

파워자바(개정3판)

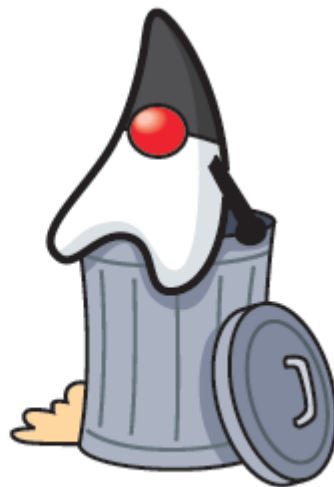


5장 클래스와 객체 II



4장의 목표

1. 객체가 언제 생성되고 언제 소멸되는지를 설명할 수 있나요?
2. 가비지 컬렉터의 역할을 설명할 수 있나요?
3. 객체가 메소드에 전달되는 메커니즘을 설명할 수 있나요?
4. 정적 멤버를 선언하고 사용할 수 있나요?





객체의 생성과 소멸

- 객체도 생성되어서 사용되다가 사용이 끝나면 파괴된다. 객체의 일생은 객체를 참조하는 변수와 밀접한 관련이 있다.



객체의 생성



객체의 사용



객체의 파괴

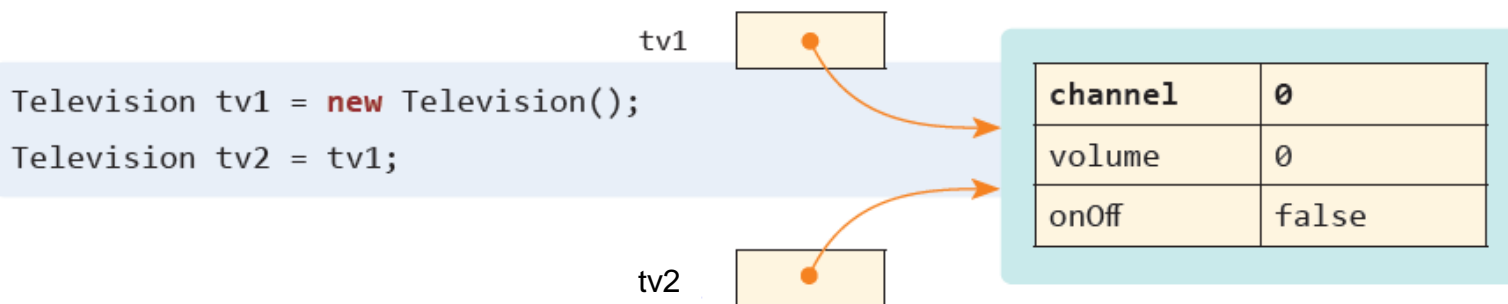
그림 5.1 객체의 일생



참조 변수와 대입 연산

- 기초 변수와 참조 변수는 대입 연산에서 상당히 다르게 동작한다.

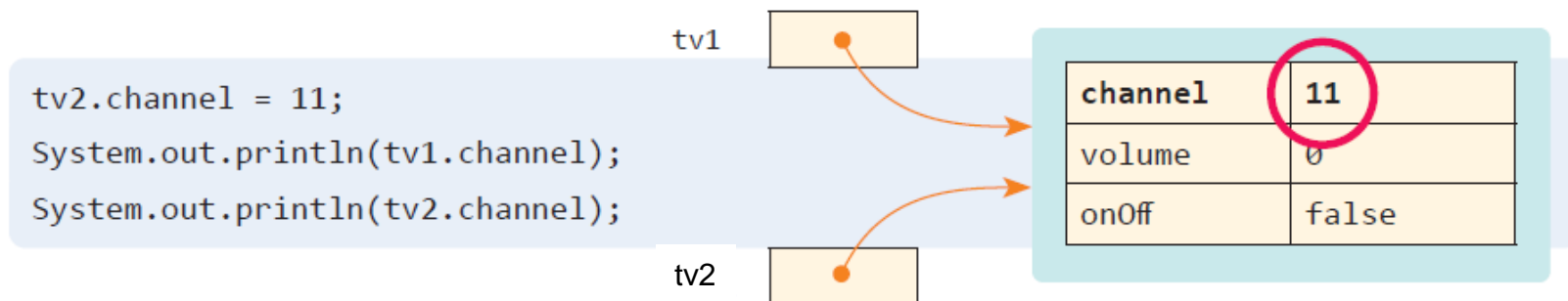
```
int x = 10, y = 20;  
y = x;           // x의 값이 y로 대입된다.
```





참조 변수와 대입 연산

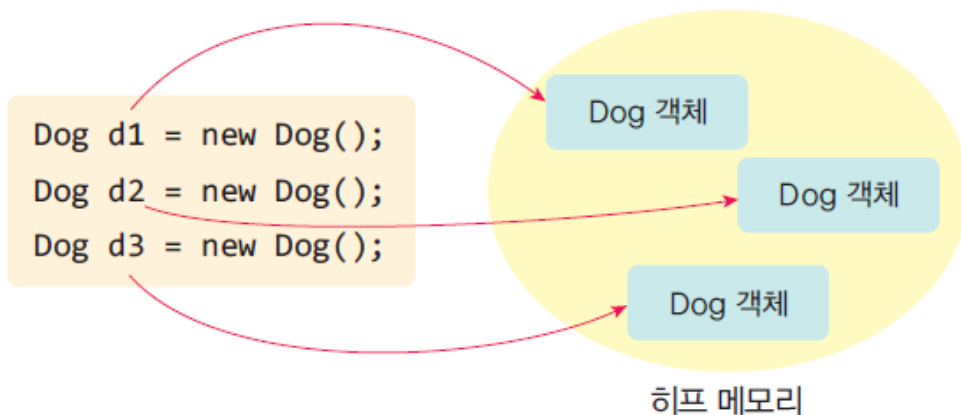
- 기초 변수와 참조 변수는 대입 연산에서 상당히 다르게 동작한다.





객체의 소멸과 가비지 컬렉션

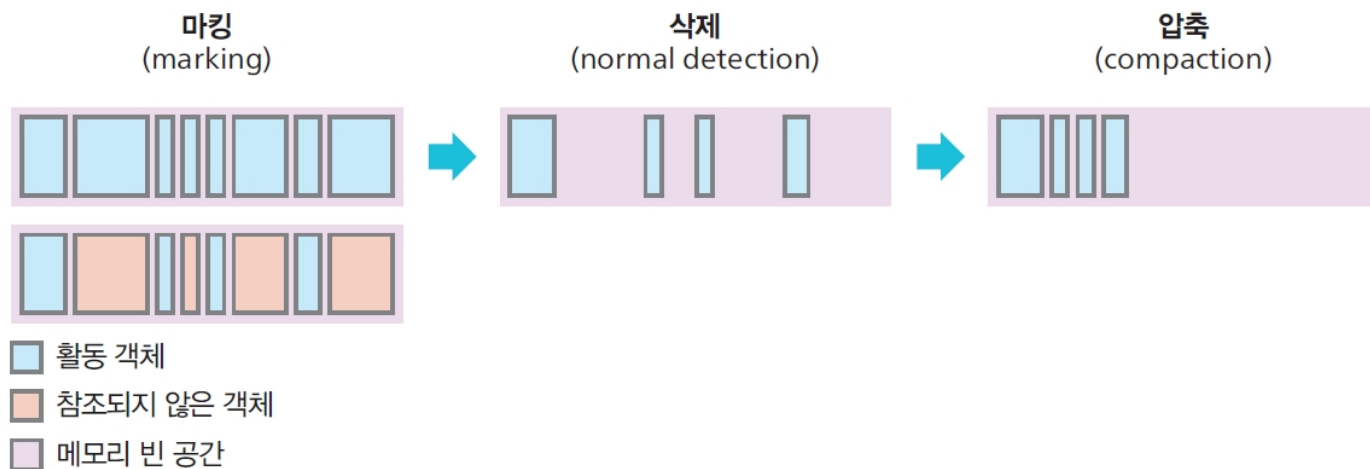
- 자바에는 객체를 생성하는 연산자는 있지만, 객체를 삭제하는 연산자는 없다.
- 자바에서는 자동 메모리 시스템을 사용하는데 이것을 가비지 컬렉션 (garbage collection)이라고 한다





가비지 컬렉터(Garbage Collector)

- 힙 메모리에서 더 이상 필요 없는 객체를 찾아 지우는 작업을 한다.
- 가비지 컬렉터는 JVM의 중요한 부분이다. JVM 중에서 가장 대표적인 것은 오라클사의 HotSpot이다. HotSpot은 많은 가비지 컬렉션 옵션을 제공한다(4개의 가비지 컬렉터가 있다).



출처: 오라클사



가비지 컬렉션 요청

- 개발자는 **System** 객체의 **gc()** 메소드를 호출하여서 가비지 컬렉션을 요청할 수 있다. 하지만 앞에서 이야기한 바와 같이 가비지 컬렉터가 수행되면 모든 다른 애플리케이션이 멈추기 때문에 가비지 컬렉터의 실행 여부는 **JVM**이 판단한다.

```
System.gc();           // 가비지 컬렉션 요청
```

- 자바 가비지 컬렉션을 조정하는 가장 좋은 방법은 JVM을 실행할 때 플래그를 설정하는 것이다.

```
C> java -XX:+UseParallelGC -jar Application.java
```




장점 단점

1. 다음의 코드를 수행할 때, JVM 안에서 발생하는 일을 설명해보자.

```
String a = new String("Hello");  
String b, c;  
b = a; c = a;
```

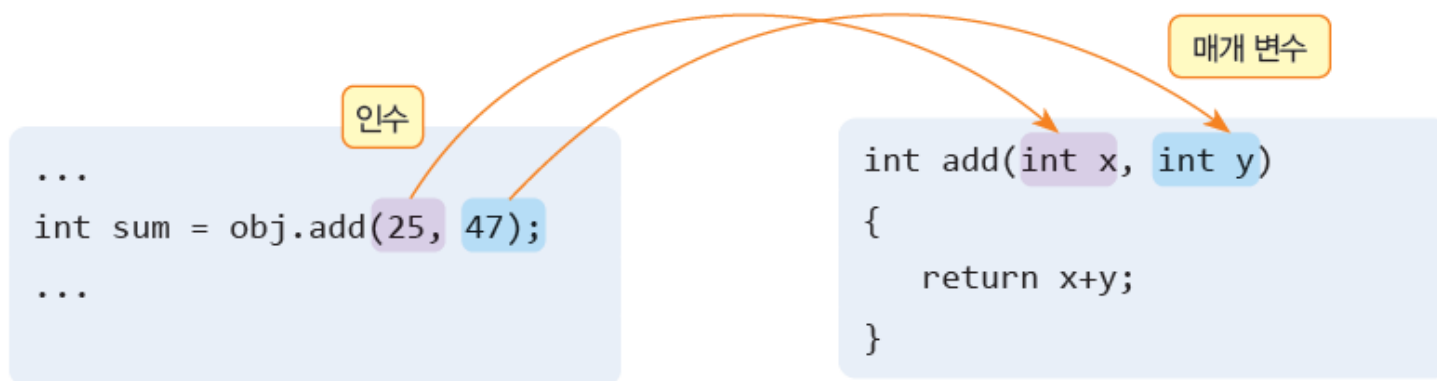
2. C++의 수동 메모리 관리와 비교하여 자바의 가비지 컬렉션 방법의 장단점은 무엇인가?
3. 다음 코드의 어디에서 가비지 컬렉션이 발생할까?

```
String a = new String("Hello");  
String b = new String("World");  
a = b;
```



인수 전달 방법

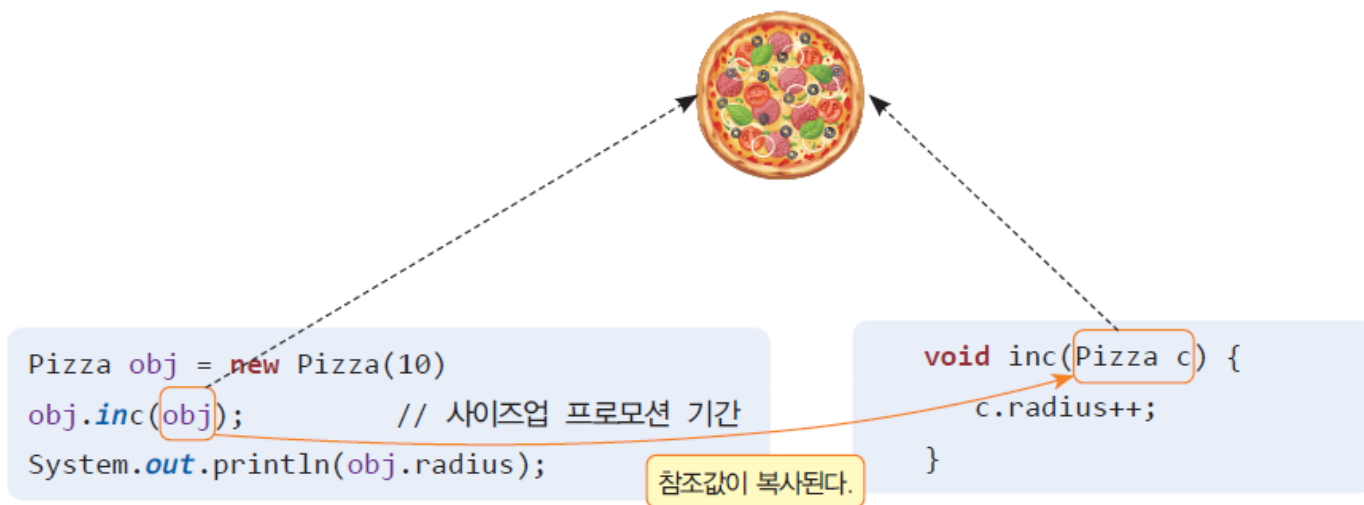
- 자바에서 메소드로 인수가 전달되는 방법은 기본적으로 “값에 의한 호출 (call-by-value)”이다





객체가 전달되는 경우

- 우리가 객체를 메소드로 전달하게 되면 객체 자체가 복사되어 전달되는 것이 아니고 객체의 참조값만 복사되어서 전달된다. 참조 변수는 참조값(주소)을 가지고 있다.
- 참조값이 매개 변수로 복사되면 메소드의 매개 변수도 동일한 객체를 참조하게 된다.





예제

- 피자 객체 2개를 받아서 더 큰 피자 객체를 반환하는 메소드 `Pizza whosLargest(Pizza p1, Pizza p2)`를 작성하고 테스트하라.

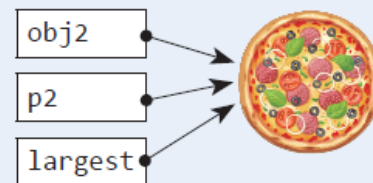
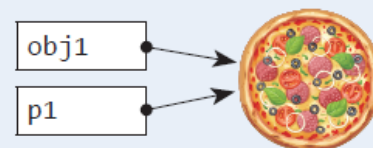
18인치 피자가 더 큼.



예제

PizzaTest.java

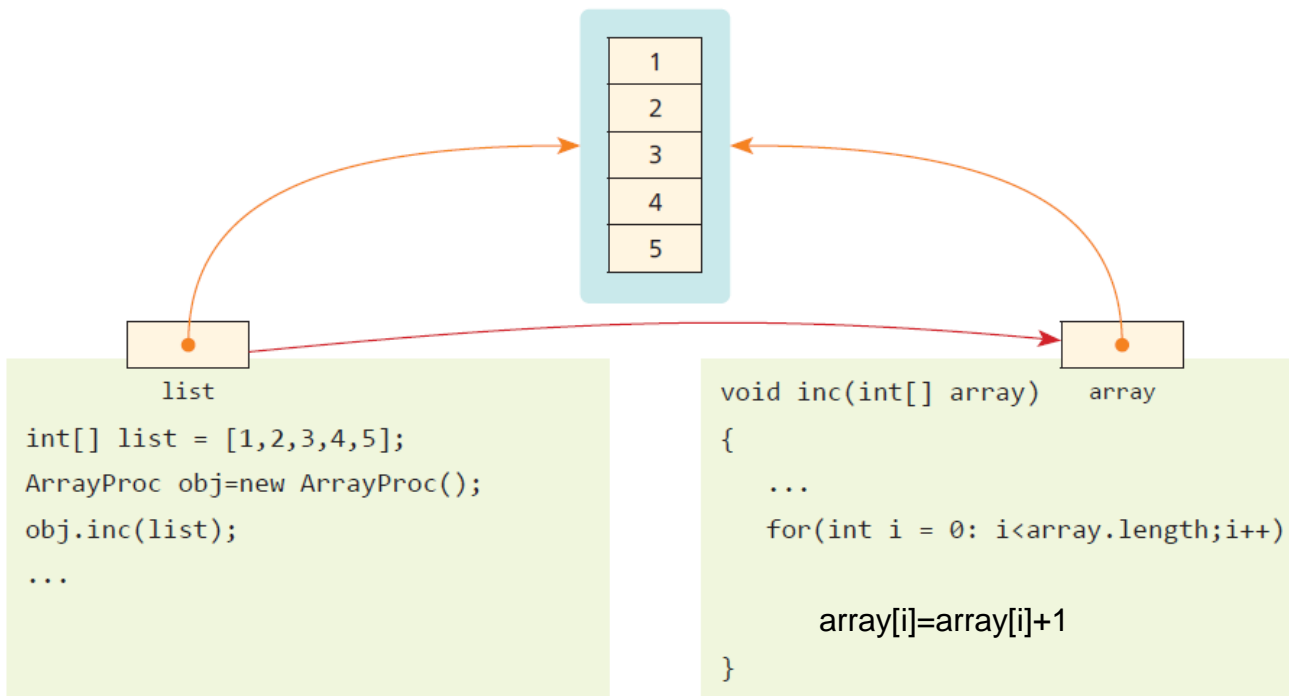
```
01  class Pizza {
02      int radius;
03
04      Pizza(int r) {
05          radius = r;
06      }
07
08      Pizza whosLargest(Pizza p1, Pizza p2) {
09          if (p1.radius > p2.radius)
10              return p1;
11          else
12              return p2;
13      }
14  }
15
16  public class PizzaTest {
17      public static void main(String args[]) {
18          Pizza obj1 = new Pizza(14);
19
20          Pizza obj2 = new Pizza(18);
21
22          Pizza largest = obj1.whosLargest(obj1, obj2);
23          System.out.println(largest.radius + "인치 피자가 더 큼.");
24      }
25  }
```





배열이 전달되는 경우

- 배열도 객체이기 때문에 배열을 전달하는 것은 배열 참조 변수를 복사하는 것이다.





예제: 배열을 받는 메소드 작성하기

배열을 받아서 최소값을 계산하여 반환하는 메소드 `minArray(double[] list)`를 작성하고 테스트해보자.

첫 번째 배열의 최소값=0.1

두 번째 배열의 최소값=-9.0



예제:

ArrayArgumentTest.java

```
01 public class ArrayArgumentTest {
02
03     public static double minArray(double[] list) {
04         double min = list[0];
05
06         for (int i = 1; i < list.length; i++) {
07             if (list[i] < min)
08                 min = list[i];
09         }
10         return (min);
11     }
12
13     public static void main(String args[]) {
14
15         double[] a = { 1.1, 2.2, 3.3, 4.4, 0.1, 0.2 };
16         double[] b = { -2.0, 3.0, -9.0, 2.9, 1.5 };
17
18         double min;
19
20         min = minArray(a);
21         System.out.println("첫 번째 배열의 최소값=" + min);
22         min = minArray(b);
23         System.out.println("두 번째 배열의 최소값=" + min);
24     }
25 }
```




중간 점검

1. 객체를 메소드에 전달하면 무엇이 전달되는가? 메소드 안에서 객체를 변경하면 전달된 객체가 변경되는가?
2. whosLargest() 메소드를 다음과 같이 변경하면 원본 객체에 영향을 끼치는가?

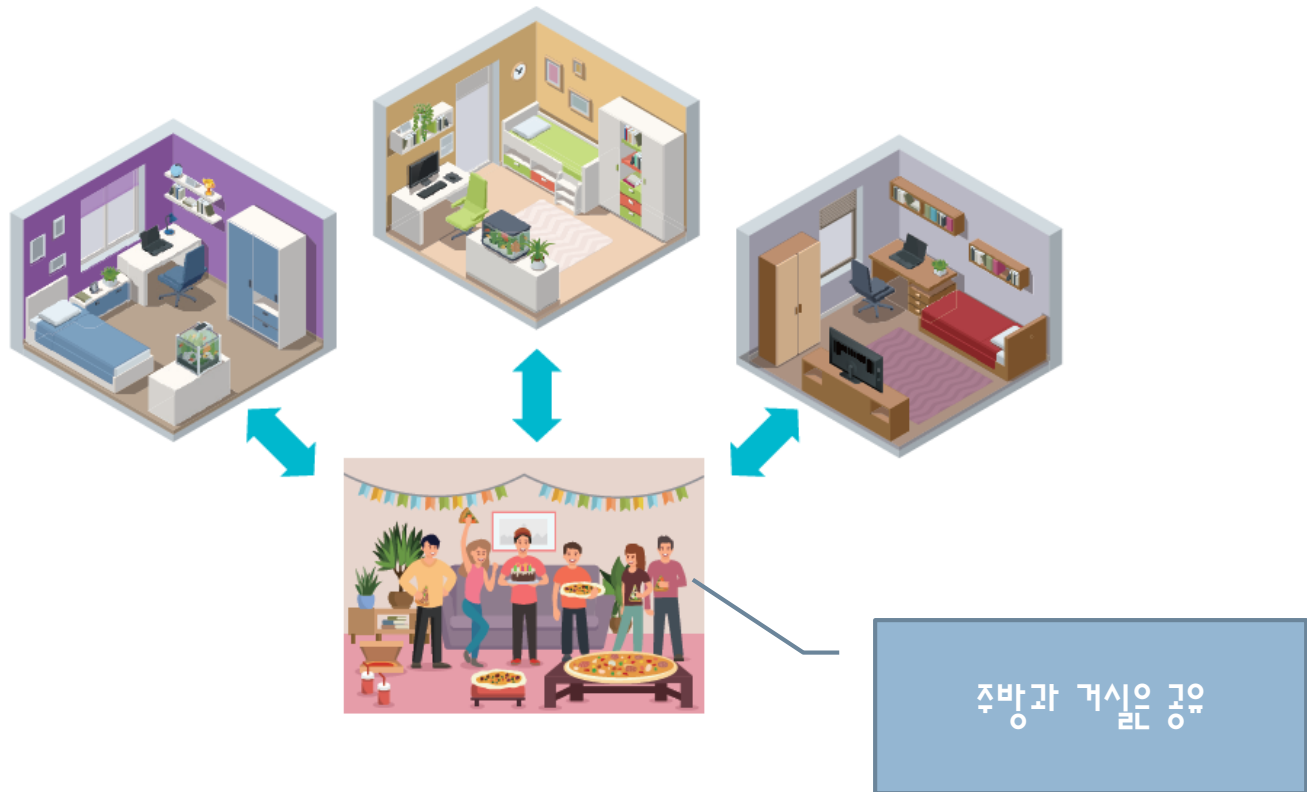
```
Pizza whosLargest(Pizza p1, Pizza p2) {  
    p1 = new Pizza(30);  
    return p1;  
}
```

3. “피자 크기 비교하기” 예제에서 작은 피자 2개와 큰 피자 1개의 면적을 비교하는 메소드를 작성해보자. 즉 20cm 피자 2개를 주문하는 게 나을지, 아니면 30cm 1개를 주문하는 것이 나을지를 결정해 주는 메소드를 작성해보자.



정적 멤버

- 프로그램을 작성하다보면 여러 개의 객체가 하나의 변수를 공유해야 되는 경우가 있다. 이러한 멤버를 정적 멤버(static member) 또는 클래스 멤버(class member)라고 한다.





인스턴스 멤버 VS 정적 멤버

```
class Television {  
    int channel;  
    int volume;  
    boolean onOff;  
    static int count;  
}
```

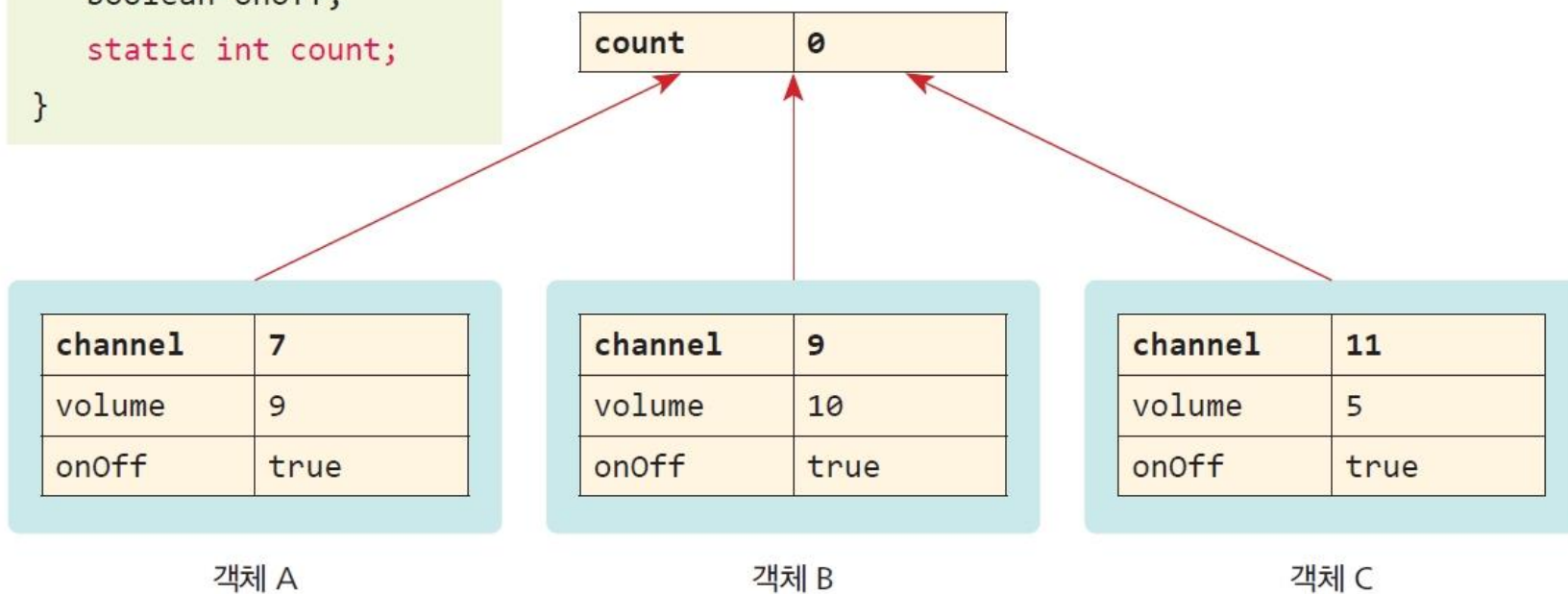


그림 5.2 정적 변수



- 정적 변수(class variable)는 클래스당 하나만 생성되는 변수이다. 정적 변수를 만들려면 변수를 정의할 때 앞에 **static**을 붙이면 된다.
- `Television.count = 100;` // 클래스를 통하여 접근
- `Television obj = new Television();`
- `obj.count = 100;` // 객체를 통하여 접근



예제: 정적 변수 사용하기

- 어떤 가게에서 하루에 판매되는 피자의 개수를 알고 싶다고 하자. 피자의 개수를 알기 위해서는 지금까지 피자가 얼마나 생성되었는지를 알아야 한다. 이러한 경우에 정적 변수를 선언하고 생성자에서 개수를 증가시키면 된다.

지금까지 판매된 피자 개수 = 3





예제: 정적 변수 사용하기

Pizza.java

```
01 public class Pizza {
02     private String toppings;
03     private int radius;
04     static final double PI = 3.141592;           // 상수 정의
05     static int count = 0;                        // 정적 필드
06
07     public Pizza(String toppings){
08         this.toppings = toppings;
09         count++;
10     }
11 }
```

PizzaTest.java

```
01 public class PizzaTest {
02     public static void main(String args[]) {
03         Pizza p1 = new Pizza("Super Supreme");
04         Pizza p2 = new Pizza("Cheese");
05         Pizza p3 = new Pizza("Pepperoni");
06         int n = Pizza.count;
07         System.out.println("지금까지 판매된 피자 개수 = " + n);
08     }
09 }
```



정적 메소드

- 정적 메소드도 정적 변수와 마찬가지로 **static** 수식자를 메소드 앞에 붙여서 만든다.

```
public class Math {  
    public static double sqrt(double a) {  
        ...  
    }  
}  
...  
double value = Math.sqrt(9.0);
```



예제: 정적 메소드 사용하기



정적 메소드 사용하기

예제 5-4

간단한 연산을 제공하는 MyMath 클래스를 작성하여 보자. MyMath 클래스는 n^k 값을 계산하는 power() 메소드와 절대값 메소드를 제공한다. 모두 정적 메소드로 정의해보자.

MyMath.java

```
01 public class MyMath {
02     public static int abs(int x) { return x>0?x:-x; }
03     public static int power(int base, int exponent) {
04         int result = 1;
05         for (int i = 1; i <= exponent; i++)
06             result *= base;
07         return result;
08     }
09 }
```

MyMathTest.java

```
01 public class MyMathTest {
02     public static void main(String args[]) {
03         System.out.println("10의 3승은 "+MyMath.power(10, 3));
04     }
05 }
```

10의 3승은 1000

실행 결과



정적 변수의 활용

- 정적 메소드는 정적 멤버만 사용할 수 있다

```
class Test {  
    int a;                // 인스턴스 변수  
    static int b;         // 정적 변수  
  
    void sub1() { a = 0; } // OK!  
    static void sub2() { a = 0; } // 오류! 정적 메소드에서는 인스턴스 멤버를 사용하면 안 됨  
}
```



정적 변수의 활용

- 정적 메소드에서 정적 메소드를 호출하는 것은 가능하다.

```
public class Test {  
    public static void main(String args[]) {  
        add(10,20); // 오류!! 정적 메소드 안에서 인스턴스 메소드 호출  
    }  
    int add(int x, int y) {  
        return x + y;  
    }  
}
```

Diagram illustrating a static method calling a non-static method, which is an error. A red 'X' marks the call to `add(10,20);` inside the `main` method. A red arrow points from the `add` method call to the `add` method definition, which is enclosed in a red box.



정적 변수의 활용

- 정적 메소드는 `this`를 사용할 수 없다.

```
class Test {  
    static int a;                // 인스턴스 변수  
  
    static void sub(int x) { this.a = x; } // 오류! 정적 메소드에서는 this 사용 안 됨  
}
```



예제: 정적 메소드 사용하기



정적 메소드 사용하기

예제 5-5

main()도 정적 메소드이기 때문에 인스턴스 메소드를 호출할 수 없다. 하지만 정적 메소드는 main()에서 호출할 수 있다. 세제곱 계산 프로그램을 정적 메소드만을 이용하여 작성해보자.

Test.java

```
01 public class Test {  
02     public static int cube(int x) {                // 정적 메소드  
03         int result = x*x*x;  
04         return result;  
05     }  
06     public static void main(String args[]) {        // 정적 메소드  
07         System.out.println("10*10*10은 "+cube(10)); // 정적 메소드 호출  
08     }  
09 }
```

10*10*10은 1000

실행 결과



final 키워드

- 어떤 필드에 **final**을 붙이면 상수가 된다. 상수를 정의할 때 **static**과 **final** 수식어를 동시에 사용하는 경우가 많다

```
public class Car {  
    ...  
    static final int MAX_SPEED = 350;  
    ..  
}
```

상수는 클래스 변수로 만들어서 공유하는 것이 메모리 공간을 절약한다.



정적 블록

- 정적 블록(static block)은 클래스가 메모리에 로드될 때 한 번만 실행되는 문장들의 집합이다.

```
01 public class Test {  
02     static int number;  
03     static String s;  
04     static {  
05         number = 23;  
06         s = "Hello World!";  
07     }  
08  
09     public static void main(String args[]) {  
10         System.out.println("number: " + number);  
11         System.out.println("s: " + s);  
12     }  
13 }
```

정적 블록

```
number: 23  
s: Hello World!
```



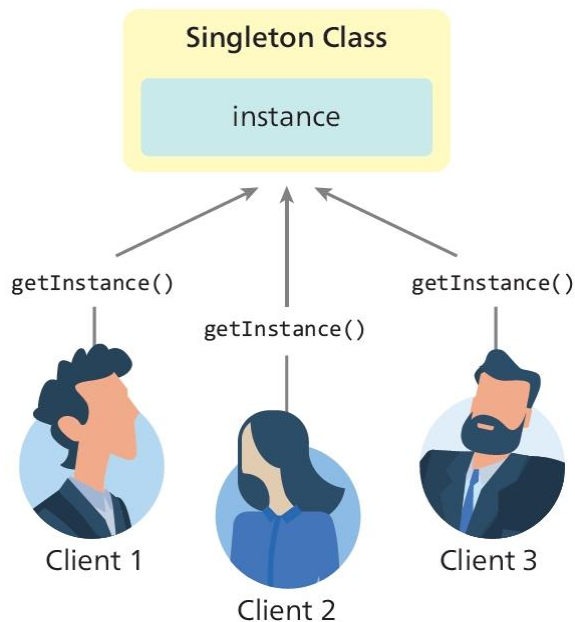
장간접거 공간접근

1. 객체마다 따로 생성되는 멤버를 무엇이라고 하는가?
2. 모든 객체를 통틀어 하나만 생성되는 멤버를 무엇이라고 하는가?
3. 왜 정적 메소드에서는 인스턴스 필드나 인스턴스 메소드에 접근할 수 없는 것인가?
4. 상수를 만들려면 어떤 키워드를 사용하는가?



Lab: 싱글톤 패턴

- 객체 중에는 전체 시스템을 통틀어서 딱 하나만 존재하여야 하는 것들이 있다. 예를 들어 환경설정 클래스나 혹은 네트워크 연결 풀(Pool), 스레드 풀(Pool)을 관리하는 클래스들이다.
- 이럴 경우에 사용할 수 있는 디자인 패턴이 있다. 싱글톤 패턴(singleton design pattern)은 하나의 프로그램 내에서 하나의 인스턴스만을 생성해야 하는 경우에 사용된다.





Sol.

SingleTest.java

```
01  class Single {  
02      private static Single instance = new Single();  
03      private Single() { }          // 전용 생성자  
04  
05      public static Single getInstance() {  
06          return instance;  
07      }  
08  }  
09  
10  public class SingleTest {  
11      public static void main(String[] args) {  
12          Single obj1 = Single.getInstance();  
13          Single obj2 = Single.getInstance();  
14          System.out.println(obj1);  
15          System.out.println(obj2);  
16      }  
17  }
```

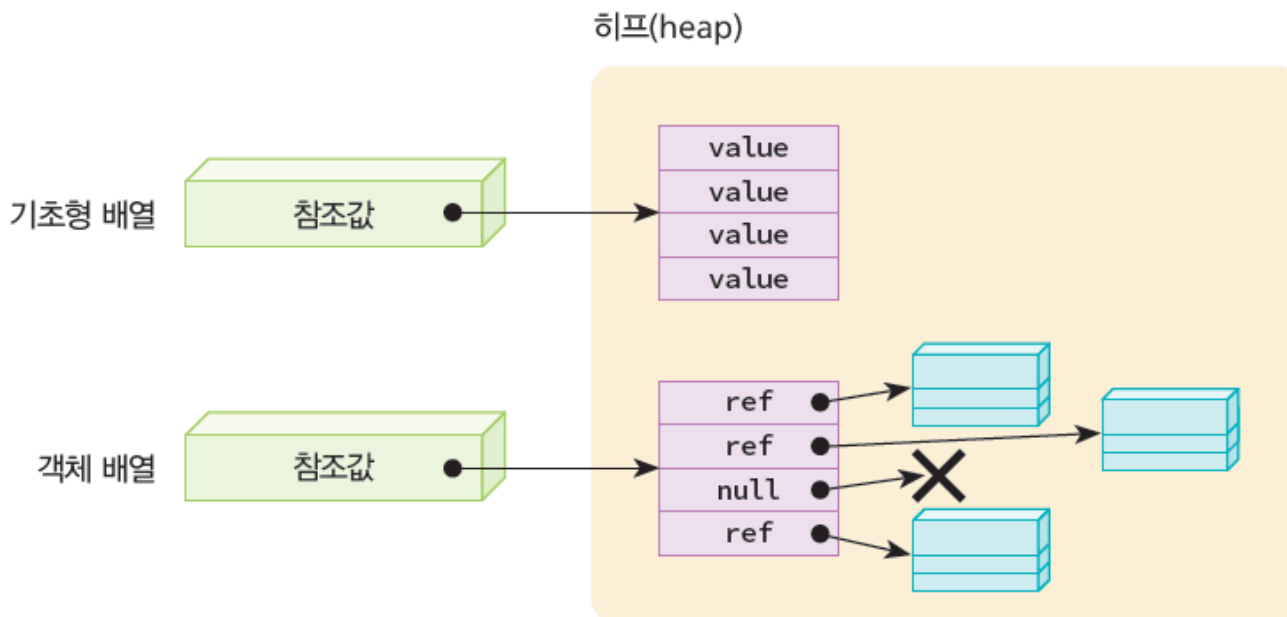
test1.Single@4926097b

test1.Single@4926097b



객체배열

- 객체를 저장하는 배열이다.
- 객체 배열에는 객체에 대한 참조값이 저장되어 있다.





예제

```
class Rect {  
    int width, height;  
  
    public Rect(int w, int h){  
        this.width=w;  
        this.height=h;  
    }  
    double getArea() {    return (double)width*height;    }  
}
```

Rect 객체를 저장하는 배열을 생성해보자.



예제

RectArrayTest.java

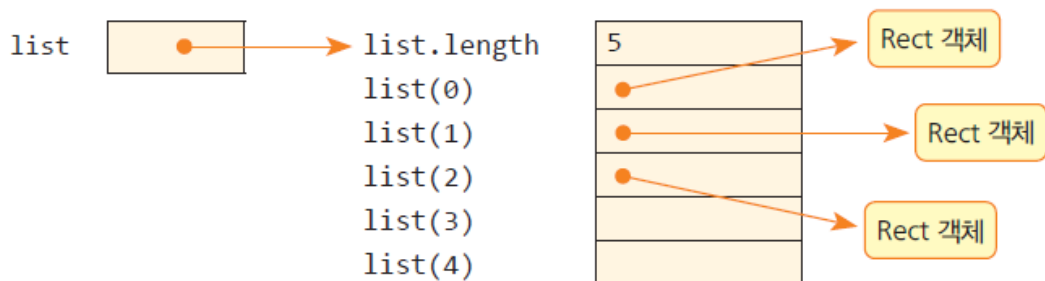
```
01 public class RectArrayTest {  
02     public static void main(String[] args) {  
03         Rect[] list; ← ① 참조 변수 선언  
04         list = new Rect[5]; ← ② 배열 객체 생성  
05  
06         for(int i=0; i < list.length; i++) ← ③ 배열 요소 생성  
07             list[i] = new Rect(i, i);  
08  
09         for(int i=0; i < list.length; i++) ← ④ 배열 요소 처리  
10             System.out.println(i+"번째 사각형의 면적="+list[i].getArea());  
11     }  
12 }
```

0번째 사각형의 면적=0.0
1번째 사각형의 면적=1.0
2번째 사각형의 면적=4.0
3번째 사각형의 면적=9.0
4번째 사각형의 면적=16.0



객체 배열의 모습

```
for(int i=0; i < list.length; i++)  
    list[i] = new Rect(i, i);
```





예제: 객체 배열 만들기

- 영화의 제목과 감독을 저장하는 **Movie** 클래스를 정의한다. 몇 개의 영화 정보를 **Movie** 객체 배열에 저장하고 다시 출력하는 프로그램을 작성해보자.

=====

제목: 백투더퓨처

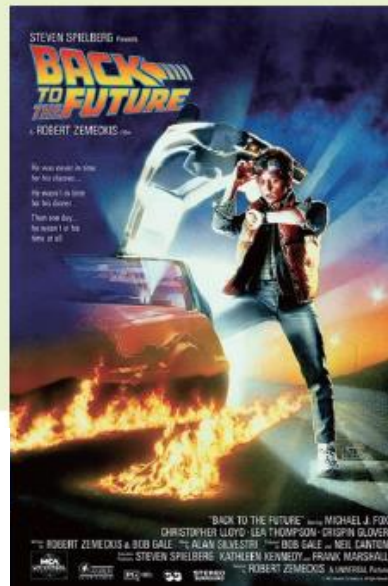
감독: 로버트 저메키스

=====

제목: 티파니에서 아침을

감독: 에드워드 블레이크

=====





예제:

MovieArrayTest.java

```
01  import java.util.Scanner;
02  class Movie {
03      String title, director;
04      static int count;
05      public Movie(String title, String director){
06          this.title=title;
07          this.director=director;
08          count++;
09      }
10  }
11
12  public class MovieArrayTest {
13      public static void main(String[] args) {
14          Scanner scanner = new Scanner(System.in);
15
16          Movie [] list= new Movie[10];
17          list[0] = new Movie("백투더퓨처", "로버트 저메키스");
18          list[1] = new Movie("티파니에서 아침을", "에드워드 블레이크");
19
20          for(int i=0;i < Movie.count; i++) {
21              System.out.println("=====");
22              System.out.println("제목: "+list[i].title);
23              System.out.println("감독: "+list[i].director);
24              System.out.println("=====");
25          }
26      }
27  }
```



동적 객체 배열

- 자바의 표준 배열은 크기가 결정되면 변경하기 힘들다. 따라서 실제 프로그래밍에서는 동적 배열을 많이 사용한다. 동적 배열 중에서 **ArrayList**에 객체들을 저장해보자. 첫 번째 예제로 여행지를 저장하고 있다가 랜덤으로 하나를 선택하여서 사용자에게 추천하는 프로그램을 작성해보자.

여행지 추천 시스템입니다.

추천 여행지는 광



도전 과제 배열

ArrayListTest.java

```
01  import java.util.ArrayList;
02
03  public class ArrayListTest {
04      public static void main(String[] args) {
05
06          ArrayList<String> list = new ArrayList<String>();           // (1)
07          list.add("홍콩");                                           // (2)
08          list.add("싱가포르");
09          list.add("괌");
10          list.add("사이판");
11          list.add("하와이");
12
13          System.out.println("여행지 추천 시스템입니다.");
14          int index = (int) (Math.random()*list.size());
15          System.out.println("추천 여행지는 "+list.get(index));      // (3)
16      }
17  }
```



예제: 연락처 정보 저장하기

이름과 전화번호를 저장하는 Person 클래스를 정의하고 Person 객체를 저장하는 동적 배열을 생성해보자. 몇 사람의 연락처를 저장해본다.

(홍길동, 01012345678)

(김유신, 01012345679)

(최자영, 01012345680)

(김영희, 01012345681)



예제: 연락처 정보 저장하기

ArrayListTest2.java

```
01  import java.util.ArrayList;
02  class Person {
03      String name;
04      String tel;
05
06      public Person(String name, String tel) {
07          this.name = name;
08          this.tel = tel;
09      }
10  };
11
12  public class ArrayListTest2 {
13      public static void main(String[] args) {
14
15          ArrayList<Person> list = new ArrayList<Person>();    // (1)
16          list.add(new Person("홍길동", "01012345678"));
17          list.add(new Person("김유신", "01012345679"));
18          list.add(new Person("최자영", "01012345680"));
19          list.add(new Person("김영희", "01012345681"));
20
21          for (Person obj : list)
22              System.out.println("(" + obj.name + ", " + obj.tel + ")");
23      }
24  }
```



Mini Project: 전기차 클래스

- 전기 자동차를 클래스로 작성해보자. 자동차는 완전(100%) 배터리로 시작한다. 자동차를 운전할 때마다 1km를 주행하고 배터리의 10%를 소모한다. 전기 자동차에는 2가지 정보를 보여주는 디스플레이가 있다. 주행한 총 거리는 “주행거리: ...km”. 남은 배터리 충전량은 “배터리: ...%”와 같이 표시된다.

1. ECar.getInstance() : 새로운 자동차를 생성하는 정적 메소드이다. 새로운 전기 자동차 인스턴스를 반환한다.

```
ECar car = ECar.getInstance();
```

2. dispDistance(): 주행 거리를 표시한다.
3. dispBattery(): 배터리 백분율 표시한다.
4. drive(): 한번 호출될 때마다 1km를 운행한다.



Mini Project: 책 정보 저장

- 사용자가 읽은 책과 평점을 저장하는 객체 배열을 생성해보자. 다음과 같은 메뉴가 제공된다.

=====

1. 책 등록
2. 책 검색
3. 모든 책 출력
4. 종료

=====

번호를 입력하시오: 1

제목: The C Language

평점: 9

...





Summary

- 객체가 자신을 가리키는 참조값을 잃으면 사용이 불가능해진다. 이런 객체들은 정리 대상으로 표시되며 나중에 가용 메모리가 부족해지면 가비지 컬렉터에 의하여 삭제된다.
- 메소드로 기초형의 값을 전달하면 복사되지만, 메소드에 객체를 전달하면 객체의 참조값이 전달된다. 메소드 안에서는 객체 참조값을 이용하여 객체 원본을 변경할 수 있다.
- 인스턴스 멤버란 객체마다 하나씩 생성되는 멤버이다.
- 정적 멤버는 모든 객체를 통틀어서 하나만 생성된다. 객체를 생성하지 않고 클래스 이름만으로 접근할 수 있다.
- 정적 멤버를 선언하려면 앞에 **static**을 붙인다.
- 정적 메소드에서는 정적 멤버에만 접근이 가능하다. 인스턴스 멤버는 사용할 수 없다.
- 객체 배열은 객체를 저장하는 배열이다. 배열 자체도 객체이므로 먼저 배열 객체를 생성하고 배열 요소에 객체를 생성하여 채워야 한다.
- 동적 객체 배열을 생성하려면 **ArrayList**를 사용한다.





Q & A

