Java

객체의 직렬화

직렬화와 역직렬화에 대하여

직렬화 가능 클래스의 선언 방법

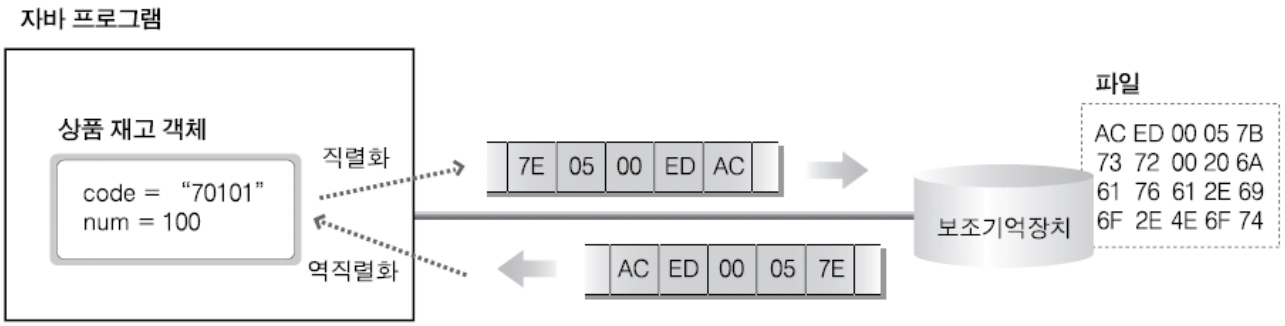
객체의 직렬화

01. 직렬화와 역직렬화에 대하여

직렬화와 역직렬화

•• 용어 설명

- **직렬화**(serialization) : 객체를 스트림으로 만드는 작업
- **역직렬화**(deserialization) : 스트림을 객체로 만드는 작업



객체의 직렬화

01. 직렬화와 역직렬화에 대하여

직렬화와 역직렬화

- [예제 17-1] 객체를 직렬화하는 프로그램과 역직렬화하는 프로그램

객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   import java.util.GregorianCalendar;
3   class ObjectOutputExample1 {
4       public static void main(String args[]) {
5           ObjectOutputStream out = null;
6           try {
7               out = new ObjectOutputStream( new FileOutputStream("output.da
8               out.writeObject(new GregorianCalendar(2006, 0, 14));
9               out.writeObject(new GregorianCalendar(2006, 0, 15));
10              out.writeObject(new GregorianCalendar(2006, 0, 16));
11          }
12          catch (IOException ioe) {
13              System.out.println("파일로 출력할 수 없습니다.");
14          }
15          finally {
16              try {
17                  out.close();
18              }
19              catch (Exception e) {
20              }
21          }
22      }
23  }
```

객체를 직렬화하는 부분

객체를 역직렬화하는 프로그램

```
1   import java.io.*;
2   import java.util.GregorianCalendar;
3   import java.util.Calendar;
4   class ObjectInputExample1 {
5       public static void main(String args[]) {
6           ObjectInputStream in = null;
7           try {
8               in = new ObjectInputStream(new FileInputStream("output.dat"));
9               while (true) {
10                  GregorianCalendar calendar = (GregorianCalendar) in.readObject();
11                  int year = calendar.get(Calendar.YEAR);
12                  int month = calendar.get(Calendar.MONTH) + 1;
13                  int date = calendar.get(Calendar.DATE);
14                  System.out.println(year + "/" + month + "/" + date);
15              }
16          }
17          catch (FileNotFoundException fnfe) {
18              System.out.println("파일이 존재하지 않습니다.");
19          }
20          catch (EOFException eofe) {
21              System.out.println("끝");
22          }
23          catch (IOException ioe) {
24              System.out.println("파일을 읽을 수 없습니다.");
25          }
26          catch (ClassNotFoundException cnfe) {
27              System.out.println("해당 클래스가 존재하지 않습니다.");
28          }
29          catch (Exception e) {
30          }
31      }
32  }
```

객체를 역직렬화하는 부분

```
명 프롬프트

ork₩chap17₩17-1>java ObjectOutputExample1

ork₩chap17₩17-1>java ObjectInputExample1
/1/14
/1/15
/1/16

ork₩chap17₩17-1>_
```

객체를 직렬화해서 파일에 저장합니다.

파일로부터 객체를 역직렬화해서 출력합니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 선언

JDK 라이브러리의 직렬화 가능 클래스 구분 방법

- java.io.Seiralizable 인터페이스를 구현하는 클래스는 직렬화 가능 클래스
- - java.io.Seiralizable 인터페이스를 구현하지 않는 클래스는 직렬화 불가능 클래스

직렬화 가능 클래스의 선언

- java.io.Seiralizable 인터페이스를 구현하는 것만으로 충분할까요?

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 선언

- [예제 17-2] 직렬화 가능 클래스를 만드는 예

직렬화 가능 클래스

```
1     class GoodsStock implements java.io.Serializable {
2         String code;
3         int num;
4         GoodsStock(String code, int num) {
5             this.code = code;
6             this.num = num;
7         }
8         void addStock(int num) {
9             this.num += num;
10        }
11        int subtractStock(int num) throws Exception {
12            if (this.num < num)
13                throw new Exception("재고가 부족합니다.");
14            this.num -= num;
15            return num;
16        }
17    }
```

객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 선언

- [예제 17-3] GoodsStock 객체를 직렬화하고 역직렬화하는 프로그램

GoodsStock 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample2 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new FileOutputStream("output
7               out.writeObject(new GoodsStock("70101", 100));
8               out.writeObject(new GoodsStock("70102", 50));
9               out.writeObject(new GoodsStock("70103", 200));
10          }
11          catch (IOException ioe) {
12              System.out.println("파일로 출력할 수 없습니다.");
13          }
14          finally {
15              try {
16                  out.close();
17              }
18              catch (Exception e) {
19              }
20          }
21      }
22  }
```

객체를 직렬화하는 부분

GoodsStock 객체를 역직렬화하는

```
1   import java.io.*;
2   class ObjectInputExample2 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output2.dat"));
7               while (true) {
8                   GoodsStock obj = (GoodsStock) in.readObject();
9                   System.out.println("상품코드:" + obj.code + "₩t상품수량:" +
10  obj.num);
11              }
12          }
13          catch (FileNotFoundException fnfe) {
14              System.out.println("파일이 존재하지 않습니다.");
15          }
16          catch (EOFException eofe) {
17              System.out.println("끝");
18          }
19          catch (IOException ioe) {
```

객체를 역직렬화 하는 부분

[직렬화가 불가능한 GoodsStock 클래스와 함께 컴

```
명령 프롬프트
E:₩work₩chap17₩17-2₩example1>javac GoodsStock.java

E:₩work₩chap17₩17-2₩example1>javac ObjectOutputExample2.java

E:₩work₩chap17₩17-2₩example1>javac ObjectInputExample2.java

E:₩work₩chap17₩17-2₩example1>java ObjectOutputExample2
파일로 출력할 수 없습니다.

E:₩work₩chap17₩17-2₩example1>
```

[직렬화가 가능한 GoodsStock 클래스와 함께 컴파일하고 실행했을 때의 결과]

```
명령 프롬프트
E:₩work₩chap17₩17-2₩example2>javac GoodsStock.java

E:₩work₩chap17₩17-2₩example2>javac ObjectOutputExample2.java

E:₩work₩chap17₩17-2₩example2>javac ObjectInputExample2.java

E:₩work₩chap17₩17-2₩example2>java ObjectOutputExample2

E:₩work₩chap17₩17-2₩example2>java ObjectInputExample2
상품코드:70101   상품수량:100
상품코드:70102   상품수량:50
상품코드:70103   상품수량:200
끝

E:₩work₩chap17₩17-2₩example2>_
```

컴파일도 정상적으로 되고,

프로그램의 실행도 정상적으로 됩니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 대상

•• 생성자, 메소드, 정적 필드는 직렬화 대상이 아닙니다.

```
class BBSItem implements java.io.Serializable {
    static int itemNum = 0;
    String writer;
    String passwd;
    String title;
    String content;
    BBSItem(String writer, String passwd,
            String title, String content) {
        this.writer = writer;
        this.passwd = passwd;
        this.title = title;
        this.content = content;
        itemNum++;
    }
    void modifyContent(String content, String passwd) {
        if (!passwd.equals(this.passwd))
            return;
        this.content = content;
    }
}
```

정적 필드는 직렬화 대상이 되지 않습니다

인스턴스 필드는 직렬화 대상이 됩니다

생성자는 직렬화 대상이 되지 않습니다

메소드도 직렬화의 대상이 되지 않습니다

직렬화 대상에서 제외시키고 싶은 인스턴스 필드가 있으면 어떻게 해야할까요?

02. 직렬화 가능 클래스의 선언 방법

직렬화 대상

- [예제 17-4] transient 필드를 포함하는 직렬화 가능 클래스의 예

```
1   class BBSItem implements java.io.Serializable {       // 게시물 클래스
2       static int itemNum = 0;     // 게시물의 수
3       String writer;              // 글쓴이
4       transient String passwd;    // 패스워드
5       String title;               // 제목
6       String content;             // 내용
7       BBSItem(String writer, String passwd, String title, String content) {
8           this.writer = writer;
9           this.passwd = passwd;
10          this.title = title;
11          this.content = content;
12          itemNum++;
13      }
14      void modifyContent(String content, String passwd) {
15          if (!passwd.equals(this.passwd))
16              return;
17          this.content = content;
18      }
19  }
```

객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 대상

- [예제 17-5] BBSItem 객체를 직렬화하고 역직렬화하는 프로그램

BBSItem 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample3 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new FileOutputStream("output3.dat"));
7               BBSItem obj = new BBSItem("최선희","sunshine","정모합시다",
                                          "이번주 주말 어떠세요?");
8               System.out.println("전체게시물의 수: " + obj.itemNum);
9               System.out.println("글쓴이: " + obj.writer);
10              System.out.println("패스워드: " + obj.passwd);
11              System.out.println("제목: " + obj.title);
12              System.out.println("내용: " + obj.content);
13              out.writeObject(obj);   ------ 객체를 직렬화하는 부분
14          }
15          catch (IOException ioe) {
16              System.out.println("파일로 출력할 수 없습니다.");
17          }
18          finally {
19              try {
20                  out.close();
21              }
22              catch (Exception e) {
23              }
24          }
25      }
26  }
```

```
명령 프롬프트

E:₩work₩chap17₩17-2-1₩example1>java ObjectOutputExample3
전체게시물의 수: 1
글쓴이: 최선희
패스워드: sunshine
제목: 정모합시다
내용: 이번주 주말 어떠세요?

E:₩work₩chap17₩17-2-1₩example1>java ObjectInputExample3
전체게시물의 수: 0
```

BBSItem 객체를 역직렬화하는 프로그램

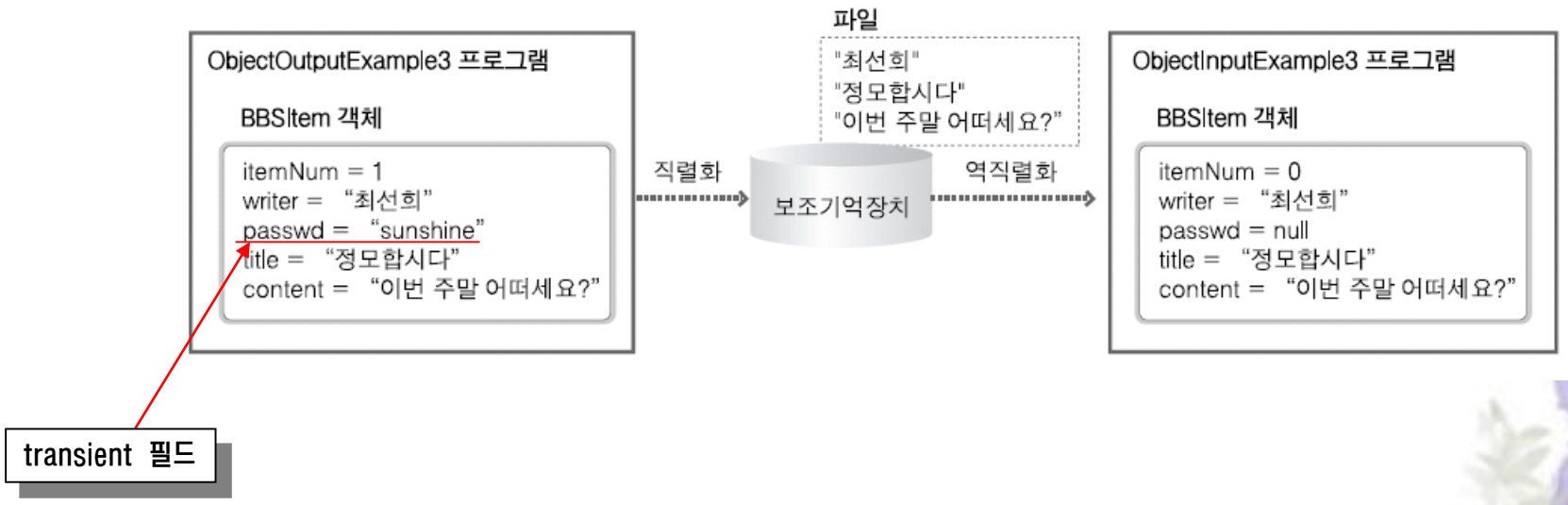
```
1   import java.io.*;
2   class ObjectInputExample3 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output3.dat"));
7               BBSItem obj = (BBSItem) in.readObject();   ----  객체를 역직렬화하는 부분
8               System.out.println("전체게시물의 수: " + obj.itemNum);
9               System.out.println("글쓴이: " + obj.writer);
10              System.out.println("패스워드: " + obj.passwd);
11              System.out.println("제목: " + obj.title);
12              System.out.println("내용: " + obj.content);
13          }
14          catch (FileNotFoundException fnfe) {
15              System.out.println("파일이 존재하지 않습니다.");
16          }
17          catch (EOFException eofe) {
18              System.out.println("끝");
19          }
20          catch (IOException ioe) {
21              System.out.println("파일을 읽을 수 없습니다.");
22          }
23          catch (ClassNotFoundException cnfe) {
24              System.out.println("해당 클래스가 존재하지 않습니다.");
25          }
26          finally {
27              try {
28                  in.close();
29              }
30              catch (Exception e) {
31              }
32          }
33      }
34  }
```

객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 대상

•• 직렬화 대상에서 제외되는 transient 필드



객체의 직렬화

직렬화 불가능한 필드 타입

- [예제 17-6] 직렬화 불가능 필드를 포함하는 직렬화 가능 클래스의 예

```
1   class BBSItem implements java.io.Serializable {       // 게시물
2   클래스
3       static int itemNum = 0;     // 게시물의 수
4       String writer;              // 글쓴이
5       transient String passwd;    // 패스워드
6       String title;               // 제목
7       String content;             // 내용
8       Object attachment;          // 첨부파일
9       BBSItem(String writer, String passwd, String title, String
10  content) {
11          this.writer = writer;
12          this.passwd = passwd;
13          this.title = title;
14          this.content = content;
15          itemNum++;
16      }
17      void modifyContent(String content, String passwd) {
18          if (!passwd.equals(this.passwd))
19              return;
20          this.content = content;
21      }
22      void addAttachment(Object attachment) {        // 파일을
23  첨부한다
```

객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 불가능한 필드 타입

- [예제 17-7] 게시물 클래스의 객체를 직렬화하고 역직렬화하는 프로그램

BBSItem 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample4 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new
7   FileOutputStream("output4.dat"));
                BBSItem obj = new BBSItem("이석영" , "moonlight", "자료
8   파일입니다.",
                                          "첨부 파일을 참고하십시오.");
9
10              obj.addAttachment(new Object());
11              System.out.println("전체게시물의 수: " + obj.itemNum);
12              System.out.println("글쓴이: " + obj.writer);
13              System.out.println("패스워드: " + obj.passwd);
14              System.out.println("제목: " + obj.title);
15              System.out.println("내용: " + obj.content);
16              System.out.println("첨부: " + obj.attachment);
17              out.writeObject(obj);
18          }
19          catch (IOException ioe) {
20              System.out.println("파일로 출력할 수 없습니다.");
21          }
22          finally {
23              try {
24                  out.close();
25              }
26              catch (Exception e) {
27              }
28          }
        }
    }
```

```
명령 프롬프트

E:\work\chap17\17-2-1\example2>java ObjectOutputExample4
전체게시물의 수: 1
글쓴이: 이석영
패스워드: moonlight
제목: 자료 파일입니다.
내용: 첨부 파일을 참고하십시오.
첨부: java.lang.Object@d9f9c3
파일로 출력할 수 없습니다.
```

에러 메시지

BBSItem 객체를 역직렬화하는

```
1   import java.io.*;
2   class ObjectInputExample4 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output4.dat"));
7               BBSItem obj = (BBSItem) in.readObject();
8               System.out.println("전체게시물의 수: " + obj.itemNum);
9               System.out.println("글쓴이: " + obj.writer);
10              System.out.println("패스워드: " + obj.passwd);
11              System.out.println("제목: " + obj.title);
12              System.out.println("내용: " + obj.content);
13              System.out.println("첨부: " + obj.attachment);
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (EOFException eofe) {
19              System.out.println("끝");
20          }
21          catch (IOException ioe) {
22              System.out.println("파일을 읽을 수 없습니다.");
23          }
24          catch (ClassNotFoundException cnfe) {
25              System.out.println("해당 클래스가 존재하지 않습니다.");
26          }
27          finally {
28              try {
29                  in.close();
30              }
31              catch (Exception e) {
32              }
33          }
34      }
35  }
```

객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 불가능한 필드 타입

- [예제 17-7] 게시물 클래스의 객체를 직렬화하고 역직렬화하는 프로그램

BBSItem 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample4 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new
7   FileOutputStream("output4.dat"));
                BBSItem obj = new BBSItem("이석영", "moonlight", "자료
8   파일입니다.",
9                                         "첨부 파일을 참고하십시오.");
10              obj.addAttachment("모카자바 500g 15500원");
11              System.out.println("전체게시물의 수: " + obj.itemNum);
12              System.out.println("글쓴이: " + obj.writer);
13              System.out.println("패스워드: " + obj.passwd);
14              System.out.println("제목: " + obj.title);
15              System.out.println("내용: " + obj.content);
16              System.out.println("첨부: " + obj.attachment);
17              out.writeObject(obj);
18          }
19          catch (IOException ioe) {
20              System.out.println("파일로 출력할 수 없습니다.");
21          }
22          finally {
23              try {
24                  out.close();
25              }
26              catch (Exception e) {
27              }
28          }
        }
    }
```

```
명령 프롬프트

E:\work\chap17\17-2-1\example2>java ObjectOutputExample4
전체게시물의 수: 1
글쓴이: 이석영
패스워드: moonlight
제목: 자료 파일입니다.
내용: 첨부 파일을 참고하십시오.
```

BBSItem 객체를 역직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectInputExample4 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output4.dat"));
7               BBSItem obj = (BBSItem) in.readObject();
8               System.out.println("전체게시물의 수: " + obj.itemNum);
9               System.out.println("글쓴이: " + obj.writer);
10              System.out.println("패스워드: " + obj.passwd);
11              System.out.println("제목: " + obj.title);
12              System.out.println("내용: " + obj.content);
13              System.out.println("첨부: " + obj.attachment);
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (EOFException eofe) {
19              System.out.println("끝");
20          }
21          catch (IOException ioe) {
22              System.out.println("파일을 읽을 수 없습니다.");
23          }
24          catch (ClassNotFoundException cnfe) {
25              System.out.println("해당 클래스가 존재하지 않습니다.");
26          }
27          finally {
28              try {
29                  in.close();
30              }
31              catch (Exception e) {
32              }
33          }
34      }
35  }
```

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 다음과 같은 희소 배열(sparse array)를 직렬화하는 경우

인덱스	0	1	2	3	4	5	6	7	8	9
0	0	2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	5	0	0	0	0
5	0	2	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	7	0	0
8	0	0	0	0	21	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

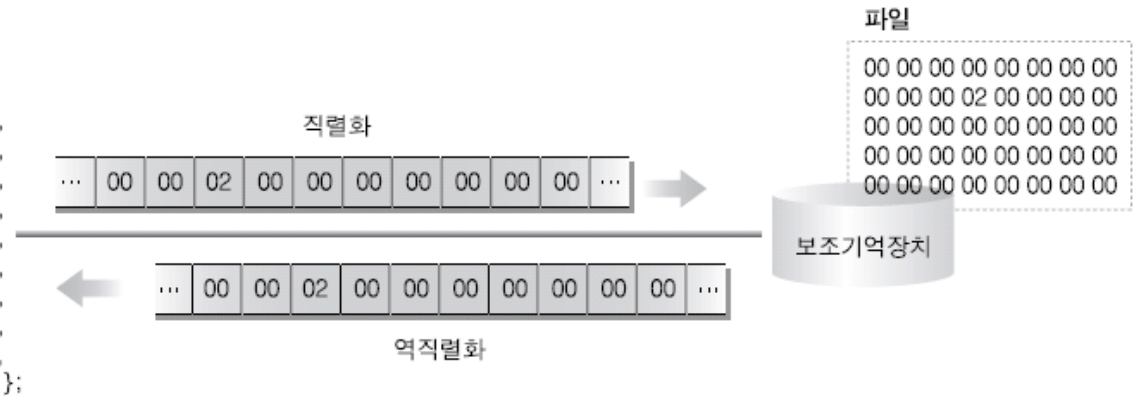
객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 디폴트 직렬화 메커니즘을 사용하면 비효율적입니다.

```
int arr[][] = {{ 0, 2, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 5, 0, 0, 0, 0},
               { 0, 2, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 7, 0, 0},
               { 0, 0, 0, 0, 21, 0, 0, 0, 0, 0},
               { 0, 0, 0, 0,  0, 0, 0, 0, 0, 0}};
```



02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 더 효율적인 방법은?

```
int arr[][] = { { 0, 2, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 5, 0, 0, 0, 0 },
                { 0, 2, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 7, 0, 0 },
                { 0, 0, 0, 0, 21, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };
```

이런 데이터만 직렬화하는 것이 경제적입니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

- [예제 17-8] 2차원 배열 필드를 포함하는 클래스

이런 클래스가 있다고 가정합시다.

```
1     import java.io.*;
2     class DistrChart implements Serializable {
3         int arr[][];
4         DistrChart() {
5             arr = new int[10][10];
6         }
7     }
```

이 배열이 희소 배열로 사용된다고 가정합시다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 직렬화 메소드의 작성 방법

- 1) writeObject라는 이름의 메소드를 선언합니다.

```
class DistrChart implements Serializable {
    ...
    private void writeObject(ObjectOutputStream out)
                                    throws IOException {
        ...
    }
}
```

직렬화 메소드

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 직렬화 메소드의 작성 방법

– 2) writeObject 메소드 안에 원하는 필드만 직렬화하는 명령문을 써넣습니다.

```
class DistrChart implements Serializable {
 int arr[][];
 ...
 private void writeObject(ObjectOutputStream out)
                                  throws IOException {
  for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[0].length; col++) {
    if (arr[row][col] != 0) {
     out.writeInt(row);
     out.writeInt(col);
     out.writeInt(arr[row][col]);
    }
   }
  }
 }
}
```

arr 배열에서 0이 아닌 항목들의 값만 행 번호, 열 번호와 함께 출력합니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 역직렬화 메소드의 작성 방법

- 1) *readObject*라는 이름의 메소드를 선언합니다.

```
class DistrChart implements Serializable {
    ...
    private void readObject(ObjectInputStream in)
            throws IOException, ClassNotFoundException {
        ...
    }
}
```

역직렬화 메서드

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 역직렬화 메소드의 작성 방법

– 2) readObject 메소드 안에 데이터를 읽어서 필드에 대입하는 명령문을 써넣습니다.

```
class DistrChart implements Serializable {
   int arr[][];
   ...
   private void readObject(ObjectInputStream in)
        throws IOException, assNotFoundException {
      arr  = new int[10][10];
      try {
         while (true) {
            int row = in.readInt();
            int col = in.readInt();
            int data = in.readInt();
            arr[row][col] = data;
         }
      }
      catch (EOFException e) {
      }
   }
}
```

in 파라미터를 통해 int 값을 3개씩 읽어서 arr 배열의 행번호, 열번호, 항목의 값으로 사용합니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

- [예제 17-9] 직렬화/역직렬화 메소드를 포함하는 직렬화 가능 클래스의 예

```
1   import java.io.*;
2   class DistrChart implements Serializable {
3       int arr[][];
4       DistrChart() {
5           arr = new int[10][10];
6       }
7       private void writeObject(ObjectOutputStream out) throws IOException {
8           for (int row = 0; row < arr.length; row++) {
9               for (int col = 0; col < arr[0].length; col++) {
10                  if (arr[row][col] != 0) {
11                      out.writeInt(row);
12                      out.writeInt(col);
13                      out.writeInt(arr[row][col]);
14                  }
15              }
16          }
17      }
18      private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
19          arr  = new int[10][10];
20          try {
21              while (true) {
22                  int row = in.readInt();
23                  int col = in.readInt();
24                  int data = in.readInt();
25                  arr[row][col] = data;
26              }
27          }
28          catch (EOFException e) {
29          }
30      }
31  }
```

직렬화 메소드

역직렬화 메소드

객체의 직렬화

직렬화 메소드와 역직렬화 메소드

- [예제 17-10] DistrChart 객체를 직렬화하고 역직렬화하는 프로그램

DistrChart 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample5 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new
7   FileOutputStream("output5.dat"));
8               DistrChart obj = new DistrChart();
9               obj.arr[0][1] = 2;
10              obj.arr[4][5] = 5;
11              obj.arr[6][1] = 2;
12              obj.arr[7][7] = 7;
13              obj.arr[8][4] = 21;
14              out.writeObject(obj);
15          }
16          catch (IOException ioe) {
17              System.out.println("파일로 출력할 수 없습니다.");
18          }
19          finally {
20              try {
21                  out.close();
22              }
23              catch (Exception e) {
24              }
25          }
26      }
    }
```

```
명령 프롬프트

E:₩work₩chap17₩17-2-2₩example2>javac DistrChart.java

E:₩work₩chap17₩17-2-2₩example2>javac ObjectOutputExample5.java

E:₩work₩chap17₩17-2-2₩example2>javac ObjectInputExample5.java

E:₩work₩chap17₩17-2-2₩example2>java ObjectOutputExample5

E:₩work₩chap17₩17-2-2₩example2>java ObjectInputExample5
  0  2  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  5  0  0  0  0
  0  0  0  0  0  0  0  0  0  0
```

DistrChart 객체를 역직렬화하는

```
1   import java.io.*;
2   class ObjectInputExample5 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output5.dat"));
7               DistrChart obj = (DistrChart) in.readObject();
8               for (int row = 0; row < obj.arr.length; row++) {
9                   for (int col = 0; col < obj.arr[0].length; col++) {
10                      System.out.printf("%3d", obj.arr[row][col]);
11                  }
12                  System.out.println();
13              }
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (EOFException eofe) {
19              System.out.println("끝");
20          }
21          catch (IOException ioe) {
22              System.out.println("파일을 읽을 수 없습니다.");
23          }
24          catch (ClassNotFoundException cnfe) {
25              System.out.println("해당 클래스가 존재하지 않습니다.");
26          }
27          finally {
28              try {
29                  in.close();
30              }
31              catch (Exception e) {
32              }
33          }
34      }
35  }
```

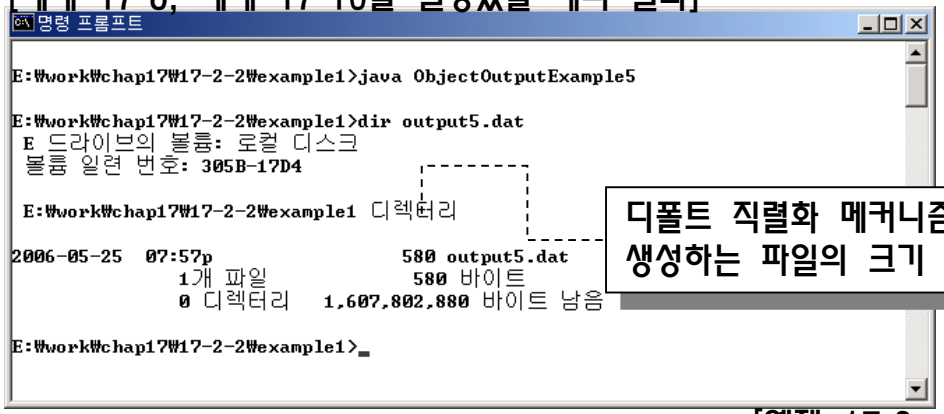
객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

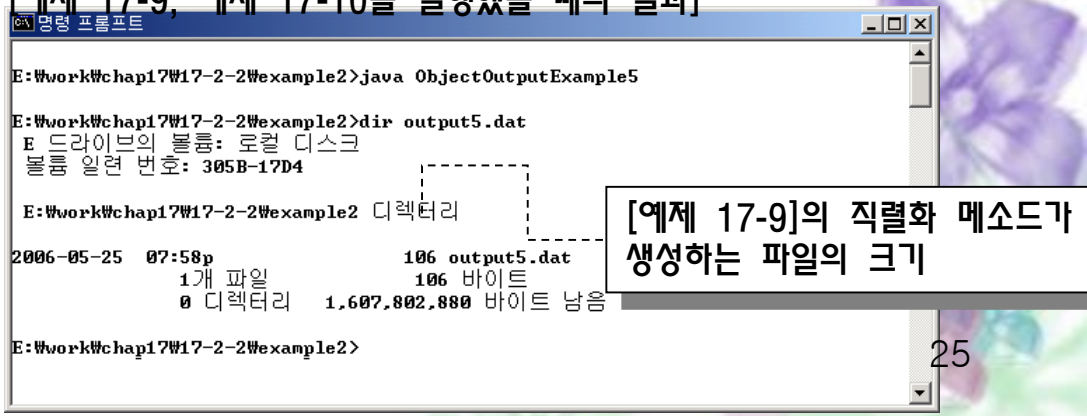
직렬화 메소드와 역직렬화 메소드

•• 디폴트 직렬화 메커니즘과의 차이점 확인

[예제 17-8, 예제 17-10을 실행했을 때의 결과]



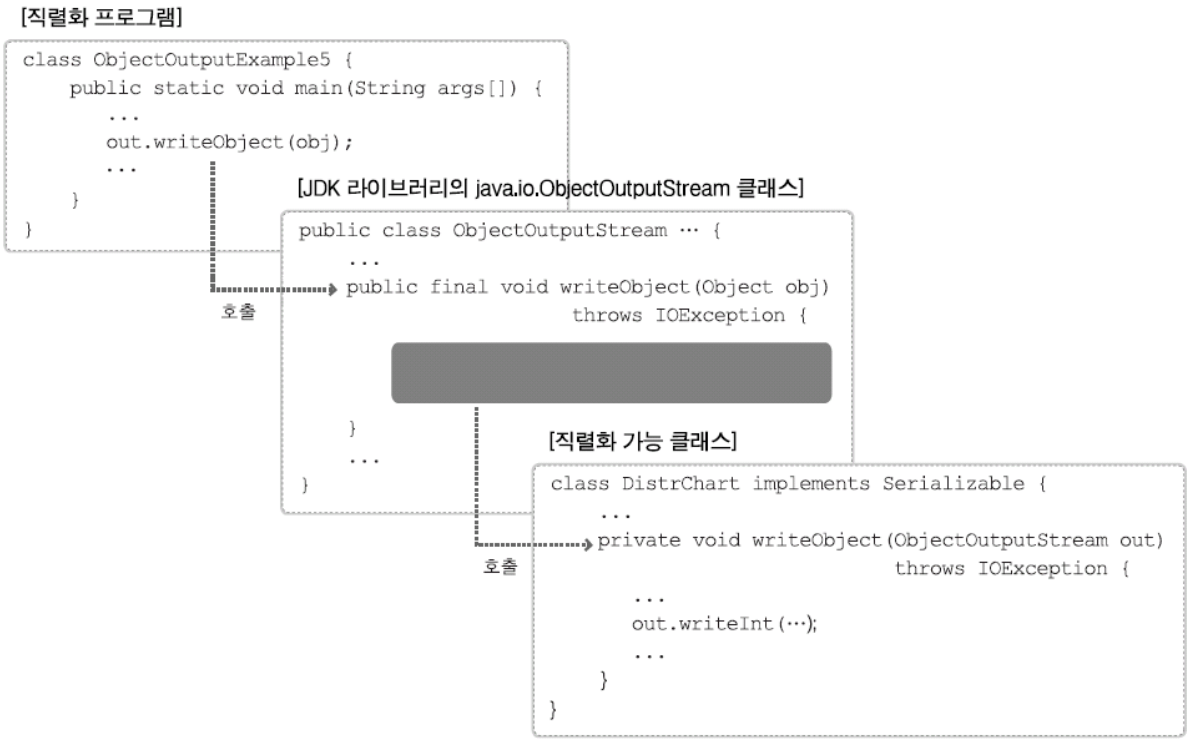
[예제 17-9, 예제 17-10을 실행했을 때의 결과]



02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

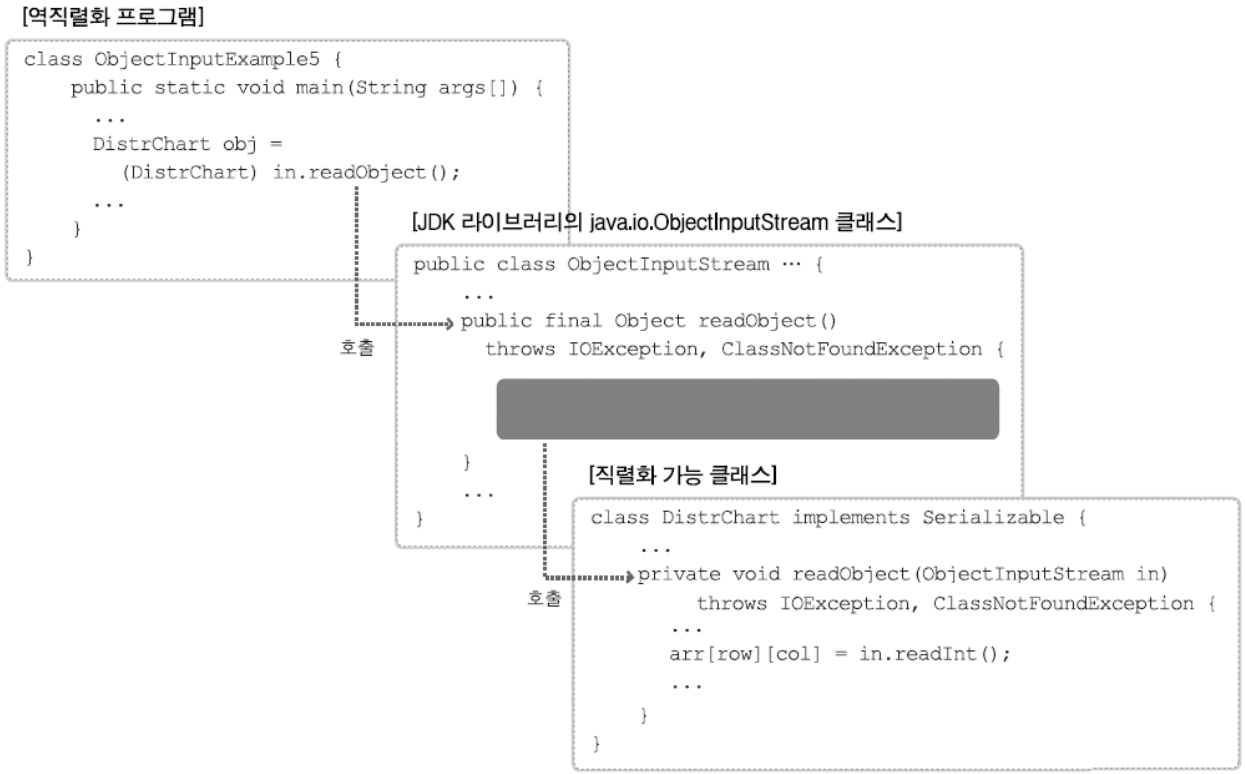
•• 직렬화 메소드가 호출되는 메커니즘



02. 직렬화 가능 클래스의 선언 방법

직렬화 메소드와 역직렬화 메소드

•• 역직렬화 메소드가 호출되는 메커니즘



객체의 직렬화

02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

- [예제 17-11] 상품 정보 클래스와 서브클래스들

상품 정보 클래스

```
1   class GoodsInfo {
2    String code;      // 상품코드
3    String name;      // 상품명
4    int price;        // 가격
5    GoodsInfo(String name, String code, int price) {
6     this.name = name;
7     this.code = code;
8     this.price = price;
9    }
10  }
```

이런 클래스들이 있다고 가정합시다.

의류 정보 클래스

```
1   class ClothingInfo extends GoodsInfo {
2    String color;     // 색상
3    char size;        // 사이즈: L M S
4    ClothingInfo(String name, String code,
5                 int price, String color, char size) {
6     super(name, code, price);
7     this.color = color;
8     this.size = size;
9    }
10  }
```

도서 정보 클래스

```
1   class BookInfo extends GoodsInfo {
2    String writer;     // 글쓴이
3    int page;          // 페이지 수
4    BookInfo(String name, String code,
5             int price, String writer, int page) {
6     super(name, code, price);
7     this.writer = writer;
8     this.page = page;
9    }
10  }
```

02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

- [예제 17-12] java.io.Serializable 인터페이스를 구현하는 도서 정보 클래스

```
1     class BookInfo extends GoodsInfo implements java.io.Serializable {
2         String writer;      // 글쓴이
3         int page;           // 페이지 수
4         BookInfo(String name, String code, int price, String writer, int page) {
5             super(name, code, price);
6             this.writer = writer;
7             this.page = page;
8         }
9     }
```

이렇게 선언하는 것만으로 충분할까요?

객체의 직렬화

다른 클래스를 상속받는 직렬화 가능 클래스

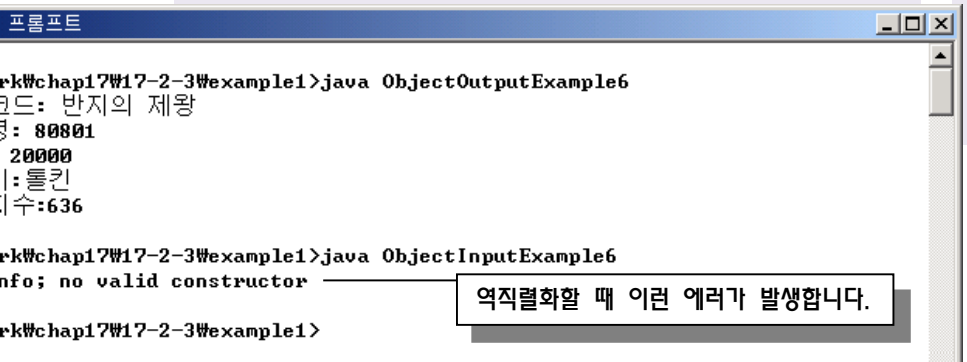
- [예제 17-13] BookInfo 객체를 직렬화하고 역직렬화하는 프로그램

BookInfo 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample6 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new
7   FileOutputStream("output6.dat"));
                BookInfo obj = new BookInfo("80801", "반지의 제왕",
8                                           20000, "톨킨", 636);
9               System.out.println("상품코드: " + obj.code);
10              System.out.println("상품명: " + obj.name);
11              System.out.println("가격: " + obj.price);
12              System.out.println("지은이:" + obj.writer);
13              System.out.println("페이지수:" + obj.page);
14              out.writeObject(obj);
15          }
16          catch (IOException ioe) {
17              System.out.println(ioe.getMessage());
18          }
19          finally {
20              try {
21                  out.close();
22              }
23              catch (Exception e) {
24              }
25          }
26      }
    }
```

BookInfo 객체를 역직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectInputExample6 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output6.dat"));
7               BookInfo obj = (BookInfo) in.readObject();
8               System.out.println("상품코드: " + obj.code);
9               System.out.println("상품명: " + obj.name);
10              System.out.println("가격: " + obj.price);
11              System.out.println("지은이:" + obj.writer);
12              System.out.println("페이지수:" + obj.page);
13          }
14          catch (FileNotFoundException fnfe) {
15              System.out.println("파일이 존재하지 않습니다.");
16          }
17          catch (EOFException eofe) {
18              System.out.println("끝");
19          }
20          catch (IOException ioe) {
21              System.out.println(ioe.getMessage());
22          }
23          catch (ClassNotFoundException cnfe) {
24              System.out.println("해당 클래스가 존재하지 않습니다.");
25          }
26          finally {
27              try {
28                  in.close();
29              }
30              catch (Exception e) {
31              }
32          }
33      }
34  }
```



02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

•• 객체를 역직렬화할 때 호출되는 생성자

직렬화 가능 클래스의
생성자 호출 메커니즘 때문입니다.

```
public class Object {
    …
}
```

이 클래스의 생성자는 서브클래스의 no-arg constructor에 의해 간접적으로 호출됩니다(5장 참조).

상속

```
class GoodsInfo {
    …
}
```

이 클래스의 no-arg constructor가 호출됩니다.

상속

```
class GoodsInfo extends GoodsInfo
        implements java.io.Serializable {
    …
}
```

이 클래스의 생성자는 호출되지 않습니다.

02. 직렬화 가능 클래스의 선언 방법

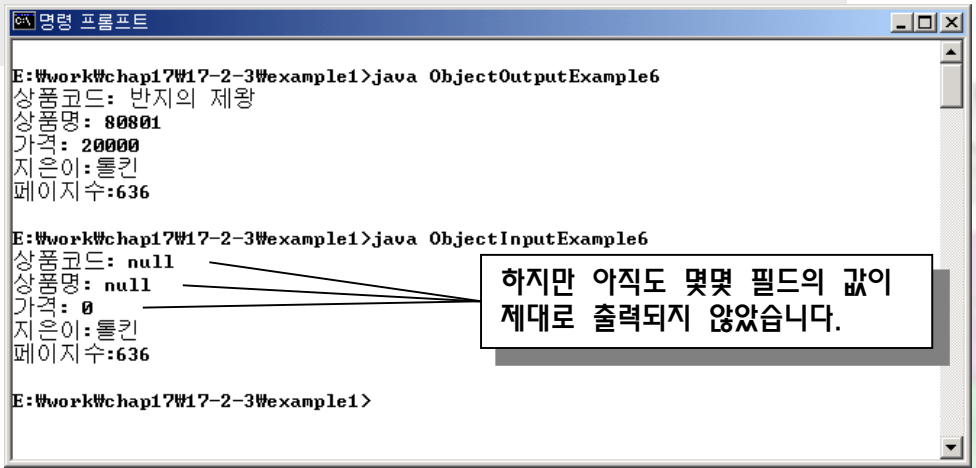
다른 클래스를 상속받는 직렬화 가능 클래스

- [예제 17-14] no-arg constructor를 추가한 GoodsInfo 클래스

```
1    class GoodsInfo {
2        String code;       // 상품코드
3        String name;       // 상품명
4        int price;         // 가격
5        GoodsInfo() {
6        }
7        GoodsInfo(String name, String code, int price) {
8            this.name = name;
9            this.code = code;
10           this.price = price;
11       }
12   }
```

no-arg constructor

이 클래스를 가지고 직렬화/역직렬화 프로그램을 실행하면 에러가 발생하지 않을 것입니다.



02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

•• 클래스의 상속 관계와 직렬화 되는 필드

상속과 관련된 직렬화/역직렬화 메커니즘 때문입니다.

```
public class Object {
    …
}
```

이 클래스의 필드도 직렬화/역직렬화되지 않습니다.

상속

```
class GoodsInfo {
    …
}
```

이 클래스의 필드는 직렬화/역직렬화되지 않습니다.

상속

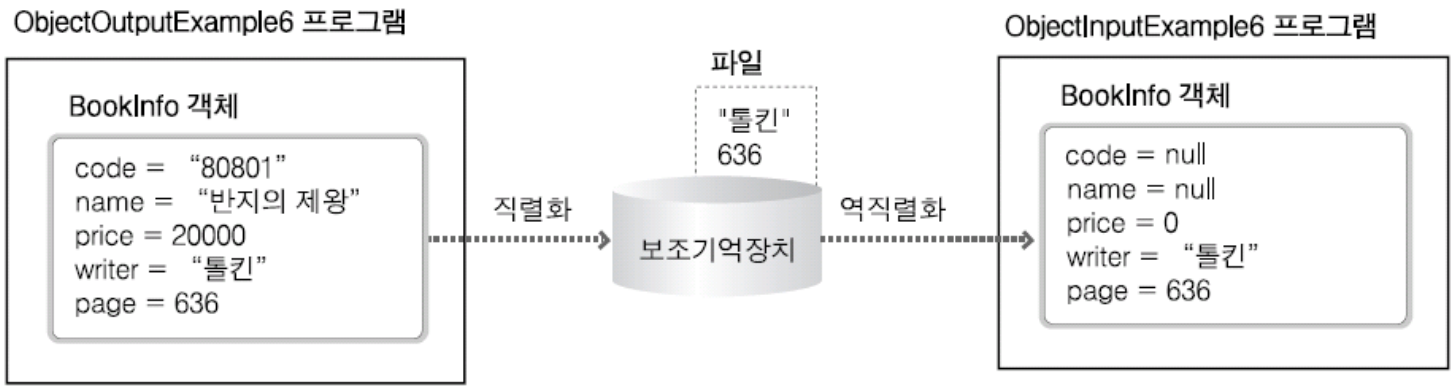
```
class GoodsInfo extends GoodsInfo
        implements java.io.Serializable {
    …
}
```

이 클래스의 필드는 직렬화/역직렬화됩니다.

02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

•• 앞 예제에서 직렬화/역직렬화 프로그램을 실행했을 때 일어난 일



객체의 직렬화

다른 클래스를 상속받는 직렬화 가능 클래스

- [예제 17-15] writeObject, readObject 메소드를 추가한 BookInfo 클래스

```
1   import java.io.*;
2   class BookInfo extends GoodsInfo implements Serializable {
3       String writer;      // 글쓴이
4       int page;           // 페이지 수
5       BookInfo(String name, String code, int price, St
6   page) {
7           super(name, code, price);
8           this.writer = wr
9           this.page = page;
10      }
11      private void writeObject(ObjectOutputStream out) throws
12  IOException {
13          out.writeUTF(cod
14          out.writeUTF(nam
15          out.writeInt(pric
16          out.writeUTF(writer);
17          out.writeInt(page);
18      }
19      private void readObject(ObjectInputStream in) throws IOException,
    ClassNotFoundException {
```

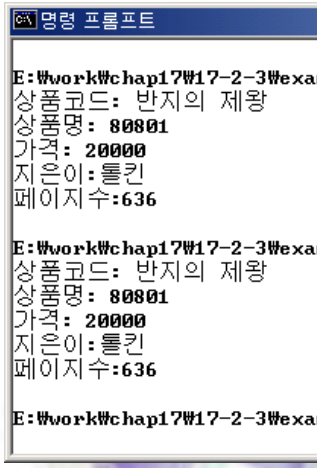
슈퍼클래스의 필드를 직렬화하는 명령문

이 클래스의 필드를 직렬화하는 명령문

슈퍼클래스의 필드를 역직렬화하는 명령문

이 클래스의 필드를 역직렬화하는 명령문

이 클래스를 가지고 실행하면 직렬화/역직렬화가 정상적으로 될 것입니다.



02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

•• 클래스 자신의 디폴트 직렬화 메커니즘을 호출하는 방법

이렇게 하면 앞 예제의 직렬화 메소드를 더 간단히 만들 수 있습니다.

```
private void writeObject(ObjectOutputStream out)
                            throws IOException {

    out.defaultWriteObject();

}
```

직렬화 가능 클래스 자신의 필드를 한꺼번에 직렬화하는 메소드

02. 직렬화 가능 클래스의 선언 방법

다른 클래스를 상속받는 직렬화 가능 클래스

- 클래스 자신의 디폴트 역직렬화 메커니즘을 호출하는 방법

앞 예제의 역직렬화 메소드도 더 간단히 만들 수 있습니다.

```
private void readObject(ObjectInputStream in)
          throws IOException, ClassNotFoundException {

      in.defaultReadObject();

}
```

직렬화 가능 클래스 자신의 필드를 한꺼번에 역직렬화하는 메소드

객체의 직렬화

다른 클래스를 상속받는 직렬화 가능 클래스

- [예제 17-16] 디폴트 직렬화/역직렬화 메소드를 호출하는 BookInfo 클래스

```
1   import java.io.*;
2   class BookInfo extends GoodsInfo implements Serializable {
3       String writer;      // 글쓴이
4       int page;           // 페이지 수
5       BookInfo(String name, String code, int price, String writer, int
6   page) {
7           super(name, code, price);
8           this.writer = wr
9           this.page = page;
1       }
0       private void writeObject(ObjectOutputStream out) throws
1   IOException {
1           out.writeUTF(code
1           out.writeUTF(name);
2           out.writeInt(price);
1           out.defaultWriteObject();
3       }
1       private void readObject(ObjectInputStream in) throws IOException,
4   ClassNotFoundException {
1           code = in.readUTF();
5           name = in.readUTF();
```

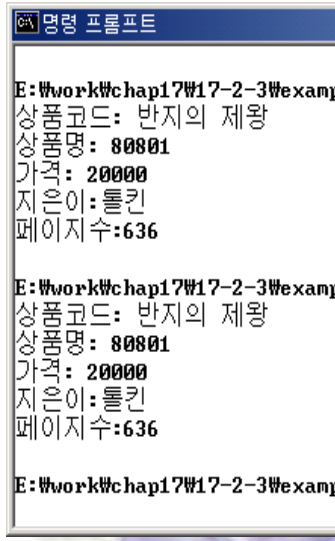
수정된 직렬화/역직렬화 메소드입니다.

슈퍼클래스의 필드를 직렬화하는 명령문

이 클래스의 필드를 직렬화하는 메소드 호출문

슈퍼클래스의 필드를 역직렬화하는 명령문

이 클래스의 필드를 역직렬화하는 메소드 호출문



02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 버전 관리

- [예제 17-17] 직렬화 가능한 사각형 클래스

```
1     class Rectangle implements java.io.Serializable {
2         int width, height;
3         Rectangle(int width, int height) {
4             this.width = width;
5             this.height = height;
6         }
7     }
```

이런 클래스가 있다고 가정합시다.

객체의 직렬화

직렬화 가능 클래스의 버전 관리

- [예제 17-18] Rectangle 클래스의 객체를 직렬화하고 역직렬화하는 프로그램

Rectangle 객체를 직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectOutputExample7 {
3       public static void main(String args[]) {
4           ObjectOutputStream out = null;
5           try {
6               out = new ObjectOutputStream(new
7   FileOutputStream("output7.dat"));
8               Rectangle obj = new Rectangle(100, 200);
9               System.out.println("넓이: " + obj.width);
10              System.out.println("높이: " + obj.height);
11              out.writeObject(obj);
12          }
13          catch (IOException ioe) {
14              System.out.println(ioe.getMessage());
15          }
16          finally {
17              try {
18                  out.close();
19              }
20              catch (Exception e) {
21              }
22          }
23      }
    }
```

```
령 프롬프트
ork₩chap17₩17-2-4₩example1>java ObjectOutputExample7
: 100
: 200

ork₩chap17₩17-2-4₩example1>java ObjectInputExample7
: 100
: 200

ork₩chap17₩17-2-4₩example1>_
```

같은 Rectangle 클래스를 가지고 실행하면 정상적인 실행 결과가 나옵니다.

Rectangle 객체를 역직렬화하는 프로그램

```
1   import java.io.*;
2   class ObjectInputExample7 {
3       public static void main(String args[]) {
4           ObjectInputStream in = null;
5           try {
6               in = new ObjectInputStream(new FileInputStream("output7.dat"));
7               Rectangle obj = (Rectangle) in.readObject();
8               System.out.println("넓이: " + obj.width);
9               System.out.println("높이: " + obj.height);
10          }
11          catch (FileNotFoundException fnfe) {
12              System.out.println("파일이 존재하지 않습니다.");
13          }
14          catch (EOFException eofe) {
15              System.out.println("끝");
16          }
17          catch (IOException ioe) {
18              System.out.println(ioe.getMessage());
19          }
20          catch (ClassNotFoundException cnfe) {
21              System.out.println("해당 클래스가 존재하지 않습니다.");
22          }
23          finally {
24              try {
25                  in.close();
26              }
27              catch (Exception e) {
28              }
29          }
30      }
31  }
```

02. 직렬화 가능 클래스의 선언 방법

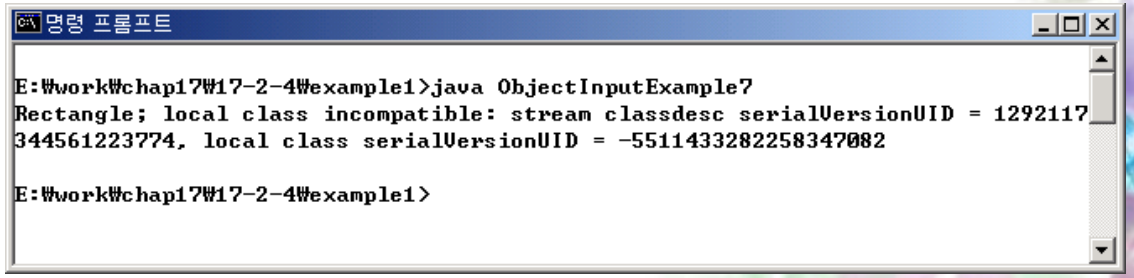
직렬화 가능 클래스의 버전 관리

- [예제 17-19] 새로운 메소드가 추가된 사각형 클래스

```
1     class Rectangle implements java.io.Serializable {
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    }
```

추가된 메소드

Rectangle 클래스를 이렇게 수정한 후에
역직렬화 프로그램을 실행하면 어떻게 될까요?



02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 버전 관리

- [예제 17-20] 버전 번호를 포함하는 사각형 클래스 (1)

```
1    class Rectangle implements java.io.Serializable {
2        static final long serialVersionUID = 100;
3        int width, height;
4        Rectangle(int width, int height) {
5            this.width = width;
6            this.height = height;
7        }
8    }
```

버전 번호

이렇게 버전 번호를 붙이면 문제를 해결할 수 있습니다.

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 버전 관리

- [예제 17-21] 버전 번호를 포함하는 사각형 클래스 (2)

```
1    class Rectangle implements java.io.Serializable {
2        static final long serialVersionUID = 100;
3        int width, height;
4        Rectangle(int width, int height) {
5            this.width = width;
6            this.height = height;
7        }
8        int getArea() {
9            return width * height;
10       }
11   }
```

버전 번호

클래스를 수정한 후에도 똑같은 버전 번호를 유지해야 합니다.

추가된 메소드

[예제 17-20]의 Rectangle 클래스를 가지고 직렬화 프로그램을 실행했을 때

```
E:\work\chap17\17-2-4\example2>java ObjectOutputExample7
넓이: 100
높이: 200

E:\work\chap17\17-2-4\example2>_
```

[예제 17-21]의 Rectangle 클래스

```
명령 프롬프트

E:\work\chap17\17-2-4\example2>java ObjectInputExamp
넓이: 100
높이: 200

E:\work\chap17\17-2-4\example2>_
```

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 버전 관리

- 버전 번호의 충돌을 최소화하는 방법

Rectangle.java - 메모장

파일(F) 편집(E) 서식(O) 도움말(H)

```
class Rectangle implements java.io.Serializable {
    int width, height;
    Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }
}
```

1) 직렬화 가능 클래스를 작성합니다.

명령 프롬프트

```
E:\work\chap17\17-2-4\example3>javac Rectangle.java

E:\work\chap17\17-2-4\example3>serialver Rectangle
Rectangle:    static final long serialVersionUID = 1292117344561223774L;

E:\work\chap17\17-2-4\example3>
```

2) 직렬화 가능 클래스를 컴파일합니다.

3) 컴파일한 디렉토리에서 serialver 명령을 실행하면 버전 번호가 생성됩니다.

버전 번호

02. 직렬화 가능 클래스의 선언 방법

직렬화 가능 클래스의 버전 관리

- [예제 17-22] serialver 명령이 생성한 버전 번호를 붙인 사각형 클래스

```
1     class Rectangle implements java.io.Serializable {
2         static final long serialVersionUID = 1292117344561223774L;
3         int width, height;
4         Rectangle(int width, int height) {
5             this.width = width;
6             this.height = height;
7         }
8         int getArea() {
9             return width * height;
10        }
11    }
```

버전 번호

원래의 Rectangle 클래스를 가지고 직렬화 프로그램을 실행했을 때

```
E:₩work₩chap17₩17-2-4₩example3>java ObjectOutputExample7
넓이: 100
높이: 200

E:₩work₩chap17₩17-2-4₩example3>_
```

getArea 메소드가 추가된 Rectangle 클래스를 가

명령 프롬프트

```
E:₩work₩chap17₩17-2-4₩example3>java ObjectInputExample7
넓이: 100
높이: 200

E:₩work₩chap17₩17-2-4₩example3>
```

열 공 합 시 다 ^^!