

Java



Object 클래스

자바 클래스의 상속 계층 구조에 대하여
Object 클래스의 메소드들



Object 클래스

- 01. 자바 클래스의 상속 계층 구조
- Object 클래스에 대하여

클래스들의 공통 특성은 추출해서
슈퍼클래스로 만듭니다.



Object 클래스는 자바의 모든 클래스들의
공통 특성을 추출해서 만든 슈퍼클래스입니다.

Object 클래스

01. 자바 클래스의 상속 계층 구조

Object 클래스에 대하여

- JDK 라이브러리의 클래스들은 Object 클래스를 상속받음

The image displays two screenshots of the Java API documentation for Java 2 Platform SE 5.0, illustrating the inheritance hierarchy of the `String` and `FileInputStream` classes.

Left Screenshot: `String` Class

- The `String` class is shown as a subclass of `Object` (indicated by a dashed line and text: "Object 클래스를 직접 상속받습니다.").
- The `String` class implements the `Serializable`, `Comparable`, and `CharSequence` interfaces.
- The `String` class is described as a public final class that represents character strings. All string literals in the class are constant and immutable.

Right Screenshot: `FileInputStream` Class

- The `FileInputStream` class is shown as a subclass of `InputStream` (indicated by a dashed line and text: "다른 클래스를 거쳐서 Object 클래스를 상속받습니다.").
- The `FileInputStream` class implements the `Closeable` interface.
- The `FileInputStream` class is described as a public class that obtains input bytes from a file in a file system.

Object 클래스

01. 자바 클래스의 상속 계층 구조

Object 클래스에 대하여

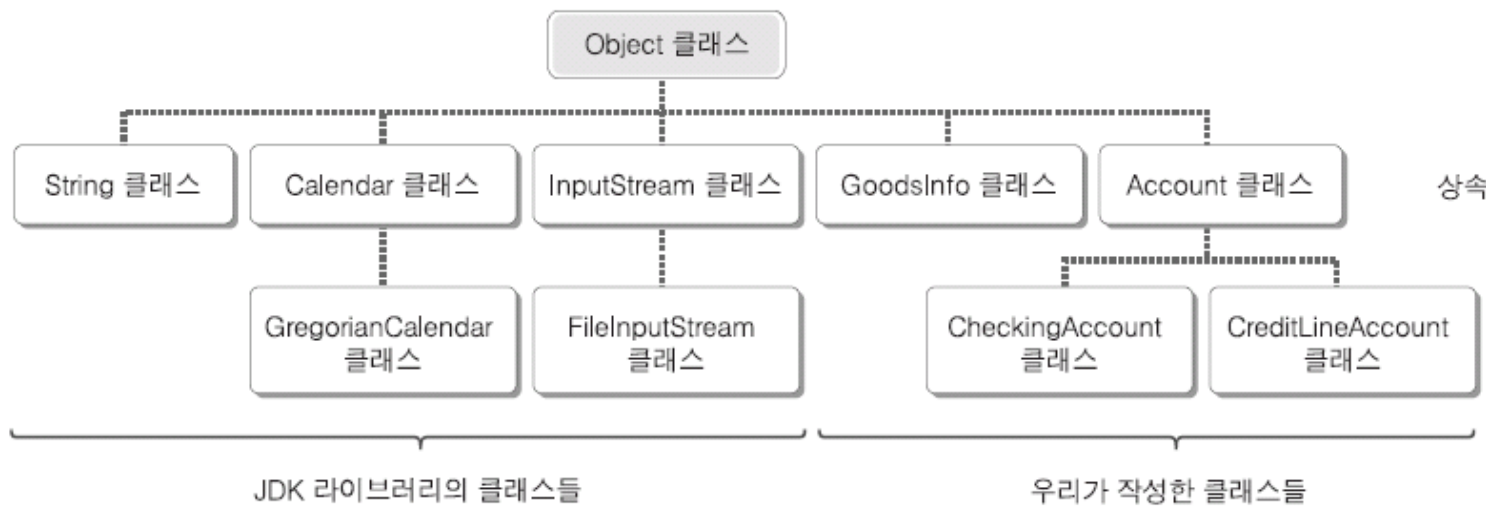
- 우리가 직접 선언한 클래스도 Object 클래스를 상속받음

```
class GoodsInfo {  
    String goodsCode;  
    String name;  
}
```

extends 절이 없는
클래스는 컴파일 과정에서
자동으로 Object의
서브클래스가 됩니다.

Object 클래스

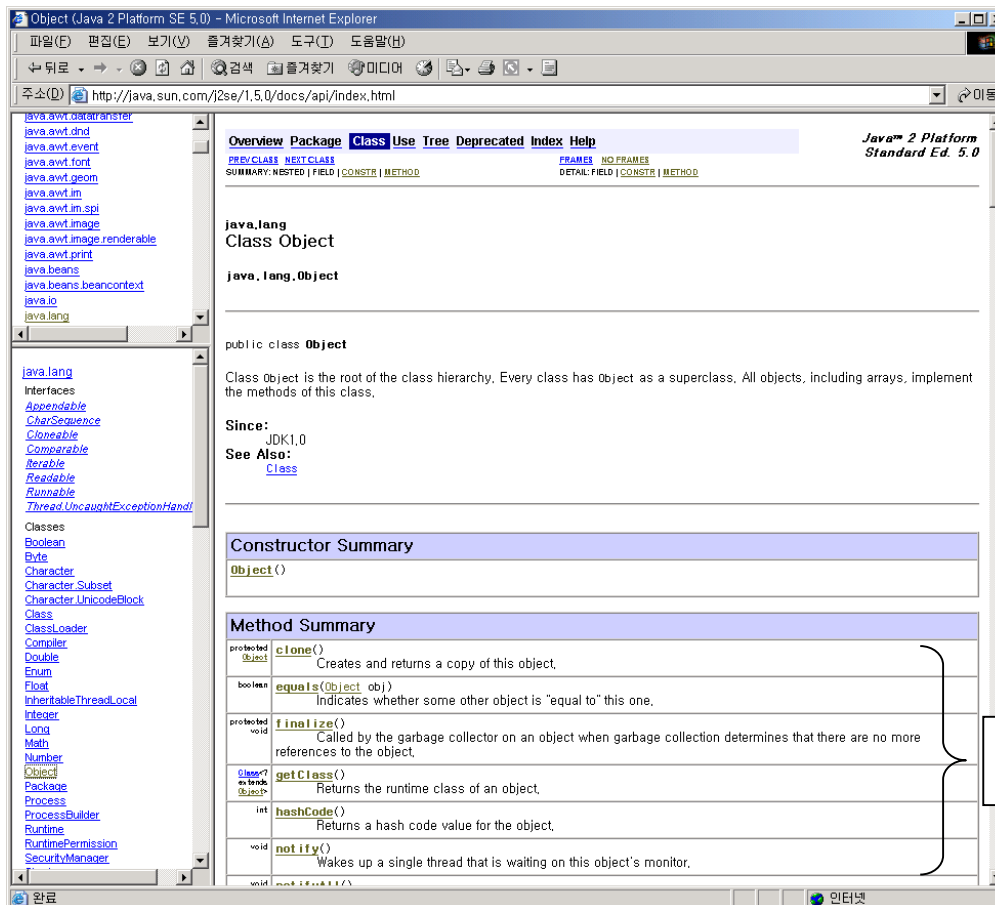
- 01. 자바 클래스의 상속 계층 구조
- 자바의 클래스의 상속 계층 구조



Object 클래스

01. 자바 클래스의 상속 계층 구조 Object 클래스

• Object 클래스의 API 규격서



자바의 다른 모든 클래스들이
상속받습니다.

Object 클래스

02. Object 클래스의 메소드들

toString 메소드

- toString 메소드 : 객체가 가지고 있는 값을 문자열로 만들어서 리턴하는 메소드
- 호출 방법

```
String str = obj.toString();
```



객체가 가진 값을
문자열로 만들어서 리턴하는 메소드

Object 클래스

02. Object 클래스의 메소드들


● toString 메소드

- [예제 11-1] toString 메소드의 사용 예 (1)

```
1 import java.io.File;
2 class ObjectExample1 {
3     public static void main(String args[]) {
4         File file = new File("C:\\www\\바꾸기");
5         String str = file.toString();
6         System.out.println(str);
7     }
8 }
```

- File 객체를 생성합니다.

File 객체에 대해 toString 메소드를 호출합니다.



명령 프롬프트

```
E:\work\chap11\11-2-1\example1>java ObjectExample1
C:배꾸기

E:\work\chap11\11-2-1\example1>
```

Object 클래스

02. Object 클래스의 메소드들

toString 메소드

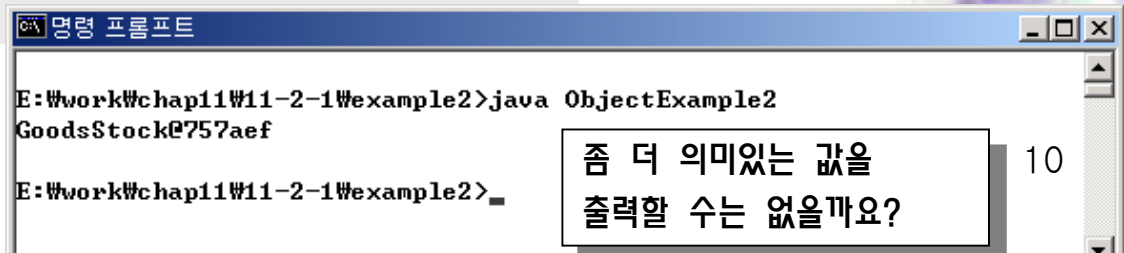
- [예제 11-2] toString 메소드의 사용 예 (2)

상품 재고 클래스를 사용하는 프로그램

```
1 class ObjectExample2 {  
2     public static void main(String args[]) {  
3         GoodsStock obj = new GoodsStock("57293", 100);  
4         String str = obj.toString();  
5         System.out.println(str);  
6     }  
7 }
```

상품 재고 클래스

```
1 class GoodsStock {  
2     String goodsCode;    // 상품코드  
3     int stockNum;        // 재고수량  
4     GoodsStock(String goodsCode, int stockNum) {  
5         this.goodsCode = goodsCode;  
6         this.stockNum = stockNum;  
7     }  
8 }
```



```
명령 프롬프트  
E:\work\chap11\11-2-1\example2>java ObjectExample2  
GoodsStock@757aef  
E:\work\chap11\11-2-1\example2>
```

좀 더 의미있는 값을
출력할 수는 없을까요?

10

Object 클래스

02. Object 클래스의 메소드들

toString 메소드

- [예제 11-3] toString 메소드의 오버라이딩 예

상품 재고 클래스를 사용하는 프로그램

```
1 class ObjectExample2 {  
2     public static void main(String args[]) {  
3         GoodsStock obj = new GoodsStock("57293", 100);  
4         String str = obj.toString();  
5         System.out.println(str);  
6     }  
7 }
```

상품 재고 클래스

```
1 class GoodsStock {  
2     String goodsCode;    // 상품코드  
3     int stockNum;        // 재고수량  
4     GoodsStock(String goodsCode, int stockNum) {  
5         this.goodsCode = goodsCode;  
6         this.stockNum = stockNum;  
7     }  
8     public String toString() {  
9         String str = "상품코드:" + goodsCode + " 재고수량:" + stockNum;  
10        return str;  
11    }  
12 }
```

명령 프롬프트

```
E:\work\chap11\11-2-1\example3>java ObjectExample2  
상품코드:57293 재고수량:100
```

```
E:\work\chap11\11-2-1\example3>
```

----- 상품코드와 재고수량을
문자열로 만들어서 리턴합니다.

Object 클래스

02. Object 클래스의 메소드들

toString 메소드의 자동 호출

- 문자열 연결 연산자 +에 의해 자동 호출됨

```
File file = new File("C:\\독수리");  
String str = "경로명:" + file;
```

```
File file = new File("C:\\독수리");  
String str = "경로명:" + file.toString();
```

똑같은 효과를 갖는 두 코드

Object 클래스

02. Object 클래스의 메소드들

toString 메소드의 자동 호출

- [예제 11-4] 문자열 연결 연산자에 의해 자동 호출되는 예

상품 재고 클래스를 사용하는 프로그램

```
1 class ObjectExample3 {  
2     public static void main(String args[]) {  
3         GoodsStock obj = new GoodsStock("57293", 100);  
4         String str = "재고 정보 = " + obj; ----- 문자열과 GoodsStock 객체를 + 연산자로 연산합니다.  
5         System.out.println(str);  
6     }  
7 }
```

상품 재고 클래스

```
1 class GoodsStock {  
2     String goodsCode;    // 상품코드  
3     int stockNum;        // 재고수량  
4     GoodsStock(String goodsCode, int stockNum) {  
5         this.goodsCode = goodsCode;  
6         this.stockNum = stockNum;  
7     }  
8     public String toString() {  
9         String str = "상품코드:" + goodsCode + " 재고수량:" + stockNum;  
10        return str;  
11    }  
12 }
```

명령 프롬프트

```
E:\work\chap11\11-2-1\example4>java ObjectExample3  
재고 정보 = 상품코드:57293 재고수량:100  
E:\work\chap11\11-2-1\example4>_
```

Object 클래스

02. Object 클래스의 메소드들

toString 메소드의 자동 호출

- [예제 11-5] System.out.println 메소드에 의해 자동 호출되는 예

상품 재고 클래스를 사용하는 프로그램

```
1 class ObjectExample4 {  
2     public static void main(String args[]) {  
3         GoodsStock obj = new GoodsStock("73527", 200);  
4         System.out.println(obj);----- GoodsStock 객체를 System.out.println 메소드에 넘겨줍니다.  
5     }  
6 }
```

상품 재고 클래스

```
1 class GoodsStock {  
2     String goodsCode;    // 상품코드  
3     int stockNum;        // 재고수량  
4     GoodsStock(String goodsCode, int stockNum) {  
5         this.goodsCode = goodsCode;  
6         this.stockNum = stockNum;  
7     }  
8     public String toString() {  
9         String str = "상품코드:" + goodsCode + " 재고수량:" + stockNum;  
10        return str;  
11    }  
12 }
```

명령 프롬프트

```
E:\work\chap11\11-2-1\example5>java ObjectExample4  
상품코드:73527 재고수량:200  
E:\work\chap11\11-2-1\example5>
```

Object 클래스

02. Object 클래스의 메소드들

equals 메소드

- equals 메소드 : 객체가 가지고 있는 값을 비교하는 메소드
- 호출 방법

obj1.equals(obj2)

이 객체와 이 객체의
값을 비교합니다.

Object 클래스

02. Object 클래스의 메소드들

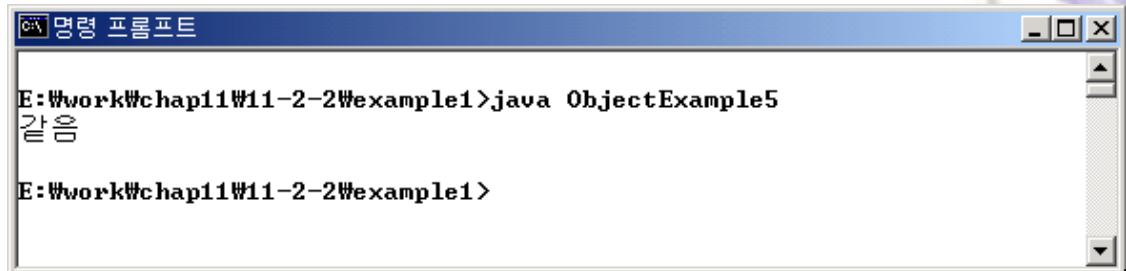
equals 메소드

- [예제 11-6] equals 메소드의 사용 예 (1)

```
1 import java.util.GregorianCalendar;
2 class ObjectExample5 {
3     public static void main(String args[]) {
4         GregorianCalendar obj1 = new GregorianCalendar(2007, 0, 1);
5         GregorianCalendar obj2 = new GregorianCalendar(2007, 0, 1);
6         if (obj1.equals(obj2))
7             System.out.println("같음");
8         else
9             System.out.println("다름");
10    }
11 }
```

똑같은 값을 갖는
두 개의 객체를 생성합니다.

두 객체의 값을 equals 메소드로 비교하여 결과를 출력합니다.



```
명령 프롬프트
E:\work\chap11\11-2-2\example1>java ObjectExample5
같음
E:\work\chap11\11-2-2\example1>
```


Object 클래스

02. Object 클래스의 메소드들

equals 메소드

- [예제 11-7] equals 메소드의 사용 예 (2)

원 클래스를 사용하는 프로그램

```
1    class ObjectExample6 {  
2        public static void main(String args[]) {  
3            Circle obj1 = new Circle(5);  
4            Circle obj2 = new Circle(5);  
5            if (obj1.equals(obj2))  
6                System.out.println("같음");  
7            else  
8                System.out.println("다름");  
9        }  
10    }
```

똑같은 값을 갖는 두 개의 Circle 객체를 생성합니다.

두 객체의 값을 equals 메소드로 비교하여 결과를 출력합니다.

원 클래스

```
1    class Circle {  
2        int radius;        // 반지름  
3        Circle(int radius) {  
4            this.radius = radius;  
5        }  
6    }
```

```
C:\> 명령 프롬프트  
E:\work\chap11\11-2-2\example2> java ObjectExample6  
다름  
E:\work\chap11\11-2-2\example2>
```

이런 결과가 나오는 이유는?

Object 클래스

02. Object 클래스의 메소드들

equals 메소드

- [예제 11-8] equals 메소드의 오버라이딩 예

원 클래스

```
1  class Circle {  
2      int radius;          // 반지름  
3      Circle(int radius) {  
4          this.radius = radius;  
5      }  
6      public boolean equals(Object obj) {  
7          if (!(obj instanceof Circle))  
8              return false;  
9          Circle circle = (Circle) obj;  
10         if (radius == circle.radius)  
11             return true;  
12         else  
13             return false;  
14     }  
15 }
```

파라미터 객체가 Circle 타입인지 검사합니다.

파라미터 객체를 Circle 타입으로 캐스트합니다.

파라미터 객체와 객체 자신의 radius 필드 값을 비교합니다.

명령 프롬프트

E:\work\chap11\11-2-2\example3>java ObjectExample6

같은

E:\work\chap11\11-2-2\example3>_

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

- clone 메소드 : 객체를 복제하는 메소드
- 복제가 가능한 클래스와 불가능한 클래스
 - - JDK 라이브러리 클래스의 경우 Cloneable 인터페이스의 구현 여부로 결정됨

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

.. 호출 방법

```
GregorianCalendar obj2 = (GregorianCalendar) obj1.clone();
```

GregorianCalendar 변수에 대입합니다.
GregorianCalendar 타입으로 캐스트 한 후
clone 메서드의 리턴 값을

Object 클래스

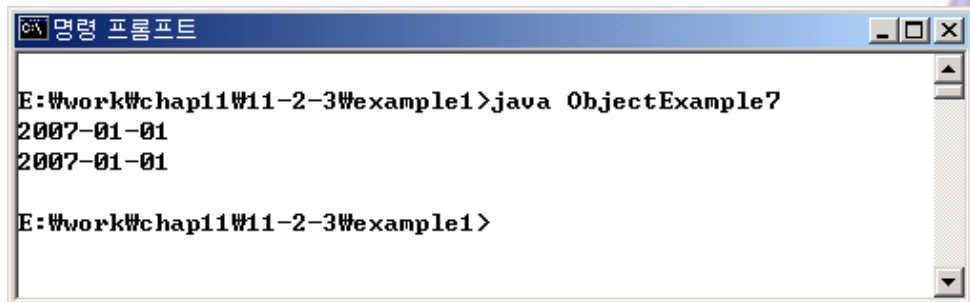
02. Object 클래스의 메소드들

clone 메소드

- [예제 11-9] clone 메소드의 호출 예

```
1  import java.util.GregorianCalendar;
2  class ObjectExample7 {
3      public static void main(String args[]) {
4          GregorianCalendar obj1 = new GregorianCalendar(2007, 0, 1); ----- 객체를 생성합니다.
5          GregorianCalendar obj2 = (GregorianCalendar) obj1.clone(); ----- clone 메소드로 객체를 복제합니다.
6          System.out.printf("%tF %n", obj1); }
7          System.out.printf("%tF %n", obj2); }
8      }
9  }
```

원본 객체와 복제 객체의 값을 출력합니다.



```
E:\work\chap11\11-2-3\example1>java ObjectExample7
2007-01-01
2007-01-01
E:\work\chap11\11-2-3\example1>
```

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

- 복제 가능한 클래스를 만드는 방법
 - 1) 다음과 같은 클래스가 있다고 가정합니다.

```
class Rectangle {  
    int width, height;  
    Rectangle(int width, int height) { // 생성자  
        this.width = width;  
        this.height = height;  
    }  
}
```

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

•• 복제 가능한 클래스를 만드는 방법

- 2) Cloneable 인터페이스를 구현하고 clone 메소드를 오버라이드하도록 만들어야 합니다.

```
class Rectangle implements Cloneable {  
    int width, height;  
    Rectangle(int width, int height) {  
        this.width = width;  
        this.height = height;  
    }  
    public Object clone() {  
        ...  
    }  
}
```

Cloneable 인터페이스를
구현해야 합니다.

clone 메소드를
오버라이드해야 합니다.

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

• 복제 가능한 클래스를 만드는 방법

- 3) clone 메소드 안에서 Object 클래스의 clone 메소드를 호출합니다.

```
public Object clone() {  
    ...  
    return super.clone();  
    ...  
}
```

슈퍼클래스의 clone 메소드를 호출해서 그 결과를 리턴합니다.

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

• 복제 가능한 클래스를 만드는 방법

- 4) clone 메소드 호출 부분을 try 문으로 묶어서 CloneNotSupportedException을 처리합니다.

```
public Object clone() {  
    try {  
        return super.clone();  
    }  
    catch (CloneNotSupportedException e) {  
        return null;  
    }  
}
```

슈퍼클래스의 clone
메소드가 발생하는
익셉션을 처리해야 합니다.

Object 클래스

02. Object 클래스의 메소드들

clone 메소드

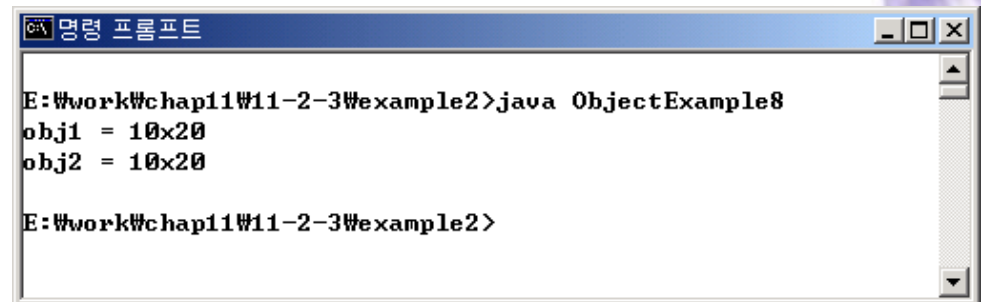
- [예제 11-10] clone 메소드의 오버라이딩 예

복제 가능한 직사각형 클래스

```
1 class Rectangle implements Cloneable {
2     int width, height;
3     Rectangle(int width, int height) {
4         this.width = width;
5         this.height = height;
6     }
7     public Object clone() {
8         try {
9             return super.clone();
10        }
11        catch (CloneNotSupportedException e) {
12            return null;
13        }
14    }
15 }
```

직사각형 클래스를 사용하는 프로그램

```
1 class ObjectExample8 {
2     public static void main(String args[]) {
3         Rectangle obj1 = new Rectangle(10, 20);
4         Rectangle obj2 = (Rectangle) obj1.clone();
5         System.out.println("obj1 = " + obj1.width + "x" + obj1.height);
6         System.out.println("obj2 = " + obj2.width + "x" + obj2.height);
7     }
8 }
```



```
E:\work\chap11\11-2-3\example2>java ObjectExample8
obj1 = 10x20
obj2 = 10x20

E:\work\chap11\11-2-3\example2>
```

Object 클래스

02. Object 클래스의 메소드들

finalize 메소드

• finalize 메소드 : 객체가 제거되기 전에 자동으로 호출되는 메소드

하지만 자바에는 객체를 제거하는 명령문이 없습니다.



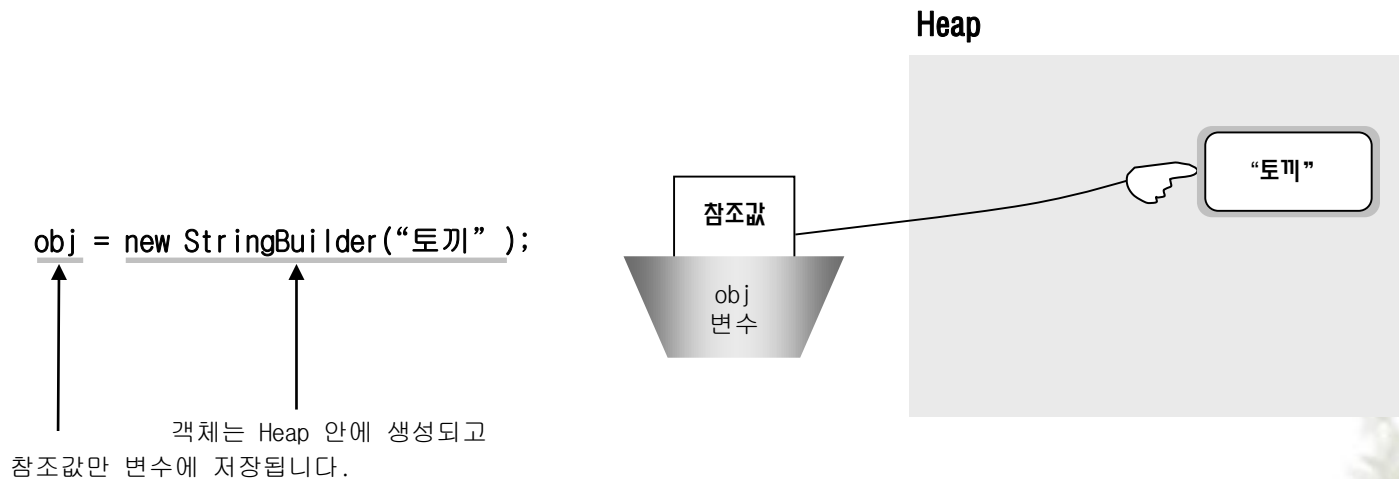
그러면 객체는 언제, 어떻게 제거될까요?

Object 클래스

02. Object 클래스의 메소드들

객체가 제거되는 방법

- 1) 객체를 생성하면 힙(heap)에 저장됩니다.

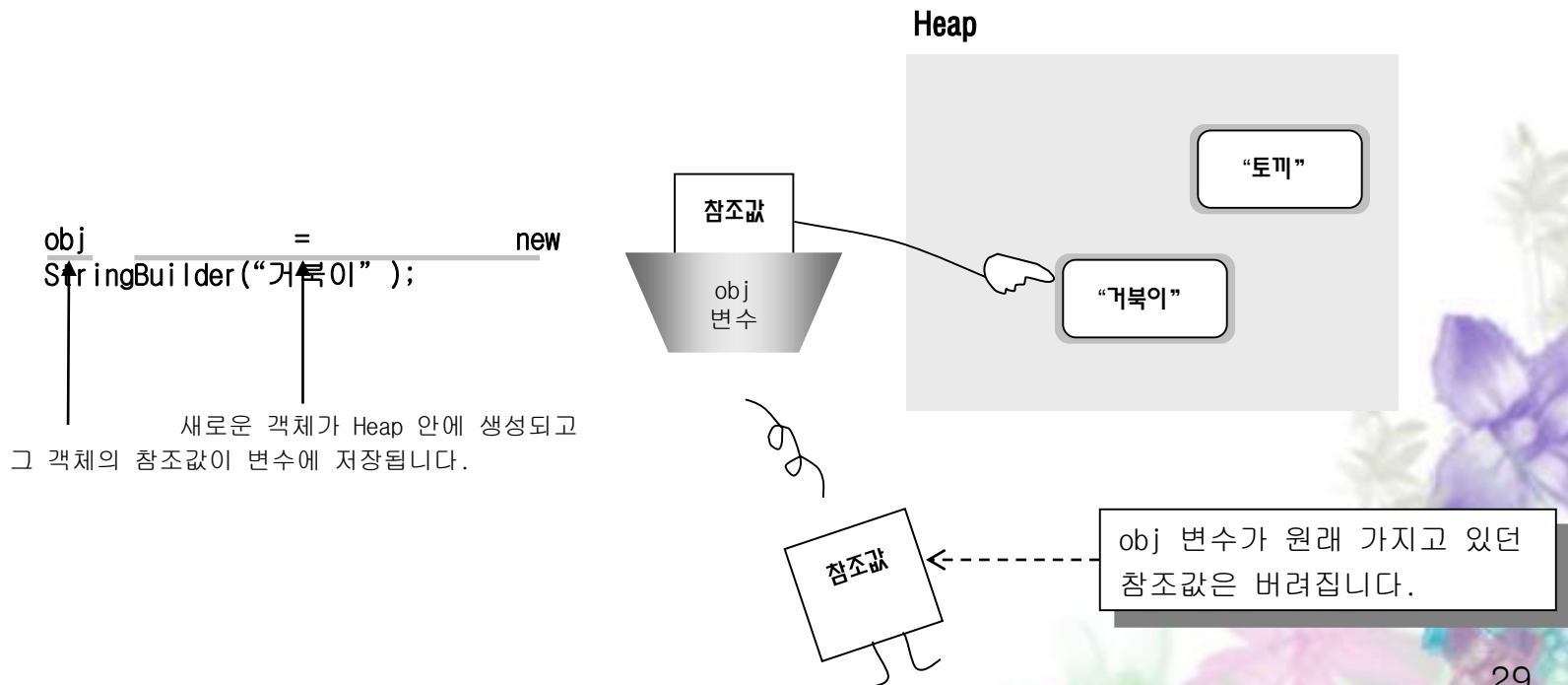


Object 클래스

02. Object 클래스의 메소드들

객체가 제거되는 방법

- 2) 참조값을 잃은 객체는 객체는 더 이상 사용될 수 없습니다.

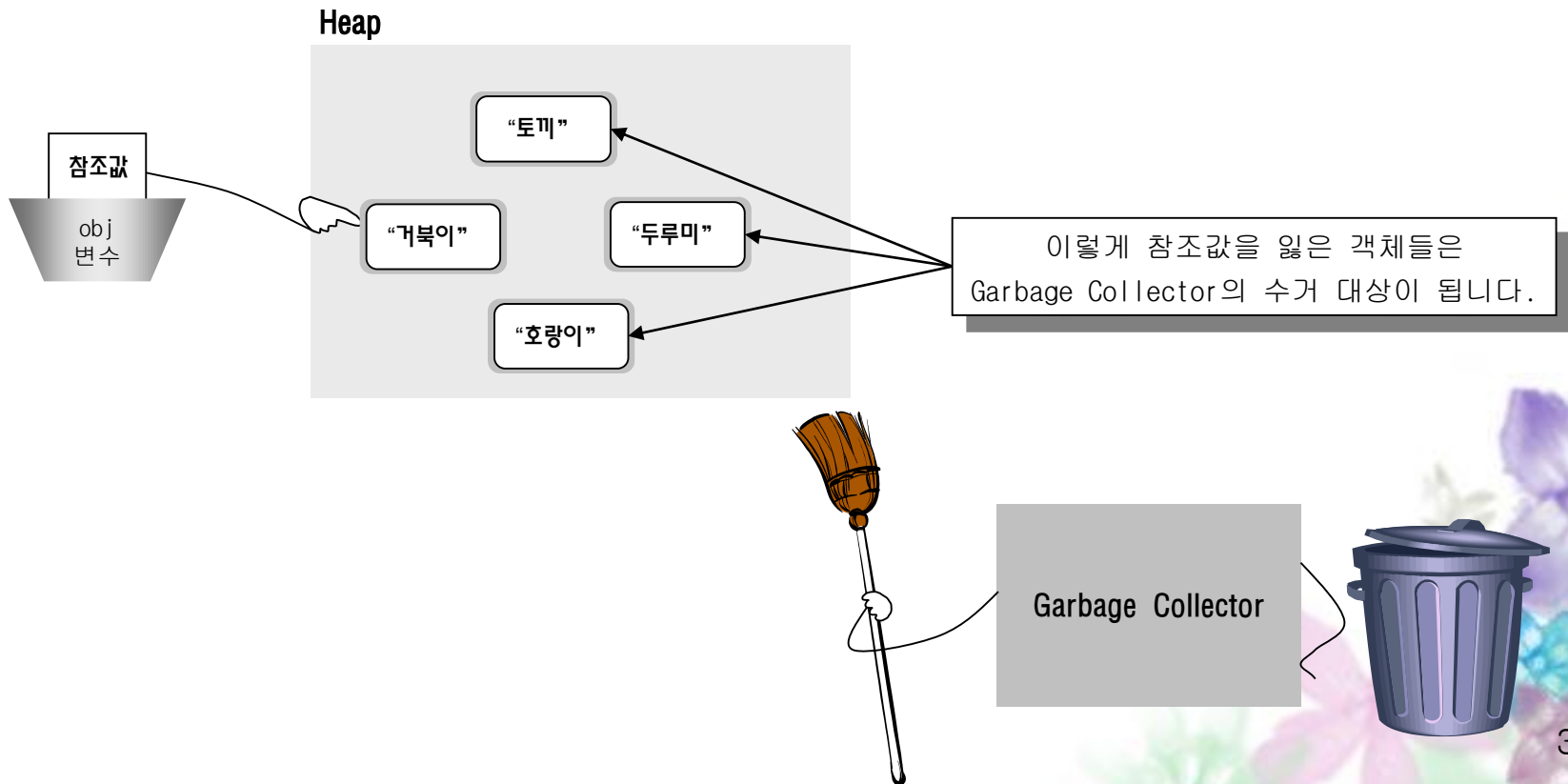


Object 클래스

02. Object 클래스의 메소드들

객체가 제거되는 방법

- 3) 가비지 컬렉터가 주기적으로 그런 객체들을 제거합니다.



Object 클래스

02. Object 클래스의 메소드들

finalize 메소드의 필요성

- [예제 11-11] 시스템 자원을 사용하는 클래스

```
1  class ResourceUser {  
2      ResourceUser() {  
3          ... ----- 생성자 안에서 시스템 자원을 할당받습니다.  
4      }  
5      void use() {  
6          ... ----- 할당받은 시스템 자원을 사용합니다  
7      }  
8      void release() {  
9          ... ----- 시스템 자원의 할당을 해제합니다.  
10     }  
11 }
```

Object 클래스

02. Object 클래스의 메소드들

finalize 메소드의 사용 예

- [예제 11-12] finalize 메소드를 오버라이드한 클래스의 예

```
1  class ResourceUser {  
2      ResourceUser() {  
3          ... ----- 생성자 안에서 시스템 자원을 할당받습니다.  
4      }  
5      void use() {  
6          ... ----- 할당받은 시스템 자원을 사용합니다  
7      }  
8      protected void finalize() {  
9          ... ----- 시스템 자원의 할당을 해제합니다.  
10     }  
11 }
```


Object 클래스

02. Object 클래스의 메소드들

getClass 메소드

- `getClass` 메소드 : 객체가 속하는 클래스의 정보를 알아내는 메소드
- 사용 방법
 - 1) `getClass` 메소드를 호출합니다.

```
Class cls = obj.getClass();
```

↑
객체가 속하는
클래스의 정보를 리턴

Object 클래스

02. Object 클래스의 메소드들

getClass 메소드

•• 사용 방법

- 2) 리턴된 Class 객체에 대해 get-메소드를 호출합니다.

```
String str = cls.getName();
```

↑
클래스의 이름을 리턴

```
Class superCls = cls.getSuperclass();
```

↑
슈퍼클래스의 정보를 리턴

```
Field field[] = cls.getDeclaredFields();
```

↑
클래스에 선언되어 있는
필드 정보를 가져오는 메소드

```
Method method[] = cls.getDeclaredMethods();
```

↑
클래스에 선언되어 있는
메서드 정보를 가져오는 메소드

Object 클래스

02. Object 클래스의 메소드들

getClass 메소드

- [예제 11-13] getClass 메소드의 사용 예

직사각형 클래스를 사용하는 프로그램

```
1  import java.lang.reflect.Field;
2  import java.lang.reflect.Method;
3  class ObjectExample9 {
4      public static void main(String args[]) {
5          Rectangle obj = new Rectangle(10, 20);
6          Class cls = obj.getClass();
7          String name = cls.getName();
8          System.out.println("클래스 이름: " + name);
9          Class superCls = cls.getSuperclass();
10         String superName = superCls.getName();
11         System.out.println("슈퍼클래스 이름: " + superName);
12         Field field[] = cls.getDeclaredFields();
13         System.out.println("필드: ");
14         for (int cnt = 0; cnt < field.length; cnt++)
15             System.out.println("    " + field[cnt]);
16         Method method[] = cls.getDeclaredMethods();
17         System.out.println("메소드: ");
18         for (int cnt = 0; cnt < method.length; cnt++)
19             System.out.println("    " + method[cnt]);
20     }
21 }
```

직사각형 클래스

```
1  class Rectangle {
2      int width, height;
3      Rectangle(int width, int height) {
4          this.width = width;
5          this.height = height;
6      }
7      int getArea() {
8          return width * height;
9      }
10 }
```

```
명령 프롬프트
E:\work\chap11\11-2-5>java ObjectExample9
클래스 이름: Rectangle
슈퍼클래스 이름: java.lang.Object
필드:
    int Rectangle.width
    int Rectangle.height
메서드:
    int Rectangle.getArea()
E:\work\chap11\11-2-5>
```

영

어

하

시

다

^^!