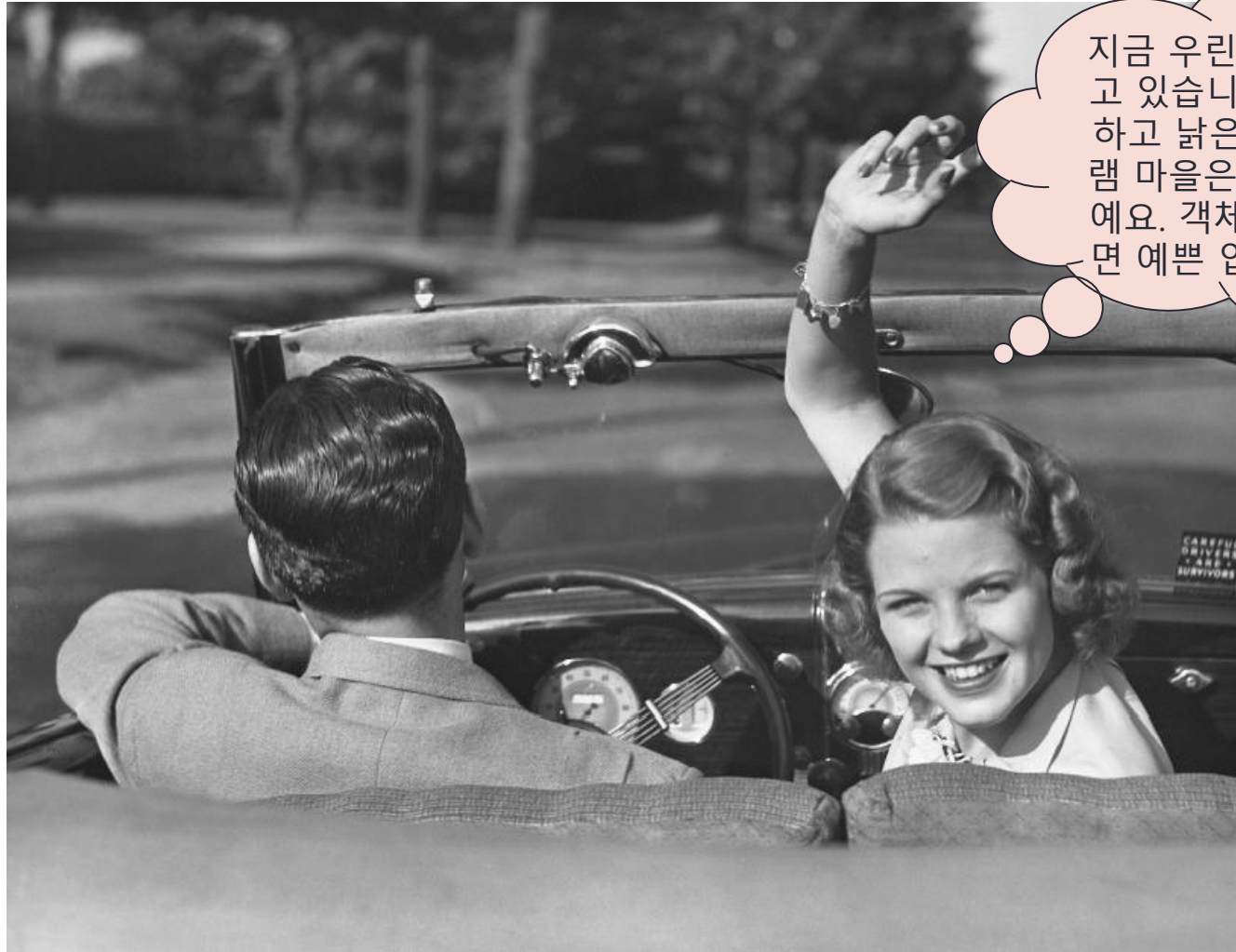


2장. 클래스와 객체

1. 객체지향적인 프로그래밍에 대한 이해
2. 클래스와 객체의 개념에 대한 이해

이제 객체 마을로 떠나봅시다.



지금 우린 객체 마을로 가고 있습니다. 이제 지저분하고 낡은 절차적 프로그램 마을은 영원히 떠날 거예요. 객체 마을에 도착하면 예쁜 엽서 보내줄게요.

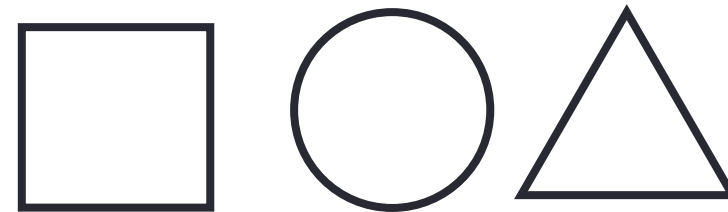
의자 전쟁

p.70

로라 – 절차적 프로그래밍
브래드 – 객체지향적 프로그래밍



GUI에 세 개의 도형이 있는데 사용자가 어떤 도형을 클릭하면 그 도형을 회전시키고 각 도형마다 정해진 AIF 사운드 파일을 재생한다.



로라의 일하는 방식

- 로라는 무슨 생각을 했을까요?
 - 이 프로그램에서 어떤 일을 해야 할 것인가?
 - 어떤 프로시저(또는 함수)가 필요할까?
- 로라는 무엇을 했을까요?
 - 사무실에 앉아서 프로시저를 만들었습니다.

분석(what)
설계(how)

```
rotate(shapeNum) {  
    // 도형을 360° 회전시킨다.  
}  
  
playSound(shapeNum) {  
    // shapeNum 값으로어떤 AIF 사운드를  
    // 재생할지 확인한다음재생한다.  
}
```

브래드의 일하는 방식

- 브래드는 무슨 생각을 했을까요?
 - 이 프로그램에는 뭐가 들어가야 할까? 분석(what)
 - 가장 중요한 역할 (또는 클래스, 객체)을 하는 것은 무엇일까? 설계(how)
- 브래드는 무엇을 했을까요?
 - 카페에 가서 도형 객체를 만들었습니다.

정사각형

```
rotate() {
    // 정사각형을 회전시키기 위
}

playSound() {
    // AIF 사운드를 재생시키기
}
```

원

```
rotate() {
    // 원을 회전시키기 위한 코드
}

playSound() {
    // AIF 사운드를 재생시키기 위
}
```

삼각형

```
rotate() {
    // 삼각형을 회전시키기 위한 코드
}

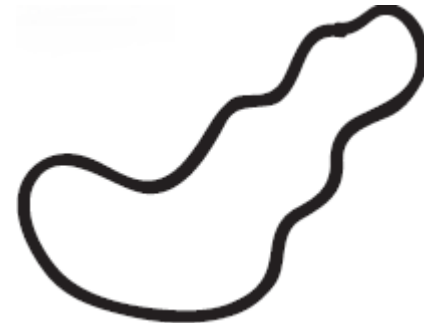
playSound() {
    // AIF 사운드를 재생시키기 위한 코드
}
```

요구사항의 변경

p.71

- 스펙 변경!!! (→ **요구사항 변경** change of requirements)

화면에 다른 도형과 함께 **아메바 모양의 도형**이 있다. 사용자가 아메바 모양을 클릭하면 다른 도형과 마찬가지로 한 바퀴 회전시키고 **.mp3 사운드 파일**을 재생해야 한다.



로라와 브래드의 수정 1

• 로라

- rotate 프로시저는 바꾸지 않아도 됨
- playSound는 .hif 파일을 재생하기 위해 수정
- (여전히 사무실에서만 작업합니다.)

• 브래드

- 아메바라는 클래스를 새로 제작
- (이번에는 해변에 앉아서 작업했군요.)

```
playSound(shapeNum) {
    // 도형이 아메바 모양이 아니면
    // shapeNum으로 재생할 AIF
    // 파일을 찾아서 재생

    // 도형이 아메바 모양이면
    // 아메바에 해당하는 .hif 파일을
    // 재생
}
```

아메바

```
rotate() {
    // 아메바를 회전시키기 위한 코드
}

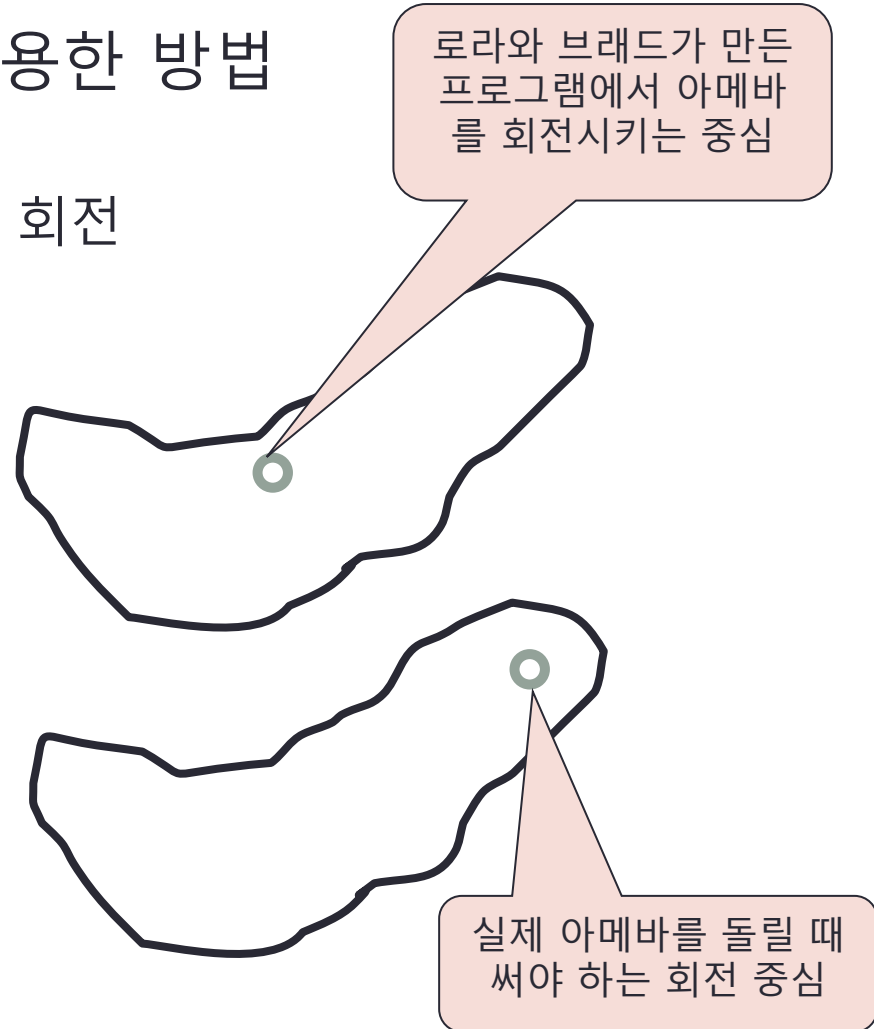
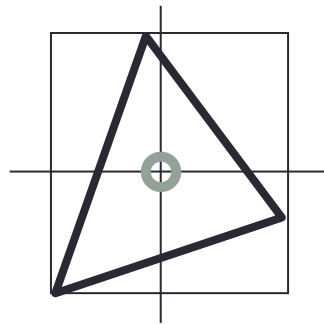
playSound() {
    // .mp3 사운드를 재생시키기 위한 코드
}
```

아메바 회전 시 주의 사항

p.72

- 로라와 브래드가 도형을 회전시킬 때 사용한 방법

- 1) 도형을 둘러싸고 있는 직사각형을 결정
- 2) 직사각형의 중심을 구해서 그 점을 중심으로 회전



로라의 수정 2

p.72

- 로라
 - 상당히 많은 코드를 수정
 - 다시 테스트, 컴파일
 - 일이 꼬이기 시작
 - 여전히 사무실 구석에 앉아서 고생하고 있습니다.

shapeNum에 따라서 도
형 모양 알고 필요한 회전
시킴 (제어 변수)

```
rotate(shapeNum, xPt, yPt) {
    // 아메바가 아닌 경우
    // 직사각형의 중심을 계산
    // 회전
    // 그렇지 않으면
    // xPt, yPt를 회전 중심으로
    // 회전
}
```

아메바만 필요한
매개변수

브래드의 수정 2

p.72

- 브래드
 - 다른 부분은 전혀 건드릴 필요가 없습니다.
 - 아메바 클래스에 있는 rotate 메서드만 수정했습니다.
 - 여유 있게 야외 콘서트장에서 작업해서 보낼 수 있었습니다.

```
아메바
int xPoint
int yPoint
rotate() {
    // 아메바를 회전시키기 위한 코드
    // xPoint와 yPoint를 사용
}

playSound() {
    // .mp3 사운드를 재생시키기 위한 코드
}
```

로라의 불만

p.73

- 불만에 가득 찬 로라, 브래드의 코드를 헐뜯기 시작하다.

로라: 코드가 중복돼 있잖아.
rotate 프로시저가 도형 거시기
에 전부 들어있네.

브래드: 그건 프로시저가 아니라
메서드라고 하는 거야. 그리고 거
시기가 아니라 클래스라고 부르지.

로라: 뭐라고 부르든지 상관없는
데, 설계가 영 이상하잖아. rotate
'메서드'라는 걸 네 개나 따로 만
들어야 하는데, 그런 설계는 좋다
고 할 수 없지.

브래드: 아, 아직 최종판을 못 봤
구나. OO에서 상속이 어떤 식으
로 돌아가는지 설명해주지.

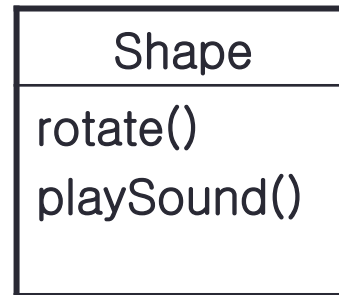
브래드의 코드 개선

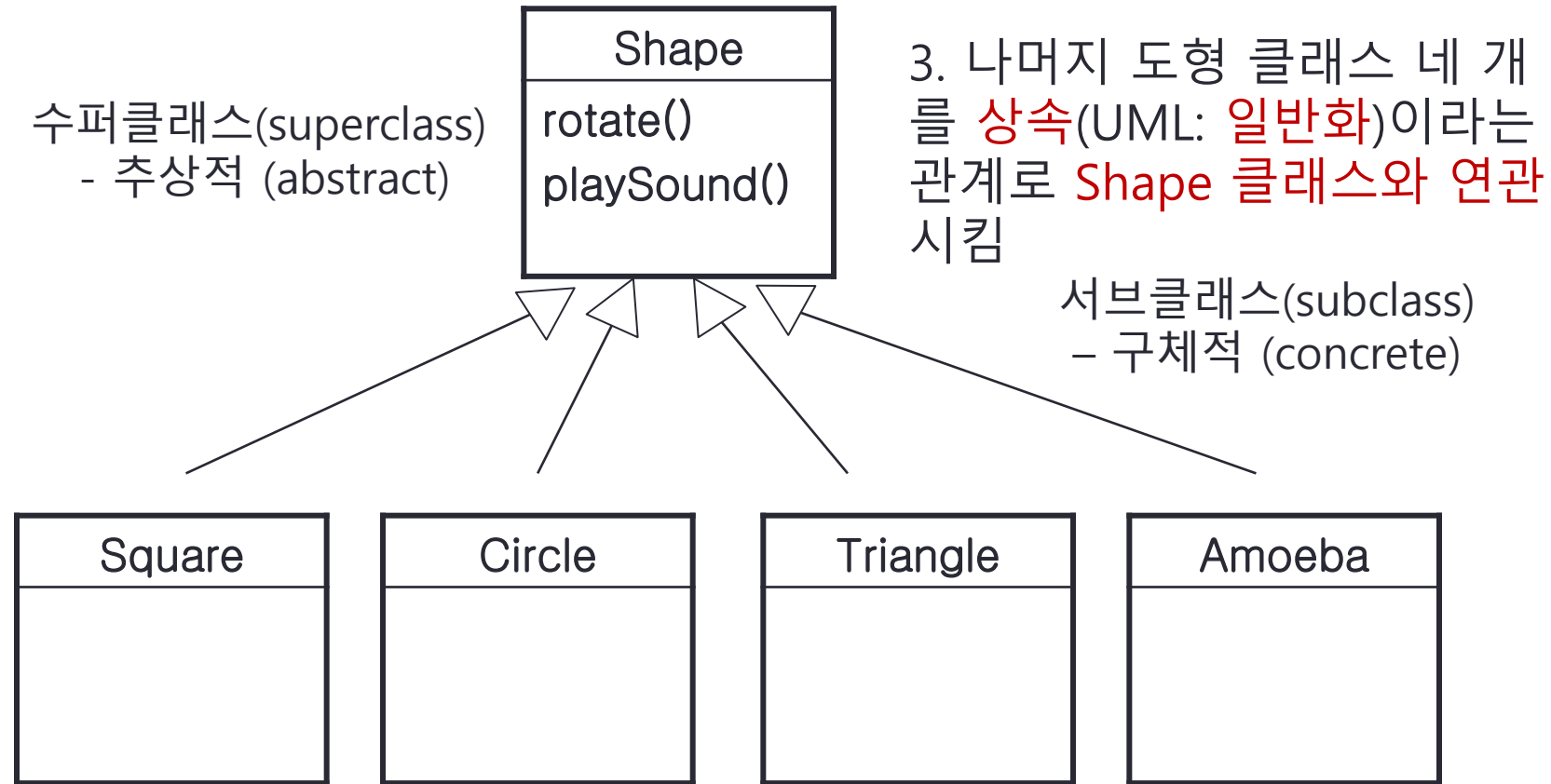
p.73

1. 네 클래스에 **공통적으로 들어있는** 것을 찾아낸다.

Square	Circle	Triangle	Amoeba
rotate() playSound()	rotate() playSound()	rotate() playSound()	rotate() playSound()

2. 모두 도형이고 회전(rotate()) 및 사운드 재생(playSound()) 기능이 있으므로 **공통적인 기능을 모두 뽑아서** Shape이라는 클래스를 만든다.



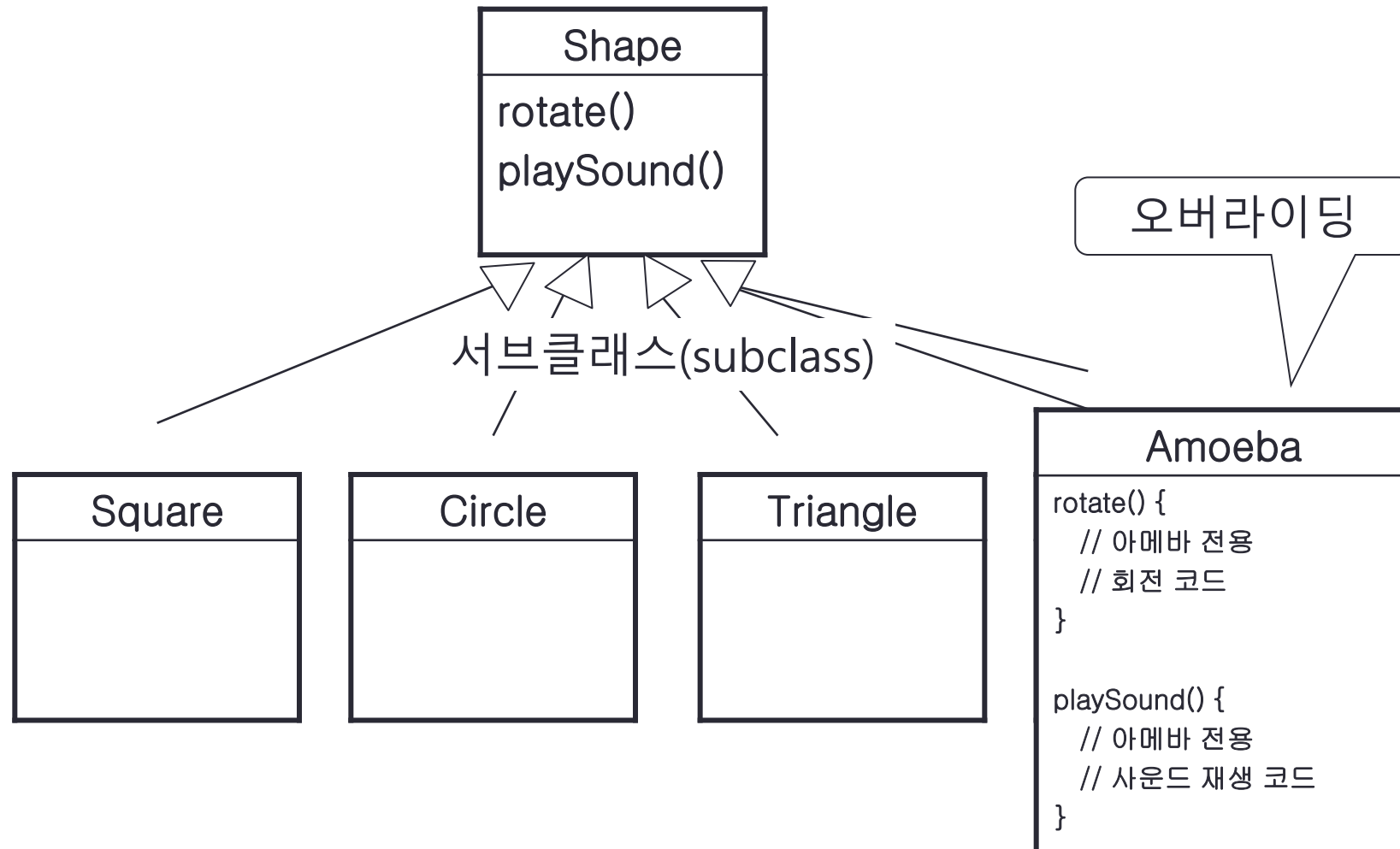


코딩을 어떻게 하는지는
아직 고민하지 않아도 됩니다!

아메바의 rotate() 처리: Overriding

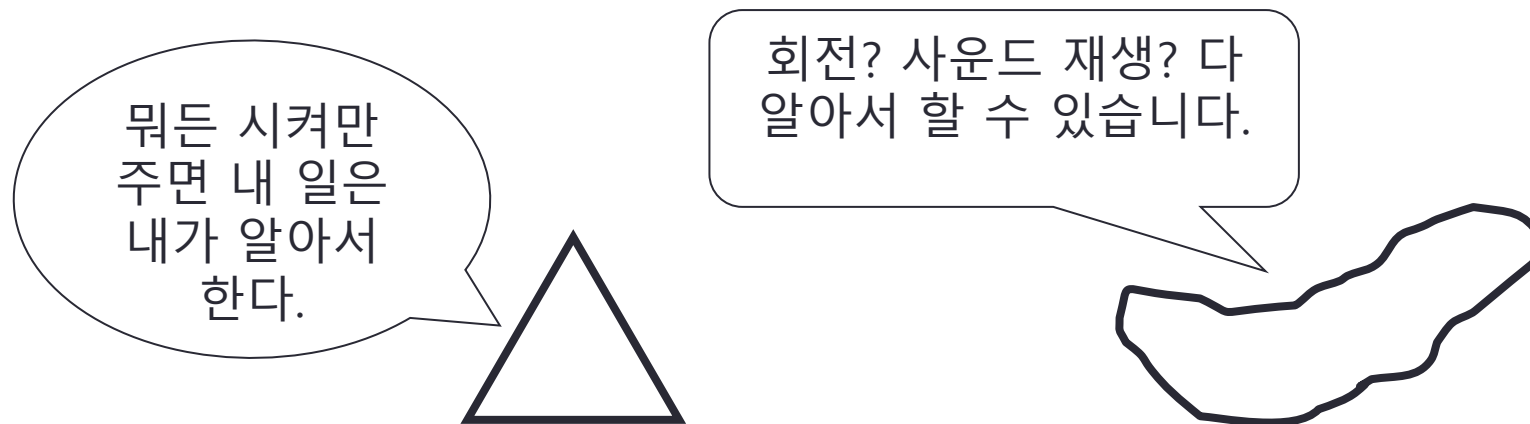
p.74

아메바의 메서드는 어떻게 처리할 수 있을까요?



OO에서의 일하는 방식

- 뭔가를 시킬 때는 어떻게 해야 할까요?
 - 각 도형에 있는 **메서드**를 **호출**합니다.
 - JVM에서는 어떤 객체의 메서드를 호출할지를 판단하여 주어진 조건에 맞는 메서드를 실행시킵니다.



프로그래머는 적절한 지시만 내리면 됩니다. 그 안에서 어떤 식으로 일이 처리되는 지 일일이 신경을 쓰지 않아도 되지요.

(**자료 추상화**: 자료 구조와 관련 행위를 같이 정의)

OO의 장점

- 더 자연스러운 방식으로 설계하는 데 도움이 됩니다.
- 새로운 기능을 추가하기 위해 이미 테스트까지 완료한 코드를 수정하지 않아도 됩니다.
- 데이터와 그 데이터로 작업을 처리하기 위한 메서드가 한 클래스에 들어갈 수 있습니다.
(자료 추상화)
- 다른 애플리케이션의 코드를 재사용할 수 있습니다.

의자는 누가 받았을까요? →



Amie

두뇌 운동

p.75

- 자바 클래스를 설계할 때 가장 근본적으로 고려해야 할 것은?
- 어떤 의문점으로부터 출발해야 할까요?
- 클래스를 설계할 때 사용할 체크리스트를 만든다면 어떤 내용을 포함시키는 게 좋을까요?

- * 공통점 분석 (Commonality Analysis)
- * 차이점 분석 (Variability Analysis)

AI: 객체지향 설계 시 공통점 분석과 차이점 분석을 적용하는 사례와 원리를 찾아서 설명해줘.

공부법에 관한 팁

- 연습문제를 풀다가 막히면 그 문제를 크게 소리 내어 말해보세요.
- 다른 사람하고 토론을 해 보세요.
- 여의치 않으면 애완동물이나 물건하고 얘기하는 것도 도움이 됩니다.



클래스 설계

p.76

- 클래스 설계 시에 고려해야 할 점
 - 객체가 아는 것 (things the object knows)
 - 객체가 하는 것 (things the object does)

ShoppingCart
cartContents
addToCart() removeFromCart() checkOut()

Button
label color
setColor() setLabel() depress() unDepress()

Alarm
alarmTime alarmMode
setAlarmTime() getAlarmTime() isAlarmSet() snooze()

클래스 설계

- 객체에서 아는 것 (**상태, state**)
 - 인스턴스 변수 (instance variable) 또는 필드(field)
- 객체에서 하는 것 (**행위, behavior**)
 - 메서드 (method)



class, object & instance

cf. <https://alfredjava.wordpress.com/2008/07/08/class-vs-object-vs-instance/>

AI: object, class, instance 개념을 설명해줘.

단일 책임 원리 & 정보 전문가 패턴

- 단일 책임 원리(Single Responsibility Principle)
 - one of **SOLID** OO design principles
 - 클래스는 하나의 책임만을 가진다.
 - 2-3줄 정도로 간단하게 클래스의 책임(또는 기능)을 **주석**으로 기술할 수 있어야 한다.
- 정보 전문가 패턴(Information Expert Pattern)
 - 일반적인 **책임 할당** 소프트웨어 패턴(GRASP; General Responsibility Assignment Software Patterns)
 - 어떤 일을 하기 위해 정보를 가장 많이 알고 있는 클래스에게 책임을 할당한다.

```
/**
 * 파일 목록 처리를 한다.
 *
 * @author Prof. Jong Min Lee
 */
public class FileListBean {
```

```
/**
 * 문제 목록을 인스턴스 변수로 초기화
 * @param problemList "n001,n002,n003" 형태의 문자열로 문제 목록을 의미.
 */
public void setProblemList(String problemList) {
    this.problemList = java.util.Arrays.asList(problemList.split(","));
}
```

참고. SOLID

- Robert C. Martin(Uncle Bob)이 명명한 'first five principles'에 대하여 Michael Feathers가 붙인 기억을 돕는 두문자어(mnemonic acronym)
- Principles of OOD(<http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod>)

The first five principles are principles of *class design*. They are:

SRP	<u>The Single Responsibility Principle</u>	<i>A class should have one, and only one, reason to change.</i>
OCP	<u>The Open Closed Principle</u>	<i>You should be able to extend a classes behavior, without modifying it.</i>
LSP	<u>The Liskov Substitution Principle</u>	<i>Derived classes must be substitutable for their base classes</i>
ISP	<u>The Interface Segregation Principle</u>	<i>Make fine grained interfaces that are client specific.</i>
DIP	<u>The Dependency Inversion Principle</u>	<i>Depend on abstractions, not on concretions.</i>

참고. GRASP

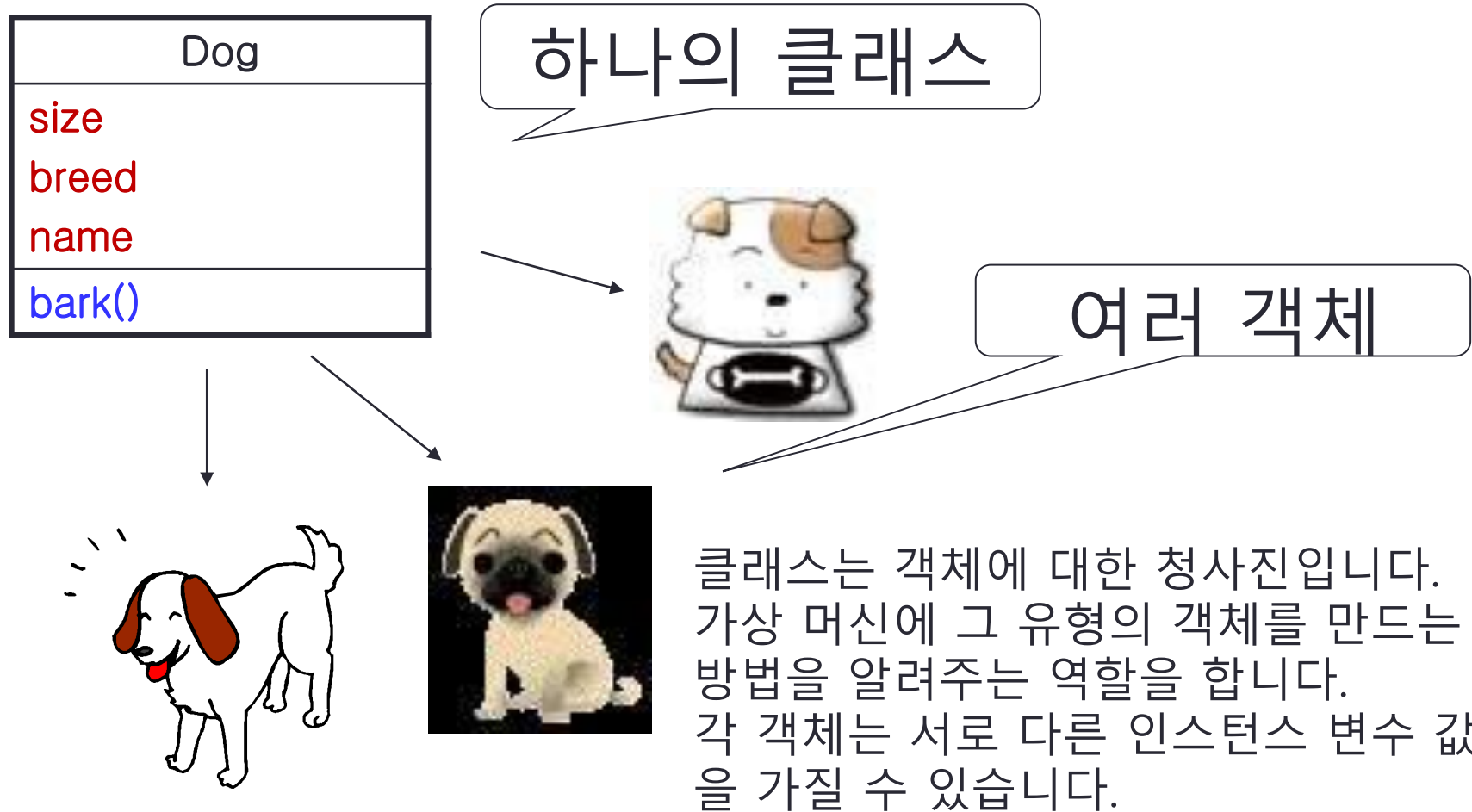
- Information Expert
- Creator
- Low Coupling
- High Cohesion
- Controller
- Indirection
- Polymorphism
- Protected variations
- Pure fabrication

- 참고

- [https://en.wikipedia.org/wiki/GRASP_\(object-oriented_design\)#Information_expert](https://en.wikipedia.org/wiki/GRASP_(object-oriented_design)#Information_expert)
- Craig Larman, Applying UML and Patterns(UML과 패턴의 적용), 3rd ed.

클래스와 객체

p.77



첫 번째 객체 만들기: 작업 순서

p.78

1. (원하는 기능을 하는) 클래스를 만듭니다.
2. 테스트용 클래스를 만듭니다.
3. 테스트용 클래스에서 객체를 만들고 그 객체의 변수와 메서드를 사용합니다.
(객체의 사용 방법 정의)

첫 번째 객체의 클래스 정의

- 1. 클래스(Dog.java)를 만듭니다.

```
class Dog {
    int size;      // 가시성: 패키지
    String breed;
    String name;

    void bark() { // 가시성: 패키지
        System.out.println("Ruff! Ruff!");
    }
}
```

Dog
size
breed
name
bark()

Dog
size: int
breed: string
name: string
bark(): void

테스트용 클래스 정의

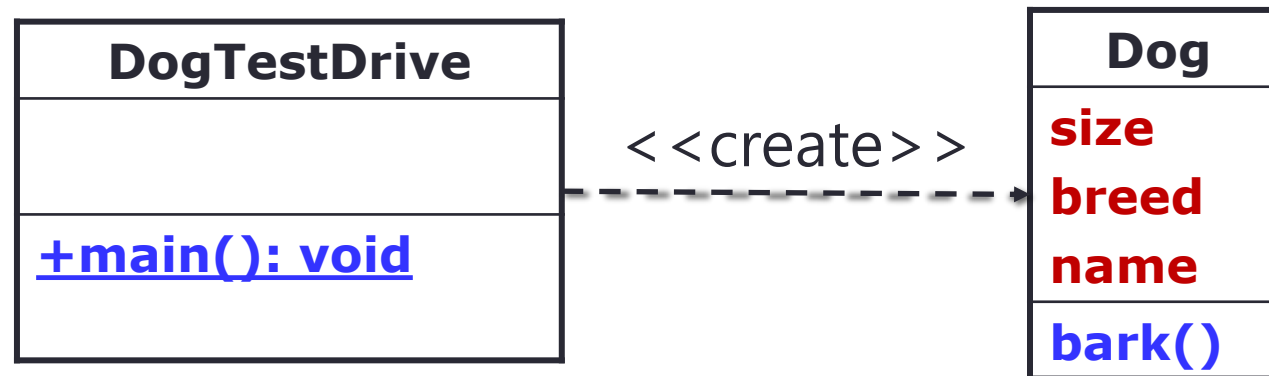
- 2. 테스트용 클래스(DogTestDrive.java)를 만듭니다.
 - 실행 가능하도록 main() 메서드를 정의해야 함!

```
class DogTestDrive {  
    public static void main (String[] args) {  
        // 클래스를 테스트하기 위한 코드  
    }  
}
```

테스트용 클래스에서 Dog 객체 사용 방법 정의

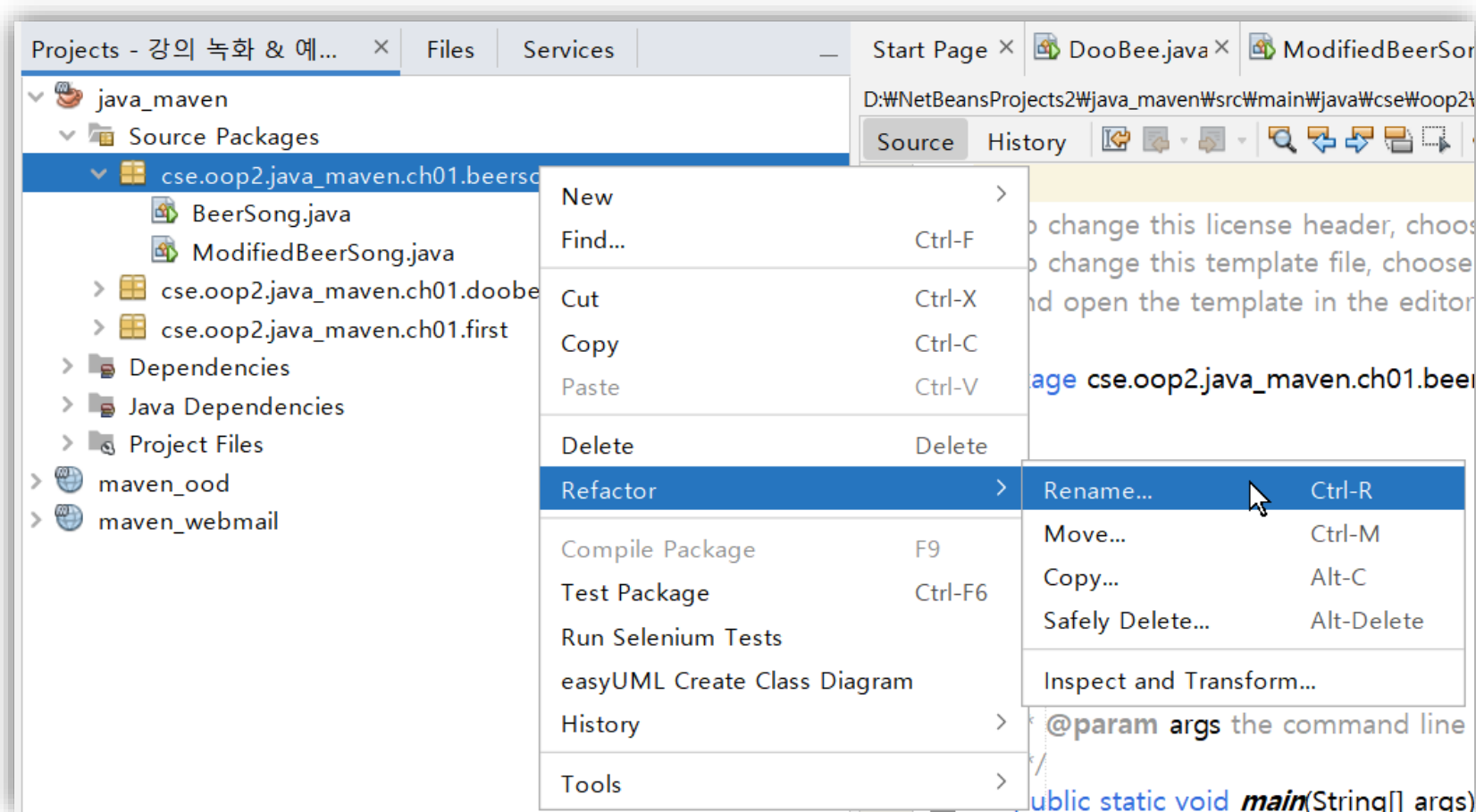
- 3. 객체를 만들고 그 객체의 변수와 메서드를 사용합니다.

```
class DogTestDrive {  
    public static void main (String[] args) {  
        Dog d = new Dog();  
        d.size = 40; // 인스턴스 변수를 바로 사용하는 것은 좋지 않음.  
        d.bark();  
    }  
}
```



패키지 이름 리팩토링

- cse.oop2.java_maven.ch01.beer[song] → cse.oop2.ch01.beer[song]



실습: Dog.java

```

1  package cse.oop2.ch02.dog;
2
3  class Dog {
4
5      int size;
6      String breed;
7      String name;
8
9      // TODO 2-1: 추가해 봅시다.
10     public Dog() {
11         this(30, "래브라도 리트리버", "새론");
12     }

```

```

14     public Dog(int size, String breed, String name) {
15         this.size = size;
16         this.breed = breed;
17         this.name = name;
18     }
19
20     void bark() {
21         System.out.println(name + "가/이 짖습니다.");
22     }
23 }

```

```
6 package cse.oop2.ch02.dog;
7
8 + /**...4 lines */
12 public class DogTestDrive {
13     - public static void main(String[] args) {
14         // TODO 2-2: 따라 해 봅시다.
15         Dog d1 = new Dog(30, "알래스카 말라뮤트", "보리");
16         d1.bark();
17
18         Dog d2 = new Dog();
19         d2.bark();
20
21         Dog d3 = new Dog();
22         d3.name = "코코";
23         d3.size = 40;
24         d3.bark();
25     }
26 }
```

어떤 실행 결과가 나올까요?

Run project 수정

Categories:

- General
- Sources
- Configurations
- JavaScript Libraries
- Build
 - Compile
 - Run
 - Java Shell
 - Actions**
- JavaScript
 - RequireJs
 - Oracle JET
 - License Headers
- Formatting
 - CheckStyle Formatting
 - Hints

Configuration:

Actions:

- <default config>
- Build project
- Clean project
- Clean and Build project
- Build with Dependencies
- Test project
- Test file
- Integration test file
- Run project**
- Run file via main()**

Execute Goals :

`process-classes org.codehaus.mojo:exec-maven-plugin:1.5.0:exec`

Activate Profiles :

Set Properties :

`exec.args=-Dfile.encoding=UTF-8 -classpath %classpath cse.oop2.ch11.M`

`exec.executable=java`

Add >

Run file via main() 수정

The screenshot shows the 'Run' configuration dialog in an IDE. The left sidebar lists categories, with 'Actions' selected. The main area shows the configuration for 'Run file via main()'. The 'Configuration' dropdown is set to '<default config>'. The 'Actions' list includes 'Build project', 'Clean project', 'Clean and Build project', 'Build with Dependencies', 'Test project', 'Test file', 'Integration test file', 'Run project', and 'Run file via main()'. The 'Execute Goals' field contains 'process-classes org.codehaus.mojo:exec-maven-plugin:1.5.0:exec'. The 'Activate Profiles' field is empty. The 'Set Properties' section shows 'exec.args=-Dfile.encoding=UTF-8 -classpath %classpath \${packageClassName}', 'exec.executable=java', and 'exec.classpathScope=\${classPathScope}'. At the bottom, there are checkboxes for 'Build Recursively (with modules)' and 'Build With Dependencies'.

Categories:

- General
- Sources
- Configurations
- JavaScript Libraries
- Build
 - Compile
- Run
- Java Shell
- Actions
- JavaScript
 - RequireJs
 - Oracle JET
- License Headers
- Formatting
 - CheckStyle Formatting
- Hints

Configuration: <default config>

Actions:

- Build project
- Clean project
- Clean and Build project
- Build with Dependencies
- Test project
- Test file
- Integration test file
- Run project
- Run file via main()

Execute Goals : process-classes org.codehaus.mojo:exec-maven-plugin:1.5.0:exec

Activate Profiles :

Set Properties :

exec.args=-Dfile.encoding=UTF-8 -classpath %classpath \${packageClassName}

exec.executable=java

exec.classpathScope=\${classPathScope}

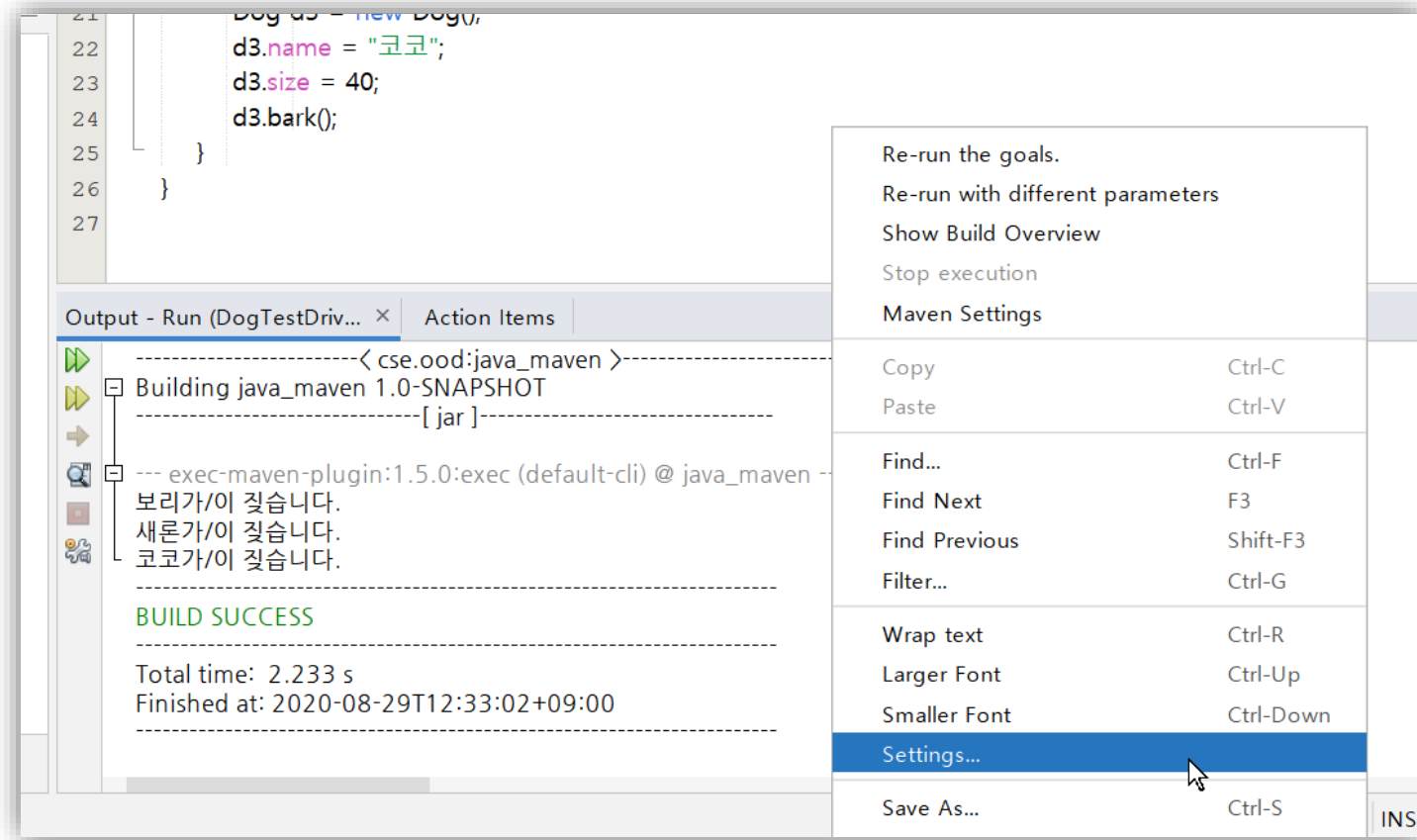
Add >

☐ Build Recursively (with modules)

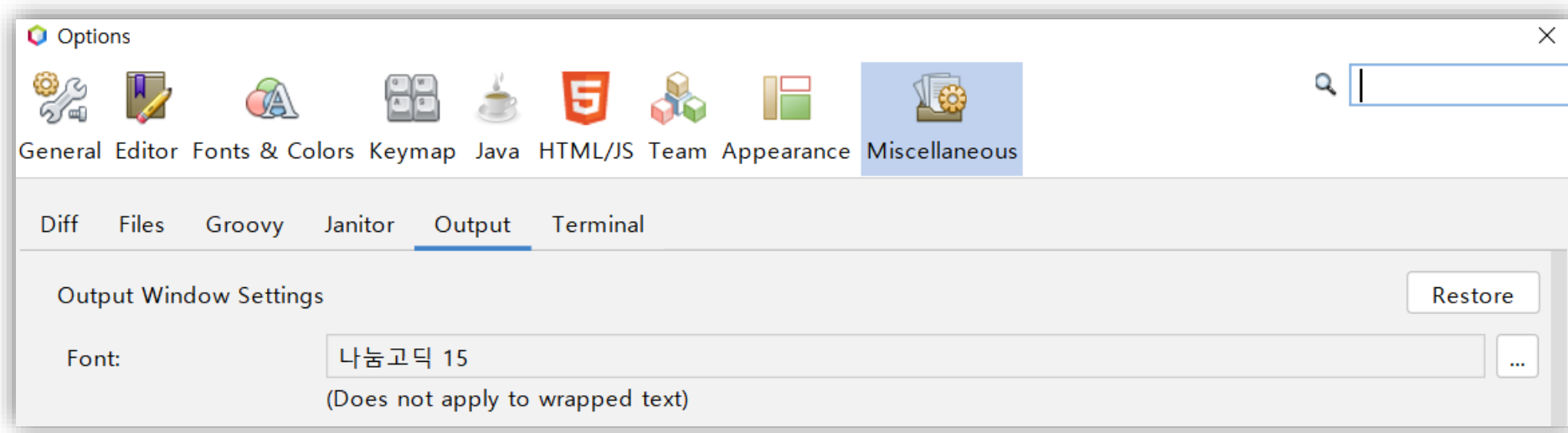
☐ Build With Dependencies

NetBeans IDE의 Output 창 한글 설정

- Output 창에서 RMB > Settings...



- 한글 폰트 설정



참고) netbeans.conf (폴더: C:\Program Files\NetBeans-12.0\netbeans\etc)

- netbeans_jdkhome 설정 `76 netbeans_jdkhome="C:\Program Files\Zulu\zulu-11"`
- netbeans_default_options 한글 설정

```
65 netbeans_default_options="-J-XX:+UseStringDeduplication -J-Xss2m -J-Djdk.gtk.version=2.2
-J-Dapple.laf.useScreenMenuBar=true -J-Dapple.awt.graphics.UseQuartz=true -J-Dsun.java2d.noddraw=true
-J-Dsun.java2d.dpiaware=true -J-Dsun.zip.disableMemoryMapping=true -J-Dplugin.manager.check.updates=false
-J-Dnetbeans.extbrowser.manual_chrome_plugin_install=yes -J--add-opens=java.base/java.net=ALL-UNNAMED
-J--add-opens=java.base/java.lang.ref=ALL-UNNAMED -J--add-opens=java.base/java.lang=ALL-UNNAMED
-J--add-opens=java.base/java.security=ALL-UNNAMED -J--add-opens=java.base/java.util=ALL-UNNAMED
-J--add-opens=java.desktop/javafx.swing.plaf.basic=ALL-UNNAMED
-J--add-opens=java.desktop/javafx.swing.text=ALL-UNNAMED
-J--add-opens=java.desktop/javafx.swing=ALL-UNNAMED -J--add-opens=java.desktop/java.awt=ALL-UNNAMED
-J--add-opens=java.desktop/java.awt.event=ALL-UNNAMED
-J--add-opens=java.prefs/java.util.prefs=ALL-UNNAMED -J--add-opens=jdk.jshell/jdk.jshell=ALL-UNNAMED
-J--add-modules=jdk.jshell -J--add-exports=java.desktop/sun.awt=ALL-UNNAMED
-J--add-exports=java.desktop/java.awt.peer=ALL-UNNAMED
-J--add-exports=java.desktop/com.sun.beans.editors=ALL-UNNAMED
-J--add-exports=java.desktop/sun.swing=ALL-UNNAMED
-J--add-exports=java.desktop/sun.awt.im=ALL-UNNAMED
-J--add-exports=jdk.internal.jvmstat/sun.jvmstat.monitor=ALL-UNNAMED
-J--add-exports=java.management/sun.management=ALL-UNNAMED
-J--add-exports=java.base/sun.reflect.annotation=ALL-UNNAMED -J-XX:+IgnoreUnrecognizedVMOptions
-J-Dfile.encoding=UTF-8 -J-locale=UTF-8"
```

Movie 객체 테스트

p.79

- 파일명: **MovieTestDrive.java**

```

4  package cse.oop2.ch02.movie;
5
6  class Movie {
7
8      String title;
9      String genre;
10     int rating;
11
12     void playIt() {
13         System.out.printf("%s(장르: %s, %d등급) "
14             + "영화를 상영합니다.%n",
15             title, genre, rating);
16     }
17 }
    
```

```

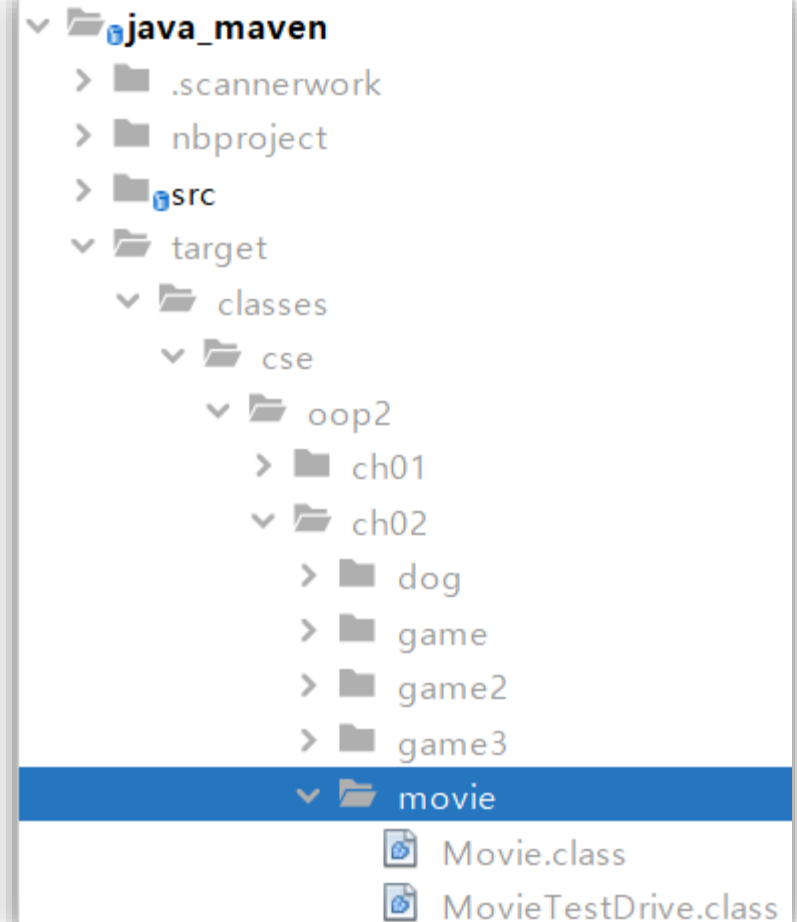
19  public class MovieTestDrive {
20
21      public static void main(String[] args) {
22          Movie one = new Movie();
23          one.title = "Gone with the Stock";
24          one.genre = "Tragic";
25          one.rating = -2;
26          Movie two = new Movie();
27          two.title = "Lost in Cubicle Space";
28          two.genre = "Comedy";
29          two.rating = 5;
30          two.playIt();
31          Movie three = new Movie();
32          three.title = "Byte Club";
33          three.genre = "Tragic but ultimately uplifting";
34          three.rating = 127;
35      }
36
37  }
    
```

Q&A

- 예상되는 실행 결과는?

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ java_maven ---
Lost in Cubicle Space(장르: Comedy, 5등급) 영화를 상영합니다.
```

- 생성되는 .class (바이트 코드) 파일은?
 - NetBeans IDE에서 Projects 탭이 아닌 Files 탭에서 확인 가능



main 메서드에서 벗어나십시오.

p.80

- main 메서드의 용도
 - 실제 클래스 테스트
 - 자바 애플리케이션 시작

자바 애플리케이션: 객체가 다른 객체와 대화를 나누는 것

다른 객체의 메서드를
호출하는 것

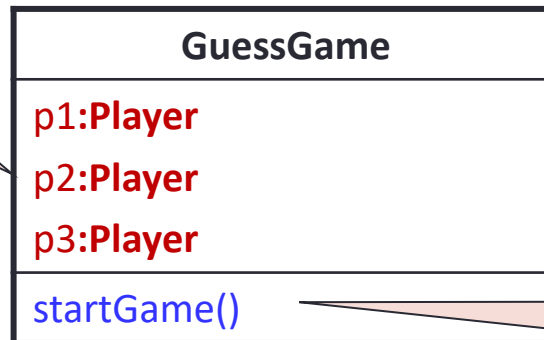
숫자 맞추기 게임 (ch02.game)

p.80

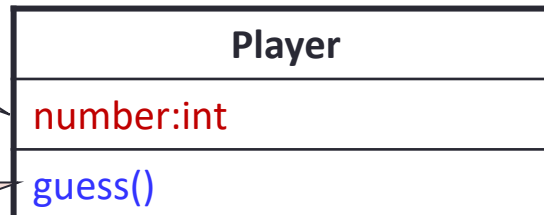
GuessGame 인스턴스를 만들고 startGame 메서드를 호출



선수 세 명을 나타내는 인스턴스 변수

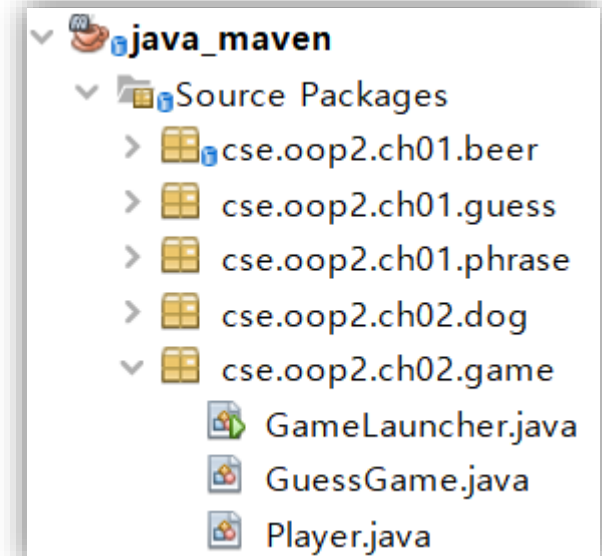


이 선수가 선택한 숫자



숫자를 선택하기 위한 메서드

실제 게임이 진행되는 부분



코드를 직접 실행시켜봅시다.

GuessGame 클래스 (1/2)

p.81

```

1  package cse.oop2.ch02.game;
2
3  public class GuessGame {
4
5      Player p1;
6      Player p2;
7      Player p3;
8
9      public void startGame() {
10         p1 = new Player();
11         p2 = new Player();
12         p3 = new Player();
13
14         int guessp1 = 0;
15         int guessp2 = 0;
16         int guessp3 = 0;
17
18         boolean p1isRight = false;
19         boolean p2isRight = false;
20         boolean p3isRight = false;

```

```

22     int targetNumber = (int) (Math.random() * 10);
23     System.out.println("0 이상 9 이하의 숫자를 맞춰보세요.");
24
25     while (true) {
26         // System.out.println("맞춰야 할 숫자는 " + targetNumber + "입니다.");
27         System.out.printf("맞춰야 할 숫자는 %d입니다.%n", targetNumber); // 수정
28
29         p1.guess();
30         p2.guess();
31         p3.guess();
32
33         guessp1 = p1.getNumber(); // p1.number;
34         System.out.printf("1번 선수가 찍은 숫자: %d%n", guessp1);
35         guessp2 = p2.getNumber(); // p2.number;
36         System.out.printf("2번 선수가 찍은 숫자: %d%n", guessp2);
37         guessp3 = p3.getNumber(); // p3.number;
38         System.out.printf("3번 선수가 찍은 숫자: %d%n", guessp3);

```

GuessGame 클래스 (2/2)

```

40         if (guessp1 == targetNumber) {
41             p1isRight = true;
42         }
43         if (guessp2 == targetNumber) {
44             p2isRight = true;
45         }
46         if (guessp3 == targetNumber) {
47             p3isRight = true;
48         }
    
```

```

52     }
53
54     if (p1isRight || p2isRight || p3isRight) {
55         System.out.println("맞춘 선수가 있습니다!");
56         System.out.printf("1번 선수: %b%n", p1isRight);
57         System.out.printf("2번 선수: %b%n", p2isRight);
58         System.out.printf("3번 선수: %b%n", p3isRight);
59         System.out.println("게임 끝");
60         break;
61     } else {
62         System.out.println("다시 시도해야 합니다.\n=====");
63     }
64 }
65 }
    
```

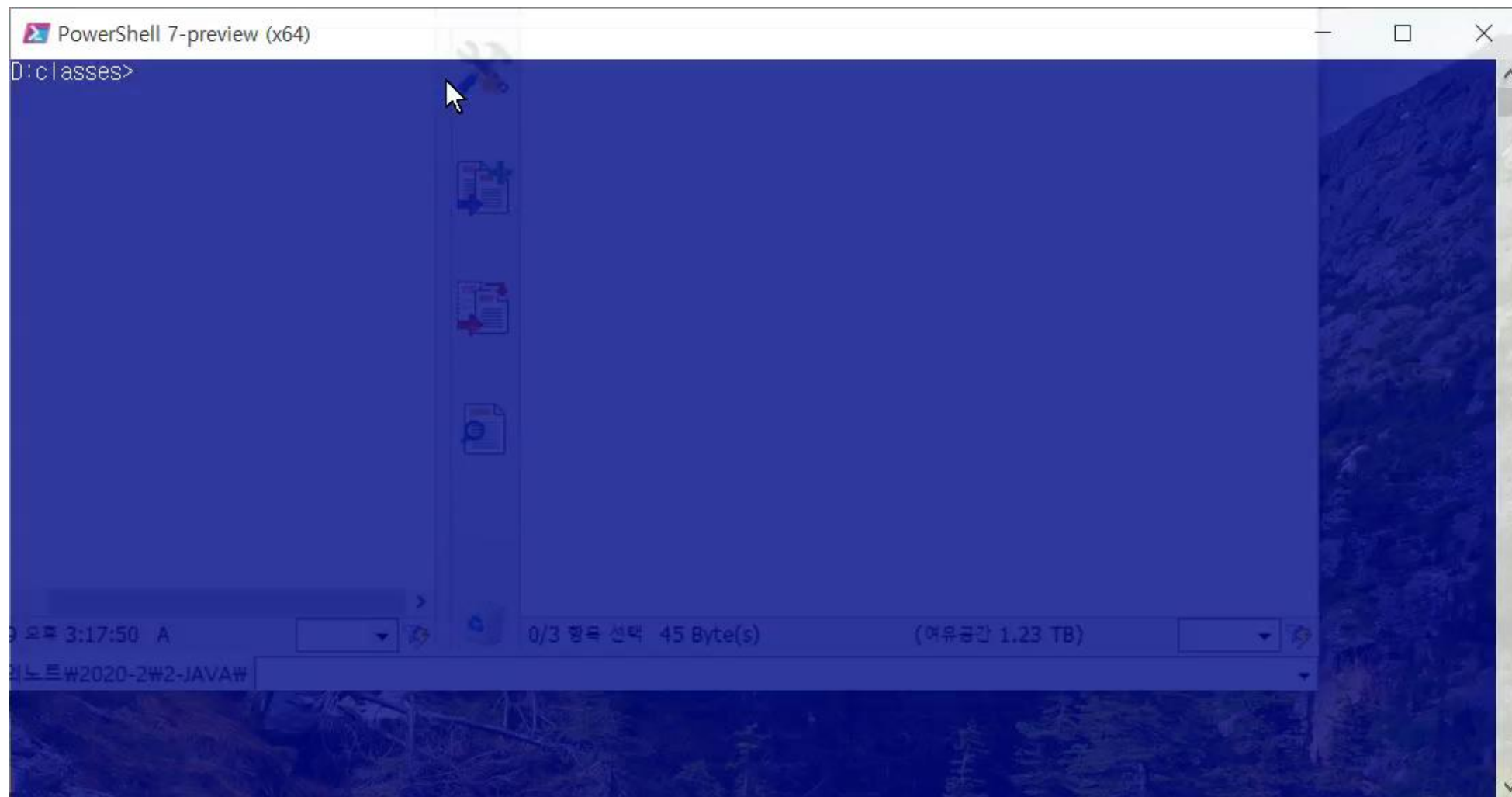
Player 클래스

```
1 package cse.oop2.ch02.game;
2
3 public class Player {
4
5     private int number = 0; // 수정
6
7     public void guess() {
8         // Math.random(): 0.0 ≤ r < 1.0인 (double)r을 반환
9         // number = (int) (java.lang.Math.random() * 10);
10        number = new java.util.Random().nextInt(10); // 수정
11        System.out.printf("찍은 숫자: %d%n", number);
12    }
13
14    // 추가
15    public int getNumber() {
16        return number;
17    }
18
19 }
```

GameLauncher 클래스

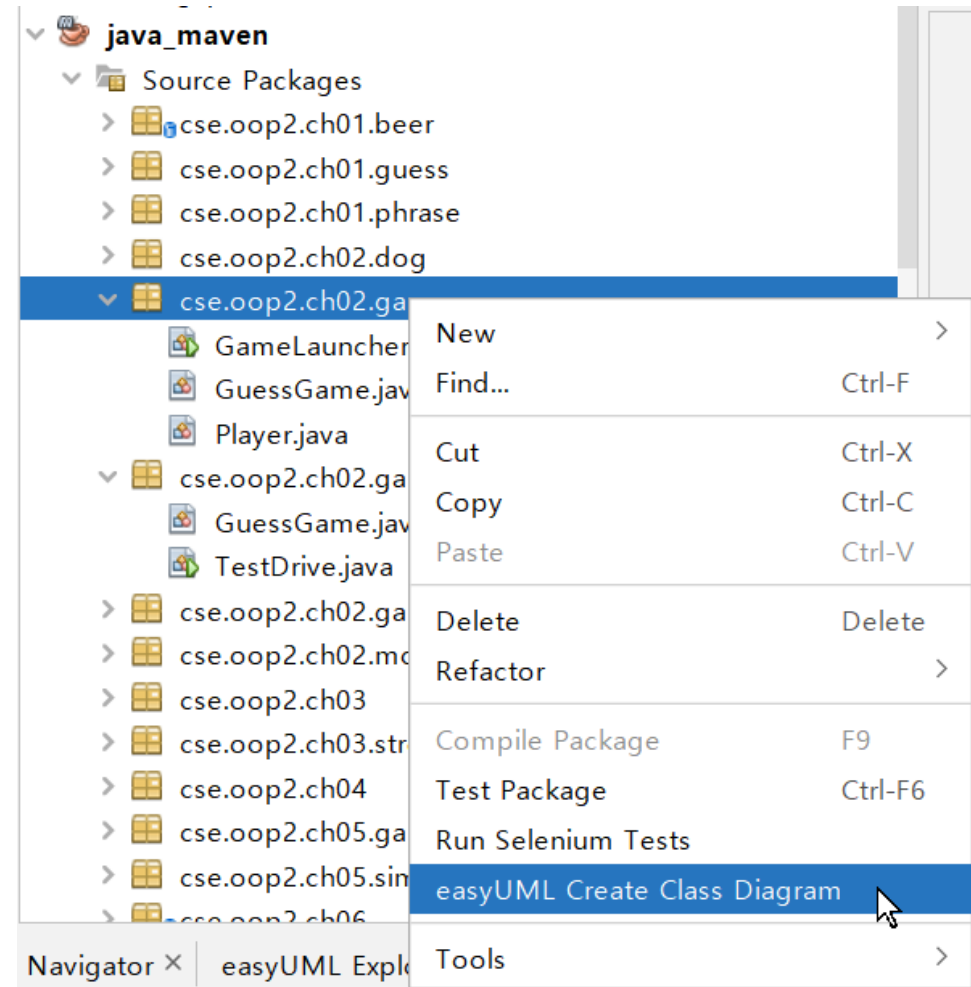
```
1 package cse.oop2.ch02.game;
2
3 public class GameLauncher {
4
5     public static void main(String[] args) {
6         GuessGame game = new GuessGame();
7         game.startGame();
8     }
9 }
```

실행 결과

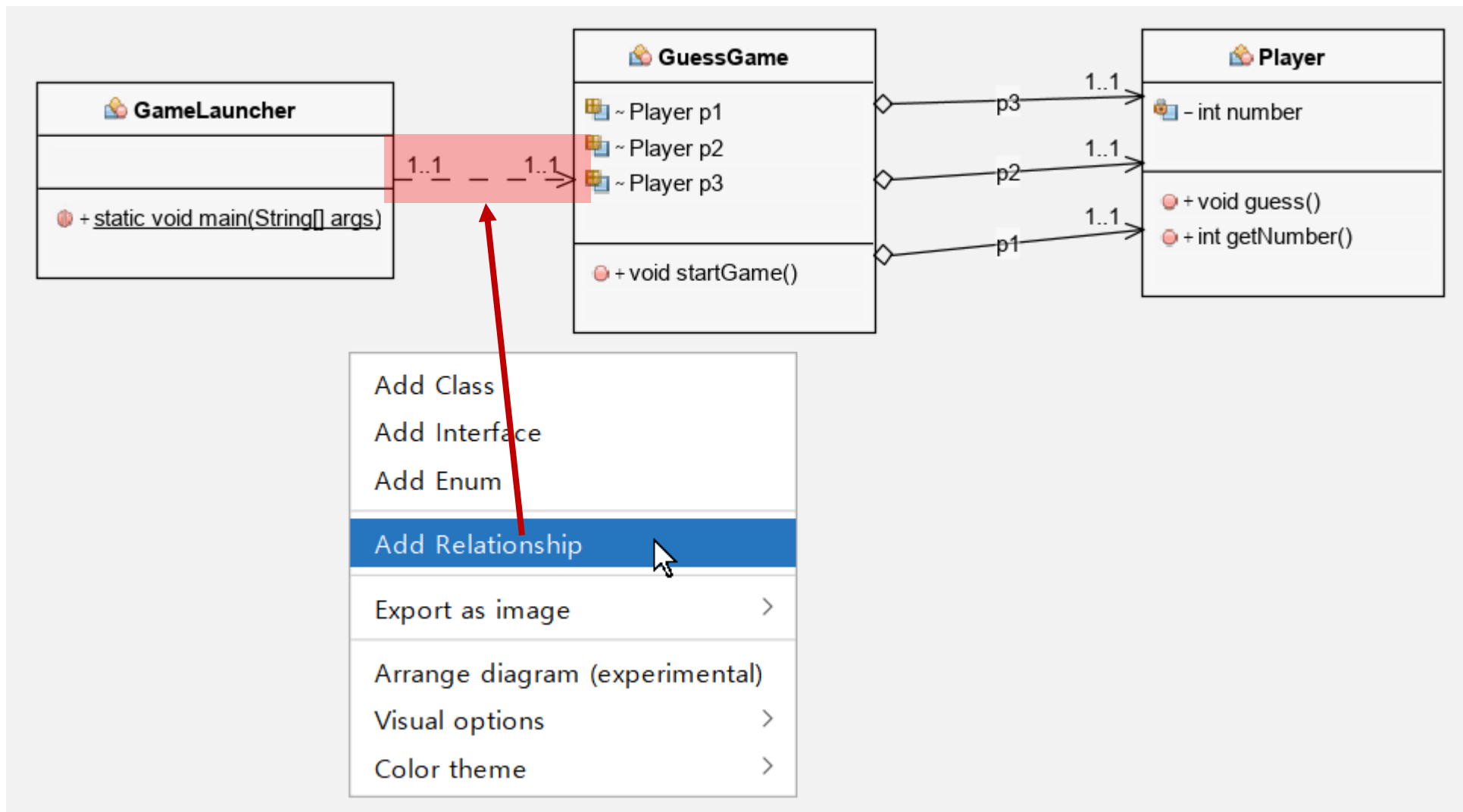


클래스 다이어그램

- easyUML 플러그인 사용
 - URL: <http://plugins.netbeans.org/plugin/55435/easyuml>
 - 또는 <https://compnet.deu.ac.kr> 홈페이지 공지사항 1018번
- 1499813117_easyuml.zip 파일을 다운로드 받아 압축 해제 후
- NetBeans IDE 메뉴의 Tools > Plugins > Downloaded 탭에서 Add Plugin... 버튼 선택하여 관련 nbm 파일을 모두 설치



클래스 다이어그램: easyUML 플러그인



클래스 다이어그램: plantUML



바보 같은 질문은 없습니다

- 전역 변수나 전역 메서드가 필요할 때는 어떻게 해야 할까요? 모든 것을 클래스에 넣어야 한다면 그런 것을 넣을 자리가 없지 않나요?
- 자바 객체지향 프로그램에는 '전역' 변수나 '전역' 메서드의 개념이 아예 없습니다. 하지만 어디에서든 쓸 수 있는 것을 만들어야 하는 경우가 있는데 그런 경우에는 `public static`(메서드), `public static final`(상수)로 선언하면 어디에서든지 사용할 수 있도록 만들 수 있습니다.

바보 같은 질문은 없습니다

- 그러면 전역 함수나 전역 데이터를 만들면서 어떻게 객체 지향적이라고 할 수 있나요?
 - 자바에서는 모든 것이 클래스 안에 들어갑니다. π 같은 상수나 `random()` 같은 메서드는 모두 `Math`라는 클래스에 들어있습니다. 사실 이런 `static`으로 정의된 것들은 예외적인 부류에 속합니다. 이런 변수와 메서드는 여러 인스턴스/객체에 들어가지 않는 매우 특별한 경우에 해당합니다.
- 참고: 유틸리티 클래스 (모든 필드와 메서드가 정적인 클래스)
 - 예) `java.lang.Math`

바보 같은 질문은 없습니다

- 자바 프로그램이 뭔가요? 실질적인 최종 결과물은 무엇인가요?
 - 자바 프로그램은 한 개 이상의 클래스가 모여있는 것이라고 할 수 있습니다. 그 중 한 클래스에는 반드시 main 메서드가 있어야 하면 그 메서드를 통해서 프로그램이 시작됩니다. 그리고 그러한 (한 개 이상의) 클래스가 바로 실질적인 최종 결과물입니다. 실제로 사용자에게 프로그램을 제공할 때는 상황에 따라 JVM 등을 함께 제공해야 할 수도 있습니다.
- 클래스가 수백, 수천 개가 된다면 어떻게 해야 할까요?
 - 클래스 파일(바이트 코드)이 많은 경우 **.jar** 또는 **.war** 파일로 만들어 배포하여 사용 (17장)
 - JAR: Java Archive
 - WAR: Web application Archive

핵심 정리

- 객체지향적인 프로그래밍을 하면 제작과 테스트 과정이 끝난 코드를 건드리지 않고도 프로그램을 확장할 수 있습니다.
- 모든 자바 코드는 클래스 내에서 정의됩니다.
- 클래스는 해당 클래스 유형의 객체를 만드는 방법을 설명하는 역할을 합니다. 클래스는 청사진 또는 템플릿과 같다고 할 수 있습니다.
- 객체는 각자 알아서 자기 할 일을 처리할 수 있습니다. 사용자는 객체에서 작업을 처리하는 방법에 대해서는 신경 쓰지 않아도 됩니다.

핵심 정리

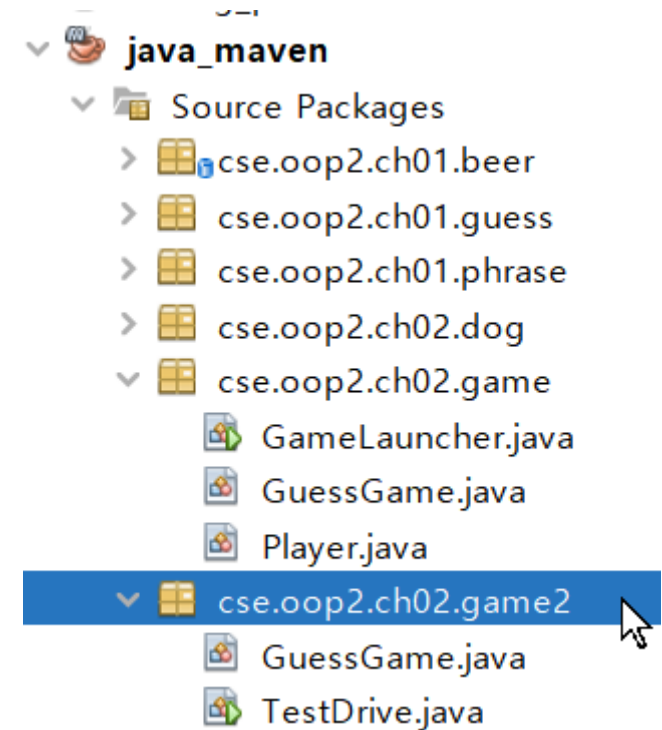
- 객체에는 알고 있는 것(**state, 상태**)과 할 수 있는 것(**행위, behavior**)이 있습니다.
- 객체가 자기 자신에 대해 알고 있는 것은 **인스턴스 변수**(또는 **필드, 멤버 변수**)라고 부릅니다. 객체의 상태를 나타내지요.
- 객체가 할 수 있는 것은 **메서드**(또는 **멤버 함수**)라고 부릅니다. 객체의 행동을 나타내지요.

핵심 정리

- 클래스를 새로 만들 때는 그 클래스 유형의 객체를 만들어서 테스트하는 테스트용 클래스를 따로 만들어보는 것이 좋습니다.
- 클래스에서는 덜 구체적인 수퍼클래스로부터 인스턴스 변수와 메서드를 상속할 수 있습니다.
- 프로그램을 실행시킬 때 보면 자바 프로그램이란 결국 다른 객체와 대화를 하는 객체에 불과합니다. → "객체 간 메시지를 통하여 의사소통한다!", "message passing"

GuessGame 변경 (ch02.game2)

- 게임 설명
 - 컴퓨터가 임의 생성한 0~9까지 한 자리 숫자를 맞추는 게임입니다.
 - 교재의 Player 클래스가 아니라 사용자가 직접 한 자리 숫자를 입력합니다.
 - 5번 안에 맞추면 당신이 이기고 그렇지 않으면 컴퓨터가 이깁니다.
- cse.oop2.ch02.game2 패키지 생성



GuessGame 클래스 소스

```

6  package cse.oop2.ch02.game2;
7
8  import java.io.BufferedReader;
9  import java.io.IOException;
10 import java.io.InputStreamReader;
11
12 /**
13  * 숫자 추측 게임
14  *
15  * @author jongmin
16  * @since 2016년 9월 20일
17  */
18 public class GuessGame {
19
20     private int machineGeneratedNumber;
21     private int userGuessNumber = -1;
22     public static final int MAX_COUNT = 5; // 상수
23
24     // 생성자 정의 없음.

```

GuessGame.
-machineGeneratedNumber: int -userGuessNumber: int = -1 -MAX_COUNT: int = 5
+getUserInput(): void -generateRandomNumber(): void +startGame(): void

```
26 public void getUserInput() throws IOException {
27     String inputLine;
28
29     System.out.print("0~9까지 한 자리 숫자를 추측해서 입력해 주세요: ");
30
31     boolean rightNumberFormat = false;
32     do {
33         BufferedReader is = new BufferedReader(new InputStreamReader(System.in));
34         inputLine = is.readLine();
35
36         if (inputLine.matches("[0-9]")) { //if (inputLine.matches("WWd")) {
37             this.userGuessNumber = Integer.parseInt(inputLine);
38             rightNumberFormat = true;
39         } else {
40             System.out.println("0~9까지의 한 자리 숫자만 입력해야 합니다! 다시 입력해 주세요: ");
41         }
42     } while (!rightNumberFormat); // rightNumberFormat == false
43 } // end of getUserInput()
44
45 private void generateRandomNumber() {
46     this.machineGeneratedNumber = (int) (Math.random() * 10);
47     // System.out.println("machineGeneratedNumber = " + machineGeneratedNumber);
48 }
```

입력의 유효성
확인

```
50 public void startGame() throws IOException {
51     int count = 0;
52     boolean matched = false;
53
54     System.out.println("컴퓨터가 임의로 생성한 0~9까지 한 자리 숫자를 맞추는 게임입니다.");
55     System.out.println("5번 안에 맞추면 당신이 이기고 그렇지 않으면 컴퓨터가 이깁니다.");
56     System.out.println("이제 시작하겠습니다.");
57     System.out.println("");
58
59     generateRandomNumber();
60
61     while (!matched && count < MAX_COUNT) {
62         getUserInput();
63         if (userGuessNumber == machineGeneratedNumber) {
64             matched = true;
65         } else {
66             count++;
67         }
68     } // end of while()
69
70     if (count < MAX_COUNT) { //or matched
71         System.out.println("당신이 이겼습니다!");
72     } else {
73         System.out.println("컴퓨터가 이겼습니다!");
74     }
75 } // end of startGame()
76 }
```

사용자 입력

- System.in으로부터 사용자 입력 받아들이 수 있음.
- BufferedReader, InputStreamReader는 **데코레이터 패턴** 사용 (3학년 소프트웨어설계공학)
- 줄 단위 문자열 입력: 엔터키 누를 때까지 문자열 읽어 들임.
 - 참고: <https://docs.oracle.com/javase/8/docs/api/>
 - 교재: p.186

```
BufferedReader is = new BufferedReader(new InputStreamReader(System.in));
inputLine = is.readLine();
```

- **BufferedReader**
 - Character input stream(문자 입력 흐름)으로부터 버퍼링을 통해 텍스트를 읽는 클래스.
- **InputStreamReader**
 - InputStream 객체가 생성하는 **byte stream(바이트 흐름)**을 **character stream(문자 흐름)**으로 바꿔주는 연결 (bridge) 클래스
 - 참고) InputStreamReader(InputStream in, String charsetName) → 문자 인코딩 지정 가능: CP949, UTF-8 등
- **System.in**
 - (static InputStream): 표준 입력흐름
- **참고**
 - **System.out**: 표준 출력 흐름
 - **System.err**: 표준 오류 흐름

정규 표현식을 이용한 문자열 비교

- 정규 표현식(regular expression)을 이용한 문자열 비교

```
36 if (inputLine.matches("[0-9]")) { //if (inputLine.matches("WWd")) {  
37     this.userGuessNumber = Integer.parseInt(inputLine);  
38     rightNumberFormat = true;  
39 } else {
```

정규 표현식 사용 방법 1

Character classes

[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-~q-z] (subtraction)

Predefined character classes

.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\h	A horizontal whitespace character: [\t\xA0\u1680\u180e\u2000-\u200a\u202f\u205f\u3000]
\H	A non-horizontal whitespace character: [^\h]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\v	A vertical whitespace character: [\n\x0B\f\r\x85\u2028\u2029]
\V	A non-vertical whitespace character: [^\v]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

정규 표현식 사용 방법 2

Boundary matchers

<code>^</code>	The beginning of a line
<code>\$</code>	The end of a line
<code>\b</code>	A word boundary
<code>\B</code>	A non-word boundary
<code>\A</code>	The beginning of the input
<code>\G</code>	The end of the previous match
<code>\Z</code>	The end of the input but for the final terminator, if any
<code>\z</code>	The end of the input

Greedy quantifiers

<code>X?</code>	<code>X</code> , once or not at all
<code>X*</code>	<code>X</code> , zero or more times
<code>X+</code>	<code>X</code> , one or more times
<code>X{n}</code>	<code>X</code> , exactly <code>n</code> times
<code>X{n,}</code>	<code>X</code> , at least <code>n</code> times
<code>X{n,m}</code>	<code>X</code> , at least <code>n</code> but not more than <code>m</code> times

// 0 ~ 99는? 단, 01형태는 안 됨.
→ “[1-9]?\\w\\w d” , “[1-9]?[0-9]”

Logical operators

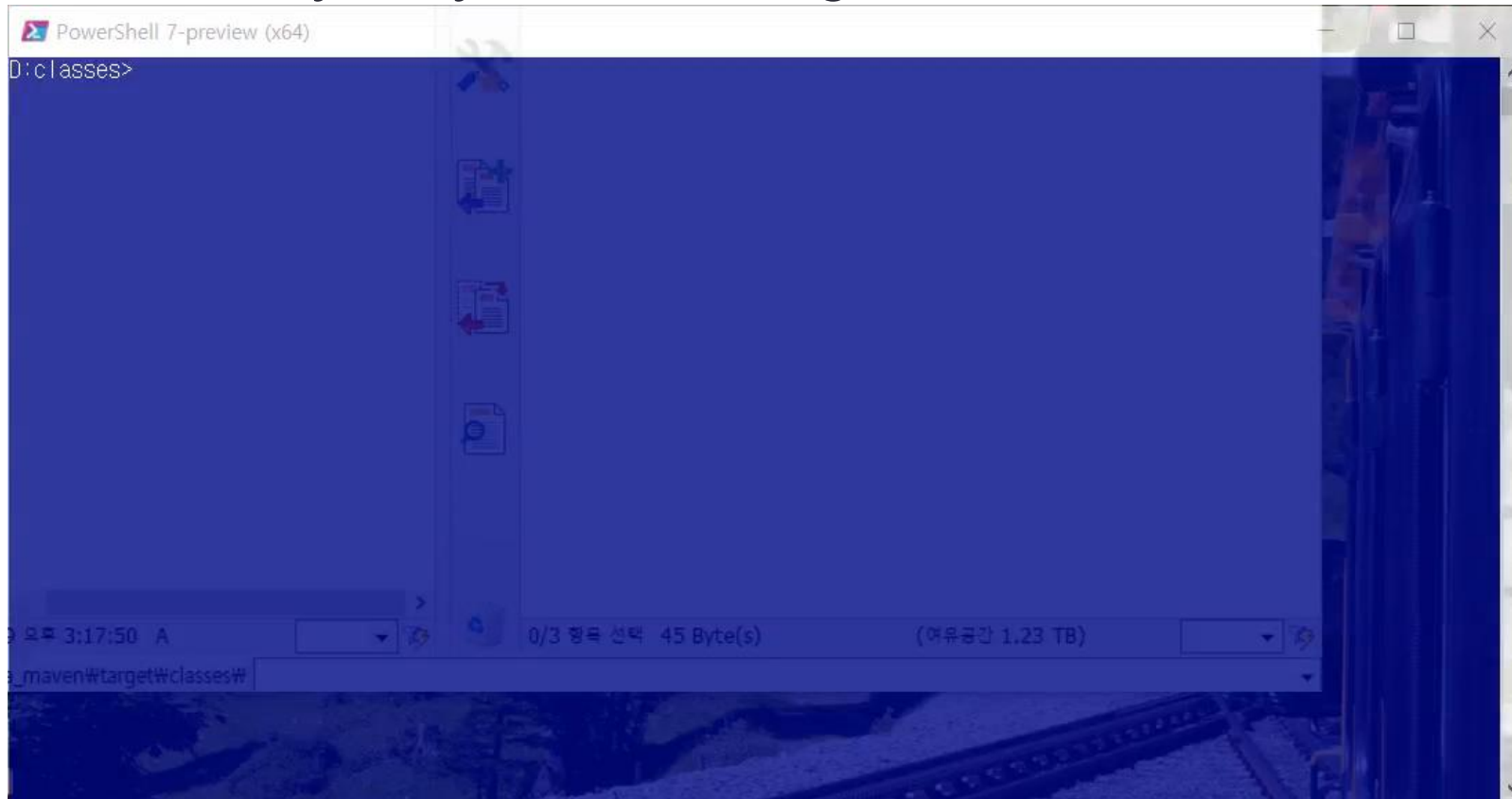
<code>XY</code>	<code>X</code> followed by <code>Y</code>
<code>X Y</code>	Either <code>X</code> or <code>Y</code>
<code>(X)</code>	<code>X</code> , as a capturing group

TestDrive 클래스

```
6      package cse.oop2.ch02.game2;
7
8      [-] import java.io.IOException;
9
10     [+] /**...4 lines */
14     public class TestDrive {
15
16     [+] /**...4 lines */
20     [-] public static void main(String[] args) throws IOException {
21         // TODO code application logic here
22         GuessGame game = new GuessGame();
23         game.startGame();
24     }
25
26 }
```

실행 결과

- NetBeans IDE에서 Run File 하여 실행 시 잘 동작 안 할 수도 있음.
- 그런 경우에는 명령창(cmd or PowerShell 등)에서 실행하여 동작 확인해 보면 됨.
- 경로명: D:\NetBeansProjects\java_maven\target\classes



HW: 고객의 요구사항 변경 (ch02.game3)

- 1) 0 ~ 99까지의 숫자를 맞추는 게임으로 바꾸고 싶네요.
- 2) 두 명의 사용자가 동시에 사용할 수 있으면 좋겠습니다. 물론 두 명의 사용자가 모두 5번 안에 못 맞추면 컴퓨터가 이깁니다.
 - 예시)

컴퓨터가 임의로 생성한 0~99까지 숫자를 맞추는 게임입니다.
5번 안에 맞추면 **맞춘 사람**이 이기고 그렇지 않으면 컴퓨터가 이깁니다.
이제 시작하겠습니다.

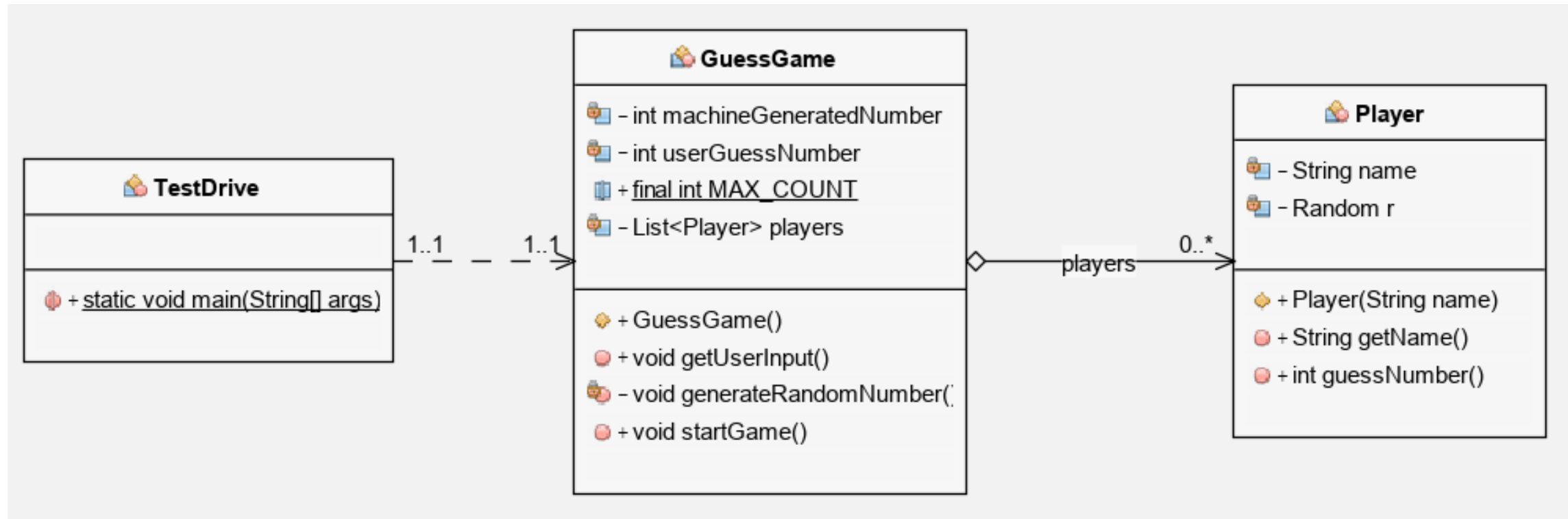
사용자 이름을 차례대로 입력해 주세요: **홍길동/전우치**

홍길동님 차례입니다:
0~99까지 숫자를 추측해서 입력해 주세요: 1
틀렸습니다.

전우치님 차례입니다:
0~99까지 숫자를 추측해서 입력해 주세요: 4
전우치님이 이겼습니다.

참고: ch02.game3의 클래스 다이어그램

- ch02.game2의 구조를 최대한 활용



예상 실행 결과

컴퓨터가 임의로 생성한 0~99까지 숫자를 맞추는 게임입니다.
5번 안에 맞추면 맞춘 사람이 이기고 그렇지 않으면 컴퓨터가 이깁니다.
이제 시작하겠습니다.

사용자 이름을 차례대로 입력해 주세요: 홍길동/전우치

machineGeneratedNumber = 68

홍길동님 차례입니다.

0~99까지의 숫자를 추측해서 입력해 주세요: 23
틀렸습니다.

전우치님 차례입니다.

0~99까지의 숫자를 추측해서 입력해 주세요: 33
틀렸습니다.

홍길동님 차례입니다.

0~99까지의 숫자를 추측해서 입력해 주세요: 88
틀렸습니다.

전우치님 차례입니다.

0~99까지의 숫자를 추측해서 입력해 주세요: 47
틀렸습니다.