5장. 메서드를 더 강력하게

준비 코드, 테스트 코드, 실제 코드

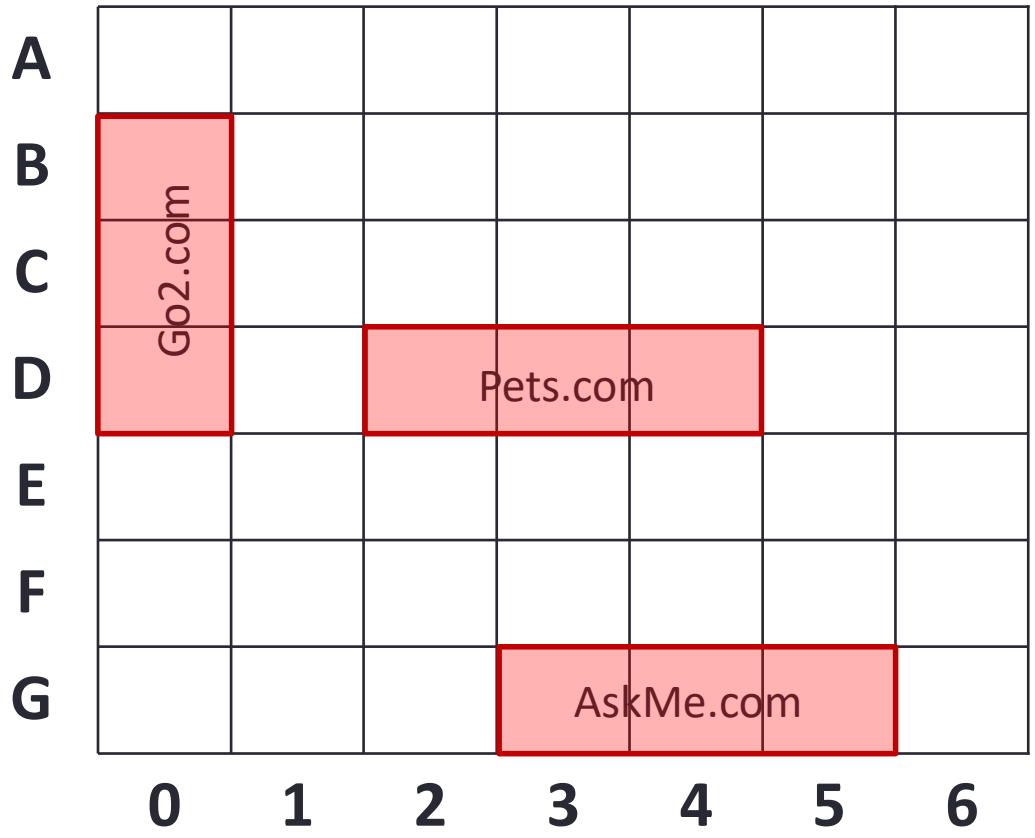
연산자 및 순환문의 이해

유형 변경 방법

간단한 닷컴 게임 제작

더 강력한 메서드를 만들기 위해...

닷 컴(.COM) 가라앉히기 게임




```
$ java StartupBust
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 Go2.com   :(
```

p.138

고수준 설계

1. 사용자가 게임을 시작시킵니다

   A. 닷컴을 세 개 만듭니다.

   B. 세 닷컴을 가상 그리드에 배치합니다.

2. 게임이 시작됩니다.

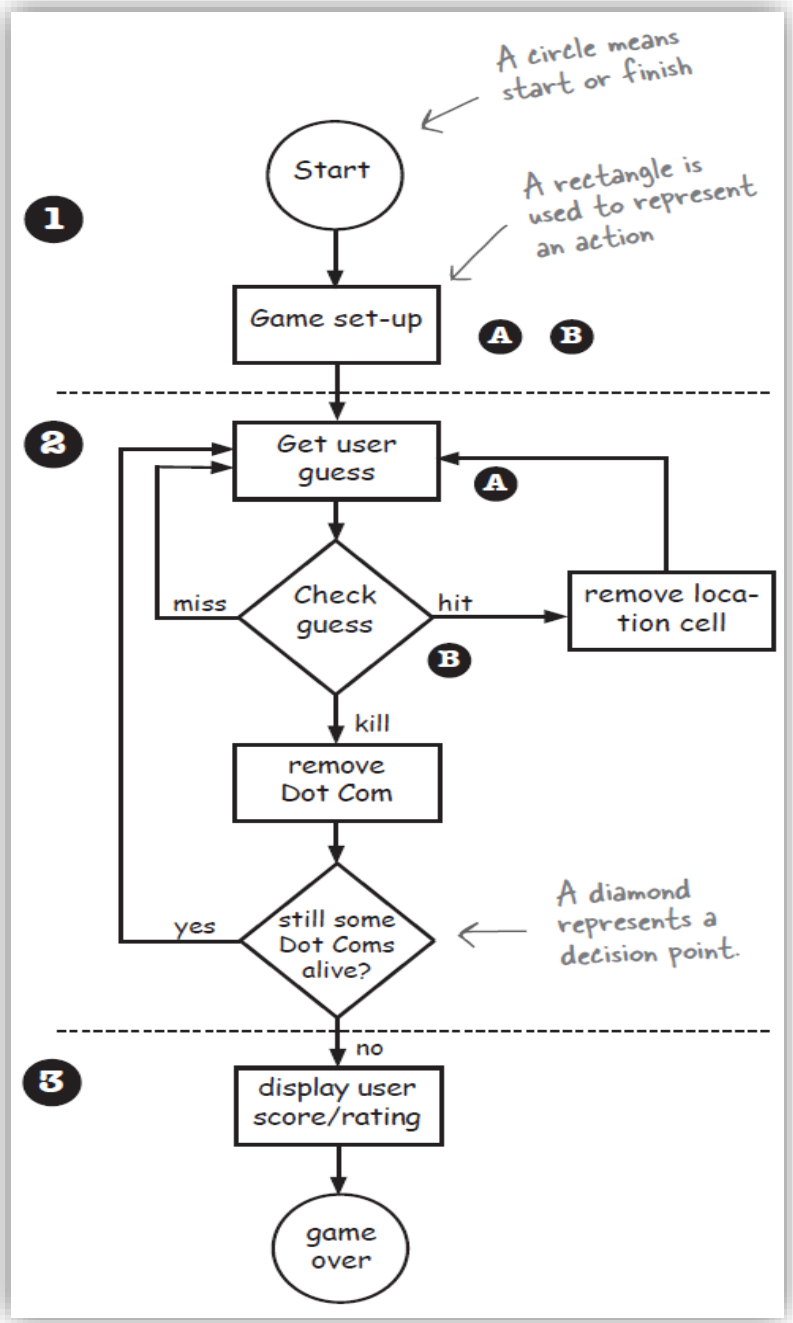
   닷컴이 모두 없어질 때까지 다음 과정 반복

   A. 위치를 입력 받는 프롬프트 출력

   B. 위치가 맞는지 확인하고 적절한 행동(miss, hit, kill)을 취함

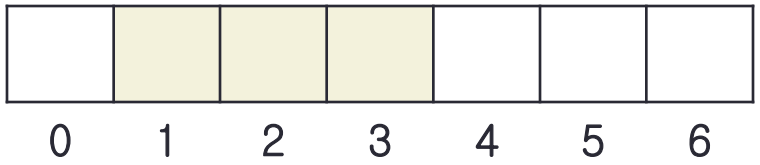
3. 게임 종료

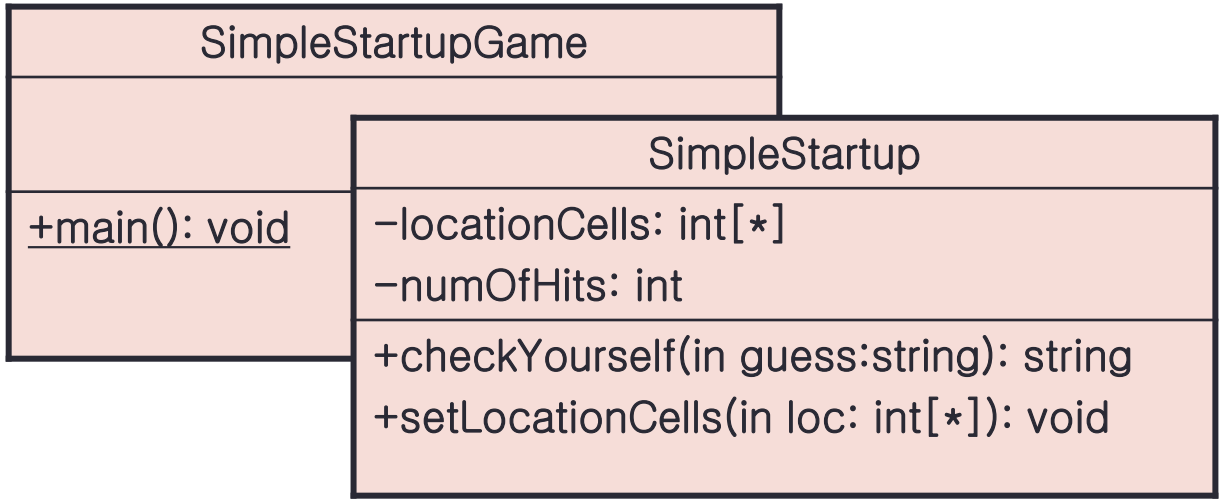
   찍은 회수를 바탕으로 사용자의 등급을 매깁니다.



p.139

간단한 닷컴 게임






```
%java SimpleStartupGame
enter a number   2
hit
enter a number   3
hit
enter a number   4
miss
enter a number   1
kill
You took 4 guesses
```

p.140

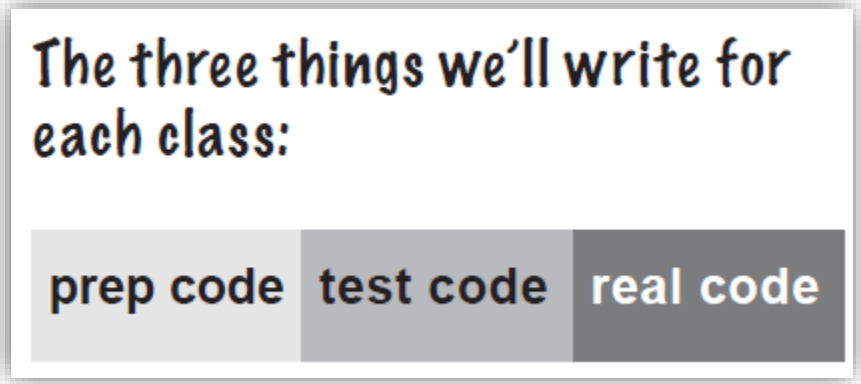
클래스 개발

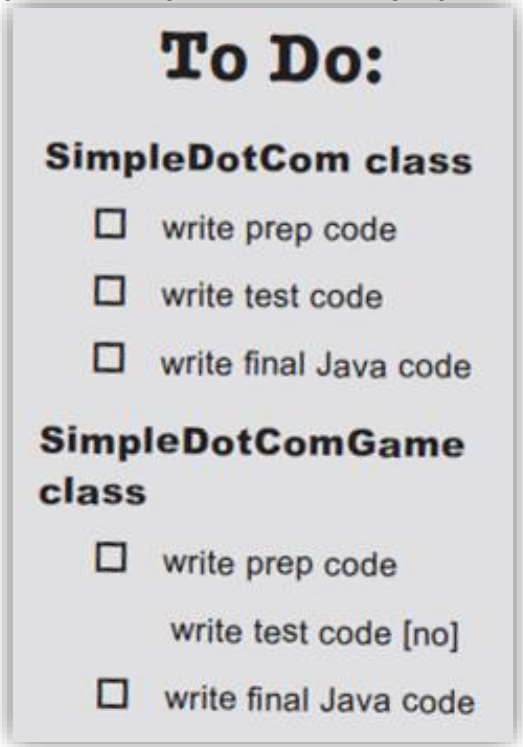
- 클래스에서 어떤 것을 해야 하는지 파악
- 인스턴스 변수/메서드 목록 작성 (객체의 상태와 행위 관점에서 생각)
- 메서드를 만들기 위한 **준비 코드** 만들기
- 메서드에 대한 **테스트 코드** 만들기
- **클래스 구현**
- 메서드 테스트
- 디버깅(debugging)/다시 구현

p.141

세 가지 종류의 코드

- **준비 코드** (자료 구조/알고리즘 설계)
  - 문법보다는 논리를 중점적으로 살펴보기 위해 유사코드 형태로 표현
- **테스트 코드**
  - 실제 코드를 테스트하고 작업이 제대로 처리되는지 확인하기 위한 클래스 또는 메서드
- **실제 코드**
  - 클래스를 실제로 구현한 코드. 실제로 사용할 자바 코드





p.141

To Do

- SimpleStartup 클래스
  - 준비 코드 만들기
  - 테스트 코드 만들기
  - 최종 자바 코드 만들기
- SimpleStartupGame 클래스
  - 준비 코드 만들기
  - ~~테스트 코드 만들기~~
  - 최종 자바 코드 만들기

p.141

p.142

SimpleStartup 클래스 준비 코드

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

참고: int[] locationCells – 세 개의 셀 위치 정보

메서드: void setLocationCells(int[] cellLocations)

셀 위치를 int 배열 매개변수로 **받아옴**

셀 위치 매개변수를 셀 위치 인스턴스 변수에 **대입**

메서드 끝

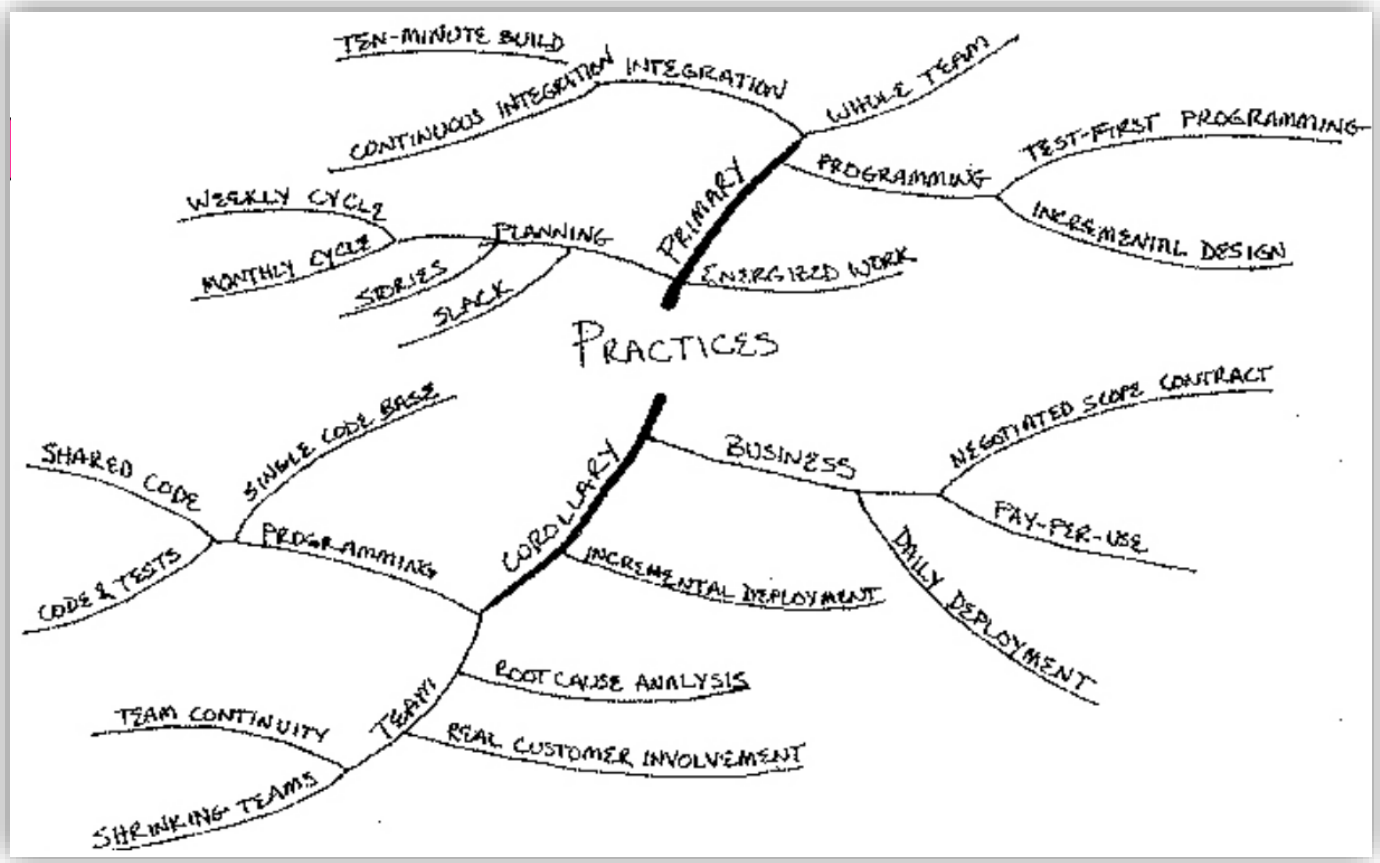
locationCells라는 int 배열 **선언**
numOfHits라는 int 배열 **선언**, 값을 0으로 **설정**

추측한 위치를 String으로 받아들이고 그 값을 확인하고 hit, miss, kill 중 하나를 나타내는 결과를 리턴하는 **checkYourself()**라는 **메서드를 선언**
int 배열을 받아들이는 **setLocationCells()**라는 세터 **메서드를 선언**

메서드: String checkYourself(String userGuess)
사용자가 추측한 위치를 String 매개변수로 **받아옴**
사용자가 추측한 위치를 int로 **변환**
int 배열의 각 셀에 대해 다음 작업 **반복**
  // 사용자가 추측한 위치를 셀과 비교
  만약 사용자가 추측한 것이 맞으면
    맞춘 개수 **증가**
    // 마지막 위치인지 확인
    만약 맞춘 회수가 3이면 kill을 리턴
    그렇지 않으면 hit 리턴
    만약 부분 끝
  그렇지 않으면 miss 리턴
  만약 부분 끝
반복 부분 끝
메서드 끝

참고. XP(eXtreme Programming) 1/2

- 초기 애자일 개발 방법론(agile methodology)
- 보다 좋은 품질의 SW를 보다 생산성 있게 만들기 위해 사람들을 조직
- 실천법(practices)



참고. XP(eXtreme Programming) 2/2

- 조금씩 자주 동작하는 코드를 배포(또는 발표)한다.
- 사이클을 반복해서 돌리면서 개발한다.
- 스펙에 없는 것은 절대 집어넣지 않는다.
- *테스트 코드를 먼저 만든다. (test-first programming)*
- 야근은 하지 않는다. 항상 정규 일과 시간에만 작업한다.
- 기회가 생기는 족족 언제 어디서든 코드를 개선한다.
- 모든 테스트를 통과하기 전에는 어떤 것도 배포(또는 발표)하지 않는다.
- 조금씩 발표하는 것을 기반으로 하여 현실적인 작업 계획을 만든다.
- 모든 일을 단순하게 처리한다.
- 두 명씩 팀을 편성하고 모든 사람이 대부분의 코드를 알 수 있도록 돌아가면서 작업한다.

SimpleStartup 테스트 코드

메서드: String checkYourself(String userGuess)
사용자가 추측한 위치를 String 매개변수로 **받아옴**
사용자가 추측한 위치를 int로 **변환**
int 배열의 각 셀에 대해 다음 작업 **반복**
  // 사용자가 추측한 위치를 셀과 비교
  만약 사용자가 추측한 것이 맞으면
    맞춘 개수 **증가**
    // 마지막 위치인지 확인
    만약 맞춘 회수가 3이면 kill을 리턴
    그렇지 않으면 hit 리턴
    만약 부분 끝
  그렇지 않으면 miss 리턴
  만약 부분 끝
반복 부분 끝
메서드 끝

테스트 사항

1. SimpleStartup 객체의 인스턴스를 만듭니다.
2. 위치를 대입합니다. ([2,3,4]와 같이 세 값이 들어있는 배열)
3. 사용자가 추측한 위치를 나타내는 String을 만듭니다.
4. 3단계에서 만들어낸 String을 전달하면서 **checkYourself() 메서드를 호출**합니다.
5. 결과를 출력하여 옳은 결과가 나왔는지 확인합니다. (결과가 맞으면 “passed”, 틀리면 “failed”)

p.144

실제 코드를 만들기 전에 테스트 코드를 먼저 만듭니다!!!
테스트 기반 개발(Test-Driven Development, TDD)

- 아직 존재하지도 않는 것을 어떻게 테스트하나요?

  - 테스트를 하는 것은 아닙니다. 아직 테스트할 대상은 없는 상태이기 때문에 뼈대만 있는 코드만이라도 미리 만들어주지 않으면 컴파일도 할 수 없습니다. 물론 이런 식으로 컴파일을 하더라도 실제 코드를 만들기 전까지는 정말로 테스트를 할 수 있는 건 아닙니다.

- 그러면 왜 실제 코드를 먼저 만들지 않고 테스트 코드를 먼저 만드나요?

  - 테스트 코드에 대해 생각해보기 위한 것입니다. 메서드의 기능과 역할에 대해 더 확실히 이해할 수 있으니까요. 그리고 코드를 완성했을 때 바로 테스트할 수 있다는 장점도 있습니다.
  - 가장 이상적인 방법은 테스트 코드를 조금씩 복잡하게 고쳐가면서 그 테스트를 통과할 수 있는 실제 코드를 만들어나가는 것입니다.

SimpleStartup 클래스용 테스트 코드

p.145

```
public class SimpleStartupTestDrive {          //  SimpleStartupTestDrive
   public static void main (String[] args) {
      SimpleStartup dot = new SimpleStartup();

      int[] locations = {2,3,4};
      dot.setLocationCells(locations);

      String userGuess = "2";
      String result = dot.checkYourself(userGuess);
      String testResult = "failed";
      if (result.equals("hit")) {      // "hit".equals(result)가 더 바람직함.
         testResult = "passed";
      }
      System.out.println(testResult);
   }
}
```

참고: JUnit

checkYourself() 메서드의 실제 코드

```
public String checkYourself(String stringGuess) {  // int guess
   int guess = Integer.parseInt(stringGuess);  // 없음
   String result = "miss";

   for (int cell : locationCells) {  // enhanced for-loop
      if (guess == cell) {
         result = "hit";
         numOfHits++;
         break;
      }
   }

   if (numOfHits == locationCells.length) {
      result = "kill";
   }

   System.out.println(result);
   return result;
}
```

코드에 아무런 문제가 없나요?

- syntax error
- semantic error

https://blogs.oracle.com/corejavatechtips/using-enhanced-for-loops-with-your-classes

checkYourself() 메서드

- Integer.parseInt("3")
  - String을 int로 변환

```
EnhancedForStatement:
    for ( {VariableModifier} LocalVariableType VariableDeclaratorId
        : Expression )
      Statement
```

- for(int cell : locationCells) { }
  - Enhanced for statement (cf. basic for statement)
  - Java 5에서 새로 도입된 문법

```
BasicForStatement:
    for ( [ForInit] ; [Expression] ; [ForUpdate] ) Statement
```

- numOfHits++
  - 후 증가 연산자(post-increment operator)

- break
  - break 명령문

- Integer.parseInt()에 숫자가 들어있지 않은 문자열을 전달하면 어떻게 되나요? “three” 같은 것도 인식이 되나요?
  - 이 메서드는 숫자를 나타내는 아스키값으로 구성된 String 객체에 대해서만 작동합니다.
  - “two”, “오~~” 같은 것을 파싱하려고 하면 맛이 갑니다.

- for 순환문이 전에 봤던 거랑 다른데, for 순환문이 두 종류 있나요?
  - 자바 5.0(타이거)부터 배열(또는 컬렉션)의 원소들에 대해서 반복작업을 하고 싶을 때 쓸 수 있는 **향상된 for** 순환문이 등장했습니다. 배열의 모든 원소에 대해 반복작업을 하려고 한다면 **기존 for 순환문(index 사용)**보다는 새로 도입된 향상된 for 순환문이 더 좋겠죠?
  - enhanced for loop: 색인 필요 없이 값만 필요할 때 유용

SimpleStartup.java (or SimpleStartup.java)

```
6   package cse.oop2.ch05.simpledotcom;
7
8   /** p ...5 lines */
13  public class SimpleDotCom {
14
15      int[] locationCells;
16      int numOfHits = 0;
17
18      public void setLocationCells(int[] locs) {
19          locationCells = locs;
20      }
```

```
22      public String checkYourself(String stringGuess) {
23          int guess = Integer.parseInt(stringGuess);
24          String result = "miss";
25          for (int cell : locationCells) {
26              if (guess == cell) {
27                  result = "hit";
28                  numOfHits++;
29                  break;
30              }
31          }
32          if (numOfHits == locationCells.length) {
33              result = "kill";
34          }
35          System.out.println(result);
36          return result;
37      }
38  }
```

p.148

SimpleStartupTestDrive.java

cf. SimpleStartupTestDrive.java

```
6   package cse.oop2.ch05.simpledotcom;
7
8   /** p ...5 lines */
13  public class SimpleDotComTestDrive {
14
15      public static void main(String[] args) {
16          SimpleDotCom dot = new SimpleDotCom();
17
18          int[] locations = {2, 3, 4};
19          dot.setLocationCells(locations);
20
21          String userGuess = "2";
22          String result = dot.checkYourself(userGuess);
23          String testResult = "failed";
24          if (result.equals("hit")) {
25              testResult = "passed";
26          }
27          System.out.println(testResult);
28      }
29
30  }
```

```
--- exec-maven-plugin:
hit
passed
```

p.148

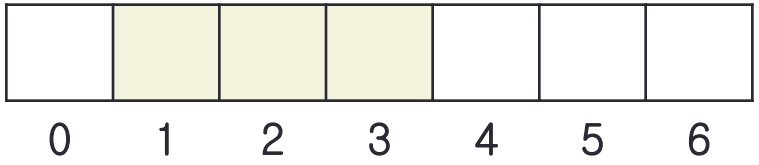
문제점

- **실행 결과를 예상해보고 직접 실행해서 확인해봅시다. 테스트 클래스와 SimpleStartup 클래스만으로는 게임을 할 수가 없네요. TT** → 항상 동일한 결과를 보임!!!

- **이제 무엇을 해야 할까요?**

SimpleStartupGame 클래스에서 할 일

1. SimpleStartup 객체 만들기
2. 위치 만들기 (일곱 개의 셀 중에서 연속된 세 개의 셀)
3. 위치 물어보기
4. 추측한 위치가 맞는지 확인
5. 닷컴이 죽을 때까지 같은 작업 반복
6. 추측 회수 출력



141 페이지에 나와있는 “연필을 깎으며” 섹션을 보고 직접 준비 코드를 만들어봅시다.

```
%java SimpleStartupGame
enter a number   2
hit
enter a number   3
hit
enter a number   4
miss
enter a number   1
kill
You took 4 guesses
```

SimpleStartupGame 클래스 준비 코드

메서드 public static void main (String[] args)
  사용자가 추측한 회수를 저장하기 위한 numOfGuesses라는 int 변수 선언
  SimpleStartup 인스턴스 만들기
  0 이상 4 이하의 난수 계산 (셀 위치 시작점, *난수, 난수+1, 난수+2 사용*)
  세 개의 int가 들어있는 배열 만들기
  SimpleStartup 인스턴스의 setLocationCells() 메서드 호출
  게임의 상태를 나타내는 isAlive라는 부울 변수 선언, true로 설정
  닷컴이 살아있는 동안(while (isAlive == true))
    명령행을 통해 위치 받기
    // 사용자가 추측한 위치 확인
    SimpleStartup 인스턴스의 chechYourself() 메서드 호출
    numOfGuesses 변수 증가
    // 닷컴이 죽었는지 확인
    만약 결과가 "kill"이면
      isAlive를 false로 설정
      (순환문 중단)
      사용자가 추측한 회수 출력
    만약 부분 끝
  while 부분 끝
메서드 끝

절차를 따라서 하면 프로그램 실행이 잘 될 것 같나요?

→ 이 모든 것을 main() 메서드 안에서 구현?

p.150

핵심 정리

- 자바 프로그램을 만들 때는 우선 고수준 설계부터 시작합니다.
- 새로운 클래스를 만들 때는 일반적으로 다음과 같은 세 가지를 만들어야 합니다.
  - **준비 코드**
  - **테스트 코드**
  - **실제 코드 (자바 코드)**
- 준비 코드에서는 어떻게 해야 할지 보다는 무엇을 해야 할지를 기술해야 합니다. 구현은 나중에 하면 됩니다.
- 테스트 코드를 설계할 때는 준비 코드를 활용하면 좋습니다.
- 메서드를 구현하기 전에 테스트 코드를 만들어야 합니다.

핵심 정리

- 순환문 코드 반복 회수를 미리 알 수 있는 경우에는 while보다는 for를 쓰는 것이 좋습니다.
- 변수에 1을 더할 때는 선/후 증가 연산자를 쓰면 됩니다. (++x; x++;)
- 변수에서 1을 뺄 때는 선/후 감소 연산자를 쓰면 됩니다. (--x; x--;)
- String을 int로 바꿀 때는 Integer.parseInt()를 쓰면 됩니다.
- Integer.parseInt()는 숫자를 나타내는 String에 대해서만 사용할 수 있습니다.
- 순환문을 중간에 무조건 빠져나올 때는 break문을 사용하면 됩니다. (continue문은?)

GameHelper를 이용한 SimpleStartupGame

- 필요한 소스 코드 (NetBeans IDE 프로젝트 내 패키지 및 파일 목록)
  - cse.oop2.ch05.**game**.GameHelper.java: 사용자로부터 콘솔 입력 받아 처리
  - cse.oop2.ch05.**game**.SimpleStartupGame.java

  - cse.oop2.ch05.**simpleStartup**.SimpleStartup.java (앞에서 구현한 클래스 재사용)

GameHelper.java

```
1   package cse.oop2.ch05.game;
2
3   // Ctrl-Shift-I Fix imports
4   import java.io.BufferedReader;
5   import java.io.IOException;
6   import java.io.InputStreamReader;
7
8   /** p ...5 lines */
13  public class GameHelper {
14
15      public String getUserInput(String prompt) {
16          String inputLine = null;
17          System.out.print(prompt + ":  ");
```

```
18          try {
19              BufferedReader is = new BufferedReader(
20                      new InputStreamReader(System.in));
21              inputLine = is.readLine();
22              if (inputLine.length() == 0) {
23                  // return null;  // NumberFormatException
24                  return "-1";
25              }
26          } catch (IOException e) {
27              System.out.println("IOException: " + e);
28          }
29          // return inputLine;
30          if (inputLine != null && inputLine.matches("\\d")) {
31              return inputLine;
32          } else {
33              return "-1";
34          }
35      }
36  }
```

한 자리 숫자를 나타내는 문자열만 반환
아닌 경우에는 “-1” 문자열을 반환

p.154 (이전 버전)

GameHelper.java (3판)

- package cse.oop2.ch05.game;

```
import java.util.Scanner;

public class GameHelper {
  public int getUserInput(String prompt) {
    System.out.print(prompt + ": ");
    Scanner scanner = new Scanner(System.in);
    return scanner.nextInt();
  }
}
```

```
jshell> Scanner sc = new Scanner(System.in);
sc ==> java.util.Scanner[delimiters=\p{javaWhitesp

jshell> sc.nextInt()
100
$2 ==> 100

jshell> sc.nextInt()
abc
|  Exception java.util.InputMismatchException
|        at Scanner.throwFor (Scanner.java:947)
|        at Scanner.next (Scanner.java:1602)
|        at Scanner.nextInt (Scanner.java:2267)
|        at Scanner.nextInt (Scanner.java:2221)
|        at (#3:1)
```

- (주의) scanner.nextInt() 사용 시 정수 외에 문자열 입력하면 오류 발생!

p.154

SimpleStartupGame.java

- (주의) cse.oop2.ch05.simplestartup.SimpleStartup.java 는 재사용!

```
1   package cse.oop2.ch05.game;
2
3   import cse.oop2.ch05.simplestartup.SimpleStartup;
4
5   /** p ...6 lines */
11  public class SimpleStartupGame {
12
13      public static void main(String[] args) {
14          int numOfGuesses = 0;
15          GameHelper helper = new GameHelper();
16
17          SimpleStartup theStartup = new SimpleStartup();
18          int randomNum = (int) (Math.random() * 5);
19          System.out.format("(randomNumber = %s)%n", randomNum);
```

```
21              int[] locations = {randomNum, randomNum + 1, randomNum + 2};
22              theStartup.setLocationCells(locations);
23              boolean isAlive = true;
24              while (isAlive == true) {
25                  String guess = helper.getUserInput("enter a number");
26                  String result = theStartup.checkYourself(guess);
27                  numOfGuesses++;
28                  if (result.equals("kill")) {
29                      isAlive = false;
30                      System.out.println("You took " + numOfGuesses + " guesses");
31                  }
32              }
33          }
34      }
```

실행 결과

- GameHelper클래스에서 **return inputLine;** 그냥 사용할 경우: NumberFormatException 발생

```
D:₩NetBeansProjects₩java_maven₩target₩classes>java cse.oop2.ch05.game.SimpleDotComGame
(randomNumber = 1)
enter a number:  ab
Exception in thread "main" java.lang.NumberFormatException: For input string: "ab"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.base/java.lang.Integer.parseInt(Integer.java:652)
        at java.base/java.lang.Integer.parseInt(Integer.java:770)
        at cse.oop2.ch05.simpledotcom.SimpleDotCom.checkYourself(SimpleDotCom.java:23)
        at cse.oop2.ch05.game.SimpleDotComGame.main(SimpleDotComGame.java:26)
```

- 수정된 GameHelper 클래스 사용할 경우: 입력에 대한 강건성(robustness)이 있다!

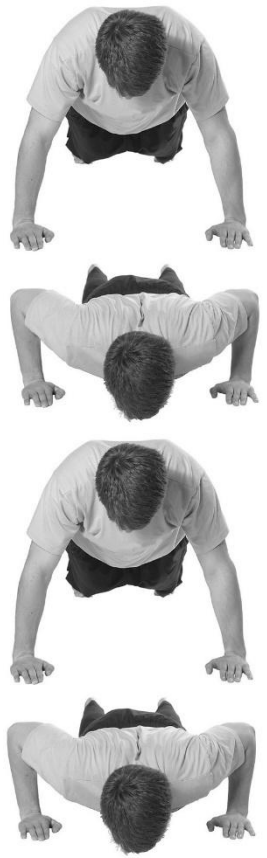
```
D:₩NetBeansProjects₩java_maven₩target₩classes>java cse.oop2.ch05.game.SimpleDotComGame
(randomNumber = 0)
enter a number:  ab
miss
enter a number:  0
hit
enter a number:  1
hit
enter a number:  2
kill
You took 4 guesses

D:₩NetBeansProjects₩java_maven₩target₩classes>
```

for 순환문

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

- **초기화 코드**
  - 변수 선언 및 초기화
  - 주로 카운터(or 색인) 변수를 선언/초기화함
- **부울 테스트 코드**
  - 조건 테스트
  - 부울값(true/false)이 나오는 코드를 써야 함
- **반복 표현식 코드**
  - 매번 순환문을 반복할 때마다 실행할 코드

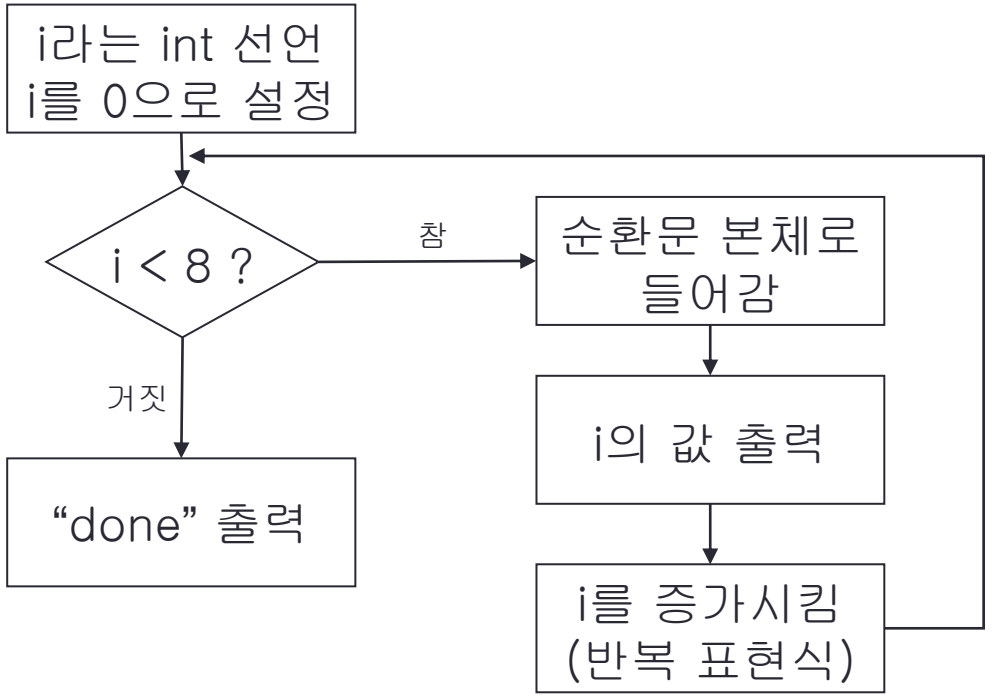


p.156

for 순환문

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

```
$ java Test
0
1
2
3
4
5
6
7
done
```



p.157

for와 while

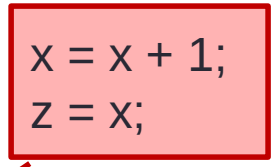
```
for (int i = 0; i < 8; i++) {
  System.out.println(i);
}
System.out.println("done");
```

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

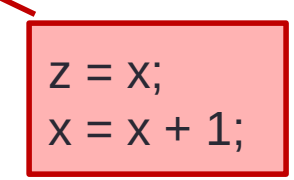
p.157

++와 --: 선/후, 증가/감소 연산자

- x++; ++x;
  - x = x + 1;
  - x의 현재 값에 1을 더한다.
- x--; --x;
  - x = x – 1;
  - x의 현재 값에서 1을 뺀다.



- int x = 0; int z = ++x; 선증가연산자 → z = 1, x = 1
- int x = 0; int z = x++; 후증가연산자 → z = 0, x = 1



p.157

향상된 for문

for (String name : nameArray) { }

In Python,

for name in nameArray:
    pass

- String name
  - 배열에 들어있는 한 원소의 값을 저장할 반복작업용 변수 선언
- 콜론 (:)
  - 영어 "in"에 해당함
- nameArray
  - 반복작업 대상이 될 원소들의 컬렉션 (List, Set, 배열 등)

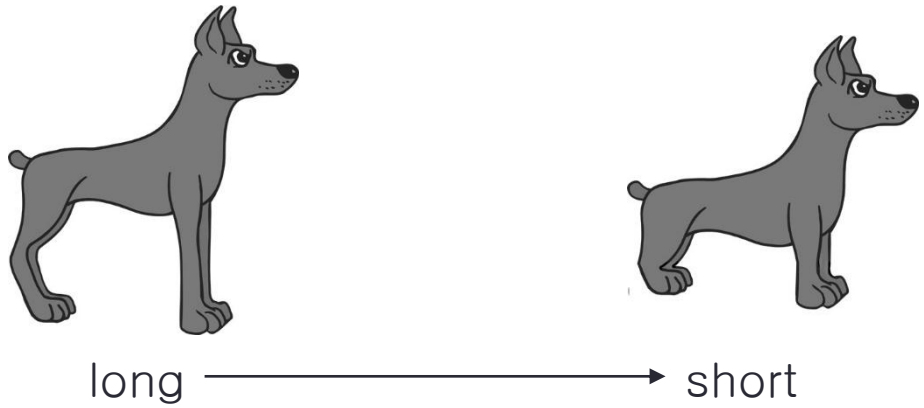
AI: Java에서 enhance for 문에서 사용 가능한 collection 유형은 어떤 것이 있는가?

- Enhanced for or for-each loop

- ```
  for (int i=0; i<nameArray.length; i++) {
      String name = nameArray[i];
      ....
  }
  ```

p.158
  ```

캐스팅(casting)



크기가 큰 자료형에서 작은 자료형으로의 묵시적 형 변환은 데이터 손실 가져올 수 있음.

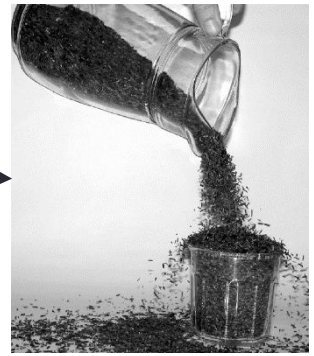
p.159

캐스팅(casting)***

~~long y = 42;~~

~~int x = y;~~

컴파일 오류!



long y = 42;

int x = (int) y; // (명시적) 형 변환, type casting

long y = 40002;

short x = (short) y; // x = -25534

float f = 3.14**f**; // 3.14?

int x = (int) f; // x = 3

```
jshell> float f = 3.14;
|  Error:
|  incompatible types: possible lossy conversion from double to float
|  float f = 3.14;
|            ^--^
```

숙제

- 교재 본문을 한 번 쭉 훑어보세요.
- 중간 중간에 "독자들이 직접 해야 하는 부분"을 꼭 해 보세요.
- 코드를 꼭 직접 입력해서 실행해보세요.
- 마지막 코드에서 어떤 문제가 있었는지, 어떻게 해결할 수 있을지 생각해보세요.
- 연습문제와 퍼즐도 꼭 시간을 내서 직접 해결해보세요.
- https://www.acmicpc.net/ 에 회원 가입해서 문제를 풀고 있나요?

정리

- 클래스 개발 시 만들어야 하는 세 가지
  - **준비 코드(prep code), 테스트 코드, 실제 코드**
- 준비 코드는 유사(pseudo, 의사) 코드 형태로 작성
- 테스트 코드는 ~TestDrive 클래스에 작성
  - JUnit 프레임워크를 사용하면 테스트 코드 작성 쉬우며, 테스트 결과도 쉽게 확인 가능
- 실제 코드는 준비 코드에 따라서 작성
- 향상된 for 문: for(int cell : locationCells)
- 묵시적 형 변환 유의 사항
  - 크기가 큰 자료형에서 작은 자료형으로의 묵시적 형 변환은 값의 손실 있을 수 있어 컴파일 오류 발생
  - 묵시적 형 변환에 따른 컴파일 오류 제거하려면 명시적 형 변환 필요

이 장에서의 주요 내용

- 고수준 설계 (알고리즘)
- 클래스 개발 순서
- 코드 유형: 준비, 테스트, 실제 코드
- 의사 코드 (pseudo code)
- 배열, 리스트 사용법
- 문자열 → 정수 변환
- 콘솔 줄단위 입력 방법: BufferedReader, InputStreamReader, System.in 사용