

# 16장. 객체 저장: 직렬화와 파일 입출력

---

객체를 저장하는 방법  
직렬화된 객체  
객체 역직렬화  
파일 저장 및 불러오기

# 객체 저장

- 객체의 파일 저장과 복구
- 직렬화(serialization)와 역직렬화(de-serialization)
- 파일 입출력
- NIO2와 java.nio.file 패키지
- try-with-resources 문



# 비트박스 패턴을 저장합시다.

- Save 버튼을 누르면 패턴을 저장할 수 있도록 만들어봅시다.
  - 데이터 저장 방법
    - 직렬화(serialization)를 사용하는 방법
      - 저장된 데이터를 자바 프로그램에서만 사용하는 경우
      - 객체를 통째로 저장한다고 생각하면 됩니다.
    - 일반 텍스트 파일로 저장하는 방법
      - 다른 프로그램에서도 사용할 수 있도록 할 때
      - 파싱하기 좋도록 적당한 구분자로 각 필드를 구분해주는 것이 좋습니다

# 상태 저장 방법

- 직렬화된 캐릭터 객체 세 개를 파일에 저장하는 방법

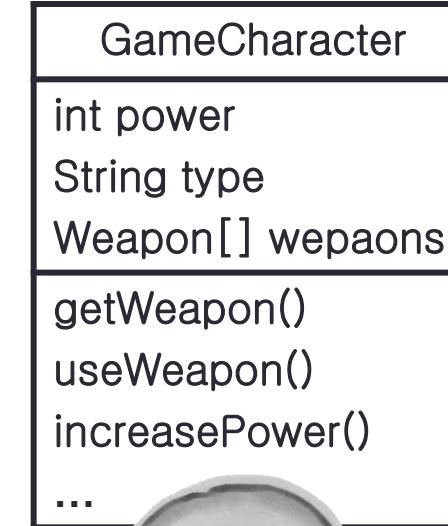
"Isr Character5 3+

- 일반 텍스트 파일로 저장하는 방법

50,Elf,bow,sword,dust

200,Troll,bare hands,big ax

120,Magician,spells,invisibility



# 직렬화된 객체를 파일에 저장하는 방법

1. FileOutputStream 만들기 (저수준 파일 출력 담당)

```
FileOutputStream fileStream = new FileOutputStream("MyGame.ser");
```

2. ObjectOutputStream 만들기 (객체 수준의 파일 출력 담당 → 저수준 파일 출력 호출하여 실제로 저장)

```
ObjectOutputStream os = new ObjectOutputStream(fileStream);
```

3. 객체 저장

```
os.writeObject(characterOne);
os.writeObject(characterTwo);
os.writeObject(characterThree);
```

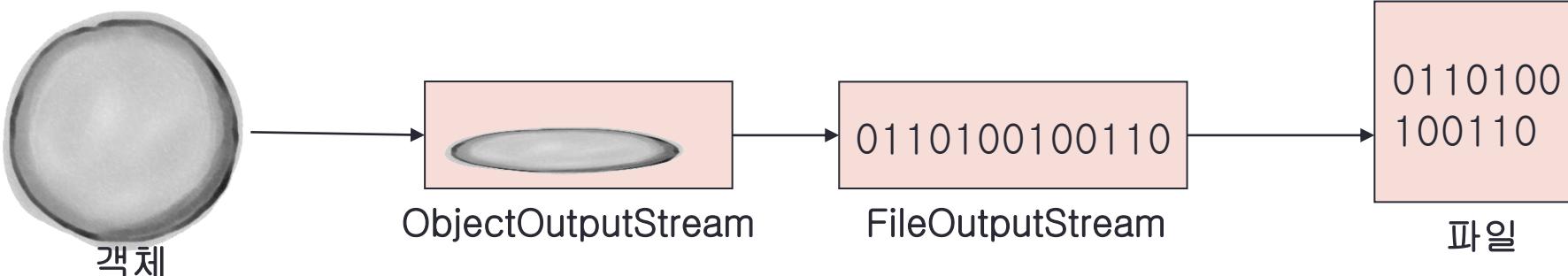
4. ObjectOutputStream 닫기

```
os.close();
```

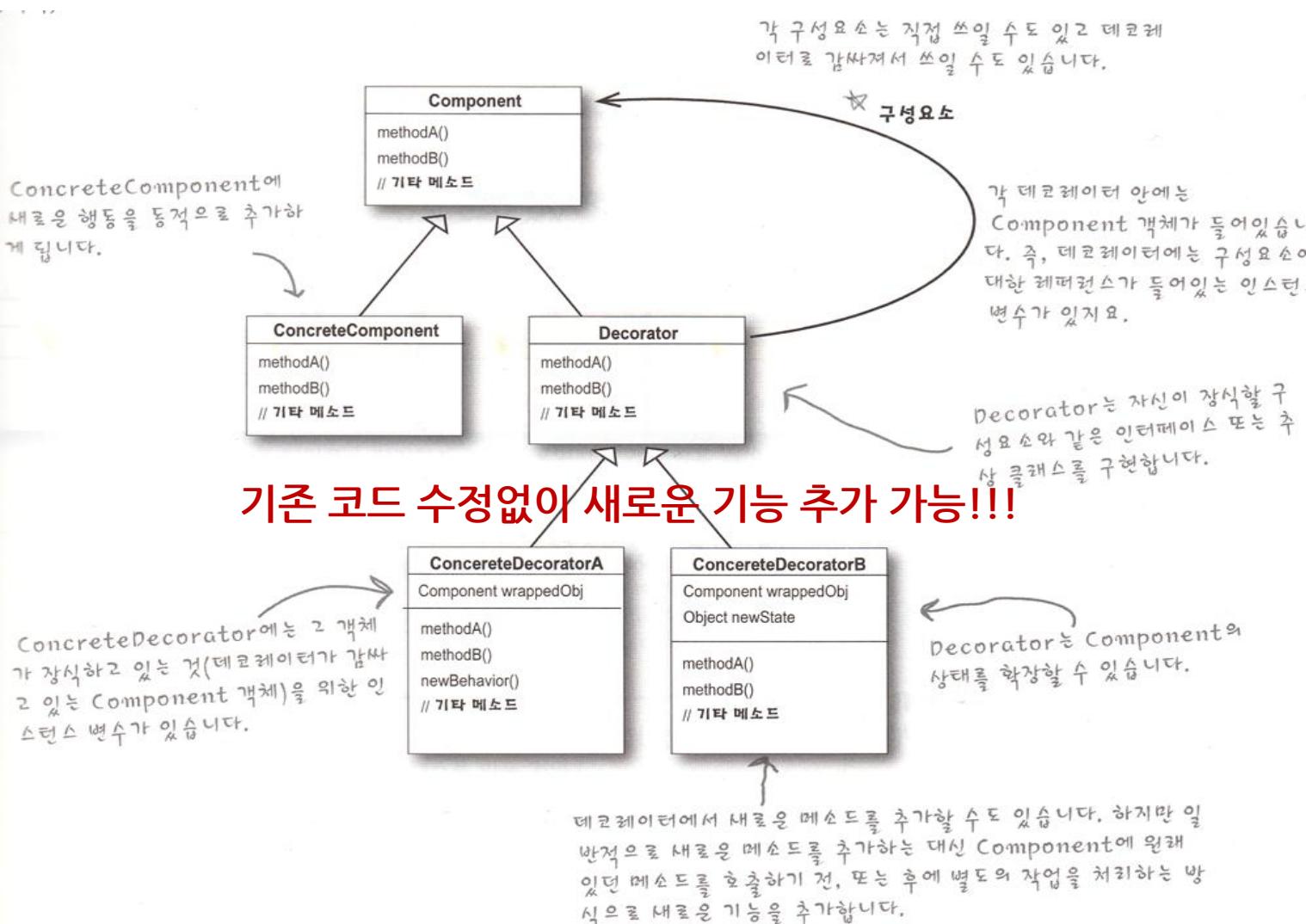
# 스트림

참고: <http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html>

- 연결 스트림(connection stream) - 데코레이터 패턴의 ConcreteComponent에 해당
  - 파일 또는 네트워크 소켓과 같은 출발지, 목적지로의 연결을 나타내는 스트림
  - 일반적으로 너무 저수준이라서 연결 스트림만 가지고는 작업을 하기 힘듭니다.
- 연쇄 스트림(chain stream) - 데코레이터 패턴의 ConcreteDecorator에 해당
  - 다른 스트림에 연쇄 시켜야만 쓸 수 있음
  - 고수준의 작업을 처리하기 위한 메서드를 제공



# 참고. 데코레이터 패턴

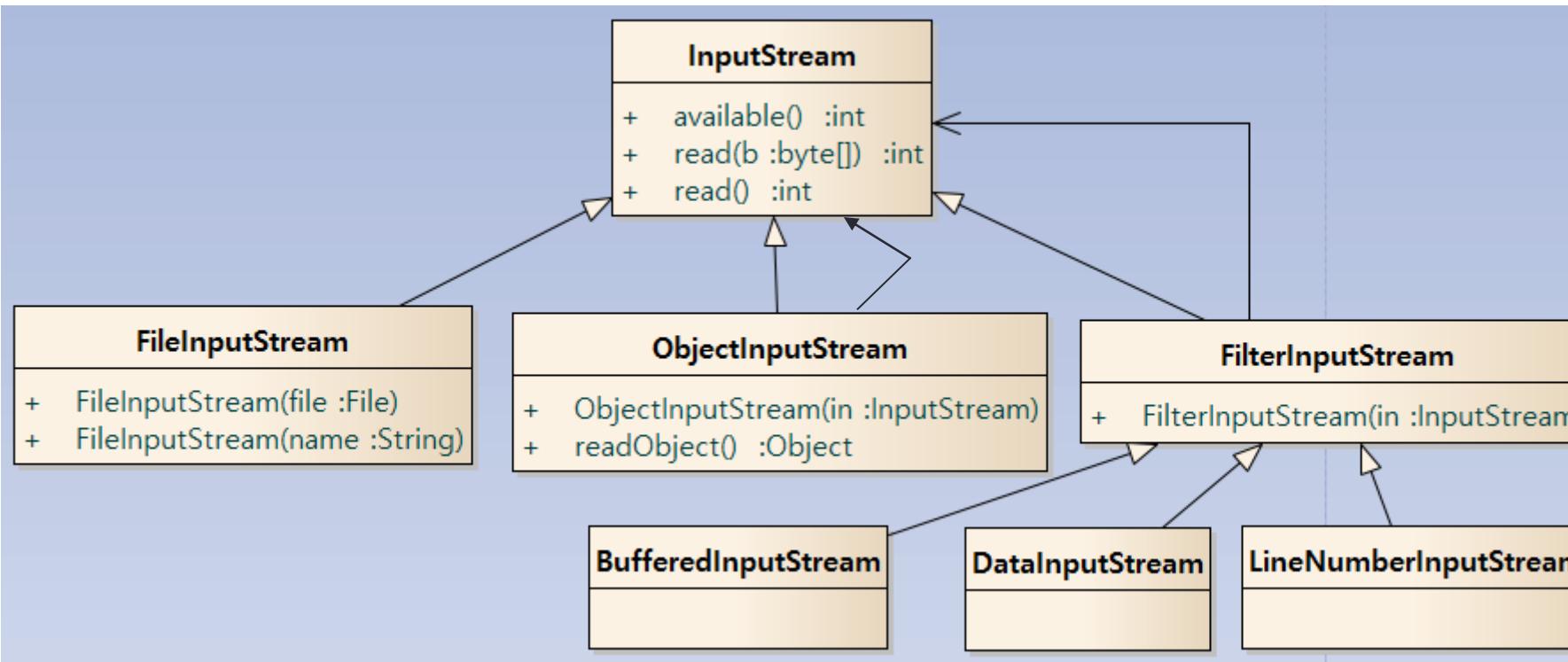


# 입출력 패키지

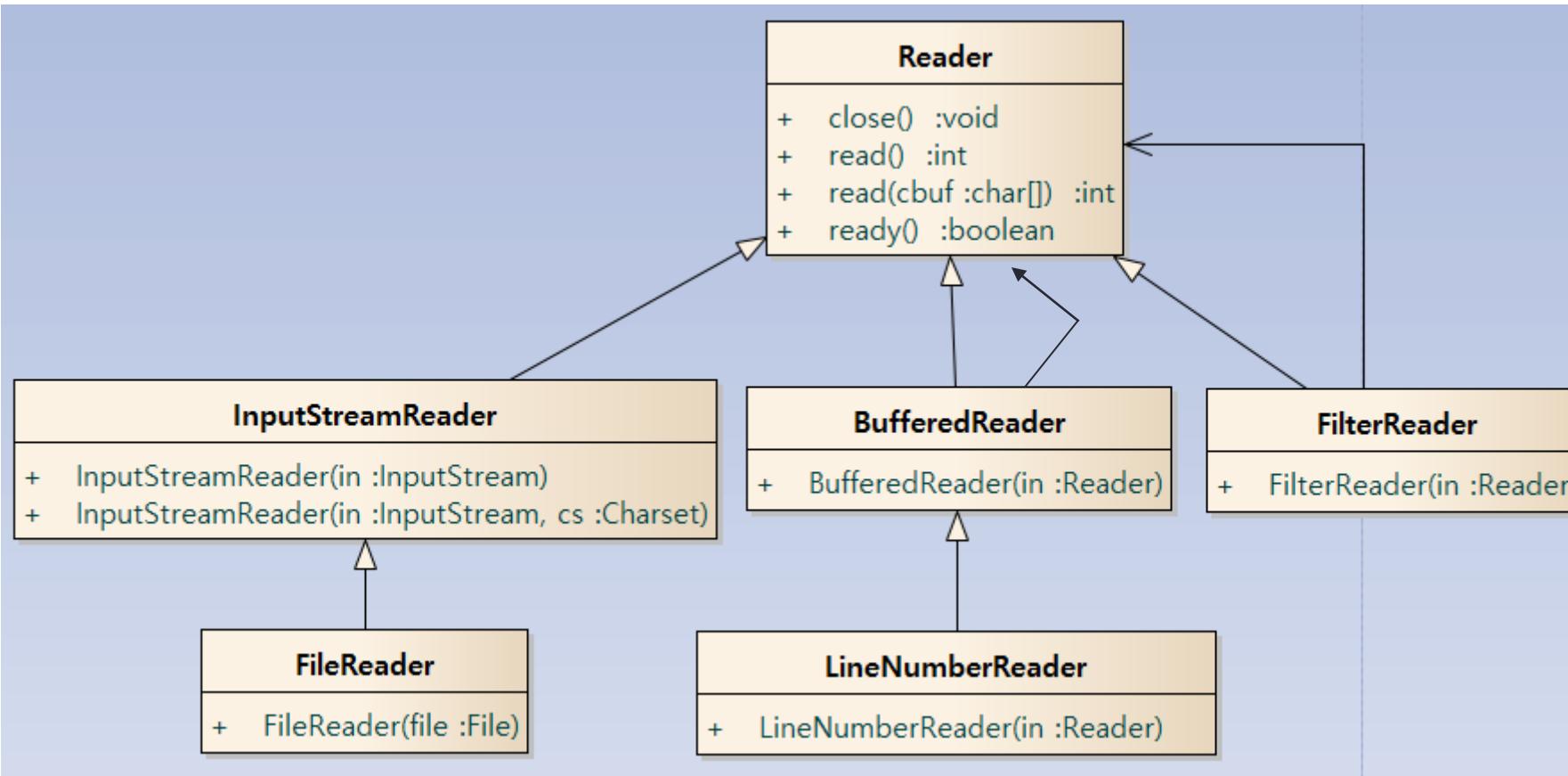
- 바이트 입출력: 바이트 단위의 입출력
- 문자 입출력: 문자 단위의 입출력
- 참고) char (or Character): UTF-16 → 2 바이트



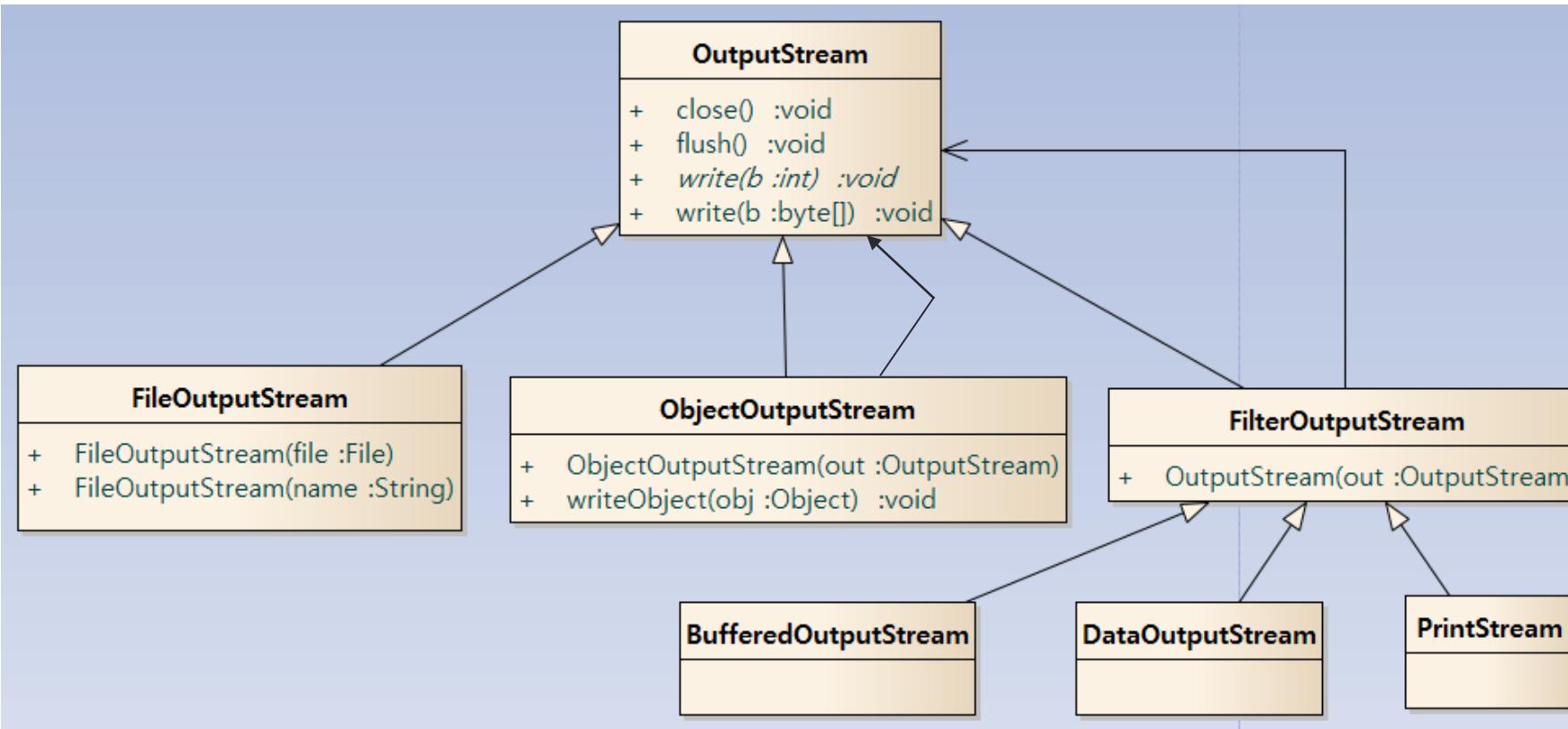
# 바이트 입력 클래스



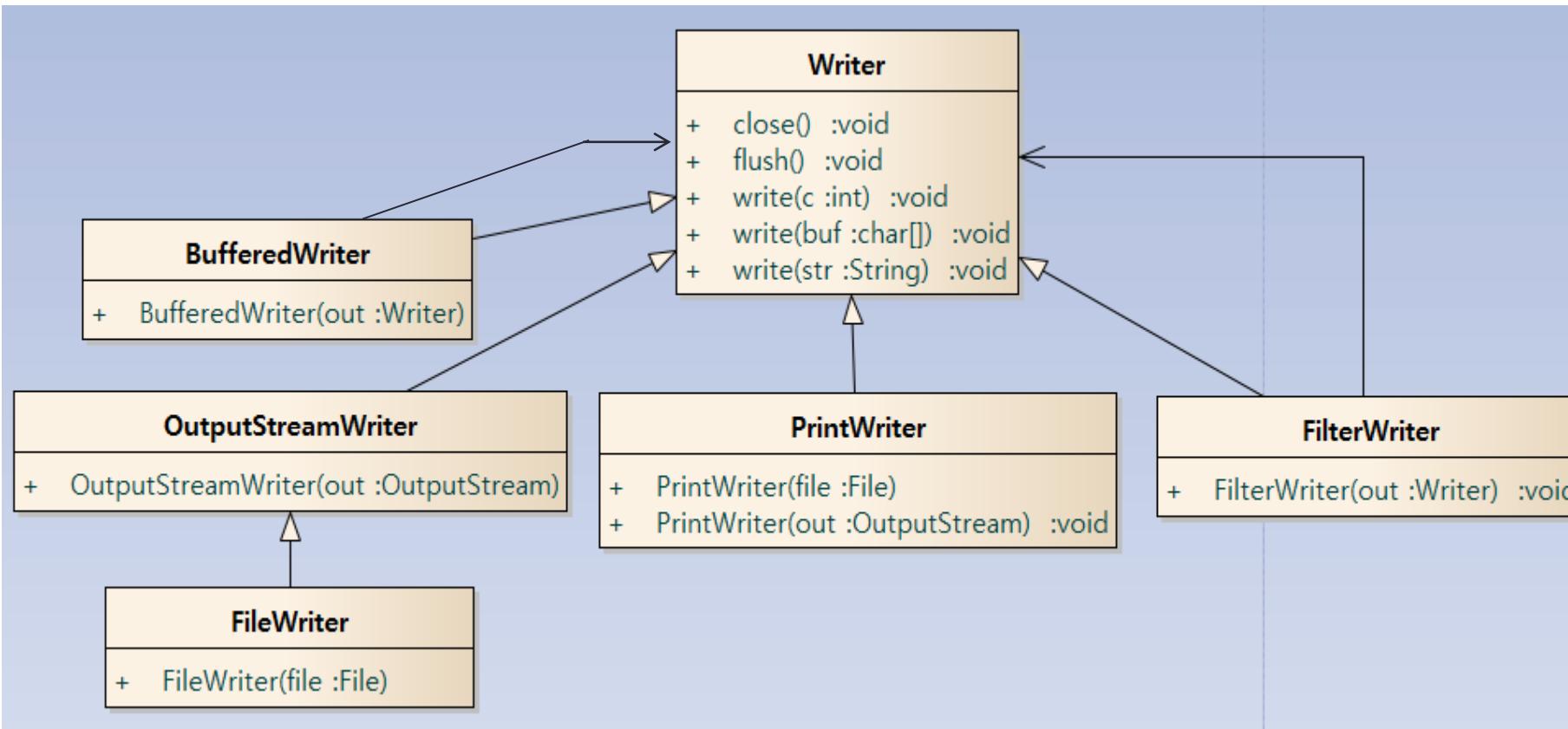
# 문자 입력 클래스



# 바이트 출력 클래스



# 문자 출력 클래스



# 자바 I/O Decorator 실습

- LowerCaseInputStream
  - 입력 스트림을 바이트 단위로 읽어 영문 대문자는 소문자로 변환
  - 한글이 깨어짐
- LowerCaseReader
  - 입력 스트림을 문자 단위로 읽어 영문 대문자는 소문자로 변환
  - 한글 출력에 문제 없음

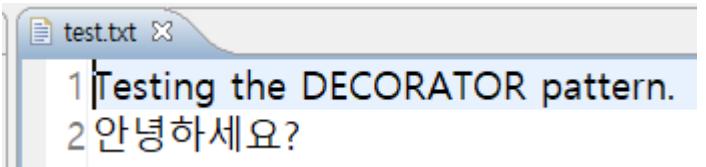
# LowerCaseInputStream.java

```
6  package cse.oop2.ch14.decorator;          패키지 이름을 cse.oop2.ch16_3.decorator로 수정해야 함.  
7  
8  import java.io.BufferedInputStream;  
9  import java.io.FileInputStream;  
10 import java.io.FilterInputStream;  
11 import java.io.IOException;  
12 import java.io.InputStream;  
13  
14 /**...4 lines */  
15 public class LowerCaseInputStream extends FilterInputStream {  
16  
17     public LowerCaseInputStream(InputStream in) {  
18         super(in);  
19     }  
20  
21     @Override  
22     public int read() throws IOException {  
23         int c = super.read();  
24         return (c == -1 ? c : Character.toLowerCase((char) c));  
25     }  
26 }  
27  
28 }
```

```
30  public static void main(String[] args) {
31    try {
32      int c;
33      InputStream is = new LowerCaseInputStream(
34        new BufferedInputStream(
35          new FileInputStream("./src/main/java/cse/oop2/ch14/decorator/test.txt")));
36      while ((c = is.read()) >= 0) {
37        System.out.print((char) c);
38      }
39    } catch (IOException ex) {
40      System.out.println("오류 발생: " + ex.getMessage());
41    }
42  }
43 }
44 }
```

# 실행 결과

- 입력 파일: test.txt



- 실행 결과



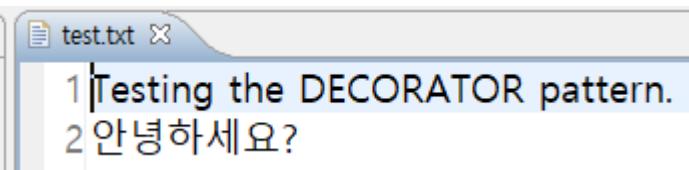
# LowerCaseReader.java

```
6  package cse.oop2.ch14.decorator;
7
8  import java.io.BufferedReader;
9  import java.io.FileReader;
10 import java.io.FilterReader;
11 import java.io.IOException;
12 import java.io.Reader;
13
14
15 public class LowerCaseReader extends FilterReader {
16
17     public LowerCaseReader(Reader in) {
18         super(in);
19     }
20
21     @Override
22     public int read() throws IOException {
23         int c = super.read();
24         return (c == -1 ? c : Character.toLowerCase((char) c));
25     }
26 }
```

```
27 public static void main(String[] args) {
28     try {
29         int c;
30         // ant로 빌드 시 UTF-8 파일 읽는 것 문제 없음. 윈도우10에서 maven 빌드 시 인코딩 문제 발생.
31         Reader is = new LowerCaseReader(
32             new BufferedReader(
33                 new FileReader("./src/main/java/cse/oop2/ch14/decorator/test.txt")));
34
35         // InputStreamReader(InputStream in, String charsetName) 사용하여 인코딩 지정.
36         /*
37             Reader is = new LowerCaseReader(
38                 new InputStreamReader(
39                     new FileInputStream("./src/main/java/cse/oop2/ch14/decorator/test.txt"), "UTF-8"));
40         */
41         while ((c = is.read()) >= 0) {
42             System.out.print((char) c);
43         }
44     } catch (IOException ex) {
45         System.out.println("오류 발생: " + ex.getMessage());
46     }
47 }
48 }
```

# 실행 결과

- 입력 파일: test.txt

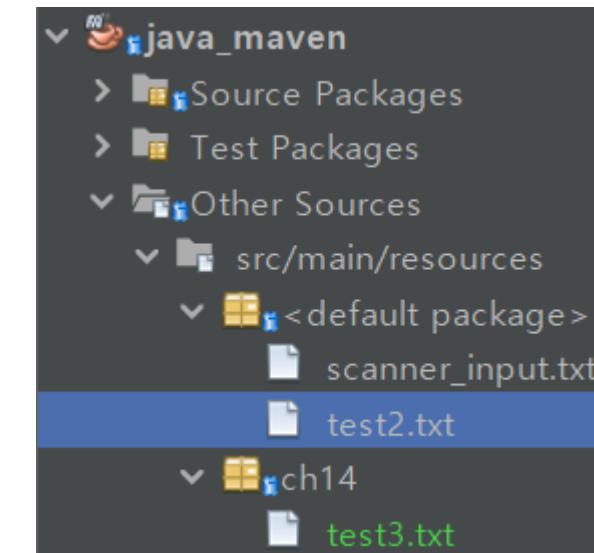
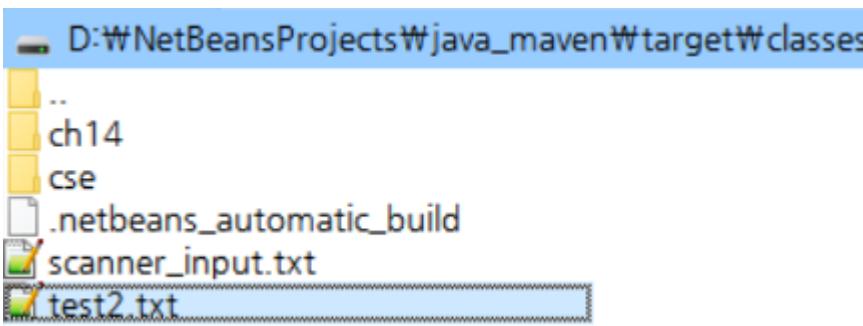
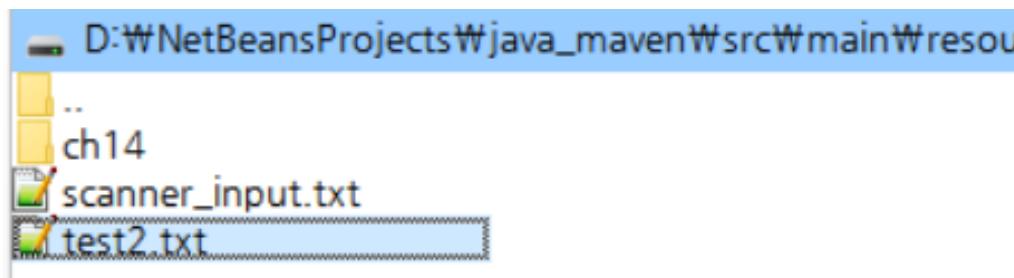


- 실행 결과



# 참고. Class.getResourceAsStream() 사용

- test.txt 파일을 복사하여 java\_maven 프로젝트 내 **src/main/resources** 폴더에 복사
- 해당 폴더가 없으면 수작업으로 생성
- (기존의 test.txt 파일을 실수로 읽지 않도록) 복사한 파일을 test2.txt로 이름 변경



# TestDrive.java\*\*\*

```
6  package cse.oop2.ch14.decorator.using_resource;
7
8  import cse.oop2.ch14.decorator.LowerCaseReader;           기존의 LowerCaseReader는 그대로 사용
9  import java.io.BufferedReader;
10 import java.io.IOException;
11 import java.io.InputStream;
12 import java.io.InputStreamReader;
13 import java.io.Reader;
14
15 /**
16  * ...4 lines ...
17 */
18
19 public class TestDrive {
20
21     public void go() {
22         System.out.printf("%s.go() called...\n", getClass().getSimpleName());
23         InputStream is = getClass().getResourceAsStream("/test2.txt");
24         int c;           "/test2.txt" 파일에 대한 InputStream 찾기
```

기존의 LowerCaseReader는 그대로 사용.  
try-with-resources 문을 사용하여 자동으로 획득한  
자원이 종료되므로 finally 블록 필요 없음!

```
26 try (Reader reader = new LowerCaseReader(  
27     new BufferedReader(  
28         new InputStreamReader(is, "UTF-8")))) {  
29  
30     while ((c = reader.read()) >= 0) {  
31         System.out.print((char) c);  
32     }  
33 } catch (IOException ex) {  
34     System.out.println("오류 발생: " + ex.getMessage());  
35 }  
36  
37  
38 /**...3 lines */  
41 public static void main(String[] args) {  
42     // TODO code application logic here  
43     new TestDrive().go();  
44 }  
45  
46 }
```

```
--- exec-maven-plugin:1.5.0:ex  
TestDrive.go() called...  
testing the decorator pattern.  
안녕하세요?
```

InputStreamReader는 인코딩 지정 가능:  
“UTF-8”, “CP949” 등

# (간단 실습) BasicReadTest.java

강의 자료 참고

```
1 package ch16;
2
3 import java.io.*;
4
5 /**
6  * 읽기 실습
7  */
8 public class BasicReadTest {
9     public static void main(String[] args) {
10         String fileName = "BasicReadTest.java";
11         try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
12             while (reader.ready()) {
13                 String line = reader.readLine();
14                 System.out.println(line);
15             }
16         } catch (Exception ex) {
17             System.err.println(ex);
18         }
19     }
20 }
```

# java.io.Serializable 인터페이스

- <https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html>

클래스의 직렬화 기능은 java.io.Serializable 인터페이스를 구현하는 클래스에 의해 사용 가능합니다. 이 인터페이스를 구현하지 않는 클래스는, 자신의 상태가 직렬화 또는 직렬화 복원되지 않습니다. 직렬화 가능 클래스의 모든 아류 형 자체는 직렬화 가능합니다. 직렬화 인터페이스에는 메서드나 필드가 없으면 직렬화 가능이라는 의미를 식별하는 데에만 사용됩니다.

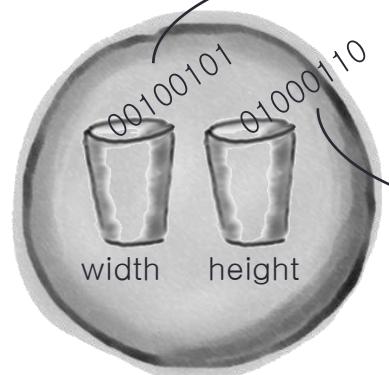
직렬화 할 수 없는 클래스의 부속 유형을 직렬화 할 수 있도록하려면 부속 유형의 상위 유형의 공용, 보호 및 액세스 가능한 패키지 필드의 상태를 저장 및 복원해야합니다. 아류 형은 그것이 확장 한 클래스가 클래스의 상태를 초기화하기 위해 접근 할 수 있는 인수없는 생성자를 갖는 경우에만이 책임을 맡을 수 있습니다. 이러한 경우가 아니라면 Serializable 클래스를 선언하는 것은 오류입니다. 런타임에 오류가 감지됩니다.

직렬화 복원 중, 직렬화 불가능한 클래스의 필드는, 클래스의 public 또는 protected의 인수 없음의 constructor를 사용해 초기화됩니다. 인수가없는 생성자는 직렬화 가능한 서브 클래스에 액세스 할 수 있어야합니다. 직렬화 가능 서브 클래스의 필드는 스트림로부터 복원됩니다.

그래프를 traversal 할 때 Serializable 인터페이스를 지원하지 않는 객체가 발생할 수 있습니다. 이 경우, NotSerializableException가 슬로우되어 직렬화 가능하지 않은 객체의 클래스를 식별합니다.

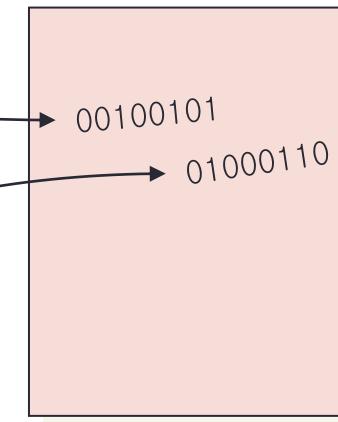
# 직렬화된 객체

힙 안에 들어있는 객체



```
Foo myFoo = new Foo();
myFoo.setWidth(37);
myFoo.setHeight(70);
```

직렬화된 객체



Foo.ser

```
FileOutputStream fs = new FileOutputStream("Foo.ser");
ObjectOutputStream os = new ObjectOutputStream(fs);
os.writeObject(myFoo);
```

# 무엇이 저장될까?

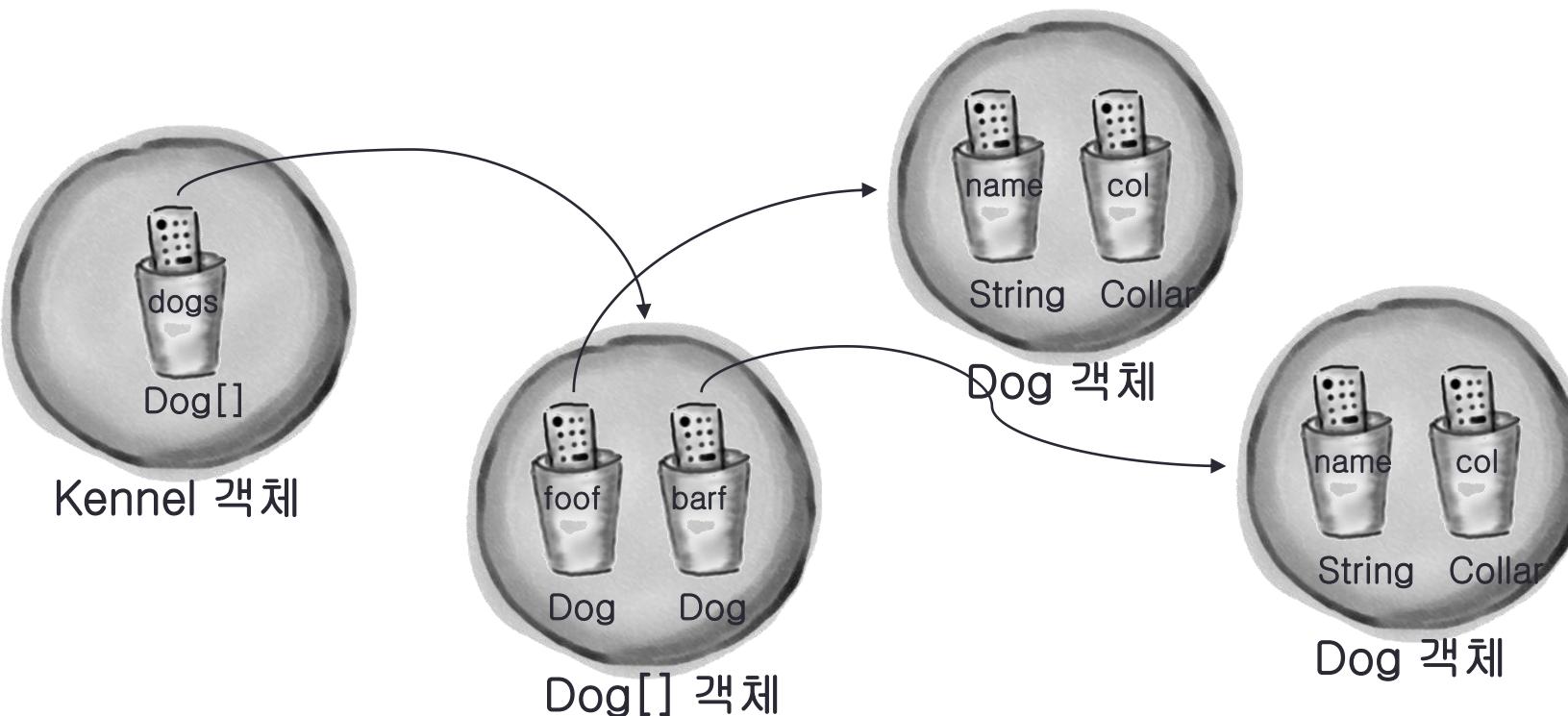
- “객체의 상태”란 정확하게 무엇을 의미할까?
  - 인스턴스 변수 중에 원시 변수가 아닌 레퍼런스 변수가 있다면?
  - 그 레퍼런스 변수가 참조하는 객체에 또 다른 객체를 참조하는 인스턴스 변수가 있다면?
- 저장했을 때와 똑같은 상태의 객체를 얻기 위해 어떤 것이 필요할지 생각해봅시다.

# 바보 같은 질문은 없습니다

- 직렬화할 때 정적 변수도 저장되나요?
  - 그렇지 않습니다. 정적 변수는 클래스마다 하나씩 있는 변수이므로 현재 정적 변수에 들어있는 값을 받게 됩니다.
  - 직렬화할 수 있는 객체를 만들 때는 동적으로 바뀔 수 있는 정적 변수에 의존하지 않도록 만들어야 합니다.

# 객체의 직렬화

- 객체를 직렬화할 때는 그 객체와 관련된 모든 것이 저장
- 인스턴스 변수로 참조된 모든 객체가 줄줄이 엮여서 저장



# Serializable 인터페이스

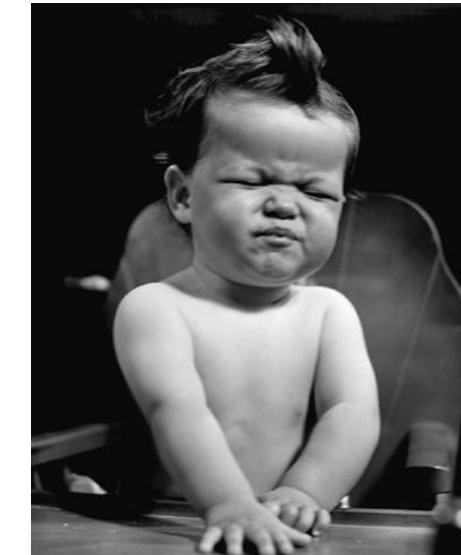
- 클래스를 직렬화할 수 있도록 하고 싶다면 Serializable 인터페이스를 구현해야 합니다.
- 재정의해야 할 메서드가 하나도 없음!
- Serializable 같은 인터페이스는 표지(marker) 인터페이스, 또는 태그(tag) 인터페이스라고 부르기도 합니다.

```
objectOutputStream.writeObject(mySquare);
```

```
import java.io.*;  
  
public class Square implements Serializable {  
    ...  
}
```

# 직렬화가 되지 않으면?

- 어떤 객체를 저장할 때 그 객체와 관련된 것들이 모두 제대로 직렬화되지 않으면 그 직렬화 작업은 제대로 완료되지 않습니다.
- 직렬화하고자 하는 모든 객체가 직렬화할 수 있는(즉 Serializable을 구현하는) 클래스에 속해야 합니다.
- 직렬화가 제대로 되지 않는 경우에는 **NotSerializableException** 예외가 발생합니다.
- 특정 변수를 생략하고 싶다면 어떻게 해야 할까요?



# transient 키워드

- 특정 인스턴스 변수가 직렬화되지 않도록 하고 싶다면 transient 키워드를 쓰면 됩니다.

```
import java.net.*;
import java.io.*;

class Chat implements Serializable {
    transient String currentID;
    String userName;
    // 나머지 코드
}
```

# 바보 같은 질문은 없습니다

- transient로 지정했던 변수는 나중에 역직렬화할 때 어떤 값으로 복구되나요?
  - 무조건 null 값을 가지게 됩니다.
- null이 들어있으면 안 되는 경우의 해결책
  1. 특정 기본값을 정해놓고 복구할 때 항상 기본값을 대입하는 방법
  2. 해당 변수에서 중요한 역할을 하는 값을 따로 저장했다가 그 값을 이용하여 복구하는 방법

# 역직렬화(deserialization)

직렬화: [강의자료 p.5](#) 참고

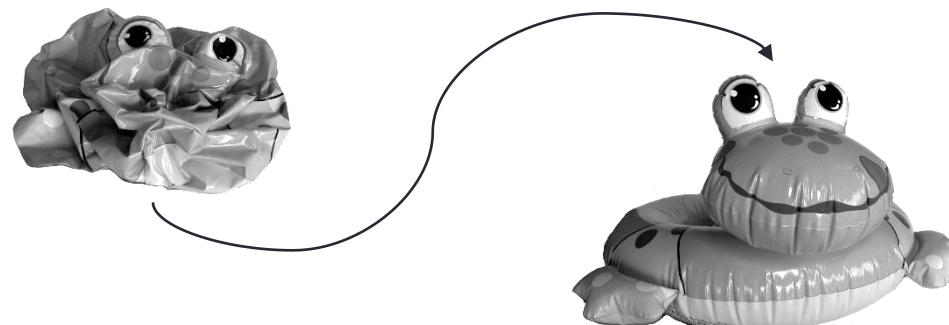
- 직렬화된 객체를 원래 상태로 돌려놓는 것

- FileInputStream 만들기

```
FileInputStream fileStream = new FileInputStream("MyGame.ser");
```

- ObjectInputStream 만들기

```
ObjectInputStream os = new ObjectInputStream(fileStream);
```



# 역직렬화(deserialization)

## 3. 객체 읽기

```
Object one = os.readObject();
```

```
Object two = os.readObject();
```

```
Object three = os.readObject();
```

## 4. 객체 캐스팅

```
GameCharacter elf = (GameCharacter) one;
```

```
GameCharacter troll = (GameCharacter) two;
```

```
GameCharacter magician = (GameCharacter) three;
```

## 5. ObjectInputStream 닫기

```
os.close();
```

# 역직렬화 과정

1. 스트림으로부터 객체를 읽어옴
2. JVM에서 객체의 클래스 유형 결정
3. 클래스를 찾아서 불러오려는 시도를 함. 클래스를 찾거나 불러올 수 없으면 예외 발생
4. 객체가 힙 안에 자리를 잡는데, 이 때 생성자는 실행되지 않습니다.
5. 상속 트리에서 윗부분에 직렬화할 수 없는 클래스가 있으면 그 클래스의 생성자가 실행됩니다.
6. 인스턴스 변수에 직렬화된 값, 또는 기본값이 대입됩니다.

# (참고) SerializationTest.java

강의 자료 참고

```
1 import java.io.*;
2
3
4 public class SerializationTest {
5     public static void main(String[] args) {
6         Person p1 = new Person();
7         Person p2 = new Person("전우치", 20);
8
9         // 객체 쓰기
10        try (FileOutputStream fos = new FileOutputStream("person.ser");
11             ObjectOutputStream oos = new ObjectOutputStream(fos))
12        {
13            oos.writeObject(p1);
14            oos.writeObject(p2);
15        } catch (Exception ex) {
16            System.err.println(ex);
17        }
18    }
19 }
```

```
19 // 객체 읽기
20 try (FileInputStream fis = new FileInputStream("person.ser");
21     ObjectInputStream ois = new ObjectInputStream(fis))
22 {
23     Person p3 = (Person) ois.readObject();
24     Person p4 = (Person) ois.readObject();
25     System.out.printf("p3 = %s, p4 = %s%n", p1, p2);
26 } catch (Exception ex) {
27     System.err.println(ex);
28 }
29 }
30 }
```

```
33 class Person implements Serializable {
34     private String name;
35     private Integer age;
36
37     public Person() {
38         name = "홍길동";
39         age = 25;
40     }
41
42     public Person(String name, Integer age) {
43         this.name = name;
44         this.age = age;
45     }
46
47     @Override
48     public String toString() {
49         return String.format("%s(%d살)", name, age);
50     }
51
52 }
```

\$ java SerializationTest.java  
p3 = 홍길동(25살), p4 = 전우치(20살)

# 버전 ID (p.598)

- 클래스를 변경했을 때 역직렬화 과정에서 **문제가 생길 수 있는 경우**
  - 인스턴스 변수를 삭제한 경우
  - 인스턴스 변수의 유형을 변경한 경우
  - transient로 지정하지 않았던 것을 transient로 지정하는 경우
  - 클래스를 상속 계층에서 위나 아래로 옮기는 경우
  - Serializable이었던 클래스를 그렇지 않은 클래스로 변경하는 경우
  - 인스턴스 변수를 정적 변수로 변경하는 경우

- 클래스를 변경해도 역직렬화와 관련된 **문제가 생기지 않는 경우**
  - 인스턴스 변수를 추가하는 경우
  - 상속 트리에 클래스를 추가하는 경우
  - 상속 트리에서 클래스를 제거하는 경우
  - 인스턴스 변수의 접근 레벨을 역직렬화 과정에서 변수에 값을 대입하는 데 문제가 없는 범위 내에서 변경하는 경우
  - transient로 지정했던 인스턴스 변수를 transient가 아닌 것으로 지정하는 경우

101101  
101101  
1010101010  
1010110101

Dog.class

클래스 버전 ID  
#343

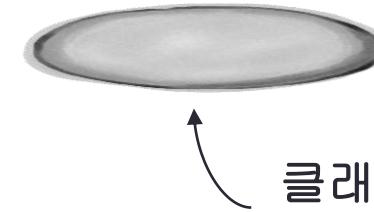


Dog 객체

101101  
10111101  
1010101110  
1010110101

Dog.class

클래스 버전 ID  
#728



클래스 버전 ID #343  
으로 직렬화된 객체

역직렬화가 되지 않습니다.

# serialVersionUID (p.599)

- serialVersionUID
  - 객체를 직렬화할 때 그 객체에 찍히는 클래스의 버전 ID
  - 클래스 구조에 대한 정보를 바탕으로 계산됨
  - serialVersionUID가 다르면 역직렬화가 되지 않음
- 클래스가 바뀔 것 같으면 클래스에 ID를 따로 저장
  - 클래스가 바뀌더라도 버전 ID가 똑같도록 하면 JVM에서 직렬화된 객체와 클래스가 서로 호환되는 것으로 간주함
  - 클래스를 변경할 때 역직렬화 과정에서 문제가 생기지 않도록 주의

## 1. serialver라는 프로그램으로 버전 ID 확인

```
D:\NetBeansProjects\java_maven\target>serialver -classpath ./classes cse.oop2.ch14.quiz.QuizCardPlayer  
Class cse.oop2.ch14.quiz.QuizCardPlayer is not Serializable.
```

```
D:\NetBeansProjects\java_maven\target>serialver -classpath ./classes cse.oop2.ch14.quiz.QuizCard  
cse.oop2.ch14.quiz.QuizCard: private static final long serialVersionUID = 5331707306129796302L;
```

## 2. 출력된 결과를 클래스에 붙여 넣음

```
public class Dog {  
  
    static final long serialVersionUID = 1624322038008032988L;  
  
    private String name;  
  
    private int size;  
  
}
```

## 3. 역직렬화할 때 문제가 생기지 않도록 주의

# 텍스트 파일에 문자열을 저장하는 방법

- 직렬화된 객체를 저장하는 방법

```
objectOutputStream.writeObject(someObject);
```

- String으로 저장하는 방법

```
fileWriter.write("My First String to Save");
```

```
import java.io.*;
class WriteAFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("Foo.txt");
            writer.write("hello foo!");
            writer.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
50,Elf,bow,sword,dust  
200,Troll,bare hands,big ax  
120,Magician,spells,invisibility
```

# 실습: 객체 저장 및 읽어 오기

- 예제1: 단순 객체 저장 실패
- 예제2: 단순 객체 저장 성공 + 단순 객체 적재
- 예제3: 복합 객체 저장과 적재

# 예제1: 단순 객체 저장 실패

- java.io.Serializable 인터페이스를 구현하지 않음
- 패키지: cse.oop2.ch16\_3.example1
  - ObjectOutputStream 객체를 사용하여 객체를 저장
  - 실제 파일에 저장은 FileOutputStream 객체에서 실행
- 클래스
  - Score
    - 필드
      - String name
      - int java
  - TestDrive
    - Score 객체를 생성하여 저장

# Score.java

```
6 package cse.oop2.ch16_3.example1;
7
8     + /**...4 lines */
9
10    public class Score {
11
12        private String name;
13
14        private int java;
15
16
17        - public Score(String name, int java) {
18            this.name = name;
19            this.java = java;
20        }
21
22        - public String getName() {
23            return name;
24        }
25
26        - public int getJava() {
27            return java;
28        }
29
30    }
```

# TestDrive.java

```
6 package cse.oop2.ch16_3.example1;
7
8 import ...4 lines
12
13 /**
17 public class TestDrive {
18
19 /**
20
21 /**
22
23 public static void main(String[] args) throws FileNotFoundException, IOException {
24     Score score1 = new Score("홍길동", 95);
25     Score score2 = new Score("전우치", 92);
26
27     FileOutputStream fos = new FileOutputStream("./src/main/java/cse/oop2/ch14/example1/scores.ser");
28     ObjectOutputStream oos = new ObjectOutputStream(fos);
29
30     oos.writeObject(score1);
31     oos.writeObject(score2);
32
33     oos.close();
34 }
```

# 실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ java_maven ---
Exception in thread "main" java.io.NotSerializableException: cse.oop2.ch14.example1.Score
  at java.base/java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1185)
  at java.base/java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:349)
  at cse.oop2.ch14.example1.TestDrive.main(TestDrive.java:29)
|Command execution failed.
```

## 예제2: 단순 객체 저장 성공

- java.io.Serializable 인터페이스를 구현함 → 실제 할 일은 없음
- 패키지: cse.oop2.ch16\_3.example2
  - ObjectOutputStream 객체를 사용하여 객체를 저장
  - 실제 파일에 저장은 FileOutputStream 객체에서 실행
- 클래스
  - Score
    - 필드
      - String name
      - int java
  - TestDrive
    - Score 객체를 생성하여 저장

# Score.java

```
6  package cse.oop2.ch16_3.example2;
7
8  /**...4 lines */
9
10 public class Score implements java.io.Serializable {
11
12     private String name;
13     private int java;
14
15
16     public Score(String name, int java) {
17         this.name = name;
18         this.java = java;
19     }
20
21
22     public String getName() {
23         return name;
24     }
25
26     public int getJava() {
27         return java;
28     }
29 }
```

# TestDrive.java

```
6  package cse.oop2.ch16_3.example2;
7
8  import java.io.File;
9  import java.io.FileInputStream;
10 import java.io.FileOutputStream;
11 import java.io.IOException;
12 import java.io.ObjectInputStream;
13 import java.io.ObjectOutputStream;
14 import java.util.LinkedList;
15 import java.util.List;
16
17 /**
18  * ...4 lines */
19 public class TestDrive {
20
21     /**
22      * ...4 lines */
23     public static void main(String[] args) throws IOException, ClassNotFoundException {
24         String fileName = "scores.ser";
25         String fileDir = System.getProperty("user.dir") + File.separator
26             + "src/main/java/cse/oop2/ch14/example2".replace("/", File.separator);
27         Score score1 = new Score("홍길동", 95);
28         Score score2 = new Score("전우치", 92);
```

```
34 // 객체 쓰기
35 FileOutputStream fos = new FileOutputStream(fileDir + File.separator + fileName);
36 ObjectOutputStream oos = new ObjectOutputStream(fos);
37 oos.writeObject(score1);
38 oos.writeObject(score2);
39 oos.close();
40
41 // 객체 읽기
42 ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName));
43
44 List<Score> scores = new LinkedList<>();
45 scores.add((Score) ois.readObject());
46 scores.add((Score) ois.readObject());
47
48 for (var score : scores) {
49     System.out.println(score.getName() + ": " + score.getJava());
50 } https://developer.oracle.com/java/jdk-10-local-variable-type-inference.html
51 /* 지역 변수 선언 시 초기화된 코드 사용하면 자료형을 지정 안 해도 됨.
52 scores.forEach(e -> { System.out.println(e.getName() + ": " + e.getJava()); });
53 */
54 ois.close();
55 }
56 }
```

--- exec-maven-p  
홍길동: 95  
전우치: 92

# 시스템 프로퍼티

- 참고 URL: [https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/java.base/java/lang/System.html#getProperties\(\)](https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/java.base/java/lang/System.html#getProperties())
- System.getProperties() : Properties

java.compiler	Name of JIT compiler to use
os.name	Operating system name
os.arch	Operating system architecture
os.version	Operating system version
file.separator	File separator ("/" on UNIX)
path.separator	Path separator ":" on UNIX)
line.separator	Line separator ("\n" on UNIX)
user.name	User's account name
user.home	User's home directory
user.dir	User's current working directory

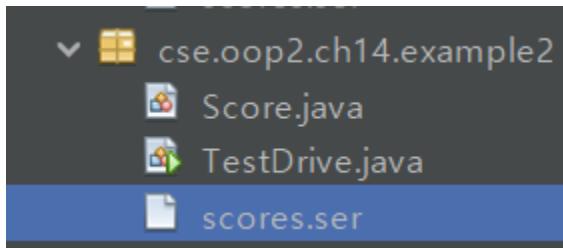
**Module** java.base  
**Package** java.util  
**Class Properties**

java.lang.Object  
java.util.Dictionary<K,V>  
java.util.Hashtable<Object, Object>  
java.util.Properties

## [참고]

```
Enumeration e = System.getProperties().propertyNames();
while (e.hasMoreElements()) {
    System.out.println(e.nextElement());
}
```

# 실행 결과

A screenshot of a Java IDE showing the contents of the 'scores.ser' file. The title bar says 'scores.ser'. The code editor displays the following byte array representation of the serialized object:

```
1  ȏ\NUL\ENOSr\NUL\FScse.oop2.ch14.example2.Score.\xFF.\xD9\ESC'喀\STX\NUL\STX\NUL\FO\TjavaL\NUL\FO\Tna
met\NUL\DC2Ljava\lang\String;xp\NUL\NUL\NUL_t\NUL    ?뚯만?\NUL~\NUL\NUL\NUL\NUL\NUL\Wt\NUL
?߮\NUL\x98
```

The content is mostly non-printable characters and some recognizable Java class names like 'Score' and 'String'.

# 예제3: 복합 객체 저장과 적재

- java.io.Serializable 인터페이스를 구현함 → 실제 할 일은 없음
- 패키지: cse.oop2.ch16\_3.example3
  - ObjectOutputStream 객체를 사용하여 객체를 저장
  - 실제 파일에 저장은 FileOutputStream 객체에서 실행
  - ObjectInputStream을 사용하여 객체 읽기
  - 실제 객체가 저장된 파일을 읽는 것은 FileInputStream 객체
- 클래스
  - Score
    - 필드
      - *Person person (String name; String id;)*
      - int java
  - TestDrive
    - Score 객체를 생성하여 저장

# Person.java

```
6  package cse.oop2.ch16_3.example3;
7
8  /**...4 lines */
12 public class Person implements java.io.Serializable {
13
14     private String name;
15     private String id;
16
17     public Person(String name, String id) {
18         this.name = name;
19         this.id = id;
20     }
21
22     public String getName() {
23         return name;
24     }
25
26     public String getId() {
27         return id;
28     }
29 }
```

# Score.java

```
6 package cse.oop2.ch16_3.example3;
7
8 /**
9  * ...4 lines ...
10 */
11
12 public class Score implements java.io.Serializable {
13
14     private Person person;
15     private int java;
16
17     public Score(Person person, int java) {
18         this.person = person;
19         this.java = java;
20     }
21
22     public Person getPerson() {
23         return person;
24     }
25
26     public int getJava() {
27         return java;
28     }
29 }
```

# TestDrive.java

```
6  package cse.oop2.ch14.example3;
7
8  import java.io.File;
9  import java.io.FileInputStream;
10 import java.io.FileNotFoundException;
11 import java.io.FileOutputStream;
12 import java.io.IOException;
13 import java.io.ObjectInputStream;
14 import java.io.ObjectOutputStream;
15 import java.util.LinkedList;
16 import java.util.List;
17
18 /**
19  * ...4 lines */
20 public class TestDrive {
21
22
23 /**
24  * ...3 lines */
25
26
27 /**
28  * ...
29  * @param args
30  * @throws FileNotFoundException
31  * @throws IOException
32  * @throws ClassNotFoundException
33  */
34 public static void main(String[] args) throws FileNotFoundException, IOException,
35                                         ClassNotFoundException {
36     String fileName = "scores.ser";
37     String fileDir = System.getProperty("user.dir") + File.separator
38             + "src/main/java/cse/oop2/ch14/example3".replace("/", File.separator);
39     System.out.println("fileDir = " + fileDir);
```

```
34     Person p1 = new Person("홍길동", "101");
35     Person p2 = new Person("전우치", "102");
36     Score score1 = new Score(p1, 95);
37     Score score2 = new Score(p2, 92);
38
39     // 객체 쓰기
40     FileOutputStream fos = new FileOutputStream(fileDir + File.separator + fileName);
41     
42     ObjectOutputStream oos = new ObjectOutputStream(fos);
43     oos.writeObject(score1);
44     oos.writeObject(score2);
45     oos.close();
46
47     // 객체 읽기
48     ObjectInputStream ois = new ObjectInputStream(
49         new FileInputStream(fileDir + File.separator + fileName));
50     List<Score> scores = new LinkedList<>();
51     scores.add((Score) ois.readObject());
52     scores.add((Score) ois.readObject());
53     
54     for (Score score : scores) {
55         System.out.println(score.getPerson().getName() + "("
56             + score.getPerson().getId() + "): " + score.getJava());
57     }
58 }
```

System.out.format("%s (%d): %d%n",  
score.getPerson().getName(),  
score.getPerson().getId(),  
score.getJava());

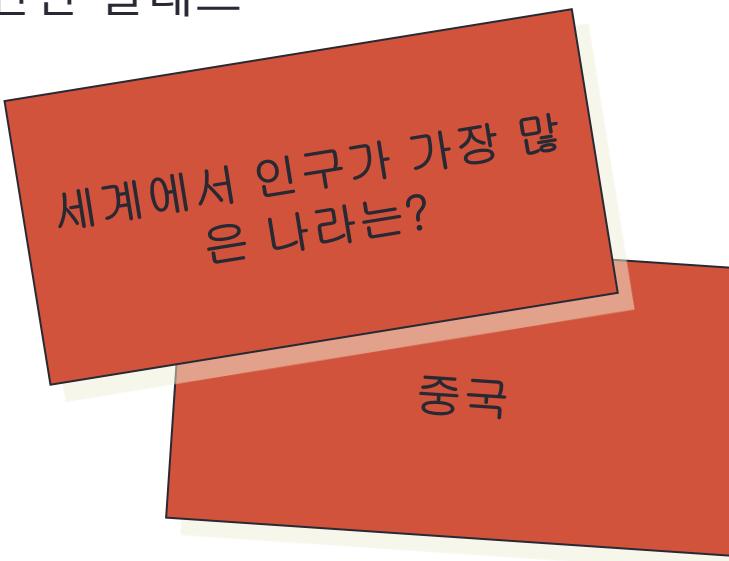
# 실행 화면

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ java_maven ---
fileDir = D:\NetBeansProjects\java_maven\src\main\java\cse\woop2\ch14\example3
홍길동 (101): 95
전우치 (102): 92
```

# 암기장 프로그램 (p.602)

- QuizCardBuilder
  - 암기장을 만들고 저장하기 위한 간단한 도구
- QuizCardReader
  - 암기장을 불러와서 사용자에게 질문을 해 주는 클래스
- QuizCard
  - 카드 데이터를 나타내는 간단한 클래스

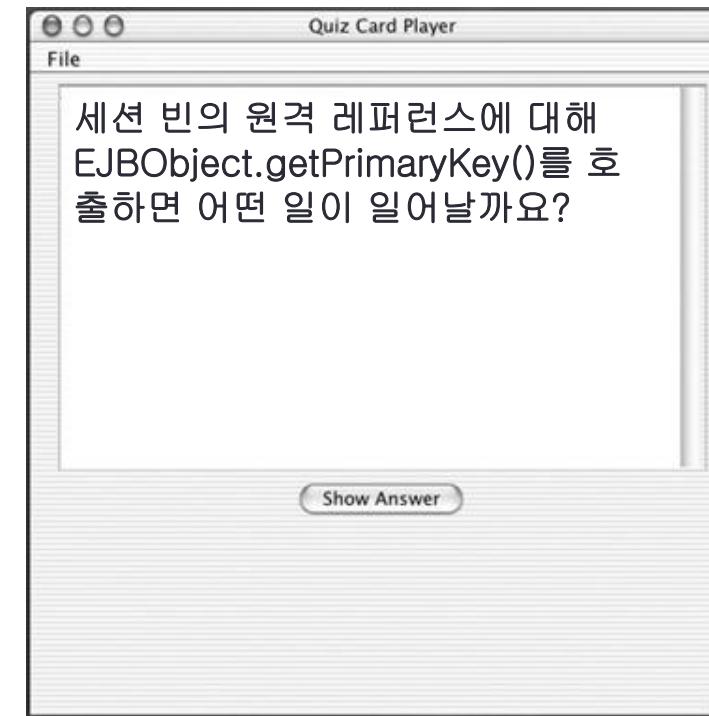
QuizCard
question
answer
QuizCard(q, a)
getQuestion()
getAnswer()



# 암기장 프로그램



QuizCardBuilder



QuizCardReader

# QuizCardBuilder 개요 (p.603)

- go() 메서드
  - GUI를 만들고 화면에 표시함
- NextCardListener 내부 클래스
  - Next Card 버튼을 클릭했을 때 필요한 메서드 정의
- SaveMenuListener 내부 클래스
  - 메뉴에서 Save를 선택했을 때 필요한 메서드 정의
- NewMenuListener 내부 클래스
  - 메뉴에서 New를 선택했을 때 필요한 메서드 정의
- saveFile() 메서드
  - 파일에 내용을 저장하는 작업을 처리

# QuizCardBuilder.java

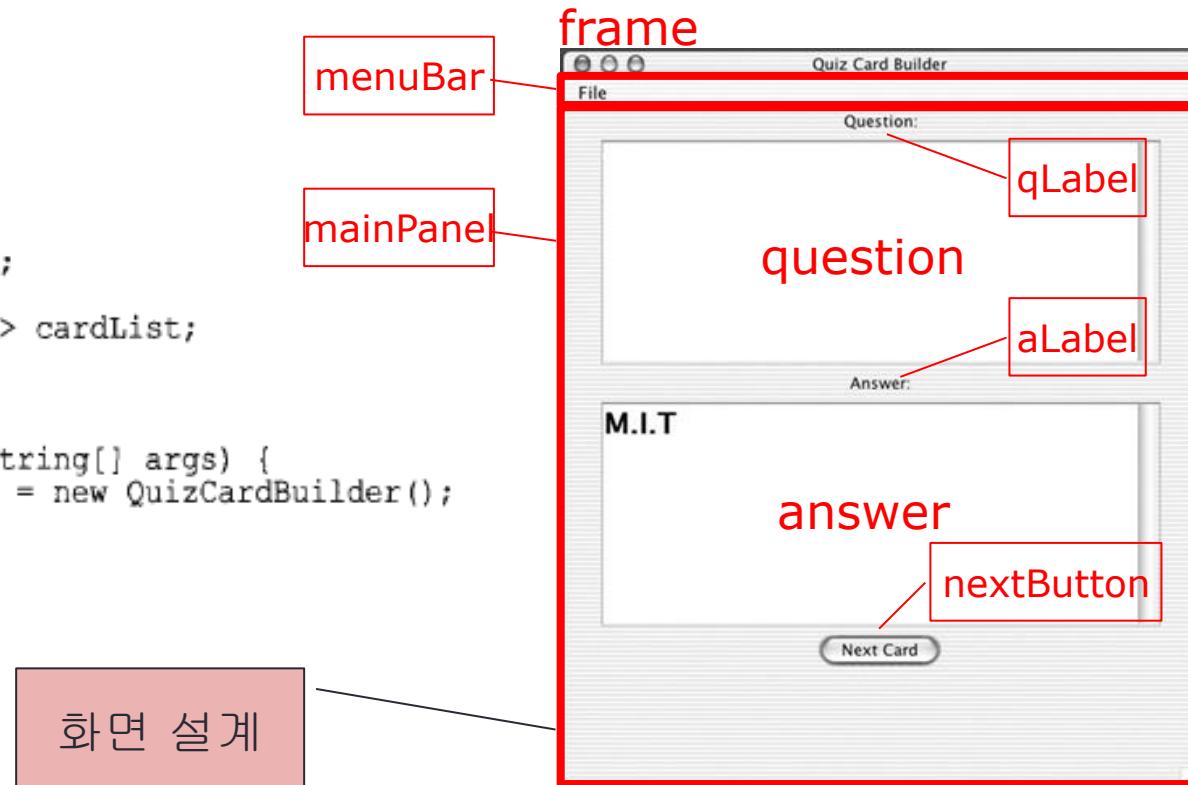
- 임포트, 클래스 정의, 필드(인스턴스 변수) 선언, main 메서드

```
import java.util.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.*;
import java.io.*;

public class QuizCardBuilder {

    private JTextArea question;
    private JTextArea answer;
    private ArrayList<QuizCard> cardList;
    private JFrame frame;

    public static void main (String[] args) {
        QuizCardBuilder builder = new QuizCardBuilder();
        builder.go();
    }
}
```



- 화면 구성 (1/2)

```
public void go() {
    // build gui

    frame = new JFrame("Quiz Card Builder");
    JPanel mainPanel = new JPanel();
    Font bigFont = new Font("sanserif", Font.BOLD, 24);
    question = new JTextArea(6,20);
    question.setLineWrap(true);
    question.setWrapStyleWord(true);
    question.setFont(bigFont);

    JScrollPane qScroller = new JScrollPane(question);
    qScroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    qScroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

    answer = new JTextArea(6,20);
    answer.setLineWrap(true);
    answer.setWrapStyleWord(true);
    answer.setFont(bigFont);

    JScrollPane aScroller = new JScrollPane(answer);
    aScroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    aScroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

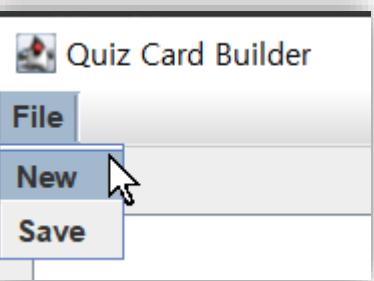
This is all GUI code here. Nothing special, although you might want to look at the `MenuBar`, `Menu`, and `MenuItem`s code.

## • 화면 구성 (2/2)

```

JButton nextButton = new JButton("Next Card");
cardList = new ArrayList<QuizCard>();
JLabel qLabel = new JLabel("Question:");
JLabel aLabel = new JLabel("Answer:");
mainPanel.add(qLabel);
mainPanel.add(qScroller);
mainPanel.add(aLabel);
mainPanel.add(aScroller);
mainPanel.add(nextButton);
nextButton.addActionListener(new NextCardListener());
JMenuBar menuBar = new JMenuBar();
JMenu fileMenu = new JMenu("File");
JMenuItem newItem = new JMenuItem("New");
JMenuItem saveMenuItem = new JMenuItem("Save");
newItem.addActionListener(new NewMenuItemListener());
saveMenuItem.addActionListener(new SaveMenuItemListener());
fileMenu.add(newItem);
fileMenu.add(saveMenuItem);
menuBar.add(fileMenu);
frame.setJMenuBar(menuBar);
frame.getContentPane().add(BorderLayout.CENTER, mainPanel);
frame.setSize(500, 600);
frame.setVisible(true);
}

```



The screenshot shows a Java Swing application window titled "Quiz Card Builder". The window has a menu bar with a single item "File". Below the menu bar, there is a toolbar with two buttons: "New" and "Save". The "New" button is highlighted with a blue selection bar.

menuBar > fileMenu > {newMenuItem, saveMenuItem}

*We make a menu bar, make a File menu, then put 'new' and 'save' menu items into the File menu. We add the menu to the menu bar, then tell the frame to use this menu bar. Menu items can fire an ActionEvent*



- ActionListener 인터페이스 구현하는 내부 클래스

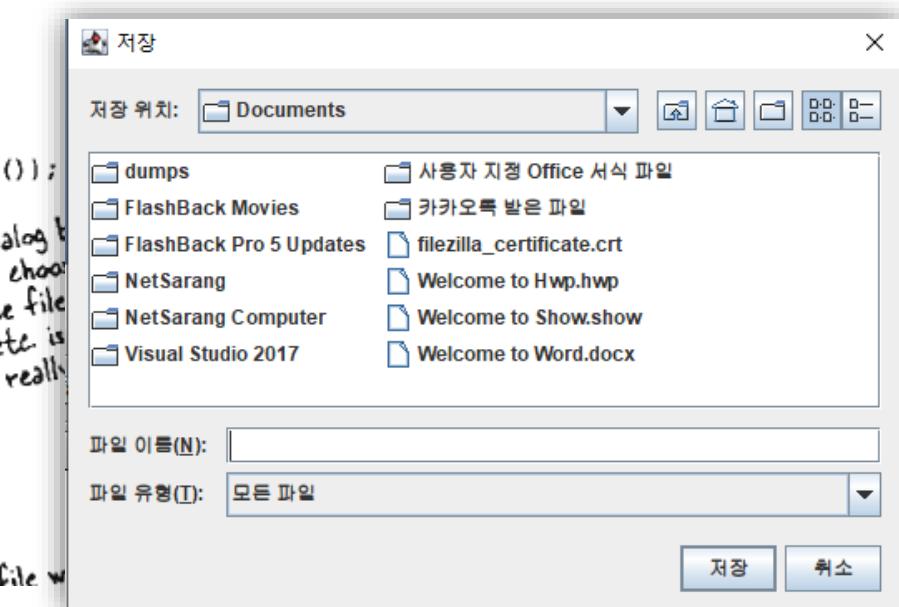
```
public class NextCardListener implements ActionListener {
    public void actionPerformed(ActionEvent ev) {
        QuizCard card = new QuizCard(question.getText(), answer.getText());
        cardList.add(card);
        clearCard();
    }
}

public class SaveMenuListener implements ActionListener {
    public void actionPerformed(ActionEvent ev) {
        QuizCard card = new QuizCard(question.getText(), answer.getText());
        cardList.add(card);

        JFileChooser fileSave = new JFileChooser();
        fileSave.showSaveDialog(frame);
        saveFile(fileSave.getSelectedFile()); ←
    }
}

public class NewMenuListener implements ActionListener {
    public void actionPerformed(ActionEvent ev) {
        cardList.clear();
        clearCard();
    }
}
```

Brings up a file dialog line until the user chooses a file. All the file dialog box. All the file selecting a file, etc. is JFileChooser! It really



- private 메서드 구현

```
        }

    private void clearCard() {
        question.setText("");
        answer.setText("");
        question.requestFocus();
    }

    private void saveFile(File file) {
        try (BufferedWriter writer = new BufferedWriter(
            new OutputStreamWriter(
                new FileOutputStream(file), "UTF-8"))) {
            // BufferedWriter writer = new BufferedWriter(new FileWriter(file));
            for (var card : cardList) {
                writer.write(card.getQuestion() + "/");
                writer.write(card.getAnswer() + "Wn");
            }
            // writer.close();
        } catch (IOException ex) {
            System.out.println("couldn't write the cardList out");
            ex.printStackTrace();
        }
    } // close method
```

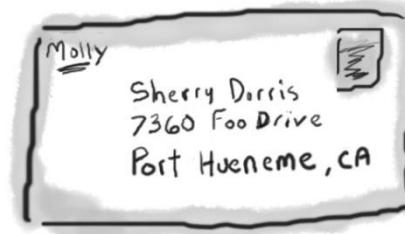
The method that does the actual file writing.  
(called by the SaveMenuItem's event handler).  
The argument is the 'File' object the user is saving.  
We'll look at the File class on the next page.

# QuizCard.java

```
1  package cse.oop2.ch14.quiz;
2
3  import java.io.*;
4
5  public class QuizCard implements Serializable {
6
7      private String question;
8      private String answer;
9
10     public QuizCard(String q, String a) {
11         question = q;
12         answer = a;
13     }
14
15     public void setQuestion(String q) {
16         question = q;
17     }
18
19     public String getQuestion() {
20         return question;
21     }
22
23     public void setAnswer(String a) {
24         answer = a;
25     }
26
27     public String getAnswer() {
28         return answer;
29     }
30 }
```

# java.io.File 클래스 (p.606)

- File 클래스
  - 디스크에 있는 “파일”을 나타내는 클래스



1. 이미 존재하는 파일을 나타내는 File 객체 만들기

```
File f = new File("MyCode.txt");
```

2. 새 디렉토리 만들기

```
File dir = new File("Chapter14");
```

**dir.mkdir();**

3. 디렉토리에 들어있는 내용의 목록 출력

```
if (dir.isDirectory()) {  
    String[] dirContents = dir.list();  
    for (int i = 0; i < dirContents.length; i++) {  
        System.out.println(dirContents[i]);  
    }  
}
```

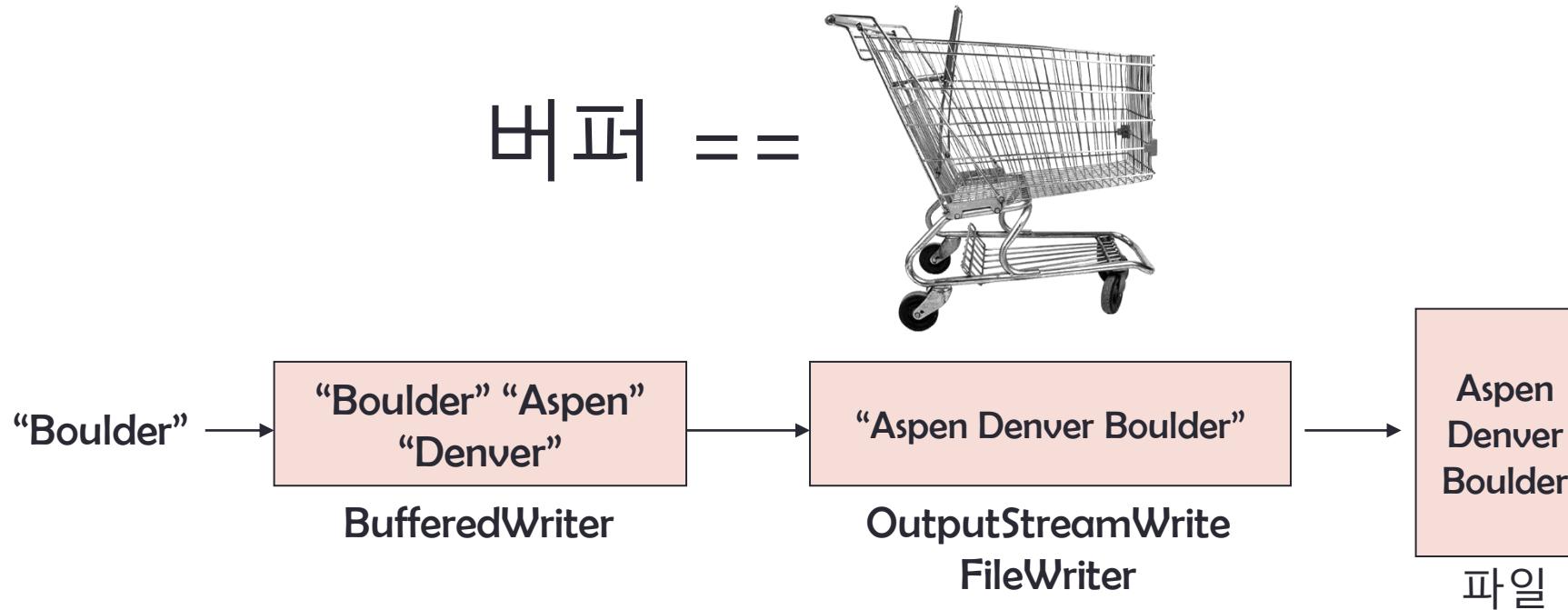
4. 파일 또는 디렉토리의 절대 경로명 구하기

```
System.out.println(dir.getAbsolutePath());
```

5. 파일 또는 디렉토리 삭제

```
boolean isDeleted = f.delete();
```

# 버퍼(buffer) (p.607)



```
BufferedWriter writer = new BufferedWriter(new FileWriter(aFile)); // or  
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(  
        new FileOutputStream(aFile), "UTF-8"));
```

# BufferedWriter 클래스의 메서드

void	<b>close()</b>	Closes the stream, flushing it first.
void	<b>flush()</b>	Flushes the stream.
void	<b>newLine()</b>	Writes a line separator.
void	<b>write(char[] cbuf, int off, int len)</b>	Writes a portion of an array of characters.
void	<b>write(int c)</b>	Writes a single character.
void	<b>write(String s, int off, int len)</b>	Writes a portion of a String.

# 텍스트 파일 읽기 (p.608)

```
import java.io.*;  
  
class ReadAFile {  
    public static void main(String[] args) {  
        try {  
            File myFile = new File("MyText.txt");  
            FileReader fileReader = new FileReader(myFile);  
            BufferedReader reader = new BufferedReader(fileReader); // or  
            /*  
             *  
             *  
             *  
             *  
             *  
             */  
            BufferedReader reader = new BufferedReader(  
                new InputStreamReader(  
                    new FileInputStream("MyText.txt"), "UTF-8"));  
        }  
  
        String line = null;                                // import java.nio.file.*; → NIO.2 사용  
        while ((line = reader.readLine()) != null)  
            System.out.println(line);  
        reader.close();  
    } catch (Exception ex) {  
        ex.printStackTrace();  
    }  
}
```

Files.lines(Path.of("MyText.txt"))  
.forEach(line -> System.out.println(line));

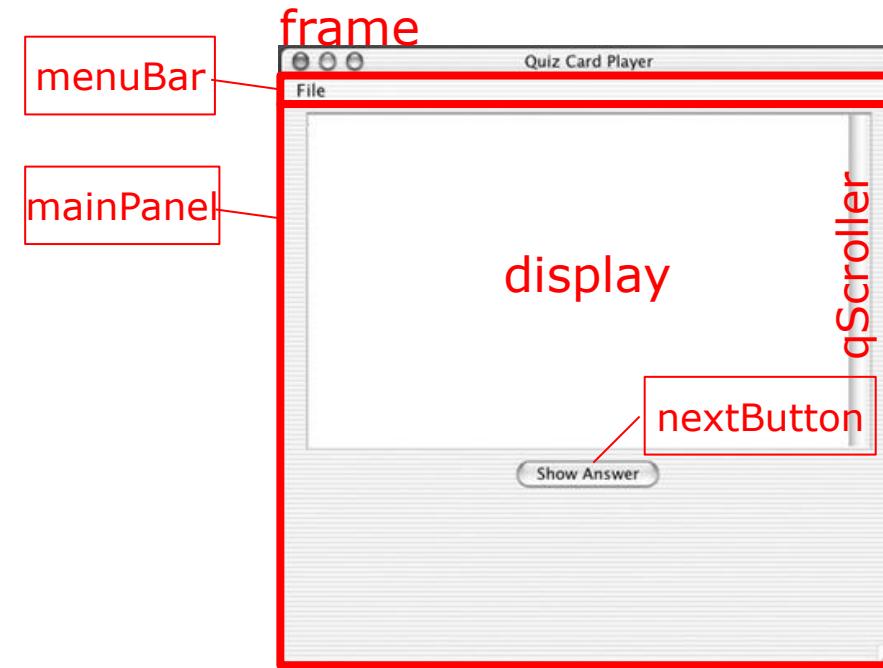
# QuizCardPlayer 개요 (p.609)

- go() 메서드
  - GUI를 만들고 화면에 표시함
- NextCardListener 내부 클래스
  - Next Card 버튼을 클릭했을 때 필요한 메서드 정의
- OpenMenuListener 내부 클래스
  - 메뉴에서 Open을 선택했을 때 필요한 메서드 정의
- loadFile() 메서드
  - 파일을 불러오고 ArrayList를 만드는 작업을 처리
- makeCard() 메서드
  - 텍스트 파일에서 한 행씩 읽어와서 문제와 답으로 파싱 후 CardList라는 ArrayList에 저장함

# QuizCardPlayer.java

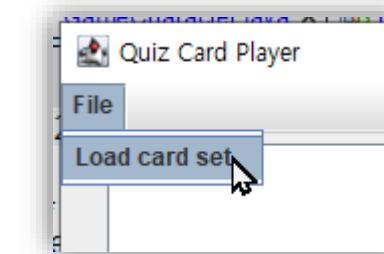
- 임포트, 클래스 정의, 필드 (인스턴스 변수) 선언, main 메서드

```
1 package cse.oop2.ch14;
2
3 import java.util.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import java.awt.*;
7 import java.io.*;
8
9 public class QuizCardPlayer {
10
11     private JTextArea display;
12     private ArrayList cardList;
13     private QuizCard currentCard;
14     private Iterator cardIterator;
15     private JFrame frame;
16     private JButton nextButton;
17     private boolean isShowAnswer;
18
19     public static void main(String[] args) {
20         QuizCardPlayer qReader = new QuizCardPlayer();
21         qReader.go();
22     }
}
```



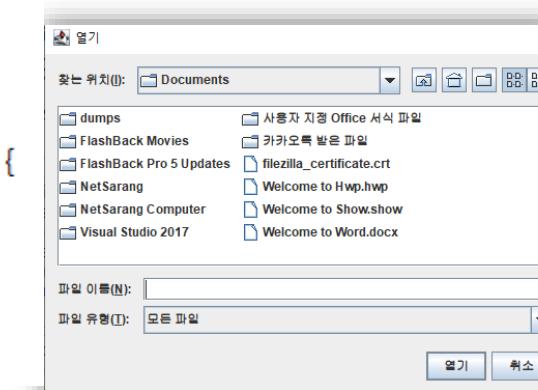
- GUI 코드

```
24 public void go() {  
25  
26     frame = new JFrame("Quiz Card Player");  
27     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
28     JPanel mainPanel = new JPanel();  
29     Font bigFont = new Font("sanserif", Font.BOLD, 24);  
30  
31     display = new JTextArea(9, 20);  
32     display.setFont(bigFont);  
33     display.setLineWrap(true);  
34     display.setWrapStyleWord(true);  
35     display.setEditable(false);  
36  
37     JScrollPane qScroller = new JScrollPane(display);  
38     qScroller.setVerticalScrollBarPolicy(  
39         ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);  
40     qScroller.setHorizontalScrollBarPolicy(  
41         ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);  
42  
43     nextButton = new JButton("Show Question");  
44  
45     mainPanel.add(qScroller);  
46     mainPanel.add(nextButton);  
47     nextButton.addActionListener(new NextCardListener());  
48  
49     JMenuBar menuBar = new JMenuBar();  
50     JMenu fileMenu = new JMenu("File");  
51     JMenuItem loadMenuItem = new JMenuItem("Load card set");  
52     loadMenuItem.addActionListener(new OpenMenuItemListener());  
53     fileMenu.add(loadMenuItem);  
54     menuBar.add(fileMenu);  
55  
56     frame.setJMenuBar(menuBar);  
57     frame.getContentPane().add(BorderLayout.CENTER, mainPanel);  
58     frame.setSize(500, 600);  
59     frame.setVisible(true);  
60 } // close go
```



- ActionListener 인터페이스 구현하는 내부 클래스

```
62  public class NextCardListener implements ActionListener {  
63      public void actionPerformed(ActionEvent ev) {  
64          if (isShowAnswer) {  
65              // show the answer because they've seen the question  
66              display.setText(currentCard.getAnswer());  
67              nextButton.setText("Next Card");  
68              isShowAnswer = false;  
69          } else {  
70              // show the next question  
71              if (cardIterator.hasNext()) {  
72                  showNextCard();  
73              } else {  
74                  // there are no more cards!  
75                  display.setText("That was last card");  
76                  // nextButton.disable();  
77                  nextButton.setEnabled(false);  
78              }  
79          } // close if  
80      } // close method  
81  } // close inner class  
  
83  public class OpenMenuListener implements ActionListener {  
84      public void actionPerformed(ActionEvent ev) {  
85          JFileChooser fileOpen = new JFileChooser();  
86          fileOpen.showOpenDialog(frame);  
87          loadFile(fileOpen.getSelectedFile());  
88      }  
89  }
```



- private 메서드 정의

```
91  private void loadFile(File file) {
92      cardList = new ArrayList();
93      try (BufferedReader reader = new BufferedReader(
94          new InputStreamReader(new FileInputStream(file), "UTF-8"))) {
95          // BufferedReader reader = new BufferedReader(new FileReader(file));
96
97          String line = null;
98          while ((line = reader.readLine()) != null) {
99              makeCard(line);
100         }
101         // reader.close();
102     } catch (Exception ex) {
103         System.out.println("couldn't read the card file");
104         ex.printStackTrace();
105     }
106     // now time to start
107     cardIterator = cardList.iterator();
108     showNextCard();
109 }
110
111 private void makeCard(String lineToParse) {
112     StringTokenizer parser = new StringTokenizer(lineToParse, "/");
113     if (parser.hasMoreTokens()) {
114         QuizCard card = new QuizCard(parser.nextToken(), parser.nextToken());
115         cardList.add(card);
116     }
117 }
118
119 private void showNextCard() {
120     currentCard = (QuizCard) cardIterator.next();
121     display.setText(currentCard.getQuestion());
122     nextButton.setText("Show Answer");
123     isShowAnswer = true;
124 }
125 } // close class
```

# java.util.StringTokenizer 클래스\*

- 문자열을 토큰으로 나누는 기능 수행
- 일반적인 사용 방법

```
StringTokenizer st = new StringTokenizer("this is a test");
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

```
// String.split()을 사용 : '\s' – tab, space, form feed 등
String[] result = "this is a test".split("\s"); // "\s+", "\s+"와의 차이점은?
for (int x=0; x<result.length; x++) {
    System.out.println(result[x]);
}
```

(참고) <https://devdocs.io/openjdk~11/java.base/java/util/regex/pattern#sum>

# String의 split() 메서드

파란색과 녹색을 더하면?

녹색

파란색과 노란색을 더하면?/녹색  
빨간색과 파란색을 더하면?/보라색

- String[] split(String regex)
  - String 객체와 구분자를 지정하면 알아서 조각을 내 주는 기능 수행

파란색과 노란색을 더하면?

/

녹색

```
String toTest = "파란색과 노란색을 더하면?/녹색"  
String[] result = toTest.split("/");  
for (String token : result) {  
    System.out.println(token);  
}
```

# NIO.2를 이용한 BufferedWriter 생성 방법

- 임포트

```
import java.nio.file.*;
```

- Path 객체 생성

- **Path myPath = Paths.get("MyFile.txt");**

- 또는 Path myPath = Paths.get("/myApp", "files", "MyFile.txt"); // "/myapp/files/MyFile.txt"

- BufferedWriter 객체 생성

- BufferedWriter writer = **Files.newBufferedWriter(myPath);**

# 디렉터리 처리

```

import java.nio.file.*;
public class Install {
    public static void main(String[] args) {
        try {
            Path myPath = Paths.get("MyApp");
            Path myPath2 = Paths.get("MyApp", "media");
            Path myPath3 = Paths.get("MyApp", "source");
            Path mySource = Paths.get("MyApp.class");
            Path myMedia = Paths.get("MyMedia.jpeg");

            Files.createDirectory(myPath);
            Files.createDirectory(myPath2);
            Files.createDirectory(myPath3);
            Files.move(mySource, myPath3.resolve(mySource.getFileName()));
            Files.move(myMedia, myPath2.resolve(myMedia.getFileName()));

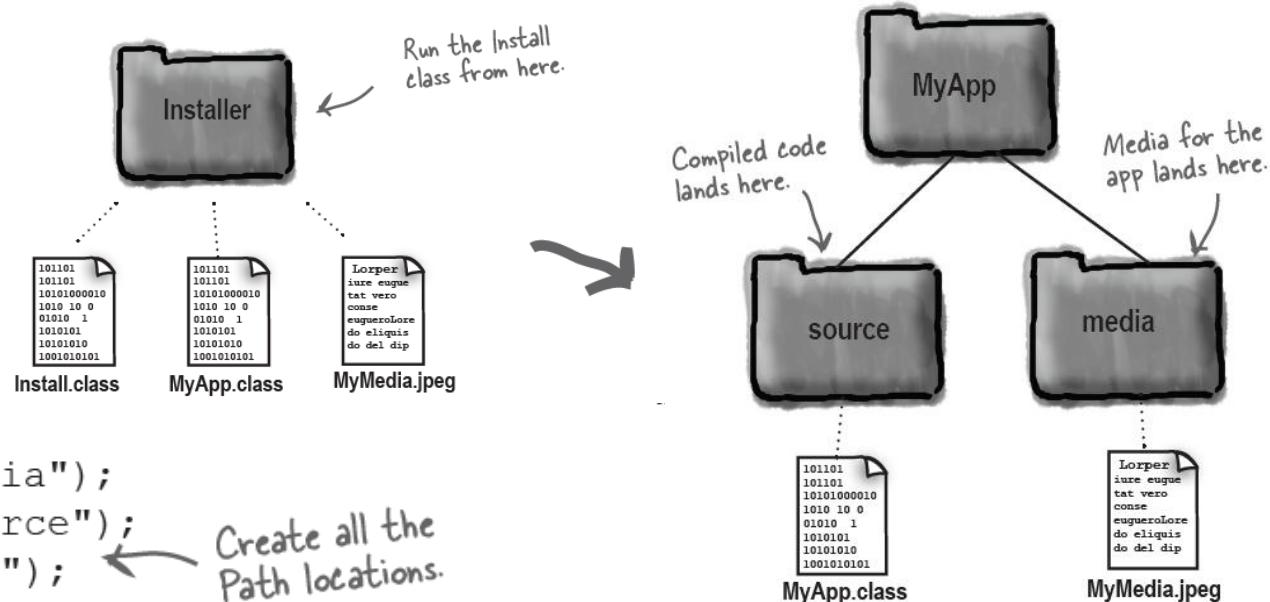
        } catch (Exception e) {
            System.out.println("Got an NIO Exception" + e.getMessage());
        }
    }
}

```

*Create all the Path locations.*

*Create the three new directories.*

*Move the two files into their new directories.*



# jshell 실습

```
$ jshell  
jshell> import java.nio.file.*;
```

```
jshell> Path myPath3 = Paths.get("MyApp", "source");  
myPath3 ==> MyApp#source
```

```
jshell> Path mySource = Paths.get("MyApp.class");  
mySource ==> MyApp.class
```

```
jshell> myPath3.resolve(mySource.getFileName());  
$4 ==> MyApp#source#MyApp.class
```

# 파일 저장

```
private void saveFile(File file) {  
    BufferedWriter writer = null;  
    try {  
        writer = new BufferedWriter(new FileWriter(file));  
  
        for (QuizCard card : cardList) {  
            writer.write(card.getQuestion() + "/");  
            writer.write(card.getAnswer() + "\n");  
        }  
  
    } catch (IOException e) {  
        System.out.println("Couldn't write the cardList out: " + e.getMessage());  
    } finally {  
        try {  
            writer.close();  
        } catch (Exception e) {  
            System.out.println("Couldn't close writer: " + e.getMessage());  
        }  
    }  
}
```

Old style,  
try-catch-finally  
code

```
private void saveFile(File file) {    Autocloseable 인터페이스!!!
    try (BufferedWriter writer =
        new BufferedWriter(new FileWriter(file))) {
        for (QuizCard card : cardList) {
            writer.write(card.getQuestion() + "/");
            writer.write(card.getAnswer() + "\n");
        }
    } catch (IOException e) {
        System.out.println("Couldn't write the cardList out: " + e.getMessage());
    }
}
```

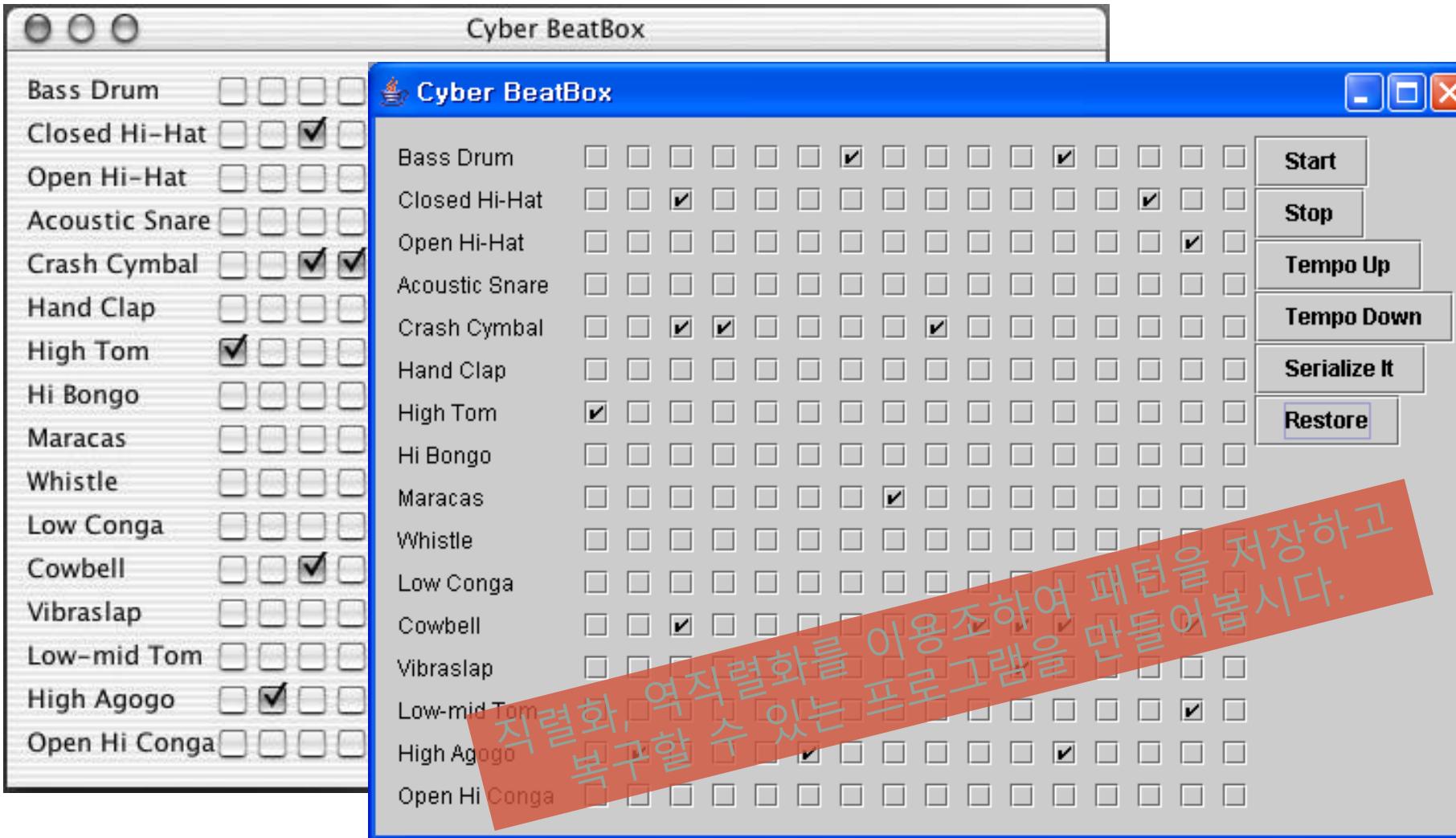
Modern,  
try-with-resources  
code

# 두 개 이상의 자원 사용 가능

```
try (BufferedWriter writer =  
      new BufferedWriter(new FileWriter(file)) ;  
    BufferedReader reader =  
      new BufferedReader(new FileReader(file)) {
```

Separate the resources  
using semicolons, ";".

# 패턴을 저장할 수 있는 비트박스 프로그램



# BeatBoxSaveOnly.java

```
1 package cse.oop2.ch14;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.io.*;
6 import javax.sound.midi.*;
7 import java.util.*;
8 import java.awt.event.*;
9
10 public class BeatBoxSaveOnly { // implements MetaEventListerner
11
12     JPanel mainPanel;
13     ArrayList<JCheckBox> checkboxList;
14     // int bpm = 120;
15     Sequencer sequencer;
16     Sequence sequence;
17     Sequence mySequence = null;
18     Track track;
19     JFrame theFrame;
20
21     String[] instrumentNames = {"Bass Drum", "Closed Hi-Hat",
22         "Open Hi-Hat", "Acoustic Snare", "Crash Cymbal", "Hand Clap",
23         "High Tom", "Hi Bongo", "Maracas", "Whistle", "Low Conga",
24         "Cowbell", "Vibraslap", "Low-mid Tom", "High Agogo",
25         "Open Hi Conga"};
26     int[] instruments = {35, 42, 46, 38, 49, 39, 50, 60, 70, 72, 64, 56, 58, 47, 67, 63};
```

```
28     public static void main(String[] args) {
29         new BeatBoxSaveOnly().buildGUI();
30     }
31
32     public void buildGUI() {
33         theFrame = new JFrame("Cyber BeatBox");
34         theFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35         BorderLayout layout = new BorderLayout();
36         JPanel background = new JPanel(layout);
37         background.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
38
39         checkboxList = new ArrayList<JCheckBox>();
40         Box buttonBox = new Box(BoxLayout.Y_AXIS);
41
42         JButton start = new JButton("Start");
43         start.addActionListener(new MyStartListener());
44         buttonBox.add(start);
45
46         JButton stop = new JButton("Stop");
47         stop.addActionListener(new MyStopListener());
48         buttonBox.add(stop);
49
50         JButton upTempo = new JButton("Tempo Up");
51         upTempo.addActionListener(new MyUpTempoListener());
52         buttonBox.add(upTempo);
53
54         JButton downTempo = new JButton("Tempo Down");
55         downTempo.addActionListener(new MyDownTempoListener());
56         buttonBox.add(downTempo);
57
58         JButton saveIt = new JButton("Serialize It"); // new button
59         saveIt.addActionListener(new MySendListener());
60         buttonBox.add(saveIt);
```

```
62     JButton restore = new JButton("Restore"); // new button
63     restore.addActionListener(new MyReadInListener());
64     buttonBox.add(restore);
65
66     Box nameBox = new Box(BoxLayout.Y_AXIS);
67     for (int i = 0; i < 16; i++) {
68         nameBox.add(new Label(instrumentNames[i]));
69     }
70
71     background.add(BorderLayout.EAST, buttonBox);
72     background.add(BorderLayout.WEST, nameBox);
73
74     theFrame.getContentPane().add(background);
75
76     GridLayout grid = new GridLayout(16, 16);
77     grid.setVgap(1);
78     grid.setHgap(2);
79     mainPanel = new JPanel(grid);
80     background.add(BorderLayout.CENTER, mainPanel);
81
82     for (int i = 0; i < 256; i++) {
83         JCheckBox c = new JCheckBox();
84         c.setSelected(false);
85         checkboxList.add(c);
86         mainPanel.add(c);
87     } // end loop
88
89     setUpMidi();
90
91     theFrame.setBounds(50, 50, 300, 300);
92     theFrame.pack();
93     theFrame.setVisible(true);
94 } // close method
```

```
96  ↴ public void setUpMidi() {
97    try {
98      sequencer = MidiSystem.getSequencer();
99      sequencer.open();
100     // sequencer.addMetaEventListener(this);
101     sequence = new Sequence(Sequence.PPQ, 4);
102     track = sequence.createTrack();
103     sequencer.setTempoInBPM(120);
104
105     ↪
106     ↪ } catch (Exception e) {
107       e.printStackTrace();
108     }
109   } // close method
110
111  /* public class MyCheckBoxListener implements ItemListener {
112    public void itemStateChanged(ItemEvent ev) {
113      // might add real-time removal or addition, probably not because of timing
114    }
115  } // close inner class
116
117  ↴ public void buildTrackAndStart() {
118    // this will hold the instruments for each vertical column,
119    // in other words, each tick (may have multiple instruments)
120    int[] trackList = null;
121
122    sequence.deleteTrack(track);
123    track = sequence.createTrack();
```

```
125     for (int i = 0; i < 16; i++) {
126         trackList = new int[16];
127
128         int key = instruments[i];
129
130         for (int j = 0; j < 16; j++) {
131             JCheckBox jc = (JCheckBox) checkboxList.get(j + (16 * i));
132
133             if (jc.isSelected()) {
134                 trackList[j] = key;
135             } else {
136                 trackList[j] = 0;
137             }
138         } // close inner
139
140         makeTracks(trackList);
141     } // close outer
142
143     track.add(makeEvent(192, 9, 1, 0, 15)); // - so we always go to full 16 beats
144
145     try {
146         sequencer.setSequence(sequence);
147         sequencer.setLoopCount(sequencer.LOOP_CONTINUOUSLY);
148         sequencer.start();
149         sequencer.setTempoInBPM(120);
150     } catch (Exception e) {
151         e.printStackTrace();
152     }
153 } // close method
```

```
156  public class MyStartListener implements ActionListener {  
157      public void actionPerformed(ActionEvent a) {  
158          buildTrackAndStart();  
159      }  
160  }  
161  
162  public class MyStopListener implements ActionListener {  
163      public void actionPerformed(ActionEvent a) {  
164          sequencer.stop();  
165      }  
166  }  
167  
168  public class MyUpTempoListener implements ActionListener {  
169      public void actionPerformed(ActionEvent a) {  
170          float tempoFactor = sequencer.getTempoFactor();  
171          sequencer.setTempoFactor((float) (tempoFactor * 1.03));  
172      }  
173  }  
174  
175  public class MyDownTempoListener implements ActionListener {  
176      public void actionPerformed(ActionEvent a) {  
177          float tempoFactor = sequencer.getTempoFactor();  
178          sequencer.setTempoFactor((float) (tempoFactor * .97));  
179      }  
180  }  
181  
182  }
```

# (p.497)

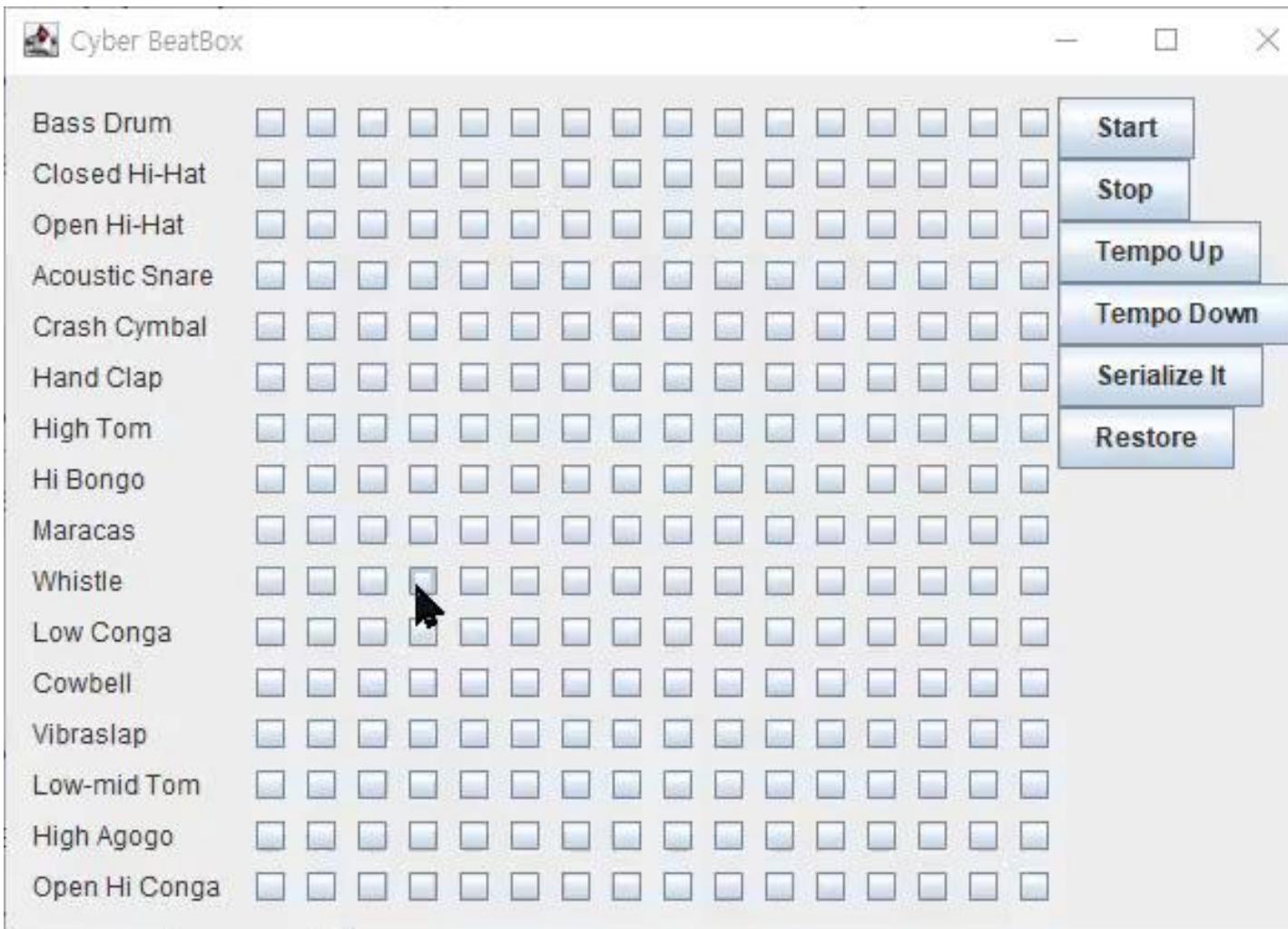
```
186  public class MySendListener implements ActionListener { // new - save
187
188  public void actionPerformed(ActionEvent a) {
189      // make an arraylist of just the STATE of the checkboxes
190      boolean[] checkboxState = new boolean[256];
191
192      for (int i = 0; i < 256; i++) {
193          JCheckBox check = (JCheckBox) checkboxList.get(i);
194          if (check.isSelected()) {
195              checkboxState[i] = true;
196          }
197      }
198
199      try {
200          FileOutputStream fileStream = new FileOutputStream(
201              new File("Checkbox.ser"));
202          ObjectOutputStream os = new ObjectOutputStream(fileStream);
203          os.writeObject(checkboxState);
204      } catch (Exception ex) {
205          ex.printStackTrace();
206      }
207
208      } // close method
209  } // close inner class
```

(p.498)

```
211 public class MyReadInListener implements ActionListener { // new - restore
212
213     boolean[] checkboxState = null;
214
215     try {
216         FileInputStream fileIn = new FileInputStream(
217             new File("Checkbox.ser"));
218         ObjectInputStream is = new ObjectInputStream(fileIn);
219         checkboxState = (boolean[]) is.readObject();
220
221     } catch (Exception ex) {
222         ex.printStackTrace();
223     }
224
225     // now reset the sequence to be this
226     for (int i = 0; i < 256; i++) {
227         JCheckBox check = (JCheckBox) checkboxList.get(i);
228         if (checkboxState[i]) {
229             check.setSelected(true);
230         } else {
231             check.setSelected(false);
232         }
233     }
234
235     // now stop sequence and restart
236     sequencer.stop();
237     buildTrackAndStart();
238 } // close method
239 } // close inner class
```

```
244     public void makeTracks(int[] list) {
245
246         for (int i = 0; i < 16; i++) {
247             int key = list[i];
248
249             if (key != 0) {
250                 track.add(makeEvent(144, 9, key, 100, i));
251                 track.add(makeEvent(128, 9, key, 100, i + 1));
252             }
253         }
254     }
255
256     public MidiEvent makeEvent(int comd, int chan, int one, int two, int tick) {
257         MidiEvent event = null;
258         try {
259             ShortMessage a = new ShortMessage();
260             a.setMessage(comd, chan, one, two);
261             event = new MidiEvent(a, tick);
262
263             } catch (Exception e) {
264             }
265             return event;
266         }
267     } // close class
```

# 실행 결과



# 숙제

- 본문을 꼼꼼하게 읽어봅시다.
- 연필을 깎으며 및 14장 끝에 있는 연습문제를 모두 각자의 힘으로 해결해봅시다.
- 모든 코드를 직접 입력해서 실행시켜보고 코드를 다시 한 번 보면서 코드를 다른 사람한테 소개하는 연습을 해 봅시다.
- API 문서에서 이 장에 나와있는 클래스 및 메서드에 대한 내용을 직접 찾아봅시다.

# QUIZCARDPLAYER 구현

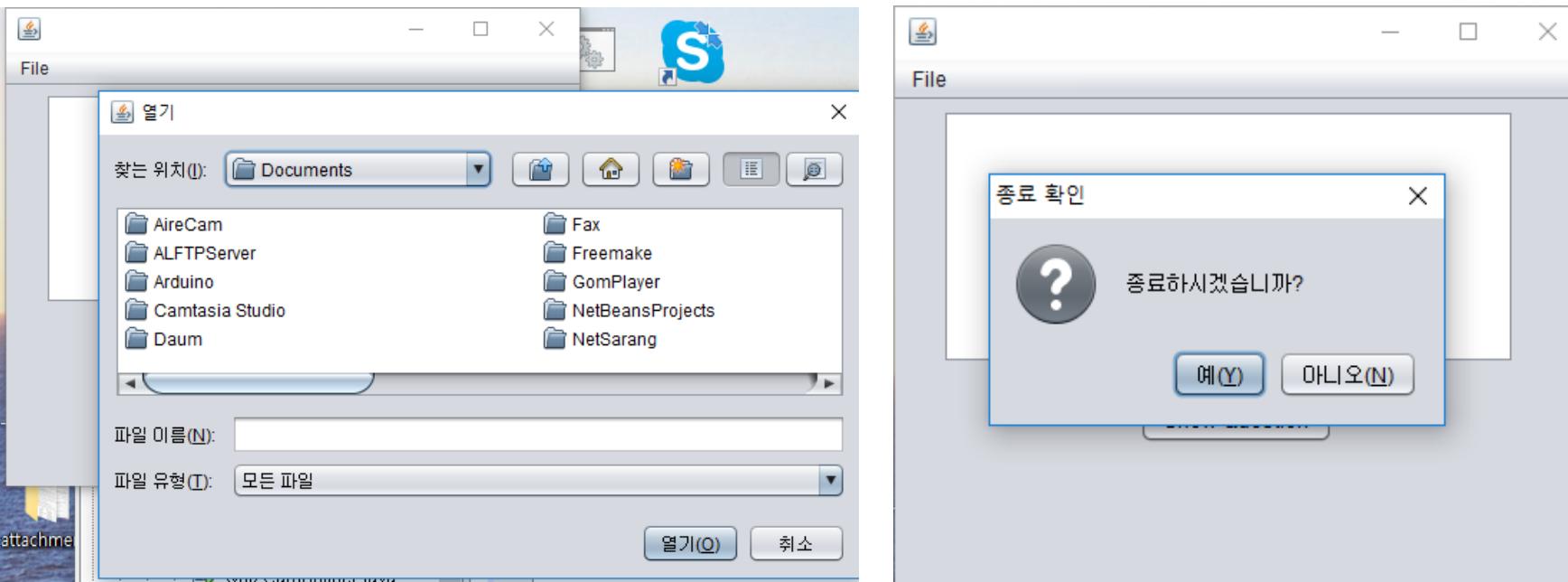
---

NetBeans IDE를 이용한 GUI 프로그래밍

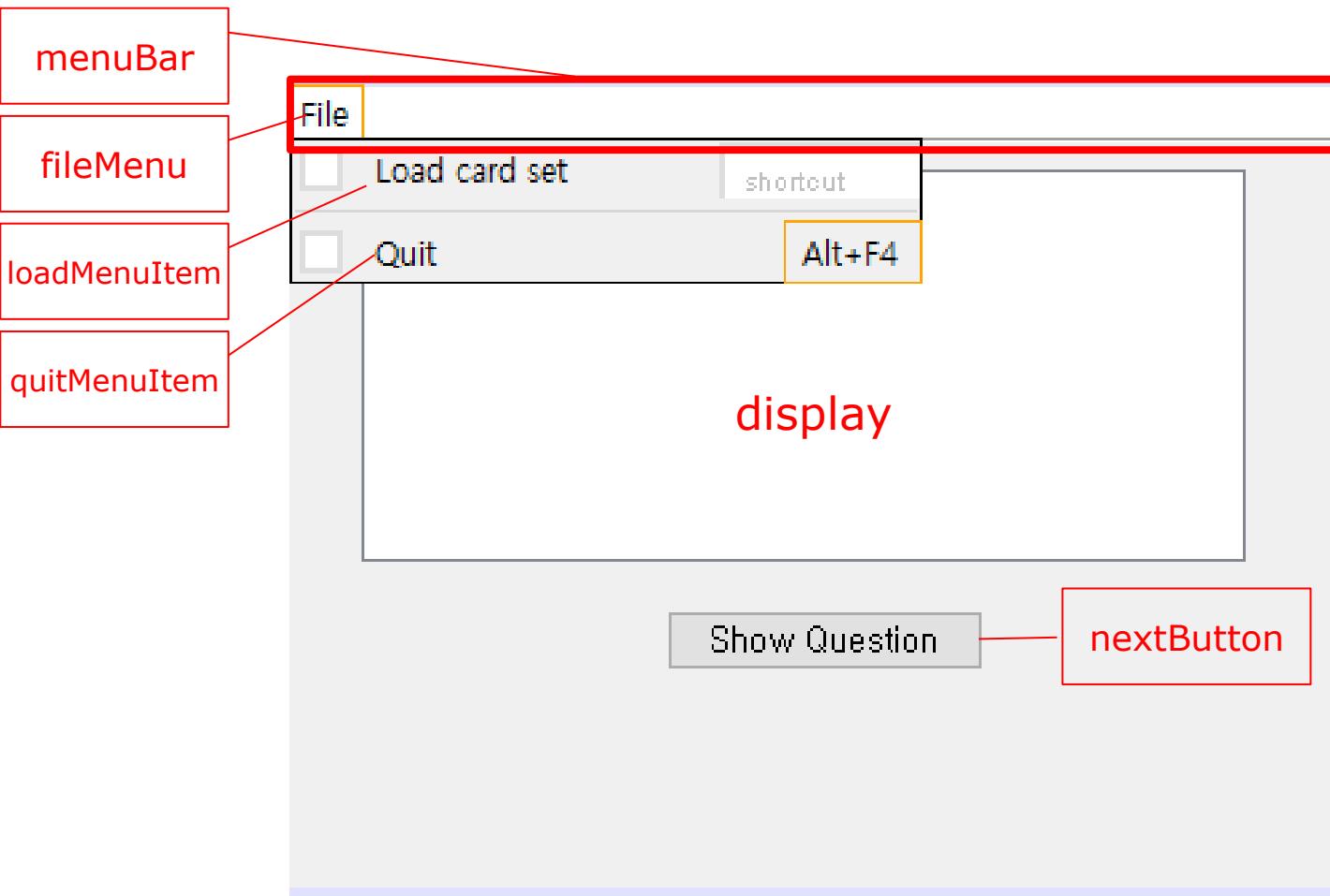
# QuizCardPlayer 개요

- 14장 pp.490-491
- 추가: 종료 버튼



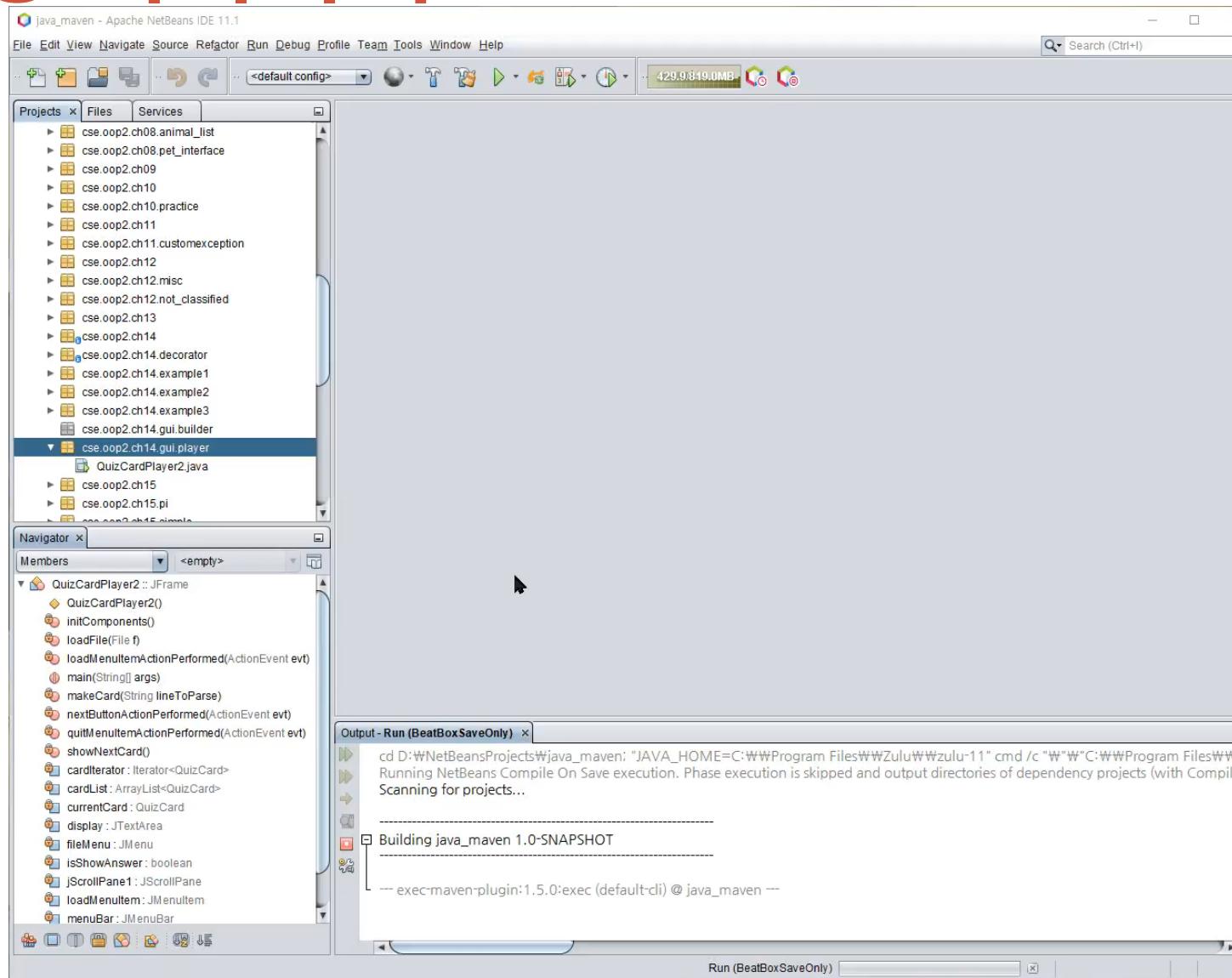


# 화면 구성하기



# 화면 구성 따라하기

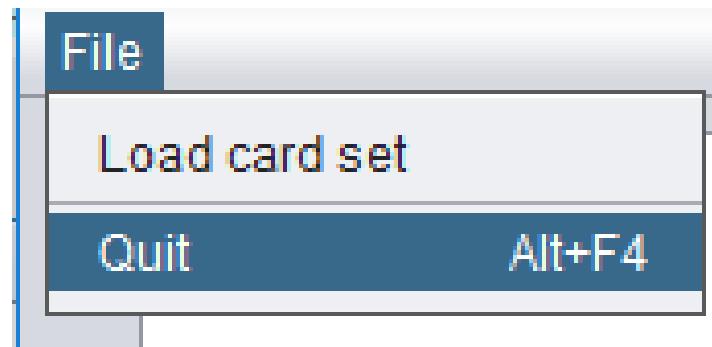
URL: <https://youtu.be/-pSeNmNSP2U>



# 사용할 변수 추가

```
private ArrayList<QuizCard> cardList;  
private QuizCard currentCard;  
private Iterator<QuizCard> cardIterator;  
private boolean isShowAnswer = false;
```

# 메뉴 추가

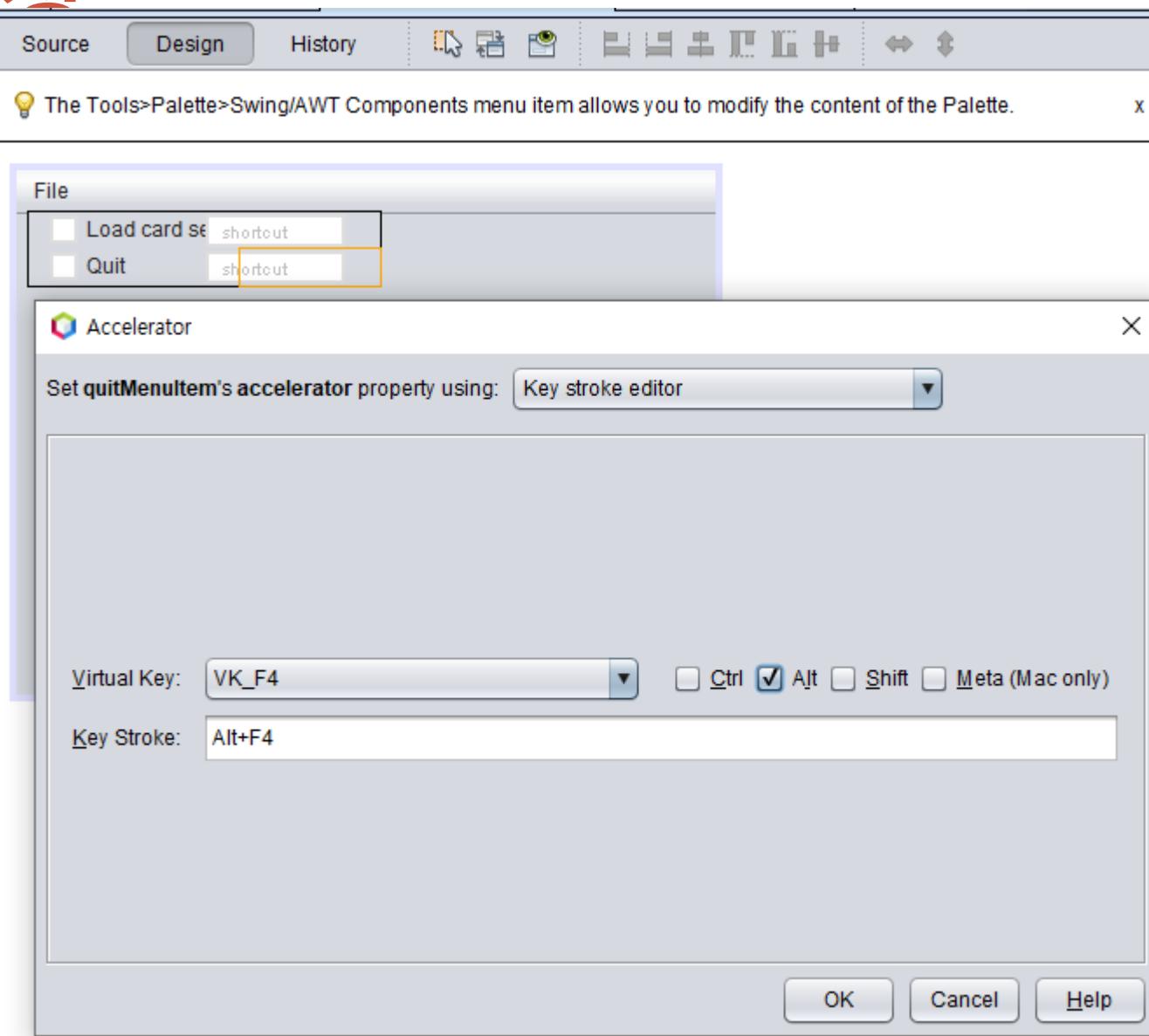


# “Quit”메뉴 구현

```
private void quitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int option = JOptionPane.NO_OPTION;  
    option = JOptionPane.showConfirmDialog(null, "종료하시겠습니까?", "종료 확인",  
                                         JOptionPane.YES_NO_OPTION);  
    if (option == JOptionPane.YES_OPTION) {  
        System.exit(NORMAL);  
    }  
}
```



# 단축키 설정



# “Load card set” 메뉴 구현

```
private void loadMenuItemActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    JFileChooser fileOpen = new JFileChooser();  
    fileOpen.showOpenDialog(this); // frame --> this  
    loadFile(fileOpen.getSelectedFile());  
}
```

```
private void loadFile(File f) {
    cardList = new ArrayList<QuizCard>();
    // try (BufferedReader reader = new BufferedReader(new FileReader(f))) {
    try (BufferedReader reader = new BufferedReader(
        new InputStreamReader(new FileInputStream(f), "UTF-8"))) {
        String line = null;
        while ((line = reader.readLine()) != null) {
            makeCard(line);
        }
    } catch (Exception ex) {
        System.err.println("카드 파일 읽을 수 없음: " + ex.getMessage());
    }

    cardIterator = cardList.iterator();
    showNextCard();
}
```

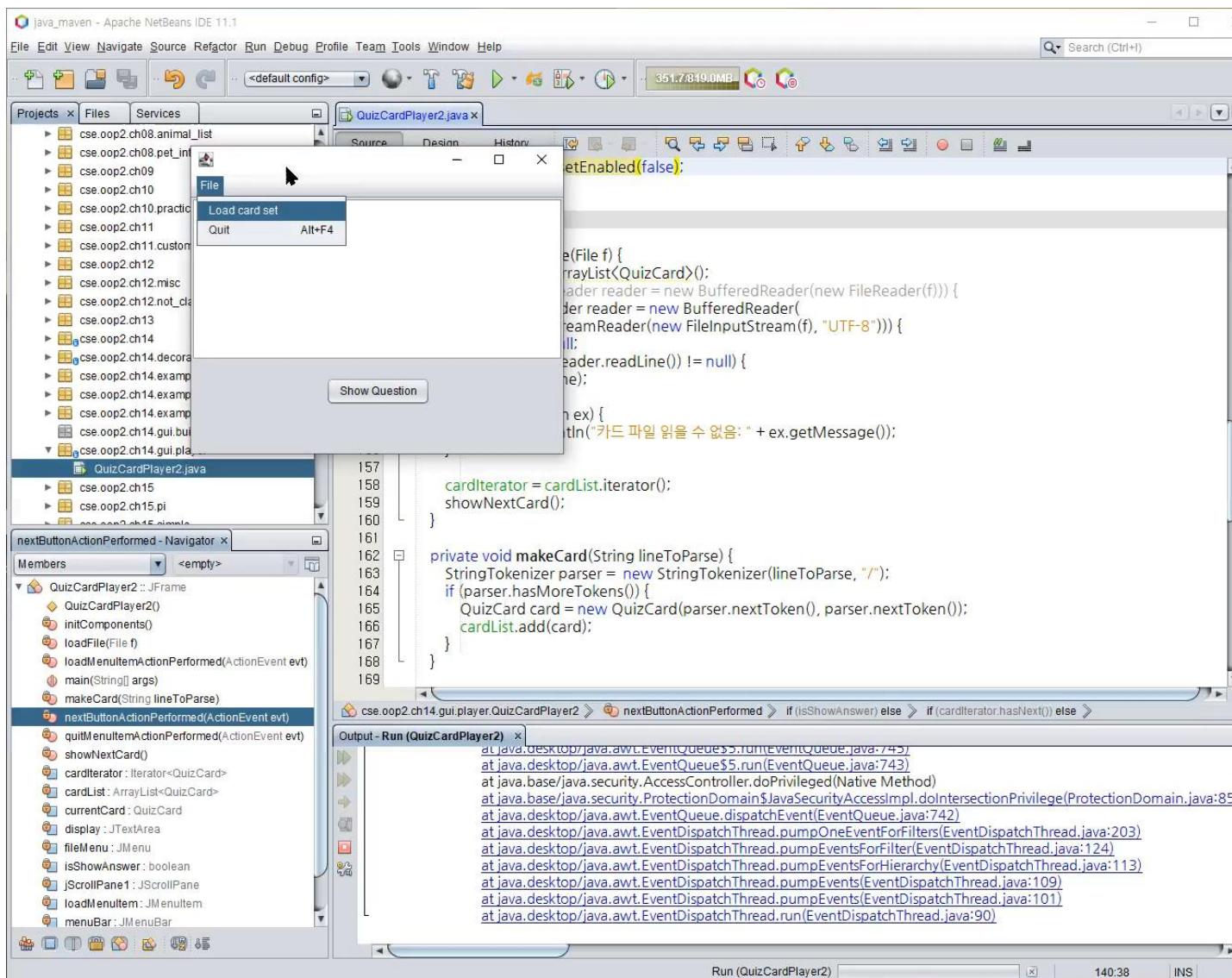
```
private void makeCard(String lineToParse) {
    StringTokenizer parser = new StringTokenizer(lineToParse, "/");
    if (parser.hasMoreTokens()) {
        QuizCard card = new QuizCard(parser.nextToken(), parser.nextToken());
        cardList.add(card);
    }
}

private void showNextCard() {
    currentCard = cardIterator.next();
    display.setText((currentCard.getQuestion()));
    nextButton.setText("Show Answer");
    isShowAnswer = true;
}
```

# nextButton 이벤트 구현

```
private void nextButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    if (isShowAnswer) {  
        display.setText(currentCard.getAnswer());  
        nextButton.setText("Next Card");  
        isShowAnswer = false;  
    } else {  
        if (cardIterator.hasNext()) {  
            showNextCard();  
        } else {  
            display.setText("That was the last card.");  
            nextButton.setEnabled(false);  
        }  
    }  
}
```

# 실행 결과



# NIO.2와 java.nio.file 패키지

- Java NIO.2 (New Input/Output 2)
  - Java 7에서 도입
  - java.nio.file 패키지: 텍스트 파일 읽기/쓰기와 디렉터리와 디렉터리 구조 조작
  - java.nio.file.attribute 패키지: 파일과 디렉터리와 연관된 메타데이터 조작
- java.nio.file 패키지
  - Path 인터페이스: 디렉터리나 파일 위치 지정에 필요한 객체
  - Paths 클래스: Path 객체 생성 시 Paths.get() 메서드 사용
  - Files 클래스: Reader/Writer 생성하고 파일 시스템에 있는 디렉터리와 파일에 대한 작업 수행

# Java NIO.2 개요 (Copilot)

Java NIO.2 (New Input/Output 2)는 Java 7에서 도입된 새로운 입출력 API로, 파일 시스템 및 비동기 I/O 작업을 보다 효율적으로 처리할 수 있도록 설계되었습니다. 주요 기능은 다음과 같습니다:

## 1. Path 클래스:

- `java.nio.file.Path` 클래스는 파일 시스템의 경로를 나타내며, 디렉토리와 파일 작업을 보다 쉽게 처리할 수 있도록 합니다.

Java

복사

```
Path path = Paths.get("/home/user/file.txt");
```

[참고]

```
Path p = Paths.get("/home/user/test");
p ==> /home/user/test
```

## 2. Files 클래스:

- `java.nio.file.Files` 클래스는 파일과 디렉토리 조작을 위한 정적 메서드를 제공합니다.

Java

복사

```
Path path = Paths.get("/home/user/file.txt");
byte[] bytes = Files.readAllBytes(path);
List<String> lines = Files.readAllLines(path, StandardCharsets.UTF_8);
```

## 3. 비동기 채널 (Asynchronous Channels):

- 비동기 파일 채널과 소켓 채널을 통해 비동기 방식으로 I/O 작업을 수행할 수 있습니다. 이를 통해 I/O 작업이 완료될 때까지 블록킹하지 않고 다른 작업을 계속할 수 있습니다.

Java

복사

```
AsynchronousFileChannel asyncFileChannel = AsynchronousFileChannel.open(path);
```

#### 4. Watch Service:

- 파일 시스템의 변경 사항을 모니터링할 수 있는 서비스로, 디렉토리 내 파일의 생성, 수정, 삭제 등을 감지할 수 있습니다.

Java

복사

```
WatchService watchService = FileSystems.getDefault().newWatchService();
Path path = Paths.get("/home/user");
path.register(watchService, StandardWatchEventKinds.ENTRY_CREATE);
```

#### 5. File Attribute View:

- 파일 시스템의 속성을 읽고 쓰기 위한 인터페이스로, 기본 파일 속성 외에도 사용자 정의 속성을 다룰 수 있습니다.

Java

복사

```
BasicFileAttributes attrs = Files.readAttributes(path, BasicFileAttributes.class);
```

# (참고) NIO.2 예제

---

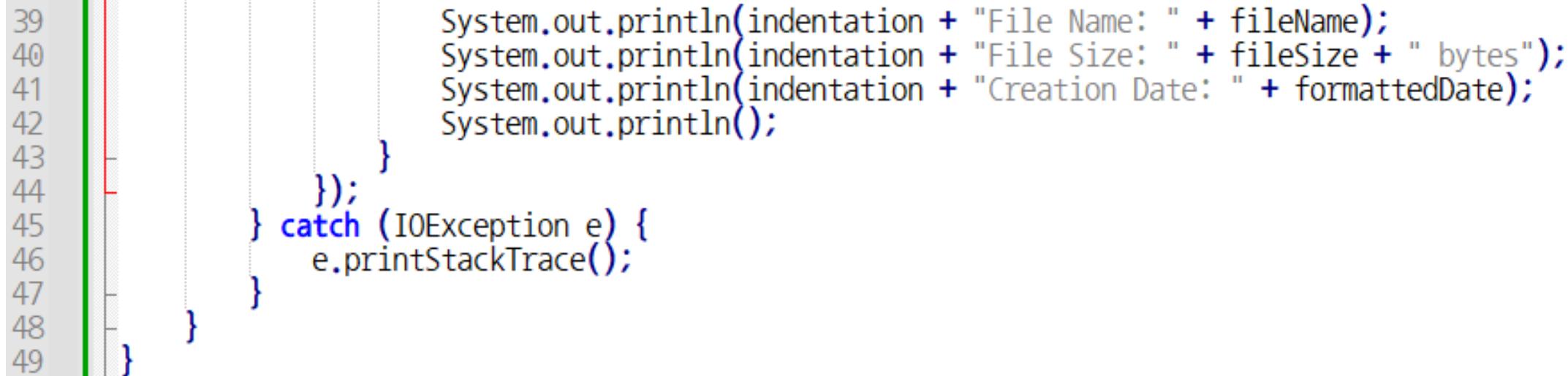
# Copilot에 의해 생성된 NIO.2 예제 프로그램

- java NIO.2를 사용하여 현재 폴더 아래에 있는 모든 파일과 디렉터리를 재귀적으로 찾아서 파일 이름과 파일 크기, 작성 날짜 정보를 보여주는 프로그램을 작성해줘.
- 파일 이름, 크기, 작성 날짜 출력 시 폴더의 depth에 따라서 4개의 공백 문자를 들여쓰는 indentation이 되면 더 보기 좋을 것 같아.

```
1 import java.io.IOException;
2 import java.nio.file.*;
3 import java.nio.file.attribute.BasicFileAttributes;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 public class DirectoryScanner2 {
8
9     public static void main(String[] args) {
10         Path startPath = Paths.get(".");
11         // 현재 폴더를 시작 경로로 설정
12
13         try {
14             Files.walkFileTree(startPath, new SimpleFileVisitor<Path>() {
15                 @Override
16                 public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) {
17                     printFileInfo(file, attrs, startPath.relativize(file).getNameCount());
18                 }
19             });
19 }
```

```
20
21     @Override
22     public FileVisitResult visitFileFailed(Path file, IOException exc) {
23         System.err.println("Failed to visit file: " + file + " (" + exc.getMessage() + ")");
24         return FileVisitResult.CONTINUE;
25     }
26
27     @Override
28     public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) {
29         printFileInfo(dir, attrs, startPath.relativize(dir).getNameCount());
30         return FileVisitResult.CONTINUE;
31     }
32
33     private void printFileInfo(Path path, BasicFileAttributes attrs, int depth) {
34         String indentation = " ".repeat(depth * 4);
35         String fileName = path.getFileName().toString();
36         long fileSize = attrs.size();
37         String formattedDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
38                         .format(new Date(attrs.creationTime().toMillis()));
```

```
39 }  
40 }  
41 }  
42 }  
43 }  
44 }  
45 } catch (IOException e) {  
46 }  
47 }  
48 }  
49 }
```



```
System.out.println(indentation + "File Name: " + fileName);  
System.out.println(indentation + "File Size: " + fileSize + " bytes");  
System.out.println(indentation + "Creation Date: " + formattedDate);  
System.out.println();
```

# 실행 결과

```
PS D:\User\jongmin\Program\AntTest> java .\DirectoryScanner2.java
File Name: .
File Size: 0 bytes
Creation Date: 2023-11-03 08:25:09

File Name: build
File Size: 0 bytes
Creation Date: 2023-11-03 08:25:09

File Name: classes
File Size: 0 bytes
Creation Date: 2023-11-03 08:25:09

File Name: HelloWorld.class
File Size: 341 bytes
Creation Date: 2023-11-03 08:25:09

File Name: jar
File Size: 0 bytes
Creation Date: 2023-11-03 08:25:09

File Name: HelloWorld.jar
File Size: 701 bytes
Creation Date: 2023-11-03 08:25:09
```

```
File Name: build.xml
File Size: 706 bytes
Creation Date: 2023-11-03 08:25:09

File Name: DirectoryScanner2.java
File Size: 2155 bytes
Creation Date: 2024-11-06 17:31:56

File Name: src
File Size: 0 bytes
Creation Date: 2023-11-03 08:25:09

File Name: HelloWorld.java
File Size: 359 bytes
Creation Date: 2023-11-03 08:25:09

PS D:\User\jongmin\Program\AntTest>
```

# 가독성 개선 버전

- SimpleFileVisitor 클래스의 메서드 오버라이딩이 별도의 클래스로 만들지 않고 바로 들어가 있어 가독성이 좋지 않을 것 같아. 가독성이 좋은 코드로 다시 작성해줘.

```
1 import java.io.IOException;
2 import java.nio.file.*;
3 import java.nio.file.attribute.BasicFileAttributes;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 public class DirectoryScanner {
8
9     public static void main(String[] args) {
10         Path startPath = Paths.get(".");
11         // 현재 폴더를 시작 경로로 설정
12
13         try {
14             Files.walkFileTree(startPath, new FileVisitorImpl(startPath));
15         } catch (IOException e) {
16             e.printStackTrace();
17         }
18     }
}
```

```
20 class FileVisitorImpl extends SimpleFileVisitor<Path> {
21
22     private final Path startPath;
23
24     public FileVisitorImpl(Path startPath) {
25         this.startPath = startPath;
26     }
27
28     @Override
29     public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) {
30         printFileInfo(file, attrs);
31         return FileVisitResult.CONTINUE;
32     }
33
34     @Override
35     public FileVisitResult visitFileFailed(Path file, IOException exc) {
36         System.err.println("Failed to visit file: " + file + " (" + exc.getMessage() + ")");
37         return FileVisitResult.CONTINUE;
38     }
```

```
40 @Override
41 public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) {
42     printFileInfo(dir, attrs);
43     return FileVisitResult.CONTINUE;
44 }
45
46 private void printFileInfo(Path path, BasicFileAttributes attrs) {
47     int depth = startPath.relativize(path).getNameCount();
48     String indentation = " ".repeat(depth * 4);
49     String fileName = path.getFileName().toString();
50     long fileSize = attrs.size();
51     String formattedDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
52         .format(new Date(attrs.creationTime().toMillis()));
53
54     System.out.println(indentation + "File Name: " + fileName);
55     System.out.println(indentation + "File Size: " + fileSize + " bytes");
56     System.out.println(indentation + "Creation Date: " + formattedDate);
57     System.out.println();
58 }
59 }
```

# 가독성 개선 버전2

- 출력되는 파일 이름, 크기, 생성날짜를 한줄로 표시해주고, 폴더 구조를 알 수 있도록 '+', '-', '|' 문자를 사용하여 리눅스 운영체제의 tree 명령이 하는 것처럼 보여주는 프로그램으로 수정해 줘.
- '|' 출력하고 나서 한줄 띄워주어야 tree 명령에서 보여지는 것처럼 출력이 될 것 같아. 수정 부탁해.
- "| \$n" 처럼 단순하게 "\$n"를 넣어주면 다음에 있는 파일이나 폴더 정보 출력시 indentation되는 공백 갯수가 맞지 않아 트리 구조가 제대로 보이지 않아. 수정 가능할까?

```
1 import java.io.IOException;
2 import java.nio.file.*;
3 import java.nio.file.attribute.BasicFileAttributes;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 public class DirectoryScanner {
8
9     public static void main(String[] args) {
10         Path startPath = Paths.get("."); // 현재 폴더를 시작 경로로 설정
11
12         try {
13             Files.walkFileTree(startPath, new FileVisitorImpl(startPath));
14         } catch (IOException e) {
15             e.printStackTrace();
16         }
17     }
18 }
19
20 class FileVisitorImpl extends SimpleFileVisitor<Path> {
21
22     private final Path startPath;
23
24     public FileVisitorImpl(Path startPath) {
25         this.startPath = startPath;
26     }
}
```

```
28     @Override
29     public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) {
30         printFileInfo(file, attrs);
31         return FileVisitResult.CONTINUE;
32     }
33
34     @Override
35     public FileVisitResult visitFileFailed(Path file, IOException exc) {
36         System.err.println("Failed to visit file: " + file + " (" + exc.getMessage() + ")");
37         return FileVisitResult.CONTINUE;
38     }
39
40     @Override
41     public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) {
42         printFileInfo(dir, attrs);
43         return FileVisitResult.CONTINUE;
44     }
45
46     private void printFileInfo(Path path, BasicFileAttributes attrs) {
47         int depth = startPath.relativize(path).getNameCount();
48         String indentation = " ".repeat(depth * 4 - 4) + (depth > 0 ? "|--- " : "");
49         String fileName = path.getFileName().toString();
50         long fileSize = attrs.size();
51         String formattedDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
52                         .format(new Date(attrs.creationTime().toMillis()));
53
54         System.out.printf("%s File Name: %-20s Size: %-10d Created: %s%n",
55                           indentation, fileName, fileSize, formattedDate);
56     }
57 }
```

# 실행 결과

```
PS D:\User\jongmin\Program\AntTest> java .\DirectoryScanner4.java
|--- File Name: .           Size: 0          Created: 2023-11-03 08:25:09
|--- File Name: build       Size: 0          Created: 2023-11-03 08:25:09
|   |--- File Name: classes  Size: 0          Created: 2023-11-03 08:25:09
|   |   |--- File Name: HelloWorld.class  Size: 341      Created: 2023-11-03 08:25:09
|   |--- File Name: jar      Size: 0          Created: 2023-11-03 08:25:09
|   |   |--- File Name: HelloWorld.jar    Size: 701      Created: 2023-11-03 08:25:09
|--- File Name: build.xml   Size: 706         Created: 2023-11-03 08:25:09
|--- File Name: DirectoryScanner2.java  Size: 2155     Created: 2024-11-06 17:31:56
|--- File Name: DirectoryScanner4.java  Size: 1840     Created: 2024-11-06 17:40:41
|--- File Name: DirectoryScanner4.java.bak Size: 2190     Created: 2024-11-06 17:42:44
|--- File Name: src         Size: 0          Created: 2023-11-03 08:25:09
|   |--- File Name: HelloWorld.java    Size: 359      Created: 2023-11-03 08:25:09
PS D:\User\jongmin\Program\AntTest>
```