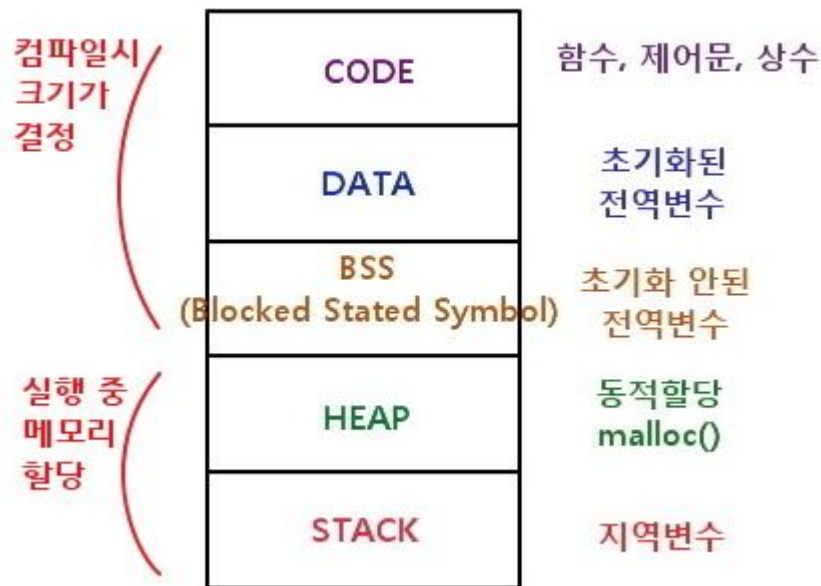


제9장 클래스

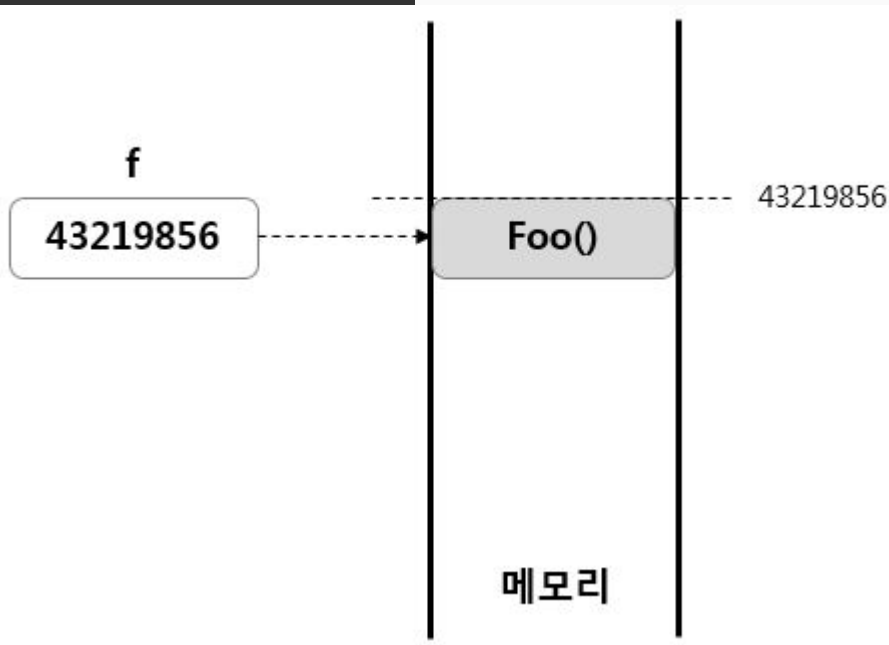
클래스 형과 참조

가상세계 = 컴퓨터의 메모리 영역

인스턴스 = heap 영역 안에 확보된 메모리



```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero;  
4         hero = new Hero();  
5         hero.hp = 100;  
6     }  
7 }
```



```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero1;  
4         hero1 = new Hero();  
5         hero1.hp = 100;  
6  
7         Hero hero2;  
8         hero2 = hero1;  
9         hero2.hp = 200;  
10  
11         System.out.println(hero1.hp);  
12     }  
13 }
```

```
1 public class Sword {  
2     String name;  
3     int damage;  
4 }
```

```
1 public class Hero {  
2     String name;  
3     int hp;  
4     Sword sword;  
5  
6     void attack() {  
7         System.out.println(this.name + "는 공격했다!");  
8         System.out.println("적에게 5포인트의 데미지를 주었다!");  
9     }
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Sword sword = new Sword();  
4         sword.name = "불의 검";  
5         sword.damage = 10;  
6  
7         Hero hero = new Hero();  
8         hero.name = "김영웅";  
9         hero.hp = 100;  
10        hero.sword = sword;  
11  
12        System.out.println("현재의 무기는 " + hero.sword.name);  
13    }  
14 }
```

```
1  public class Wizard {  
2      String name;  
3      int hp;  
4  
5      void heal(Hero hero) {  
6          hero.hp += 10;  
7          System.out.println(hero.name + "의 HP를 10회복했다!");  
8      }  
9  }
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero1 = new Hero();  
4         hero1.name = "스칼";  
5         hero1.hp = 100;  
6  
7         Hero hero2 = new Hero();  
8         hero2.name = "아서스";  
9  
10        Wizard wizard = new Wizard();  
11        wizard.name = "제이나나";  
12        wizard.hp = 50;  
13  
14        wizard.heal(hero1);  
15        wizard.heal(hero2);  
16        wizard.heal(hero2);  
17    }  
18 }
```


String 타입의 진실

int 형이나 double 형 같은 기본형 (primitive type)이 아니라,

Hero 와 같은 “클래스 타입” 이다

```
1 public class Main {  
2     public static void main(String[] args) {  
3         String str = new String("안녕하세요");  
4         System.out.println(str);  
5     }  
6 }
```

생성자 (constructor)

new 했을 때 아무 값도
설정되지 않아 귀찮음.
줄이 길어짐.

```
Hero hero1 = new Hero();  
hero1.name = "스칼";  
hero1.hp = 100;  
  
Hero hero2 = new Hero();  
hero2.name = "아서스";  
  
Wizard wizard = new Wizard();  
wizard.name = "제이나";  
wizard.hp = 50;  
  
wizard.heal(hero1);  
wizard.heal(hero2);  
wizard.heal(hero2);
```

int, short, long	0
char	\u0000
boolean	false
int[]	null
String	null

```
1  public class Hero {  
2      String name;  
3      int hp;  
4      Sword sword;  
5  
6      // 생성자 (constructor)  
7      Hero() {  
8          this.hp = 100;  
9      }  
10  
11     void attack() {...}  
15
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero = new Hero();  
4         System.out.println(hero.hp);  
5     }  
6 }
```

```
1 public class Hero {  
2     String name;  
3     int hp;  
4     Sword sword;  
5  
6     Hero(String name) {  
7         this.hp = 100;  
8         this.name = name; // 인수 값으로 name 필드 초기화  
9     }  
10  
11     void attack() {...}
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero = new Hero("스칼");  
4  
5         System.out.println(hero.hp);  
6         System.out.println(hero.name);  
7     }  
8 }
```



```
1  public class Hero {  
2      String name;  
3      int hp;  
4      Sword sword;  
5  
6      // 기본 생성자  
7      Hero() {  
8          this.hp = 100;  
9          this.name = "김영웅";  
10     }  
11     // 생성자 오버로드  
12     Hero(String name) {  
13         this.hp = 100;  
14         this.name = name;  
15     }
```


```
1 public class Main {  
2     public static void main(String[] args) {  
3         // 두번째 생성자  
4         Hero hero1 = new Hero("스칼");  
5         System.out.println(hero1.name);  
6  
7         // 기본 생성자  
8         Hero hero2 = new Hero();  
9         System.out.println(hero2.name);  
10    }  
11 }
```

리스트 9-14 모든 클래스는 반드시 1개 이상의 생성자를 가진다 = 반드시 정의해야 한다? survivalcooling.com

```
1  public class Map {  
2  
3      Map() {  
4          // 정의할 것이 아무것도 없다  
5      }  
6  }
```

생성자를 1개도 작성하지 않으면 기본 생성자가 있는 것으로 본다.

```
1 public class Hero {  
2     String name;  
3     int hp;  
4     Sword sword;  
5  
6     Hero() {  
7         this.Hero("김영웅"); // 두번째 생성자 호출  
8     }  
9     Hero(String name) {  
10        this.hp = 100;  
11        this.name = name;  
12    }  
13  
14    void attack() {...}
```

```
1  public class Hero {  
2      String name;  
3      int hp;  
4      Sword sword;  
5  
6      Hero() {  
7           this("김영웅");    // 두번째 생성자 호출  
8      }  
9      Hero(String name) {  
10         this.hp = 100;  
11         this.name = name;  
12     }  
13  
14     void attack() {...}
```

리스트 9-17 같은 클래스에서 작성 해도, 각각의 인스턴스에서 별도의 필드를 가짐

```
1 public class Hero {  
2     String name;  
3     int hp;  
4     int money;  
5 }
```

```
1 public class Hero {  
2     String name;  
3     int hp;  
4     static int MONEY;  
5 }
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero1 = new Hero();  
4         Hero hero2 = new Hero();  
5         System.out.println(hero1.hp);  
6         System.out.println(Hero.MONEY); // 공유 자원 접근  
7     }  
8 }
```



```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero hero1 = new Hero();  
4         Hero hero2 = new Hero();  
5         Hero.MONEY = 100;  
6  
7         System.out.println(Hero.MONEY);  
8         System.out.println(hero1.MONEY);  
9  
10        hero1.MONEY = 300;  
11        System.out.println(hero2.MONEY);  
12    }  
13 }
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         // 영웅을 1개도 생성하지 않은 상화  
4         Hero.MONEY = 100;  
5         System.out.println(Hero.MONEY);  
6     }  
7 }
```

```
1  public class Hero {  
2      String name;  
3      int hp;  
4      static int MONEY;  
5      Sword sword;  
6  
7      static void setRandomMoney() {  
8          Hero.MONEY = (int) (Math.random() * 1000);  
9      }  
10
```

```
1 public class Main {  
2     public static void main(String[] args) {  
3         Hero.setRandomMoney();  
4         System.out.println(Hero.MONEY);  
5  
6         Hero hero = new Hero();  
7         System.out.println(hero.MONEY);  
8     }  
9 }
```

```
1 public class Hero {
2     String name;
3     int hp;
4     static int money;
5     Sword sword;
6
7     static void setRandomMoney() {
8         Hero.money = (int) (Math.random() * 1000);
9         System.out.println(this.name + "의 소지금을 초기화했습니다");
10    }
11 }
```

클래스형의 참조

- 클래스형 변수의 안에는 "인스턴스의 정보가 담겨있는 메모리 번지"가 들어 있다
- 어떤 클래스형 변수를 다른 변수에 대입하면, 번지정보만 복사 된다.
- 클래스형은 필드나 메소드의 인수, 리턴 값의 형으로서도 이용된다

생성자

- "클래스명과 동일 명칭으로, 리턴 값의 타입이 명시되어 있지 않은 메소드"는 생성자로 사용된다.
- 생성자는 **new** 에 의한 인스턴스화의 직후에 자동적으로 실행 된다.
- 인수를 가지는 생성자를 정의하면, **new** 를 할 때에 인수를 지정하여 생성자를 실행할 수 있다
- 생성자는 오버로드에 의한 복수 정의가 된다.
- 클래스에 생성자 정의가 1개도 없는 경우에 한해, 컴파일러가 "인수없음, 처리내용없음"의 기본 생성자를 자동정의 해 준다.
- **this()** 를 사용하면, 동일 클래스의 다른 생성자를 호출 할 수 있다

정적 멤버

- **static** 키워드가 붙어 있는 정적 멤버(필드 또는 메소드) 는
 1. 각 인스턴스가 아닌, 클래스에 실체가 준비된다.
 2. "클래스명, 멤버명", "인스턴스 변수명, 멤버명" 의 어디에도 같은 실체에 액세스하게 된다
 3. 인스턴스를 1개도 생성하지 않아도 이용 가능
- 정적 메소드는 그 내부에 정적이지 않은 메소드나 필드를 이용하는 것이 불가능하다



연습문제 9-1

8장의 연습문제에서 작성한 **Cleric** 클래스에 관하여, 2가지 수정을 행하시오.

1. 현시점의 **Cleric** 클래스의 정의에는, 각 인스턴스별로 최대 **HP**와 최대 **MP** 필드에 정보를 가지고 있습니다. 하지만, 모든 성직자의 최대 **HP**는 **50**, 최대 **MP**는 **10**으로 정해져 있어, 각 인스턴스가 각각의 정보를 가지는 것은 메모리 낭비이다.
그래서, 최대 **HP**, 최대 **MP**의 필드가 각 인스턴스별로 있지 않도록, 필드 선언에 적절한 키워드를 추가 하시오.

2. 아래의 방침에 따라, 생성자를 추가 하시오

- A) 이 클래스는 `new Cleric("아서스", 40, 5)` 와 같이, 이름, HP, MP 를 지정하여 인스턴스화 할 수 있다
- B) 이 클래스는 `new Cleric("아서스", 35)` 와 같이, 이름과 HP만으로 지정하여 인스턴스화 할 수 있다. 이 때, MP는 최대 MP와 같은 값이 초기화 된다
- C) 이 클래스는 `new Cleric("아서스")` 와 같이 이름만을 지정하여 인스턴스화 할 수 있다. 이 때, HP 와 MP 는 최대 HP와 최대 MP로 초기화 된다
- D) 이 클래스는 `new Cleric()` 과 같이 이름을 지정하지 않는 경우에는 인스턴스화 할 수 없다고 한다. (이름이 없는 성직자는 존재 할 수 없음)
- E) 생성자는 가능한 한 중복되는 코드가 없도록 작성한다