Java

네스티드 클래스와 네스티드 인터페이스

네스티드 클래스와 네스티드 인터페이스에 대하여

네스티드 클래스의 선언 방법과 이용 방법

네스티드 인터페이스의 선언 방법과 이용 방법

01. 네스티드 클래스와 네스티드 인터페이스

네스티드 클래스란?

•• 네스티드 클래스의 예

– – 다음 성적표를 클래스로 표현하는 경우를 가정합시다

성적표	성적표	성적표
이름: 임꺽정	이름: 성춘향	이름: 홍길동
국어: 70	국어: 95	국어: 100
영어: 50	영어: 99	영어: 95
수학: 80	수학: 75	수학: 88

이렇게 표현할 수 있습니다.

```
class ExamResult {    // 성적표 클래스
    String name;      // 이름
    ItemResult result[]; // 과목별 성적
    ExamResult() {        // 생성자
        result = new ItemResult[3];
    }
}
```

```
class ItemResult {   // 과목별 성적 클래스
    String subject; // 과목명
    int points;     // 점수
}
```

하지만 이 클래스의 존재 이유가 오로지 ExamResult 클래스를 위한 것이라면?

01. 네스티드 클래스와 네스티드 인터페이스

네스티드 클래스란?

•• 네스티드 클래스의 예

네스티드 클래스(nested class)로 선언하는 것이 좋습니다.

```
class ExamResult {     // 성적표 클래스
   String name;
     ItemResult result[];
    class ItemResult {
         String subject;
         int points;
    }
}
```

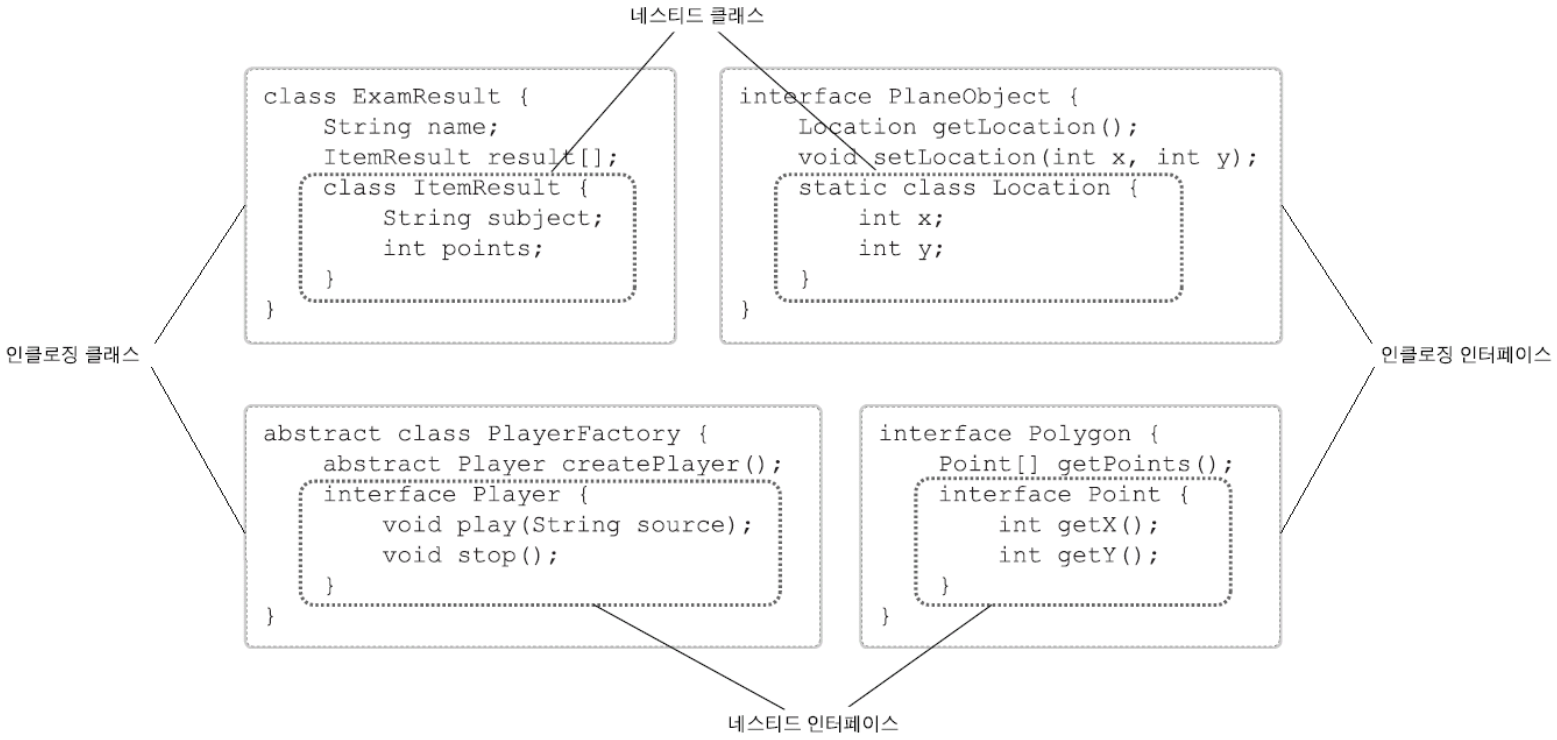
네스티드 클래스

네스티드 클래스와 네스티드 인터페이스

01. 네스티드 클래스와 네스티드 인터페이스

네스티드 클래스와 네스티드 인터페이스

•• 용어 설명

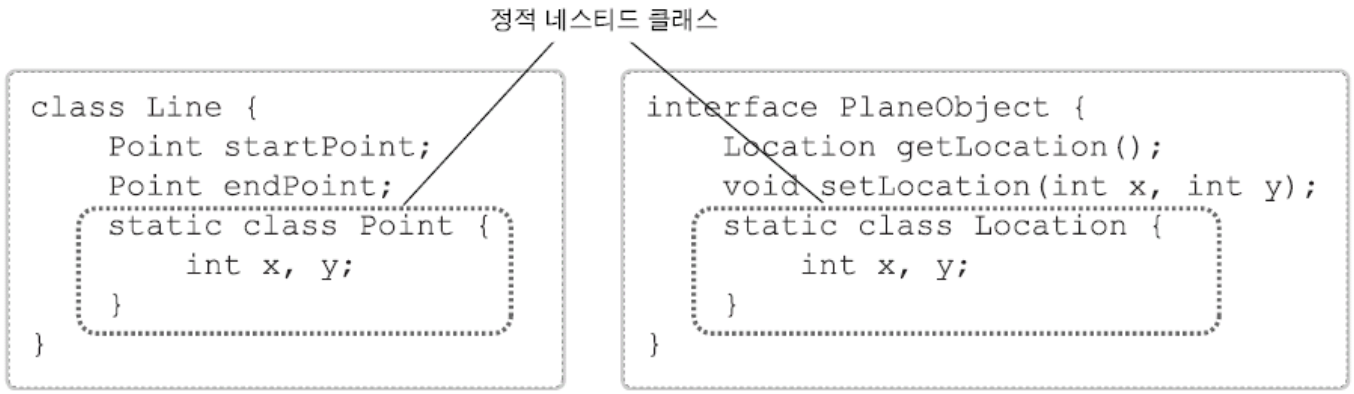


네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

네스티드 클래스의 종류

- 정적 네스티드 클래스



네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

네스티드 클래스의 종류

•• 이너 클래스

```
class ExamResult {
    String name;
    ItemResult result[];
    class ItemResult {
        String subject;
        int points;
    }
}
```

이너 클래스

02. 네스티드 클래스의 선언과 이용

네스티드 클래스의 종류

•• 로컬 이너 클래스

```
class MyProgram {
    public static void main(String args[]) {
        class NamedValue {
            String name;
            int value;
        }
        NamedValue obj = new NamedValue();
            .
            .
            .
    }
}
```

로컬 이너 클래스

02. 네스티드 클래스의 선언과 이용

이너 클래스

•• 다음과 같은 장바구니 데이터를 클래스로 표현하는 경우를 가정해 봅시다.

상품명	수량	단가	금액
초콜렛	3	1000	3000
케이크	1	25000	25000
샴페인	1	7000	7000
총계			35000

이런 데이터들은 이너 클래스로 선언하는 것이 좋습니다.

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

이너 클래스

- [예제 16-1] 장바구니 클래스와 상품 항목 클래스(이너 클래스)

```
1   import java.util.ArrayList;
2   class Cart {     // 장바구니 클래스
3       ArrayList<Item> list = new ArrayList<Item>();
4       void addItem(String name, int num, int unitPrice) {
5           list.add(new Item(name, num, unitPrice));    --------  이너 클래스의 생성자 호출
6       }
7       void removeItem(int index) {
8           list.remove(index);
9       }
10      int getItemNum() {
11          return list.size();
12      }
13      Item getItem(int index) {
14          return list.get(index);
15      }
16      int getTotalPrice() {
17          int total = 0;
18          for (Item item : list)
19              total += item.getPrice();     ---------------  이너 클래스의 메소드 호출
20          return total;
21      }
22      void chnageItemNumber(int index, int num) {
23          Item item = list.get(index);
24          item.num = num;    -------------------------  이너 클래스의 필드 사용
25      }
26      class Item {     // 상품 항목 클래스
27          String name;
28          int num;
29          int unitPrice;
30          Item(String name, int num, int unitPrice) {
31              this.name = name;
32              this.num = num;                          }  이너 클래스
33              this.unitPrice = unitPrice;
34          }
35          int getPrice() {
36              return num * unitPrice;
37          }
38      }
39  }
```

```
명령 프롬프트

E:₩work₩chap16₩16-2-1₩example1>javac Cart.java

E:₩work₩chap16₩16-2-1₩example1>dir
 E 드라이브의 볼륨: 로컬 디스크
 볼륨 일련 번호: 305B-17D4

 E:₩work₩chap16₩16-2-1₩example1 디렉터리

2006-05-18  05:06p      <DIR>          .
2006-05-18  05:06p      <DIR>          ..
2006-05-18  05:07p                 1,000 Cart.java
2006-05-18  05:07p                   509 Cart$Item.class   ---- 상품 항목 클래스의 컴파
2006-05-18  05:07p                 1,082 Cart.class
               3개 파일               2,591 바이트
               2 디렉터리   1,651,826,688 바이트 남음

E:₩work₩chap16₩16-2-1₩example1>
```

장바구니 클래스의 컴파일

02. 네스티드 클래스의 선언과 이용

이너 클래스

•• 이너 클래스의 사용 방법

- 1) 이너 클래스 이름 앞에 인클로징 클래스 이름을 붙여야 합니다.

```
Cart.Item item = cart.getItem(cnt);
```

인클로징 클래스의 이름 — Cart

이너 클래스 이름 — Item

02. 네스티드 클래스의 선언과 이용

이너 클래스

•• 이너 클래스의 사용 방법

– 2) 필드, 메소드의 사용 방법은 일반 클래스와 마찬가지입니다.

```
String str = item.name;
```

이너 클래스 객체 — 필드 이름

```
int num = item.getPrice();
```

이너 클래스 객체 — 메소드 이름

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

이너 클래스

- [예제 16-2] 장바구니 클래스와 상품 항목 클래스를 사용하는 프로그램 (1)

```
1   class NestedClassExample1 {
2       public static void main(String args[]) {
3           Cart cart = new Cart();
4           cart.addItem("쵸콜렛", 3, 1000);
5           cart.addItem("케이크", 1, 25000);
6           cart.addItem("샴페인", 1, 7000);
7           printCart(cart);
8       }
9       static void printCart(Cart cart) {
10          int num = cart.getItemNum();
11          System.out.println("      상품명    수량    단가     금액");
12          System.out.println("-----------------------------------");
13          for (int cnt = 0; cnt < num; cnt++) {
14              Cart.Item item = cart.getItem(cnt);
15              System.out.printf("%3d %5s %5d %7d %7d %n", cnt+1,
                        item.name, item.num, item.unitPrice, item.getPrice());
16          }
17          System.out.println("-----------------------------------");
18          System.out.printf("      총계                %10d %n", cart.getTotalPrice());
19      }
20  }
```

장바구니를 생성해서 세 개의 상품 항목을 추가합니다.

장바구니에 있는 상품 항목을 순서대로 가져와서 출력합니다.

```
명령 프롬프트

E:\work\chap16\16-2-1\example1>java NestedClas
      상품명    수량    단가     금액
-----------------------------------
  1   쵸콜렛       3    1000     3000
  2   케이크       1   25000    25000
  3   샴페인       1    7000     7000
-----------------------------------
      총계                    35000

E:\work\chap16\16-2-1\example1>_
```

02. 네스티드 클래스의 선언과 이용

- 이너 클래스
  - 인클로징 클래스 외부에서 이너 클래스 객체를 생성하는 방법

```
Cart.Item item = cart.new Item("꽃다발", 1, 50000);
```

인클로징 객체

이너 클래스의 생성자 호출

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

이너 클래스

- [예제 16-3] 장바구니 클래스와 상품 항목 클래스를 사용하는 프로그램 (2)

```
1   class NestedClassExample2 {
2       public static void main(String args[]) {
3           Cart cart = new Cart();
4           cart.addItem("초콜렛", 3, 1000);
5           cart.addItem("케이크", 1, 25000);
6           cart.addItem("샴페인", 1, 7000);
7           Cart.Item item = cart.new Item("꽃다발", 1, 50000);
8           cart.list.add(item);
9           printCart(cart);
10      }
11      static void printCart(Cart cart) {
12          int num = cart.getItemNum();
13          System.out.println("      상품명    수량    단가     금액");
14          System.out.println("---------------------------------");
15          for (int cnt = 0; cnt < num; cnt++) {
16              Cart.Item item = cart.getItem(cnt);
17              System.out.printf("%3d %5s %5d %7d %7d %n", cnt+1, item.name, item.num, item.unitPrice, item.getPrice());
18          }
19          System.out.println("---------------------------------");
20          System.out.printf("      총계              %10d %n", cart.getTotalPrice());
21      }
22  }
```

상품 항목 객체를 생성해서 장바구니에 추가합니다.

```
명령 프롬프트

E:₩work₩chap16₩16-2-1₩example1>java NestedCla
      상품명    수량    단가     금액
---------------------------------
  1   초콜렛       3    1000     3000
  2   케이크       1   25000    25000
  3   샴페인       1    7000     7000
  4   꽃다발       1   50000    50000
---------------------------------
      총계                    85000

E:₩work₩chap16₩16-2-1₩example1>
```

02. 네스티드 클래스의 선언과 이용

이너 클래스

•• new 키워드 앞에 인클로징 객체 이름을 써야 하는 이유는?

– -> 이너 클래스 객체와 인클로징 객체 사이의 연관 관계를 맺기 위해입니다.

```
Cart.Item item = cart.new Item("꽃다발", 1, 50000);
```

이렇게 생성된 이너 클래스 객체는
이 인클로징 객체와 연관지어 집니다.

인클로징 클래스 안에서 이너 클래스 객체를 생성할 경우에는
그 명령문이 속하는 인클로징 객체와 자동으로 연관 관계가
맺어지므로 이렇게 할 필요가 없습니다. (예제 16-1 참조)

02. 네스티드 클래스의 선언과 이용

이너 클래스

- [예제 16-4] 돼지 저금통 클래스와 입구 클래스(이너 클래스)

```
1   class PiggyBank {   // 돼지 저금통 클래스
2       int total;  <----------------
3       Slot slot;                   |
4       PiggyBank() {                |
5           total = 0;               |
6           slot = new Slot();       |
7       }                            |
8       class Slot {   // 입구 클래스 |
9           void put(int amount) {   |
10              total += amount;  ---
11          }
12      }
13  }
```

인클로징 클래스의 필드를 직접 사용합니다.

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

이너 클래스

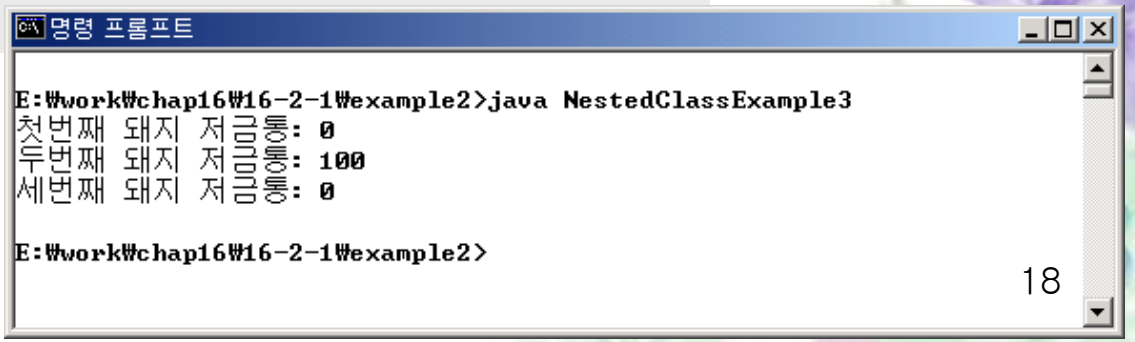
- [예제 16-5] 돼지 저금통 클래스를 사용하는 프로그램 (1)

```
1    class NestedClassExample3 {
2        public static void main(String args[]) {
3            PiggyBank bank1 = new PiggyBank();
4            PiggyBank bank2 = new PiggyBank();
5            PiggyBank bank3 = new PiggyBank();
6            bank2.slot.put(100);
7            System.out.println("첫번째 돼지 저금통: " + bank1.total);
8            System.out.println("두번째 돼지 저금통: " + bank2.total);
9            System.out.println("세번째 돼지 저금통: " + bank3.total);
10       }
11   }
```

세 개의 돼지 저금통 객체를 생성합니다.

두번째 돼지 저금통에 100원을 넣습니다.

세 개의 돼지 저금통에 있는 액수를 출력합니다.



02. 네스티드 클래스의 선언과 이용

이너 클래스

- [예제 16-6] 돼지 저금통 클래스를 사용하는 프로그램 (2)

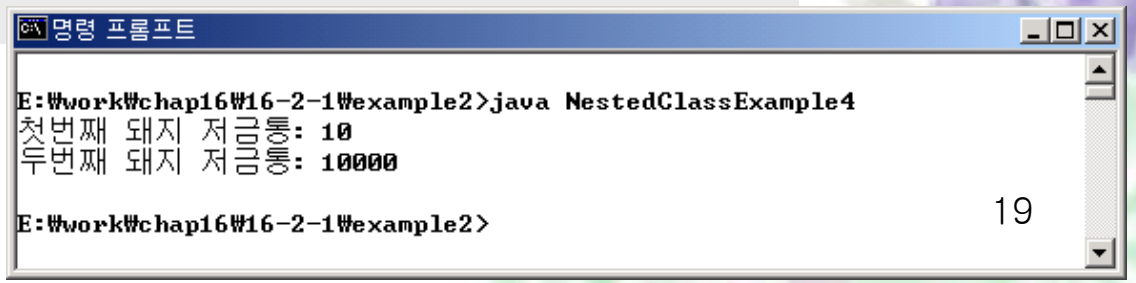
```
1   class NestedClassExample4 {
2       public static void main(String args[]) {
3           PiggyBank bank1 = new PiggyBank();
4           PiggyBank bank2 = new PiggyBank();
5           PiggyBank.Slot temp = bank1.slot;
6           bank1.slot = bank2.slot;
7           bank2.slot = temp;
8           bank1.slot.put(10000);
9           bank2.slot.put(10);
10          System.out.println("첫번째 돼지 저금통: " + bank1.total);
11          System.out.println("두번째 돼지 저금통: " + bank2.total);
12      }
13  }
```

두 개의 돼지 저금통 객체를 생성합니다.

돼지 저금통의 입구를 뒤바꿉니다.

돼지 저금통에 서로 다른 금액을 넣습니다.

두 돼지 저금통에 있는 액수를 출력합니다.



02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

•• 정적 네스티드 클래스(static nested class)

– - 필드, 메소드와 동일 수준으로 선언된 static 키워드가 붙은 네스티드 클래스

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

- [예제 16-7] 직선 클래스와 점 클래스(정적 네스티드 클래스)

```
1   class Line {   // 직선 클래스
2       Point point1, point2;
3       Line(int x1, int y1, int x2, int y2) {
4           point1 = new Point(x1, y1);
5           point2 = new Point(x2, y2);
6       }
7       void move(int offsetX, int offsetY) {
8           point1.x += offsetX;
9           point1.y += offsetY;
10          point2.x += offsetX;
11          point2.y += offsetY;
12      }
13      static class Point {   // 점 클래스
14          int x, y;
15          Point(int x, int y) {
16              this.x = x;
17              this.y = y;
18          }
19      }
20  }
```

정적 네스티드 클래스의 생성자 호출

정적 네스티드 클래스의 필드 사용

정적 네스티드 클래스

```
명령 프롬프트

E:₩work₩chap16₩16-2-2₩example1>javac Line.java

E:₩work₩chap16₩16-2-2₩example1>dir
 E 드라이브의 볼륨: 로컬 디스크
 볼륨 일련 번호: 305B-17D4

 E:₩work₩chap16₩16-2-2₩example1 디렉터리

2006-05-19  03:59p      <DIR>          .
2006-05-19  03:59p      <DIR>          ..
2006-05-19  04:01p                   315 Line$Point.class
2006-05-19  04:01p                   519 Line.class
2006-05-19  04:00p                   483 Line.java
               3개 파일           1,317 바이트
               2 디렉터리   1,672,830,976 바이트 남음

E:₩work₩chap16₩16-2-2₩example1>_
```

Point 클래스의

Line 클래스의

02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

•• 인클로징 클래스 외부에서 정적 네스티드 클래스를 사용하는 방법

```
Line.Point point = line.point1;
```

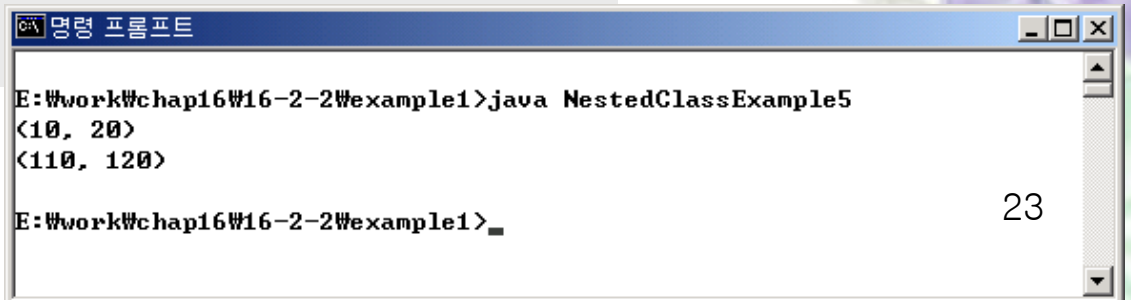
인클로징 클래스 이름 — 정적 네스티드 클래스 이름

02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

- [예제 16-8] 직선 클래스와 점 클래스를 사용하는 프로그램

```
1    class NestedClassExample5 {
2        public static void main(String args[]) {
3            Line line = new Line(0, 0, 100, 100);
4            line.move(10, 20);
5            printPoint(line.point1);
6            printPoint(line.point2);
7        }
8        static void printPoint(Line.Point point) {
9            System.out.printf("(%d, %d) %n", point.x, point.y);
10       }
11   }
```



02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

•• 인클로징 클래스 외부에서 정적 네스티드 클래스의 객체를 생성하는 방법

```
Line.Point point = new Line.Point(100, 200);
```

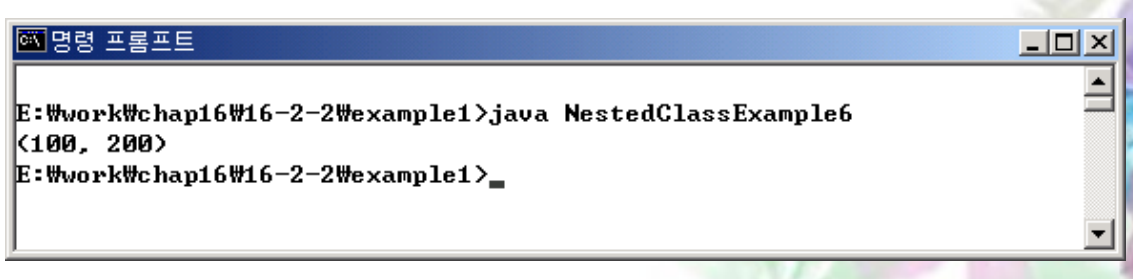
인클로징 클래스의 이름 — 정적 네스티드 클래스의 이름

02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

- [예제 16-9] 정적 네스티드 클래스의 객체를 생성하는 프로그램

```
1    class NestedClassExample6 {
2        public static void main(String args[]) {
3            Line.Point point = new Line.Point(100, 200);  ------- 정적 네스티드 클래스의 객체 생성
4            System.out.printf("(%d, %d)", point.x, point.y);
5        }
6    }
```



02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

- [예제 16-10] 평면 도형 인터페이스와 위치 클래스(정적 네스티드 클래스)

```
1    interface PlaneObject {     // 평면 도형 인터페이스
2        Location getLocation();
3        void setLocation(int x, int y);
4        static class Location {     // 위치 클래스
5            int x, y;
6            Location(int x, int y) {
7                this.x = x;
8                this.y = y;
9            }
10       }
11   }
```

} 정적 네스티드 클래스

02. 네스티드 클래스의 선언과 이용

정적 네스티드 클래스

- [예제 16-11] 평면 도형 인터페이스를 구현하는 사각형 클래스

```
1   class Rectangle implements PlaneObject {     // 사각형 클래스
2       Location location;
3       int width, height;
4       Rectangle(int x, int y, int width, int height) {
5           this.location = new Location(x, y);
6           this.width = width;
7           this.height = height;
8       }
9       public Location getLocation() {
10          return location;
11      }
12      public void setLocation(int x, int y) {
13          location.x = x;
14          location.y = y;
15      }
16  }
```

PlaneObject 인터페이스로부터 상속받은 정적 네스티드 클래스의 생성자 호출

정적 네스티드 클래스의 필드 사용

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

•• 로컬 이너 클래스(local inner class)

– - 메소드 본체 안에 선언된 네스티드 클래스

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

- [예제 16-12] 연락처 프로그램과 연락처 클래스

이런 클래스들이 있다고 가정합시다.

연락처 클래스

```
1       class ContactInfo {
2           String address;
3           String phoneNo;
4           ContactInfo(String address, String phoneNo) {
5               this.address = address;
6               this.phoneNo = phoneNo;
7           }
8       }
```

연락처 프로그램

```
1       import java.util.HashMap;
2       class ContactInfoExample {
3           public static void main(String args[]) {
4               HashMap<String, ContactInfo> hashtable = new HashMap<String, ContactInfo>();
5               hashtable.put("이순희", new ContactInfo("서울시 강남구", "02-547-0000"));
6               hashtable.put("한지영", new ContactInfo("서울시 성북구", "02-920-0000"));
7               hashtable.put("박철규", new ContactInfo("경기도 고양시", "031-915-0000"));
8               ContactInfo obj = hashtable.get("한지영");
9               System.out.println("<한지영의 연락처>");
10              System.out.println("주소:" + obj.address);
11              System.out.println("전화번호:" + obj.phoneNo);
12          }
13      }
```

이 클래스가 오로지 왼쪽 클래스의 main 메소드 내에서만 필요하다면?

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

- [예제 16-13] 연락처 프로그램과 연락처 클래스(로컬 이너 클래스)

```
1   import java.util.HashMap;
2   class NestedClassExample7 {
3       public static void main(String args[]) {
4           class ContactInfo {
5               String address;
6               String phoneNo;
7               ContactInfo(String address, String phoneNo) {
8                   this.address = address;
9                   this.phoneNo = phoneNo;
10              }
11          }
12          HashMap<String, ContactInfo> hashtable = new HashMap<String, ContactInfo>();
13          hashtable.put("이순희", new ContactInfo("서울시 강남구", "02-547-0000"));
14          hashtable.put("한지영", new ContactInfo("서울시 성북구", "02-920-0000"));
15          hashtable.put("박철규", new ContactInfo("경기도 고양시", "031-915-0000"));
16          ContactInfo obj = hashtable.get("한지영");
17          System.out.println("<한지영의 연락처>");
18          System.out.println("주소:" + obj.address);
19          System.out.println("전화번호:" + obj.phoneNo);
20      }
21  }
```

로컬 이너 클래스

```
명령 프롬프트

E:\work\chap16\16-2-3\example2>java NestedClassExample7
<한지영의 연락처>
주소:서울시 성북구
전화번호:02-920-0000

E:\work\chap16\16-2-3\example2>
```

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

- [예제 16-14] 메시지 송신 클래스와 서브클래스들

메시지 송신 클래스

```
1   abstract class MessageSender {
2       abstract void send(String message);
3   }
```

이런 클래스들이 있다고 가정합시다.

이메일 송신 클래스

```
1   class EMailSender extends MessageSender {
2       String sender;
3       String receiver;
4       EMailSender(String sender, String receiver) {
5           this.sender = sender;
6           this.receiver = receiver;
7       }
8       void send(String message) {
9           System.out.println("보내는 사람:" + sender);
10          System.out.println("받는 사람:" + receiver);
11          System.out.println("내용:" + message);
12          System.out.println();
13      }
14  }
```

문자 메시지 송신 클래스

```
1   class SMSSender extends MessageSender {
2       String phoneNo;
3       String responsePhoneNo;
4       SMSSender(String phoneNo, String responsePhoneNo) {
5           this.phoneNo = phoneNo;
6           this.responsePhoneNo = responsePhoneNo;
7       }
8       void send(String message) {
9           System.out.println("전화번호:" + phoneNo);
10          System.out.println("내용:" + message);
11          System.out.println("회신전화번호:" + responsePhoneNo);
12          System.out.println();
13      }
14  }
```

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

•• 다음과 같은 클래스가 추가로 필요하다면?

```
class SatelliteSender extends MessageSender {
 void send(String message) {
  System.out.println("발신: 마이다스");
  System.out.println("수신: 빌 게이츠");
  System.out.println("메시지:" + message);
  System.out.println();
 }
}
```

인공 위성을 통한 메시지 송신 클래스

그리고 이 클래스가 특정 프로그램의 특정 메소드 내에서만 필요하다면?

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

- [예제 16-15] 다른 클래스를 상속 받는 로컬 이너 클래스의 예

```
1   class NestedClassExample8 {
2       public static void main(String args[]) {
3           class SatelliteSender extends MessageSender {
4               void send(String message) {
5                   System.out.println("발신: 마이다스");
6                   System.out.println("수신: 빌 게이츠");
7                   System.out.println("메시지: " + message);
8                   System.out.println();
9               }
10          }
11          SatelliteSender obj = new SatelliteSender();
12          obj.send("굿 모닝");
13      }
14  }
```

로컬 이너 클래스의 선언

로컬 이너 클래스의 객체 생성

로컬 이너 클래스의 메소드 호출

```
명령 프롬프트

E:₩work₩chap16₩16-2-3₩example3>java NestedClassExample
발신: 마이다스
수신: 빌 게이츠
메시지: 굿 모닝

E:₩work₩chap16₩16-2-3₩example3>
```

02. 네스티드 클래스의 선언과 이용

로컬 이너 클래스

•• 앞 프로그램은 다음과 같은 방법으로 더 간단하게 만들 수 있습니다.

```
class SatelliteSender extends MessageSender {
    void send(String message) {
        System.out.println("발신: 마이다스");
        System.out.println("수신: 빌 게이츠");
        System.out.println("메시지: " + message);
        System.out.println();
    }
}
SatelliteSender obj = new SatelliteSender();
```

이 부분과 이 부분을 삭제합니다

슈퍼클래스 타입으로 변수를 선언합니다

슈퍼클래스의 생성자를 호출합니다

```
MessageSender obj = new MessageSender() {
    void send(String message) {
        System.out.println("발신: 마이다스");
        System.out.println("수신: 빌 게이츠");
        System.out.println("메시지: " + message);
        System.out.println();
    }
};
```

클래스 본체는 그대로 둡니다

마지막에 세미콜론을 써주어야 합니다

이런 이너 클래스를 이름 없는 이너 클래스(anonymous inner class)라고 부릅니다.

02. 네스티드 클래스의 선언과 이용

이름 없는 이너 클래스

- [예제 16-16] 이름 없는 이너 클래스의 예 (1)

```
1   class NestedClassExample9 {
2       public static void main(String args[]) {
3           MessageSender obj = new MessageSender() {
4               void send(String message) {
5                   System.out.println("발신: 마이다스");
6                   System.out.println("수신: 빌 게이츠");
7                   System.out.println("메시지: " + message);
8                   System.out.println();
9               }
10          };
11          obj.send("굿 모닝");
12      }
13  }
```

이름 없는 이너 클래스의 선언 및 객체 생성

이름 없는 이너 클래스의 메소드 호출

```
명령 프롬프트
E:₩work₩chap16₩16-2-3₩example4>java NestedClassExample9
발신: 마이다스
수신: 빌 게이츠
메시지: 굿 모닝

E:₩work₩chap16₩16-2-3₩example4>
```

02. 네스티드 클래스의 선언과 이용

이름 없는 이너 클래스

- [예제 16-17] 이름 없는 이너 클래스의 예 (2)

```
1   interface Player {
2       void play(String source);
3       void stop();
4   }
```

```
1   class NestedClassExample10 {
2       public static void main(String args[]) {
3           Player obj = new Player() {
4               public void play(String source) {
5                   System.out.println("플레이 시작: " + source);
6               }
7               public void stop() {
8                   System.out.println("플레이 종료" );
9               }
10          };
11          obj.play("LetItBe.mp3");
12          obj.stop();
13      }
14  }
```

Player 인터페이스를 구현하는 이름 없는 이너 클래스

이름 없는 이너 클래스의 메소드 호출

```
명령 프롬프트

E:\work\chap16\16-2-3\example5>java NestedClassExam
플레이 시작: LetItBe.mp3
플레이 종료

E:\work\chap16\16-2-3\example5>_
```

03. 네스티드 인터페이스의 선언과 이용

네스티드 인터페이스의 종류

•• 정적 네스티드 인터페이스 (1)

```
abstract class PlayerFactory {
    abstract Player createPlayer();
    static interface Player {
        void play(String source);
        void stop();
    }
}
```

```
interface Polygon {
    Point[] getPoints();
    static interface Point {
        int getX();
        int getY();
    }
}
```

정적 네스티드 인터페이스

네스티드 클래스와 네스티드 인터페이스

02. 네스티드 클래스의 선언과 이용

네스티드 인터페이스의 종류

•• 정적 네스티드 인터페이스 (2)

```
abstract class PlayerFactory {
    abstract Player createPlayer();
    interface Player {
        void play(String source);
        void stop();
    }
}
```

```
interface Polygon {
    Point[] getPoints();
    interface Point {
        int getX();
        int getY();
    }
}
```

static 키워드를 붙이지 않아도
정적 네스티드 인터페이스가 됩니다

네스티드 인터페이스는 정적 네스티드 인터페이스 한 종류만 있습니다.

네스티드 클래스와 네스티드 인터페이스

03. 네스티드 인터페이스의 선언과 이용

네스티드 인터페이스

- [예제 16-18] 네스티드 인터페이스를 포함하는 클래스와 서브클래스

```
1   abstract class PlayerFactory {
2       abstract Player createPlayer();
3       interface Player {
4           void play(String source);
5           void stop();
6       }
7   }
```

네스티드 인터페이스

상속

```
1   class MP3PlayerFactory extends PlayerFactory {
2       public Player createPlayer() {
3           return new MP3Player();
4       }
5       class MP3Player implements Player {
6           public void play(String source) {
7               System.out.println("플레이 시작: " + source);
8           }
9           public void stop() {
10              System.out.println("플레이 종료");
11          }
12      }
13  }
```

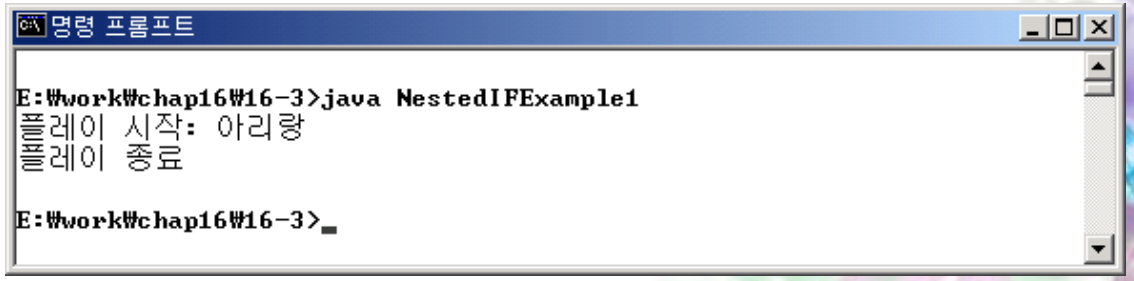
상속받은 네스티드 인터페이스를 구현하는 네스티드 클래스

03. 네스티드 인터페이스의 선언과 이용

네스티드 인터페이스

- [예제 16-19] MP3PlayerFactory 클래스를 사용하는 프로그램

```
1   class NestedIFExample1 {
2       public static void main(String args[]) {
3           MP3PlayerFactory factory = new MP3PlayerFactory();
4           PlayerFactory.Player player = factory.createPlayer();
5           player.play("아리랑");
6           player.stop();
7       }
8   }
```



열 공 합 시 다 ^^!