

```
int[ ] age = new age[3];
age[0] = 17;
age[1] = 25;
age[3] = 32;
```

또는

Int[ ] age = {17,25,32}

반복문으로 처리

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

for(int i=0; i < array.length; i++) {

    array[i] = i;

    System.out.println(array[i]);

}
```

5.4.1 연습문제

[5-1] 다음은 배열을 선언하거나 초기화한 것이다. 잘못된 것을 고르고 그 이유를 설명하시오.

```
a. int[] arr[];
b. int[] arr = {1,2,3,};
c. int[] arr = new int[5];
d. int[] arr = new int[5]{1,2,3,4,5};
e. int arr[5];
f. int[] arr[] = new int[3][];
```