

Writing a Program: Extra-Strength Methods

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hankyong.ac.kr>

“Sink a Dot Com” 게임을 만들어보자

목표: 컴퓨터가 가지고 있는 모든 닷컴명을 최소한의 추측으로 가라앉히기.

닷컴을 모두 가라앉히고 나면 성적에 따라 등급이 매겨진다.

설정: 게임이 시작되면 컴퓨터에서 닷컴 3개를 가상의 7x7 그리드 상에 배치한다.

이 작업이 끝나면 사용자에게 닷컴 위치를 추측하도록 요청한다.

게임 방법:

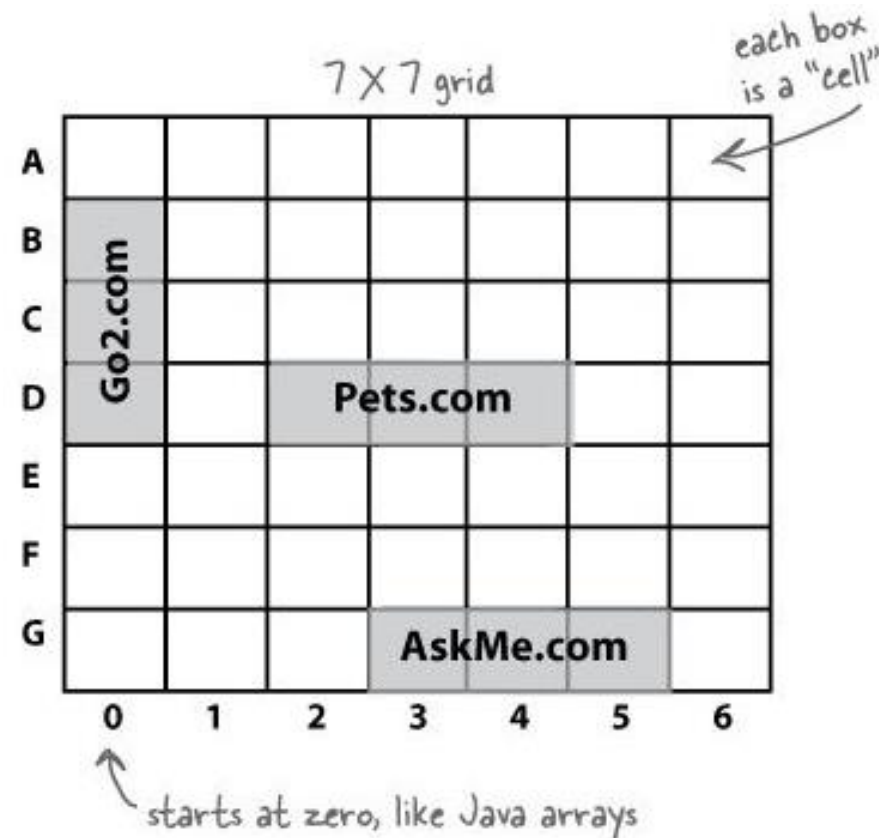
현재는 명령행에서 실행하는 버전으로 만든다.
(GUI 버전은 나중에...)

컴퓨터에서 명령행에 위치를 추측해 보라고
프롬프트 한다.

사용자는 “A3”, “C5” 이런 식으로 명령행에 위치
를 입력하면 된다.

컴퓨터에서는 맞으면 “Hit” 틀리면 “Miss”라고
결과를 알려준다.

닷컴 3개를 모두 가라앉히면 등급이 출력된다.



게임 진행 화면

```
File Edit Window Help Sell
%java DotComBust
Enter a guess A3
miss
Enter a guess B2
miss
Enter a guess C4
miss
Enter a guess D2
hit
Enter a guess D3
hit
Enter a guess D4
Ouch! You sunk Pets.com : (
kill
Enter a guess B4
miss
Enter a guess G3
hit
Enter a guess G4
hit
Enter a guess G5
Ouch! You sunk AskMe.com : (
```

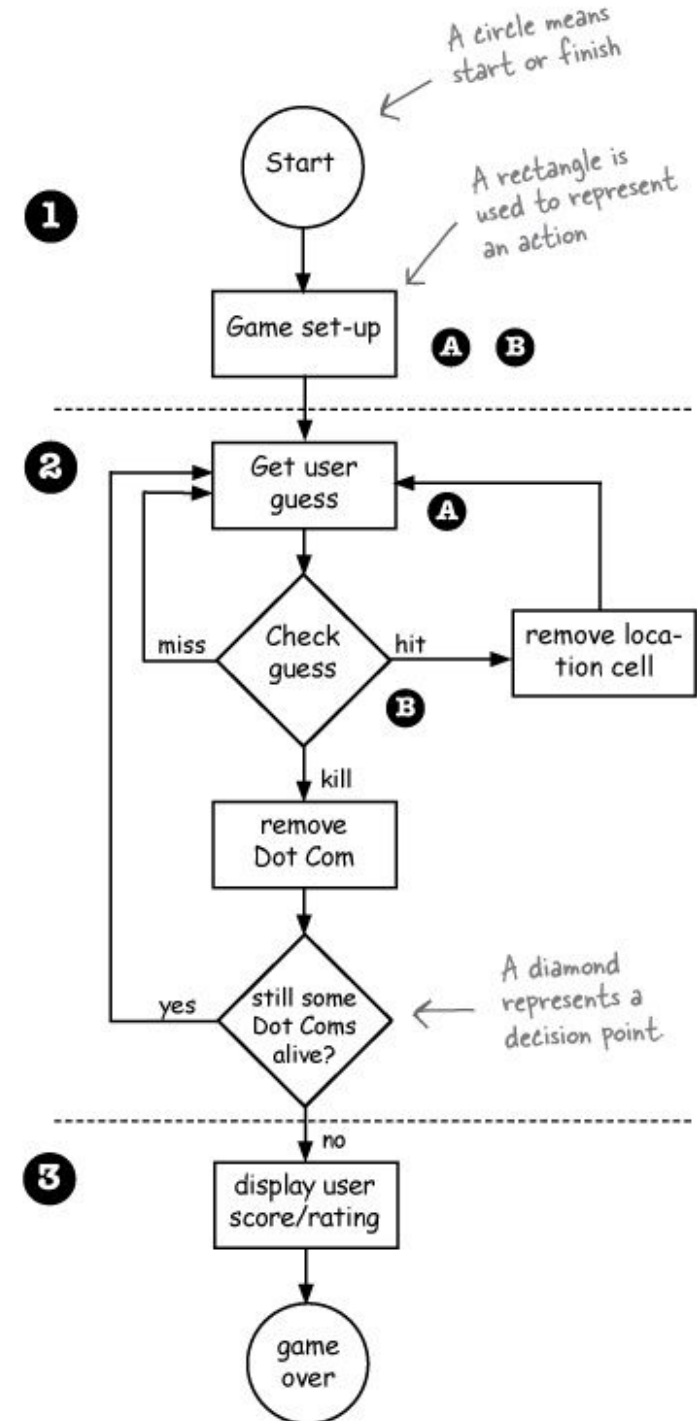
First, a high-level design

1. 사용자가 게임을 시작한다. (게임 셋업)
 - A. Game이 닳컴 3개를 생성한다
 - B. Game이 닳컴 3개를 가상 그리드에 배치시킨다
2. 게임 플레이가 시작된다.

닳컴이 하나도 남지 않게 될 때까지 다음 과정을 반복한다:

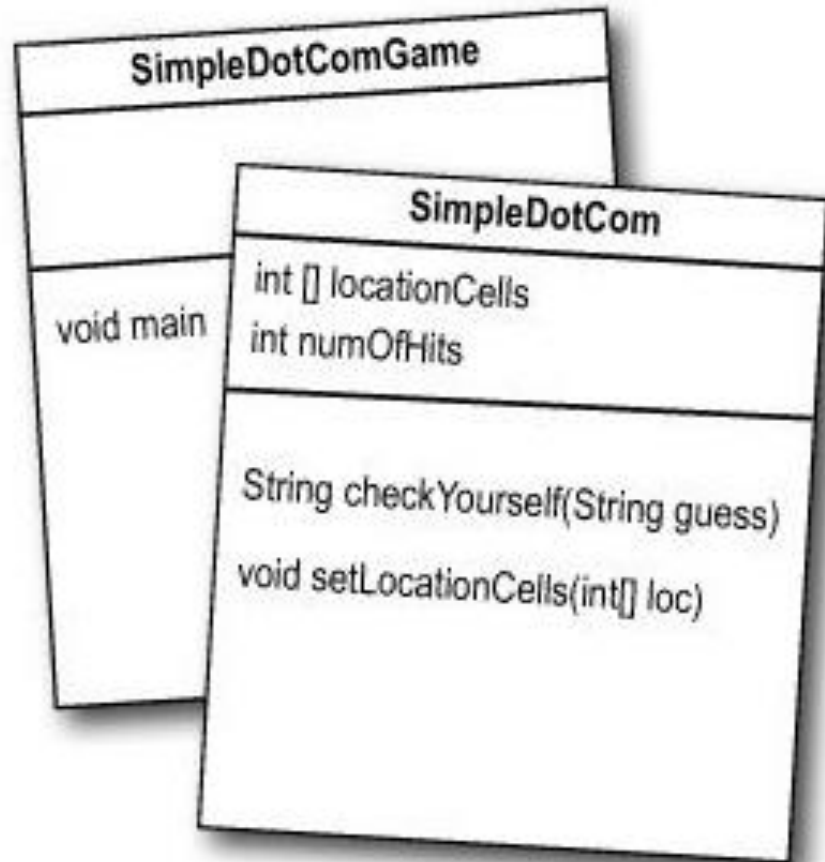
 - A. 사용자가 예상 위치 ("A2", "C0" 등)를 입력하도록 프롬프트 한다.
 - B. 사용자가 입력한 위치가 맞는지, 틀리는지를 체크하여 히트이면 셀을 삭제하고, 킬이면 닳컴을 가라앉힌다.

킬 (kill) => 닳컴의 해당 셀 3개가 모두 삭제된 경우
3. 게임을 종료한다.
 - A. 사용자가 추측한 총 횟수를 근거로 등급을 매긴다.



“Simple Dot Com Game” 소개

완전한 닷컴 가라앉히기 게임을 만들기 전에 **Simple** 닷컴 게임을 만들어보자.
(게임을 최대한 단순하게 만들어본다)



닷컴을 2D 그리드가 아닌 한 줄에 배치하고 닷컴 개수도 한 개로 제한한다.

목표는 같다:

닷컴 인스턴스를 만들어서 행의 어딘가에 위치시키고, 사용자가 입력한 내용을 받아오고, 닷컴의 모든 셀이 맞았으면 게임을 종료한다.

이 **Simple** 버전에서는 게임 클래스에 어떤 인스턴스 변수도 없고 모든 게임 코드는 **main()** 메소드에 있다.

즉, 프로그램이 시작되고 **main()**이 실행되면 닷컴 인스턴스를 한 개만 만들고 그 위치(7개의 셀을 가진 가상의 행에서 연속적인 셀 3개)를 고르고, 사용자에게 추측하도록 하고, 그 추측을 체크하고 모든 3개의 셀이 히트될 때까지 반복한다.

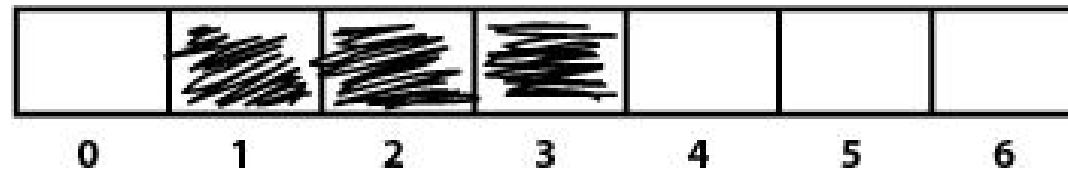
가상의 행임을 기억하자. 즉, 이 행은 프로그램의 어디에도 존재하지 않는다.

(게임 객체와 사용자가 7개의 셀 어딘가에 닳컴이 들어 있다는 것을 알고 있다면 구태여 코드에서 그 행을 표현할 필요가 없다)

1. 게임을 시작한다.

닳컴 하나를 만들고 셀 7개 가운데 연속된 셀 3개에 위치시킨다.

(위치 표시는 1,2,3과 같이 정수로만 표시한다. 즉 "A2", "C4" 대신에 1, 2, 3과 같이 표시)



2. 게임을 진행한다.
사용자에게 추측하도록 프롬프트하고
닷컴의 3개 셀 중에 하나와 일치하는지 체크하고
일치한다면 **numOfHits** 값을 증가시킨다.
3. 셀 3개를 모두 맞추면 (**numOfHits** 변수 값이 3이
되면) 게임이 종료된다.
사용자에게 몇 번의 추측 끝에 닷컴을 가라앉혔는지
알려준다.

```
File Edit Window Help Destroy
%java SimpleDotComGame
enter a number 2
hit
enter a number 3
hit
enter a number 4
miss
enter a number 1
kill
You took 4 guesses
```

클래스 개발

코드 작성에 있어서 자신만의 방법론이 있을 수 있지만 여기서는 배우는 과정의 일부분이므로 자바 클래스를 만들 때 다음 방식을 따르는 것을 권장한다:

- 클래스에서 무엇을 수행해야 하는지 파악
- 인스턴스 변수와 메소드 목록 작성
- 메소드 작성을 위한 준비 코드 (prepcode) 작성
- 메소드에 대한 테스트 코드 작성
- 클래스 구현
- 메소드 테스트
- 필요하다면 디버그 수행 또는 다시 구현

각 클래스별로 작성해야 할 3가지

prep code test code real code

준비 코드(prepare code)

문법보다는 논리를 중점적으로 살펴보기 위해 유사코드 형태로 표현한다.

테스트 코드(test code)

실제 코드를 테스트하고 작업이 제대로 동작하는지 확인하기 위한 클래스 또는 메소드

실제 코드(real code)

클래스를 실제로 구현한 코드

To Do:

SimpleDotCom class

- write prep code
- write test code
- write final Java code

SimpleDotComGame class

- write prep code
- write test code [no]
- write final Java code

SimpleDotCom
int [] locationCells int numOfHits
String checkYourself(String guess) void setLocationCells(int[] loc)

준비 코드 구성:

- 인스턴스 변수 선언
- 메소드 선언
- 메소드 로직 <= 가장 중요한 부분

⇒ 실제 코드를 작성할 때 이 메소드 로직을 바탕으로 결정하게 된다.

선언

셀의 위치를 저장하기 위한 **locationCells**라는 int 배열을 선언한다.

맞춘 셀의 개수를 저장하기 위한 **numOfHits**라는 **int**를 선언하고 값은 0으로 설정한다.

사용자가 추측한 위치를 문자열로 받아들이고 그 값을 체크하여 "hit", "miss", "kill" 중 하나를 나타내는 결과를 리턴하는 **checkYourself()** 라는 메소드를 선언한다.

int 배열 (셀 위치 3개를 나타내는 **ints**(2, 3, 4, etc.))을 받아들여 닷컴을 배치시키는 **setLocationCells()** 라는 세터 메소드를 선언한다.

METHOD: String checkYourself(String userGuess)

GET the user guess as a String parameter

CONVERT the user guess to an *int*

REPEAT with each of the location cells in the *int* array

// COMPARE the user guess to the location cell

IF the user guess matches

INCREMENT the number of hits

// FIND OUT if it was the last location cell:

IF number of hits is 3, **RETURN** "kill" as the result

ELSE it was not a kill, so **RETURN** "hit"

END IF

ELSE the user guess did not match, so **RETURN** "miss"

END IF

END REPEAT

END METHOD

METHOD: void setLocationCells(int[] cellLocations)

GET the cell locations as an *int array* parameter

ASSIGN the cell locations parameter to the cell locations instance variable

END METHOD

메소드 구현

이제 실제 메소드 코드를 작성해보자.

메소드를 코딩 하기 전에 미리 테스트 코드를 만들어 놓는 것이 좋다.

이 방법은 **익스트림 프로그래밍** (XP, eXtreme Programming) 규칙 중의 하나이다.



Extreme Programming (XP)

Extreme Programming(XP) is a newcomer to the software development methodology world. Considered by many to be **"the way programmers really want to work"**, XP emerged in the late 90's and has been adopted by companies ranging from the two-person garage shop to the Ford Motor Company.

XP is based on a set of proven practices that are all designed to work together, although many folks do pick and choose, and adopt only a portion of XP's rules. These practices include things like:

- Make small, but frequent, releases.
- Develop in iteration cycles.
- Don't put in anything that's not in the spec (no matter how tempted you are to put in functionality "for the future").
- Write the test code first.
- No killer schedules; work regular hours.
- Refactor (improve the code) whenever and wherever you notice the opportunity.
- Don't release anything until it passes all the tests.
- Set realistic schedules, based around small releases.
- Keep it simple.
- Program in pairs, and move people around so that everybody knows pretty much everything about the code.

SimpleDotCom 클래스를 위한 테스트 코드

Based on this precode:

```
METHOD String checkYourself(String userGuess)
GET the user guess as a String parameter
CONVERT the user guess to an int
REPEAT with each of the location cells in the int array
// COMPARE the user guess to the location cell
IF the user guess matches
    INCREMENT the number of hits
    // FIND OUT if it was the last location cell
    IF number of hits is 3, RETURN "Kill" as
    ELSE it was not a kill, so RETURN "Hit"
    END IF
ELSE the user guess did not match, so RETURN
END IF
END REPEAT
END METHOD
```

SimpleDotCom 객체를 만들고 이 메소드를
실행하는 테스트 코드를 만들어야 한다.

다음과 같은 것을 테스트해야 한다:

1. **SimpleDotCom** 객체의 인스턴스를 생성한다.
2. 닷컴 위치를 할당한다.
3. 사용자가 추측한 위치를 나타내는 **String**을 만든다.
4. 3단계에서 만들었던 **String**을 전달하면서 **checkYourself()** 메소드를 호출한다.
5. 결과를 출력하여 옳은 결과가 나왔는지 확인한다.

SimpleDotCom 클래스용 테스트 코드

```
public class SimpleDotComTestDrive {  
    public static void main (String[] args) {  
        SimpleDotCom dot = new SimpleDotCom();  
  
        int[] locations = {2,3,4};  
        dot.setLocationCells(locations);  
  
        String userGuess = "2";  
        String result = dot.checkYourself(userGuess);  
        String testResult = "failed";  
        if (result.equals("hit") ) {  
            testResult = "passed";  
        }  
  
        System.out.println(testResult);  
    }  
}
```

instantiate a SimpleDotCom object

make an int array for the location of the dot com (3 consecutive ints out of a possible 7).

invoke the setter method on the dot com.

make a fake user guess

invoke the checkYourself() method on the dot com object, and pass it the fake guess.

if the fake guess (2) gives back a "hit", it's working

print out the test result (passed or failed)

다음과 같은 것을 테스트해야 한다:

1. **SimpleDotCom** 객체의 인스턴스를 생성한다.
2. 닷컴 위치를 할당한다.
3. 사용자가 추측한 위치를 나타내는 **String**을 만든다.
4. 3단계에서 만들었던 **String**을 전달하면서 **checkYourself()** 메소드를 호출한다.
5. 결과를 출력하여 옳은 결과가 나왔는지 확인한다.

checkYourself() 메소드

GET the user
guess

CONVERT
the user guess to
an int

REPEAT with
each cell in the int
array

IF the user guess
matches

INCREMENT
the number of
hits

```
public String checkYourself(String stringGuess) {
```

```
    int guess = Integer.parseInt(stringGuess);
```

① ← convert the String to an int

```
    String result = "miss";
```

← make a variable to hold the result we'll return. put "miss" in as the default (i.e. we assume a "miss")

```
    for (int cell : locationCells) {
```

② ← repeat with each cell in the locationCells array (each cell location of the object)

```
        if (guess == cell) {
```

← compare the user guess to this element (cell) in the array

```
            result = "hit";
```

```
            numOfHits++;
```

③ ← we got a hit!

```
            break;
```

④ ← get out of the loop, no need to test the other cells

```
        } // end if
```

```
    } // end for
```


// FIND OUT if
it was the last cell
IF number of hits
is 3,
RETURN "kill"
as the result
ELSE it was
not a kill, so
RETURN "hit"
ELSE
RETURN
"miss"

```
if (numOfHits == locationCells.length) {  
    result = "kill";  
} // end if  
  
System.out.println(result);  
  
return result;  
} // end method
```

← we're out of the loop, but let's see if we're now 'dead' (hit 3 times) and change the result String to "Kill"

← display the result for the user
("Miss", unless it was changed to "Hit" or "Kill")

← return the result back to
the calling method

새로운 내용 설명

① Converting a String to an int

Integer.parseInt("3")

A class that ships with Java.

A method in the Integer class that knows how to "parse" a String into the int it represents.

Takes a String.

Read this for loop declaration as "repeat for each element in the 'locationCells' array: take the next element in the array and assign it to the int variable 'cell'."

The colon (:) means "in", so the whole thing means "for each int value IN locationCells..."

② The for loop

```
for (int cell : locationCells) { }
```

Declare a variable that will hold one element from the array. Each time through the loop, this variable (in this case an int variable named "cell"), will hold a different element from the array, until there are no more elements (or the code does a "break"... see #4 below).

The array to iterate over in the loop. Each time through the loop, the next element in the array will be assigned to the variable "cell". (More on this at the end of this chapter.)

③ The post-increment operator

numOfHits++

The ++ means add 1 to whatever's there (in other words, increment by 1).

numOfHits++ is the same (in this case) as saying `numOfHits = numOfHits + 1`, except slightly more efficient.

④ break statement

break;

Gets you out of a loop. Immediately. Right here.
No iteration, no boolean test, just get out now!

실습과제 5-1

여기에 숨어있는 사소한 버그가 있다.
컴파일되고 실행은 되지만 가끔은 ...

지금 당장은 넘어가도 되지만, 조금 후에는
문제에 직면하게 될 것이다.

```
public class SimpleDotComTestDrive {  
  
    public static void main (String[] args) {  
        SimpleDotCom dot = new SimpleDotCom();  
        int[] locations = {2,3,4};  
        dot.setLocationCells(locations);  
        String userGuess = "2";  
        String result = dot.checkYourself(userGuess);  
    }  
}
```

```
public class SimpleDotCom {  
  
    int[] locationCells;  
    int numOfHits = 0;  
  
    public void setLocationCells(int[] locs) {  
        locationCells = locs;  
    }  
  
    public String checkYourself(String stringGuess) {  
        int guess = Integer.parseInt(stringGuess);  
        String result = "miss";  
        for (int cell : locationCells) {  
            if (guess == cell) {  
                result = "hit";  
                numOfHits++;  
                break;  
            }  
        } // out of the loop  
  
        if (numOfHits ==  
            locationCells.length) {  
            result = "kill";  
        }  
        System.out.println(result);  
        return result;  
    } // close method  
} // close class
```

SimpleDotComGame 클래스 준비 코드

SimpleDotComGame에서는 다음과 같은 것을 수행해야 한다:

1. **SimpleDotCom** 객체를 만든다.
2. 위치(가상 셀 7개 중에서 연속된 셀 3개)를 만든다.
3. 사용자에게 위치를 물어본다.
4. 사용자가 추측한 위치를 확인한다.
5. 닷컴을 가라앉힐 때까지 같은 작업을 반복한다.
6. 몇 번의 추측 끝에 닷컴을 가라앉혔는지 알려준다.

Everything happens in main()

```
public static void main (String [] args)
```

```
    DECLARE an int variable to hold the number of user guesses, named numOfGuesses, set it to 0.
```

```
    MAKE a new SimpleDotCom instance
```

```
    COMPUTE a random number between 0 and 4 that will be the starting location cell position
```

```
    MAKE an int array with 3 ints using the randomly-
```

```
generated number, that number incremented by 1, and that number incremented by 2 (example: 3,4,5)
```

```
    INVOKE the setLocationCells() method on the SimpleDotCom instance
```

```
    DECLARE a boolean variable representing the state of the game, named isAlive. SET it to true
```

```
    WHILE the dot com is still alive (isAlive == true) :
```

```
        GET user input from the command line
```

```
        // CHECK the user guess
```

```
        INVOKE the checkYourself() method on the SimpleDotCom instance
```

```
        INCREMENT numOfGuesses variable
```

```
        // CHECK for dot com death
```

```
        IF result is "kill"
```

```
            SET isAlive to false (which means we won't enter the loop again)
```

```
            PRINT the number of user guesses
```

```
        END IF
```

```
    END WHILE
```

```
END METHOD
```




Howdy from Ghost Town

게임의 main() 메소드

DECLARE a variable to hold user guess count, set it to 0

MAKE a SimpleDotCom object

COMPUTE a random number between 0 and 4

MAKE an int array with the 3 cell locations, and

INVOKE setLocationCells on the dot com object

DECLARE a boolean isAlive

WHILE the dot com is still alive

GET user input

// CHECK it

INVOKE checkYourself() on dot com

INCREMENT numOfGuesses

IF result is "kill"

SET gameAlive to false

PRINT the number of user guesses

```
public static void main(String[] args) {
```

```
    int numOfGuesses = 0;
```

```
    GameHelper helper = new GameHelper();
```

```
    SimpleDotCom theDotCom = new SimpleDotCom();
```

```
    int randomNum = (int) (Math.random() * 5);
```

```
    int[] locations = {randomNum, randomNum+1, randomNum+2};
```

```
    theDotCom.setLocationCells(locations);
```

```
    boolean isAlive = true;
```

```
    while(isAlive == true) {
```

```
        String guess = helper.getUserInput("enter a number");
```

```
        String result = theDotCom.checkYourself(guess);
```

```
        numOfGuesses++;
```

```
        if (result.equals("kill"))
```

```
            isAlive = false;
```

```
            System.out.println("You took " + numOfGuesses + " guesses");
```

```
        } // close if
```

```
    } // close while
```

```
} // close main
```

make a variable to track how many guesses the user makes

this is a special class we wrote that has the method for getting user input. for now, pretend it's part of Java

make the dot com object

make a random number for the first cell, and use it to make the cell locations array

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

random()과 getUserInput()

1. 난수를 만든다.

This is a 'cast', and it forces the thing immediately after it to become the type of the cast (i.e. the type in the parens). Math.random returns a double, so we have to cast it to be an int (we want a nice whole number between 0 and 4). In this case, the cast lops off the fractional part of the double.

The Math.random method returns a number from zero to just less than one. So this formula (with the cast), returns a number from 0 to 4. (i.e. 0 - 4.999..., cast to an int)

```
int randomNum = (int) (Math.random() * 5)
```

↑
We declare an int variable to hold the random number we get back.

↑
A class that comes with Java.

↑
A method of the Math class.

2. **GameHelper** 클래스를 사용하여 사용자가 입력한 내용을 받아온다.

String guess = helper.getUserInput("enter a number");

An instance we made earlier, of a class that we built to help with the game. It's called GameHelper and you haven't seen it yet (you will).

This method takes a String argument that it uses to prompt the user at the command-line. Whatever you pass in here gets displayed in the terminal just before the method starts looking for user input.

We declare a String variable to hold the user input String we get back ("3", "5", etc.).

A method of the GameHelper class that asks the user for command-line input, reads it in after the user hits RETURN, and gives back the result as a String.

마지막 클래스: GameHelper

이 클래스에 `getUserInput()` 메소드가 들어있다.

READY-BAKE CODE

```
import java.io.*;

public class GameHelper {
    public String getUserInput(String prompt) {
        String inputLine = null;
        System.out.print(prompt + " ");
        try {
            BufferedReader is = new BufferedReader(
                new InputStreamReader(System.in));
            inputLine = is.readLine();
            if (inputLine.length() == 0 ) return null;
        } catch (IOException e) {
            System.out.println("IOException: " + e);
        }
        return inputLine;
    }
}
```



실습과제 5-2

지금까지 구현한 **SimpleDotComGame**을 실행시켜 보자.

A complete game interaction

```
"C:\Program Files\Java\jdk1.8.0_101\bin>java SimpleDotComGame
enter a number 1
miss
enter a number 2
miss
enter a number 3
hit
enter a number 4
hit
enter a number 5
kill
You took 5 guesses

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_101\bin>java SimpleDotComGame
enter a number 1
hit
enter a number 1
hit
enter a number 1
kill
You took 3 guesses

Process finished with exit code 0
```

What's this? A bug?

Gasp!

Here's what happens when we enter 1,1,1.

A different game interaction
(yikes)

이유를 추측해 보시오.



More about for loops

Regular (non-enhanced) for loops

`for(int i = 0; i < 100; i++) { }`

The diagram shows the code `for(int i = 0; i < 100; i++) { }` with several handwritten annotations. Brackets under the code identify three parts: `int i = 0` is labeled 'initialization', `i < 100` is labeled 'boolean test', and `i++` is labeled 'iteration expression'. An arrow points from the text 'post-increment operator' to the `i++` part. Another arrow points from the text 'the code to repeat goes here (the body)' to the opening curly brace `{`.

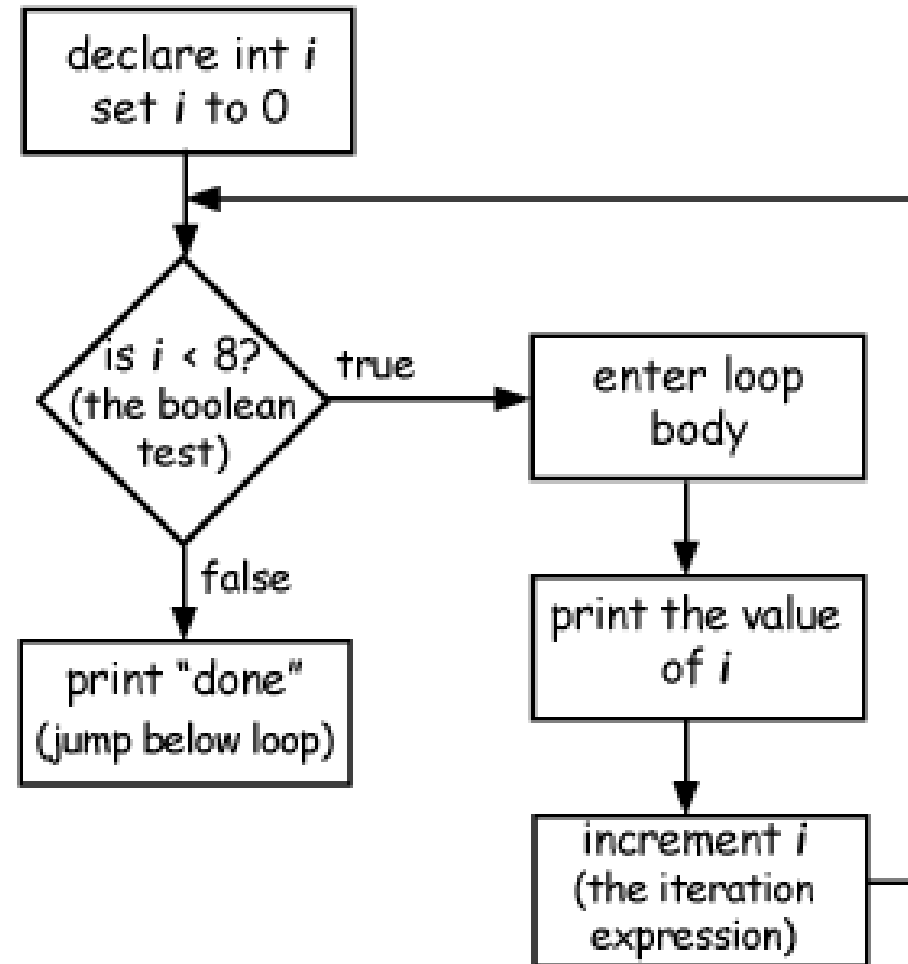
What it means in plain English: "Repeat 100 times."

How the compiler sees it:

- create a variable `i` and set it to 0.
- repeat while `i` is less than 100.
- at the end of each loop iteration, add 1 to `i`

루프 여행

```
for (int i = 0; i < 8; i++) {  
    System.out.println(i);  
}  
System.out.println("done");
```



for와 while의 차이점

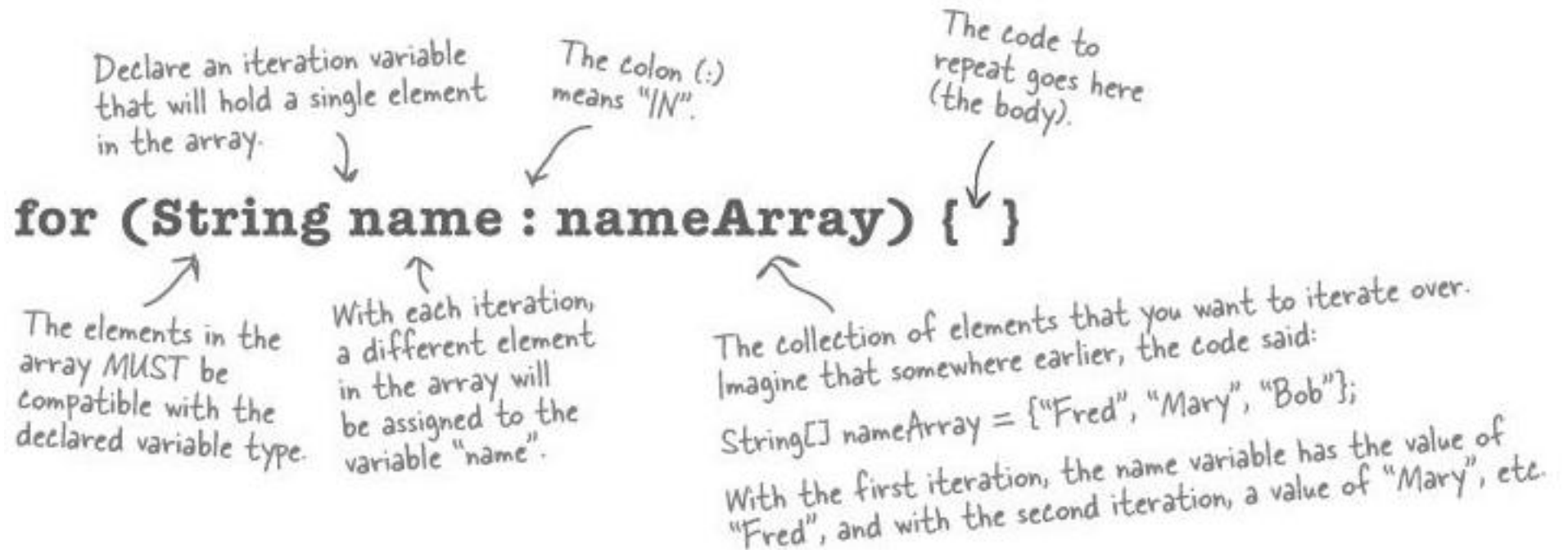
while 루프에는 부울 테스트만 있다.

```
int i = 0; ← we have to declare and  
while (i < 8) { initialize the counter  
    System.out.println(i);  
    i++; ← we have to increment  
        the counter  
}  
System.out.println("done");
```

```
File Edit Window Help Repeat  
%java Test  
0  
1  
2  
3  
4  
5  
6  
7  
done
```

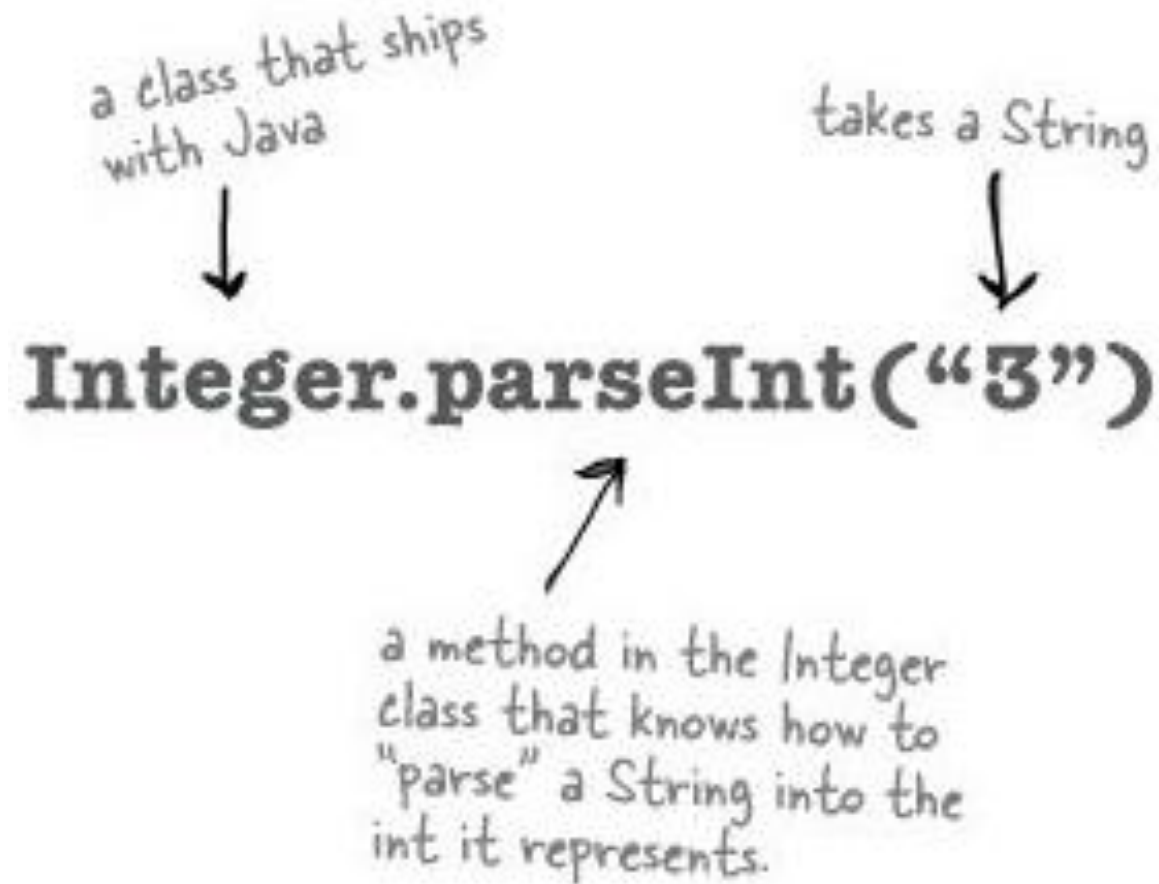
향상된 for 루프

Java 5.0 (Tiger)부터 향상된 **for** 루프라는 새로운 유형의 **for** 루프가 추가되었다.

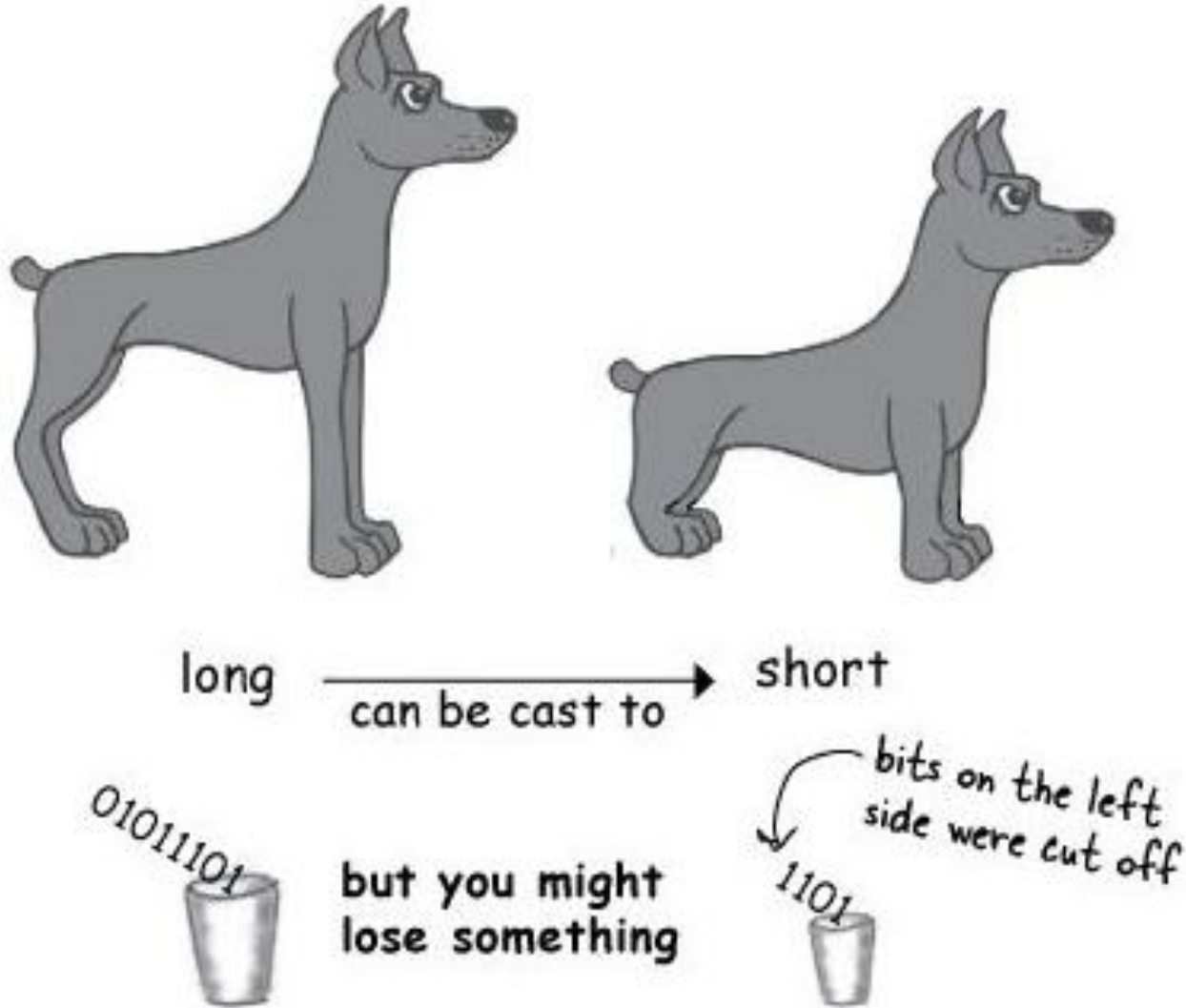


What it means in plain English: "For each element in nameArray, assign the element to the 'name' variable, and run the body of the loop."

Converting a STRING to an INT



Casting primitives



실습과제 5-3 Code Magnets

x++;

if (x == 1) {

System.out.println(x + " " + y);

class MultiFor {

for(int y = 4; y > 2; y--) {

for(int x = 0; x < 4; x++) {

public static void main(String [] args) {

File Edit Window Help Raid

% java MultiFor

0 4

0 3

1 4

1 3

3 4

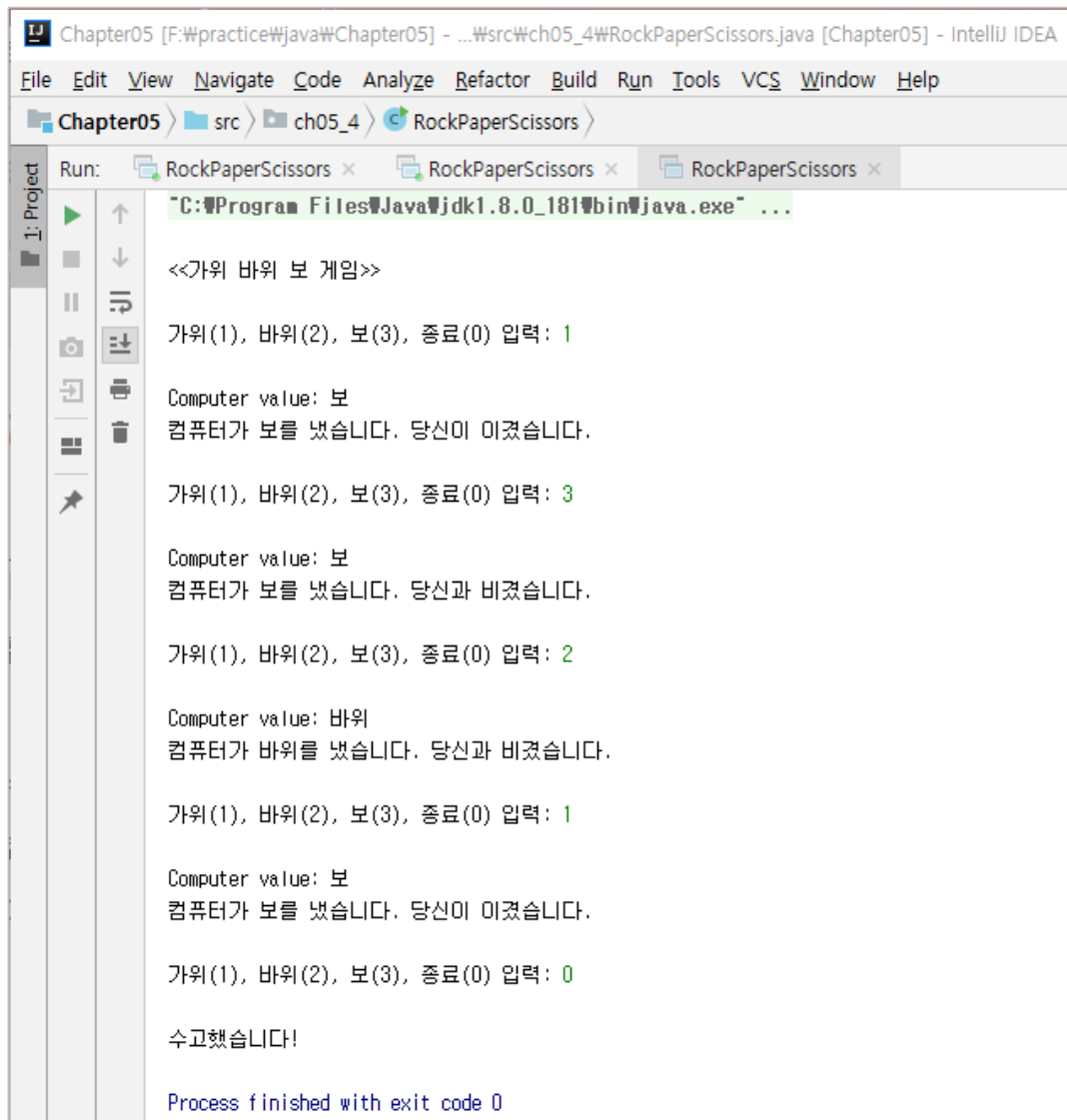
3 3

실습과제 5-4 가위바위보 게임

구현조건:

1. "가위/바위/보" 게임을 난수 발생기 클래스 Random을 이용하여 구현한다.
2. 사용자와 컴퓨터가 대결하는 것으로 하고 컴퓨터는 0부터 2까지의 난수를 발생한다.
3. 1은 가위, 2는 바위, 3은 보로 간주하고 사용자가 입력한 수를 비교하여 승부를 결정한다.
4. 사용자 입력은 Scanner 클래스를 이용하여 처리한다.
5. 적어도 2개의 클래스를 이용해야 한다.
6. 사용자가 종료(0)할 때까지 게임을 반복한다.

Sample run:



```
Chapter05 [F:\practice\java\Chapter05] - ...src\ch05_4\RockPaperScissors.java [Chapter05] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Chapter05 > src > ch05_4 > RockPaperScissors >
Run: RockPaperScissors x RockPaperScissors x RockPaperScissors x
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...

<<가위 바위 보 게임>>

가위(1), 바위(2), 보(3), 종료(0) 입력: 1

Computer value: 보
컴퓨터가 보를 냈습니다. 당신이 이겼습니다.

가위(1), 바위(2), 보(3), 종료(0) 입력: 3

Computer value: 보
컴퓨터가 보를 냈습니다. 당신과 비겼습니다.

가위(1), 바위(2), 보(3), 종료(0) 입력: 2

Computer value: 바위
컴퓨터가 바위를 냈습니다. 당신과 비겼습니다.

가위(1), 바위(2), 보(3), 종료(0) 입력: 1

Computer value: 보
컴퓨터가 보를 냈습니다. 당신이 이겼습니다.

가위(1), 바위(2), 보(3), 종료(0) 입력: 0

수고했습니다!

Process finished with exit code 0
```

