**자바지향프로그래밍2 실습과제**

**제6주 예외 처리, 리스트 구현**

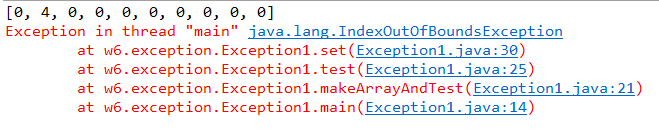
이클립스에서 자신의 이름으로 프로젝트를 만드세요. 그 아래에 w6라는 이름의 패키지를 만들고 (6주차) 그 아래에 아래 문제의 클래스들을 작성하세요.

문제 1

예외처리를 학습하는 문제입니다. w6.exception 패키지에서 강의자료에 있는 예를 실행해 봅니다.

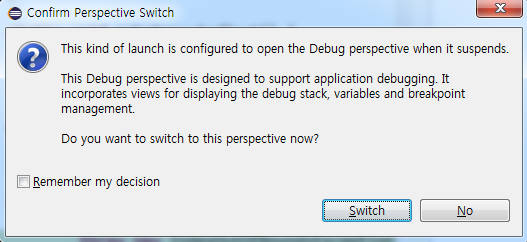
1. [Exception.java](src\Exception.java)를 읽고 실행해 보세요. 이 프로그램은 예외상황에 대처하는 고전적인 방법을 보여줍니다.
2. [Exception1.java](src\Exception1.java)를 읽고 실행해 보세요. 이 프로그램은 예외상황에서 예외를 만들어 던지는 예외처리 방법을 보여줍니다.

이 프로그램을 실행(control-F11)하면, 콘솔 창에 아래와 같은 출력이 나타납니다. 첫 줄은 정상적인 경우의 출력입니다. 그 아래에는 프로그램 실행 중 예외적인 상황이 발생했을 때의 출력입니다. 프로그램 실행 중에 예외가 던져졌는데 그 예외를 아무도 잡지 않아서 예외가 시스템까지 전파되었습니다. 그래서 **시스템(의 예외처리기)이 예외를 잡아 처리하면서 에러 메세지를 표준에러출력(콘솔)으로 내보낸 것입니다.**



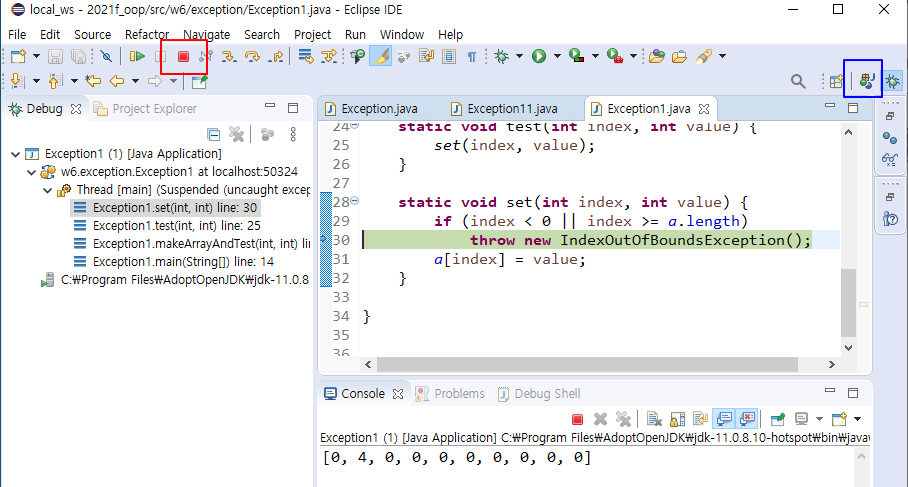
빨간 색 에러메세지 중 첫 줄은 프로그램 실해 중에 java.lang.IndexOutOfBoundsException이 튀어 나왔다는 말입니다. 둘째 줄부터 다섯째 줄까지는 이 예외가 어디에서 만들어져서 어디를 거쳐 나왔는지, 예외가 전파된 과정을 차례로 보여 줍니다. main인 14번 줄에서 예외가 발생했는데, 이 예외는 makeAndTest인 21번 줄에서 만들어져서 14번 줄로 전파된 것이고, 이것은 다시 test인 25번 줄에서 만들어져서 21번 줄로 전파된 것이고, 이것은 다시 set인 30번 줄에서 만들어져 25번 줄로 전파된 것이라는 의미입니다.

이 프로그램을 디버거 모드에서 실행하면 (F11) 아래와 같은 대화창이 나타납니다.



시스템으로 예외가 전파되어 시스템에 의해 프로그램이 강제로 종료되기 전에 이클립스가 **디버거 모드**로 전환하려고 합니다. 실제로 디버거 모드로 전환하기 전에 전환해도 좋은지 사용자에게 물어보는 창입니다. Switch 버튼을 누르면 디버거 모드로 전환됩니다.

아래 디버거 모드 화면을 보면 왼쪽에 디버그 구역이 보이고 오른쪽에 소스코드가 보입니다. 디버그 구역에에 **Call Stack**이 보입니다. Call Stack에 항목이 네 개 있는데 그 중 하나를 선택하면 그 예외가 소스코드의 어디에서 튀어나왔는지 소스코드에 표시됩니다. 아래 그림 예를 보면, set 메소드 실행 중에 IndexOutOfBoundsException이 튀어나왔는데 그게 30번 줄에서 튀어나왔다는 의미입니다.



예외가 어디에서 왜 튀어나오게 됐는지 파악을 한 후에는 위 그림의 빨간 네모 부분을 눌러 프로그램 실행을 종료시키고, 파란 네모 부분을 눌러 디버그 모드에서 **편집 모드**로 돌아옵니다.

1. [Exception2.java](src\Exception2.java)를 읽어보세요. 또, 실행해 보세요.

이 프로그램에서는 예외가 튀어나올 수 있는 코드를 try 블록으로 감싸고, try 블록에서 예외가 튀어나오는 경우, catch 블록에서 그 예외를 잡아 처리하도록 했습니다. 이 때, catch 블록 안에 있는 코드가 바로 예외처리기입니다. catch 절의 파라미터 e에는 try 블록에서 튀어나와 catch 블록에 잡힌 예외 객체가 저장됩니다. 예외 객체는 예외가 발생한 순간의 call stack 정보를 내부에 갖고 있습니다. 예외 객체에게 printStackTrace 메소드를 호출하면 예외 객체가 갖고 있는 (예외가 발생한 순간의) call stack 정보를 화면에 출력합니다. 예외처리기가 작동한 후에는 프로그램 실행이 계속되어 "예외처리기 작동 후 프로그램 계속 실행"이 출력된다는 점에 유의하세요.

1. [Exception3.java](src\Exception3.java)를 읽어보세요. 또, 실행해 보세요.

예외 객체를 생성할 때는 객체 내부에 (어디에서 왜 예외가 발생했나 등) 필요한 정보를 적어 넣을 수 있습니다. 이 프로그램의 39번 줄에서 그 예를 볼 수 있습니다. 예외 객체가 간직하고 있는 이 정보를 읽으려면 예외 객체에게 getMessage 메소드를 호출합니다. 프로그램의 22번 줄에서 그 예를 볼 수 있습니다.

1. [Exception4.java](src\Exception4.java)를 읽어보세요. 또, 실행해 보세요.

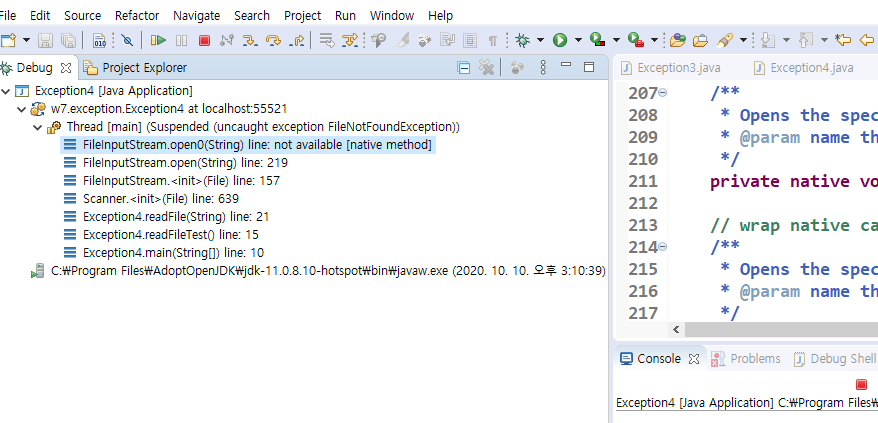
22번 줄에서 Scanner 인스턴스를 하나 생성하는데 Scanner 생성자는 FileNotFoundException을 던지는 수가 있습니다. [API 문서](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html#%3Cinit%3E(java.io.File))에서 확인하세요.

우리의 경우 "data.txt" 라는 이름의 파일을 미리 만들어 놓지 않았으므로 FileNotFoundException이 던져집니다.

[FileNotFoundException](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/FileNotFoundException.html)은 확인예외(Checked Exception)입니다. (Exception 중 RuntimeException(RuntimeException의 서브클래스 포함)을 제외한 모든 예외는 확인예외입니다.)

그래서 readFile, readFileTest, main 메소드 각각의 헤더에 throws FileNotFoundException절을 붙여 주었습니다.

프로그램을 디버거 모드에서 실행하면(F11), 시스템이 FileNotFoundException을 잡아 처리합니다. 그리고 시스템이 내보내는 표준에러출력을 이클립스가 감지하고 디버거 화면으로 넘어갑니다.



Call Stack에 일곱 개의 항목이 보입니다. 아랫부분 세 개는 우리가 작성한 메소드입니다. 나머지 네 개는 자바 표준라이브러리에 있는 코드들입니다. 아래에서부터 네 번째 것은 Scanner 생성자입니다. Scanner 생성자가 실행될 때, call stack의 바로 위에 있는 FileInputStream 생성자가 호출됩니다. FileInputStream 생성자가 실행되는 중에 다시 FileInputStream의 open 메소드가 호출되고, 그것은 또 FileInputStream의 open0 메소드를 호출합니다. Call stack의 맨 위에 있는 이 open0 메소드가 FileNotFoundException 인스턴스를 만들어 던졌습니다.

Call stack의 일곱 개의 항목 중 하나를 선택하면 화면 오른쪽에 소스코드가 나타납니다.

프로그램 실행을 종료하고 자바 편집 모드로 돌아갑니다.

1. [Exception5.java](src\Exception5.java)를 읽어보세요. 또, 실행해 보세요.

프로그램 내에서 예외처리를 해 준 경우입니다.

프로젝트 폴더에 data.txt 파일을 만들어 넣고 실행하면 예외 발생 없이 정상적으로 프로그램이 실행됩니다.

**문제 1에서는 아무것도 제출하지 않습니다.**

문제 2

Collection에 친숙해지기 위해 리스트를 이용하는 문제를 더 풀어 봅니다. w6 패키지에서 아래 프로그램을 작성하세요.

2.1 텍스트 파일인 [words.txt](data/words.txt)에 있는 단어를 모두 읽은 후 순서를 바꿔 출력하는 아래 프로그램을 완성하세요. readFile, backwards 등 두 개의 메소드를 구현하고, main 메소드의 녹색 주석에 맞춰 네모 부분을 완성해야 합니다. 출력은 Figure 1과 같아야 합니다.

A screenshot of a computer program

Description automatically generated



Figure 1

<힌트>

1. readFile 메소드에서는 이래와 같은 일을 해야 합니다.
   1. 리스트를 하나 생성한다. List<String> list = new ArrayList<>()
   2. 주어진 파일에 연결되는 Scanner를 생성한다. Scanner input = new Scanner(new File(fileName));
   3. Scanner로부터 한 단어씩 읽어 차례로 리스트에 추가한다. while (input.hasNext()) { ...
   4. Scanner를 닫는다.
   5. 리스트를 반환한다.
2. backwards 메소드에서는 아래와 같은 일을 해야 합니다.
   1. 새 리스트를 하나 생성한다.
   2. 파라미터로 주어진 list의 맨 끝 원소부터 하나씩 차례로 읽어 새 리스트에 추가한다.

ListIterator<String> it = list.listIterator(list.size());

while (it.hasPrevious()) { ...

[List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)로부터 [ListIterator](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ListIterator.html)를 얻기 위해서는 아래 두 가지 중 하나를 사용합니다.

A close-up of a computer screen

Description automatically generated

두 번째 메소드에 대한 설명은 아래와 같습니다.

A screenshot of a computer code

Description automatically generated

list.listIterator(10)이라고 호출하면 list의 10번 항목 바로 앞(9번과 10번 사이)을 가리키는 ListIterator가 반환됩니다.

이 ListIterator에게 next()를 호출하면 10번 항목이 반환되고, previous()를 호출하면 9번 항목이 반환됩니다.

list.listIterator(index)를 호출할 때, index는 0 이상이고 list.size() 이하이어야 합니다. 가령 list에 0번부터 9번까지 10개의 항목이 있는 경우, index는 0이상 10 이하이어야 합니다. 그러니까 list.listIterator(10)을 호출하면 마지막 항목 다음의 빈 곳을 가리키는 ListIterator가 반환됩니다. 이 놈에게 previous()를 연달아 호출하면 9번부터 8번, 7번의 순서로 항목이 반환됩니다.

* 1. 새 리스트를 반환한다.

1. main 메소드의 헤더 부분에도 throws FileNotFoundException 절을 붙여 주어야 합니다.
2. main 메소드에서 파일의 경로명을 "words.txt"라고 적었으므로 이에 맞춰 데이터 파일을 넣어주어야 합니다.

프로그램에서 파일을 읽으려면 작업디렉토리(working directory)를 기준으로 파일의 상대경로를 적어주면 됩니다. “프로젝트 디렉토리”가 작업디렉토리입니다.

이클립스 화면 왼쪽, 패키지 익스플로러 구역에서 프로젝트 이름을 오른쪽 버튼 클릭하여 나타나는 팝업메뉴에서 New – File을 선택하고 File name으로 words.txt를 입력하면 빈 words.txt 파일이 만들어지고 편집창이 열립니다. 편집창에 단어들을 복사해 넣고 control-s를 눌러 저장하세요.

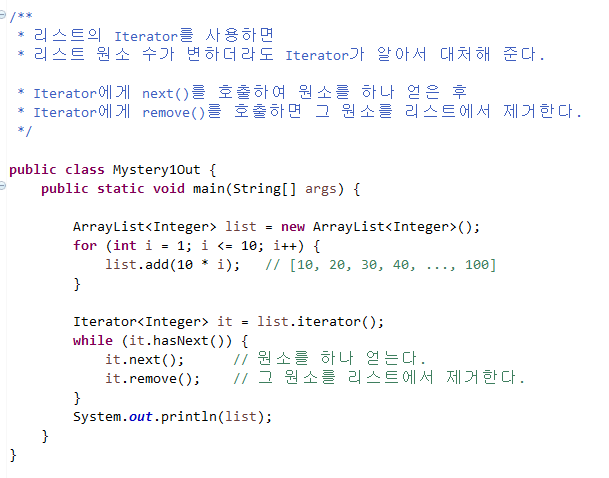
2.2 아래 코드를 실행하면 어떤 출력이 나올지 종이에 적어 보세요.

A computer screen shot of a code

Description automatically generated

[Mystery1.java](src\Mystery1.java)를 실행하여 확인해 보세요. 틀렸다면 왜 틀렸는지 생각해 보세요.

2.3 아래 코드는 Iterator를 사용하여 리스트 원소를 삭제하는 프로그램입니다. 출력이 어떻게 될지 종이에 적으세요. Iterator 사용법을 잘 살펴보세요.

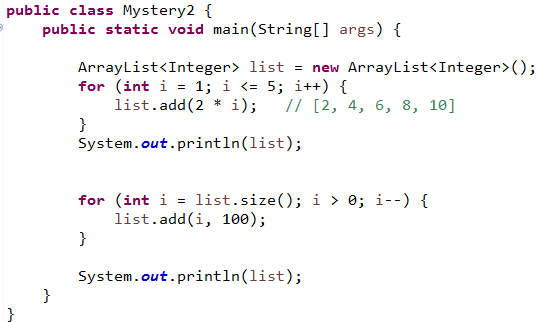


it.remove()를 호출하려면 반드시 그 전에 it.next()를 호출했어야 합니다.

it.next()를 호출한 후에it.remove()를 한 번만 호출할 수 있습니다.

[Mystery1Out.java](src\Mystery1Out.java)를 실행하여 확인해 보세요. 틀렸다면 왜 틀렸는지 생각해 보세요.

2.4 아래 코드의 출력이 어떻게 될지 종이에 적으세요.

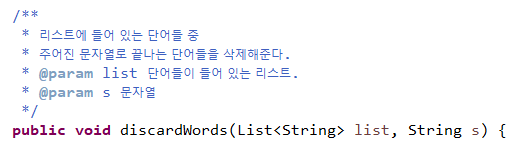


[Mystery2.java](src\Mystery2.java)를 실행하여 확인해 보세요. 틀렸다면 왜 틀렸는지 생각해 보세요.

2.5 위 2.1에 이어지는 문제입니다. FileProcessor를 복사하여 FileProcessor2를 만들고 FileProcessor2에서 아래 작업을 수행하세요.

문자열이 들어 있는 리스트에서 주어진 문자열로 끝나는 단어를 제외한 리스트를 만들어주는 discardWords 메소드를 FileProcessor2 클래스에 추가하세요. 예를 들어, list에 “국화”, “진달래”, “해당화” 등 세 개의 문자열이 들어 있을 때, “화”로 끝나는 단어를 제외하면 “진달래”만 남게 됩니다. “당화”로 끝나는 단어를 제외하면 "국화"와 “진달래”가 남게 됩니다. 이 메소드를 구현할 때 String의 endsWith 메소드를 사용하면 편리합니다. [API 문서](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)를 참고하세요. API 문서를 자꾸 들여다보아야 합니다. 익숙해져야 합니다.

리스트의 원소들을 차례로 읽을 때는 Iterator를 사용하세요. iterator.hasNext를 이용하는 while 반복문 안에서 iterator.next로 항목을 읽고, 그 항목이 조건에 맞는 경우에만 iterator.remove하면 됩니다.



또, main 메소드 코드에 아래 네모에 해당하는 코드를 추가하세요. 출력은 Figure 3와 같아야 합니다.

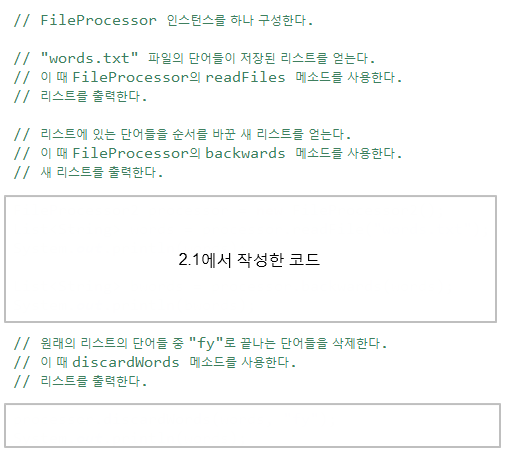
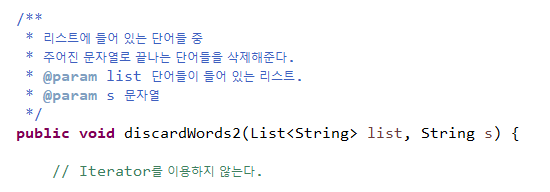




Figure 3

2.6 위 2.5에 이어지는 문제입니다. FileProcessor2를 복사하여 FileProcessor3을 만들고 FileProcessor3에서 아래 작업을 수행하세요.

위 2.5번 문제에서 구현한 discardWords와 같은 기능을 하는 discardWords2 메소드를 FileProcessor3 클래스에 추가하세요. 단, 이 메소드를 구현할 때 Iterator를 사용하지 말고 for 문장을 사용하세요. Iterator를 이용하지 않고 직접 for 문장을 사용해 구현하는 경우 인덱스 변화를 잘 따져야 합니다. (이 문제는 Iterator가 편리한 도구라는 것은 인식하기 위한 문제입니다.)



또, main 메소드에 아래 네모에 해당하는 코드를 추가하세요. 출력은 Figure 4와 같아야 합니다.

A screenshot of a computer program

Description automatically generated

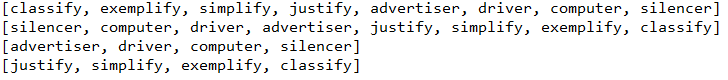


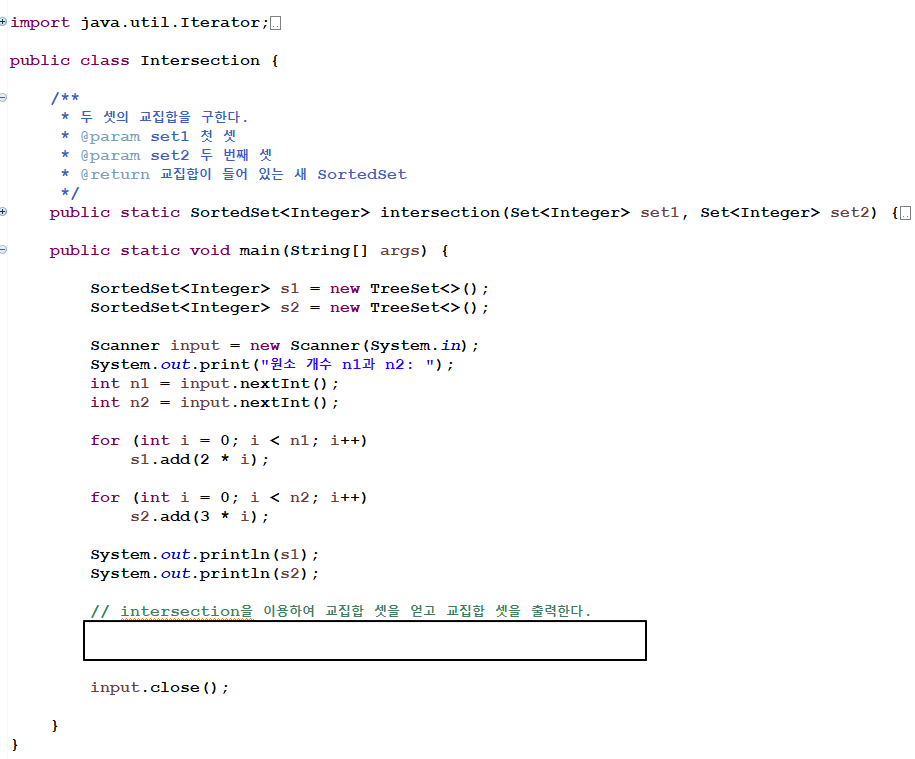
Figure 4

**문제 2에서 작성한 FileProcessor3를 강온실 문제 w6p2\_FileProcessor3에 제출하세요. 강온실의 작업디렉토리에 words.txt 파일이 이미 저장되어 있습니다. 학생들은 데이터 파일에는 신경쓰지 않고 FileProcessor3 클래스만 제출하면 됩니다.**

문제 3

w6 패키지에서 아래 과제를 수행하세요.

두 개의 Set<Integer>를 받아 그 교집합인 Set<Integer>를 반환하는 static 메소드를 작성하고 테스트하세요. 두 셋의 교집합을 구한다는 것은 첫 셋에도 들어 있고 두 번째 셋에도 들어 있는 원소들을 찾아낸다는 의미입니다. **Set의 retainAll 메소드를 사용하지 말고 직접 두 개의 셋으로부터 원소들을 골라 새 셋에 넣는 방식으로 구현하세요.** 클래스 이름은 Intersection으로 짓고 아래 뼈대 코드를 이용하여 완성하세요. 출력은 Figure 5와 같아야 합니다. 수자들이 오름차순으로 나타나게 하세요.



A number on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure 5

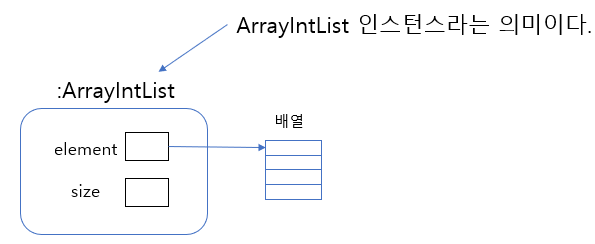
**문제 3에서 작성한 Intersection 클래스를 강온실 w6p3\_Intersection 문제의 답으로 제출하세요.**

문제 4

우리는 지금까지 표준 라이브러리에 들어 있는 java.util.ArrayList 클래스를 이용해 프로그램을 작성했습니다. 이번에는 ArrayList 클래스를 우리 나름대로 작성해 보겠습니다. java.util.ArrayList는 많은 기능을 지원하는 복잡한 클래스이므로 우리는 이를 간단히 줄여 정수만 저장할 수 있고 몇 개의 메소드들만 지원하는 단순한 클래스를 만들어 봅니다. 클래스 이름은 ArrayIntList로 짓겠습니다.

4.1 w6.arraylist1 패키지를 만들고 그 아래에서 작업하세요.

ArrayIntList는 아래와 같은 구조로 만들 수 있습니다. ArrayIntList는 배열을 필드(상태변수)로 갖고 있어서(아래 그림의 element) ArrayIntList에 정수들을 넣으면 그것들이 배열에 저장됩니다. 또한, 배열에 몇 개의 정수가 들어 있는지 알기 위해 size라는 이름의 int 타입 상태변수를 필드로 두었습니다. 그림의 “:ArrayIntList”는 ArrayIntList 인스턴스라는 의미입니다.



우선 ArrayIntList가 어떤 기능을 갖도록 할 것인지 정해야 하겠습니다. 우리가 작성할 [ArrayIntList의 API 문서](doc\w6\arraylist1\ArrayIntList.html)를 보세요. 한 개의 생성자와 여덟 개의 메소드가 있습니다. 이 기능들을 갖춘 클래스를 구현하세요. Figure 6의 뼈대를 이용하세요.

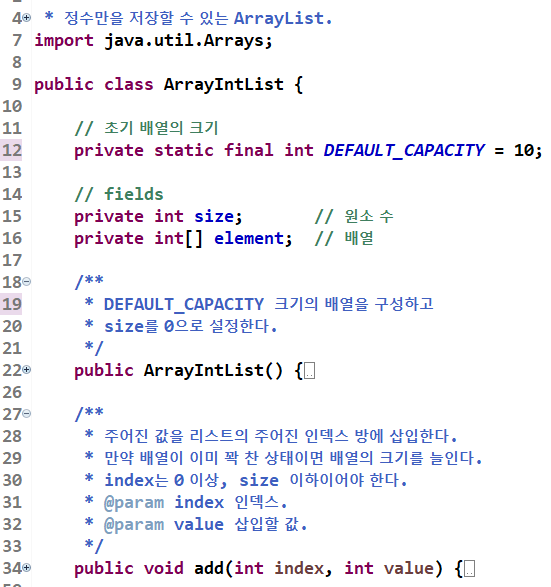
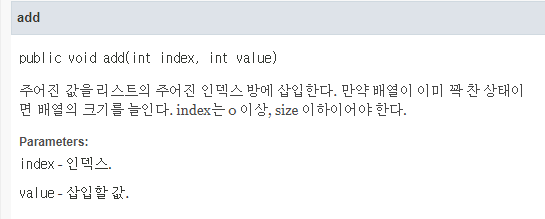


Figure 6. ArrayIntList 클래스의 뼈대.

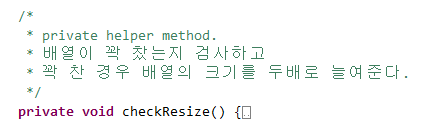
우선은 필드를 선언하고, 생성자와 add 메소드 하나를 구현합니다. 12번 줄에 있는 필드는 ArrayIntList의 기본용량입니다. 이 값은 클래스 공통의 값으로서 개별 ArrayIntList 인스턴스마다 각각 가지고 있을 필요가 없는 클래스 공통의 데이터이므로 static 필드로 선언했습니다. 15번, 16번 줄에 있는 두 개의 필드는 각 인스턴스마다 각각 가져야 하는 데이터이므로 static이 없는 인스턴스 필드로 선언했습니다. 생성자는 쉽게 작성할 수 있습니다. add 메소드가 다소 복잡합니다. 아래 설명을 보세요.

* + 1. add(int, int) 메소드는 어떤 값을 지정된 위치(index)에 삽입하는 기능입니다.



값을 삽입하기 전에 삽입할 위치에 빈 공간을 만들어야 합니다. 즉, index 이후의 값들을 한 칸씩 뒤로 옮겨야 합니다. 그런데 만약 배열이 이미 꽉 찬 상태였다면 맨 마지막 값을 잃게 되므로 값들을 뒤로 옮기기 전에 배열의 크기를 늘여 줘야 합니다. 배열의 크기를 늘일 때는 기존 크기의 두 배가 되도록 합시다. 배열은 한 번 생성하고 나면 그 크기가 바뀌지 않으므로 배열의 크기를 늘여 준다는 말은 사실은 더 큰 배열을 새로 생성하고 기존 배열에 있던 모든 값들을 새 배열로 복사해 준다는 말입니다. 그리고 기존 배열을 가리키던 element 상태변수가 새 배열을 가리키게 합니다. 그러면 기존 배열 메모리는 참조변수를 잃게 되므로 자바 시스템에 의해 재활용 쓰레기로 회수(garbage collection)됩니다. 값을 삽입한 후에는 원소 수를 관리하는 데 사용하는 size 상태변수를 1만큼 증가시켜 줍니다.

배열이 이미 꽉 찬 상태에서 배열의 크기를 늘여 주는 기능을 add 메소드 내에 구현하면 add 메소드 코드가 복잡하고 길어집니다. 그래서 이 기능을 따로 떼어 별도의 private method로 구현하고 add 메소드에서는 이 메소드를 호출하는 편이 좋겠습니다. 이 기능을 하는 private 메소드 이름을 checkResize로 지읍시다. 배열의 크기를 검사하고 필요한 경우 크기를 조정한다는 의미입니다. Element와 size는 필드이므로 클래스 안의 모든 메소드에서 접근할 수 있으므로 이 checkResize 메소드는 파라미터가 필요 없습니다.



checkResize 메소드를 구현하면 add 메소드를 구현할 때 그 첫 문장으로 checkResize()를 호출하면 됩니다. 일단은checkResize 메소드의 헤더만 적어 넣고 중괄호 속의 구현은 미루어 둡시다. (Top-Down 방법!)

셋에 원소가 10개 있는 경우,

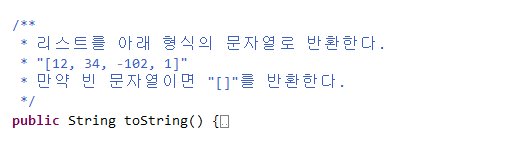
get(int index) 메소드에 줄 수 있는 인덱스는 0부터 9까지입니다.

set(int index, int value) 메소드에 줄 수 있는 인덱스도 0부터 9까지입니다.

그러나, add(int index, int value) 메소드에 줄 수 있는 인덱스는 0부터 10까지입니다.

그러니까, add 메소드의 index가 0이상, size 이하가 아니면 오류입니다. 이에 대한 예외처리도 해야 하지만 일단은 이것도 미루어 둡시다. 우선은 index가 올바른 범위에 있다고 가정합시다.

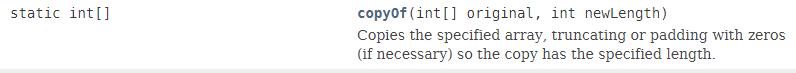
* + 1. 다음으로는 아래 그림의 toString 메소드를 구현합니다.



생성자와 add, toString 메소드를 구현하고 나면 main 메소드를 삽입하고 간단한 테스트를 해 볼 수 있습니다. 메소드들을 하나 하나 구현해 나갈 때마다, 뭔가 수정을 할 때마다, main에 적당한 테스트 코드를 넣어 확인해 보는 것이 좋습니다.

add와 toString 메소드의 기본 기능을 구현하고 나면 checkResize 메소드도 구현합니다.

어떤 크기의 배열을 새로 생성하고 기존 배열에 있던 모든 값들을 새 배열로 복사해 주려면 [java.util.Arrays](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Arrays.html) 클래스의 아래 메소드를 사용하면 편리합니다.



* + 1. UseList1 클래스를 만들고 그 안에 main 메소드를 두어 지금까지 만든 ArrayIntList의 기능을 아래와 같이 테스트합니다.
* ArrayIntList 객체를 하나 생성한다.
* 리스트에 0부터 9까지 10개의 숫자를 0번 방에 차례로 add한다. (초기 배열 크기가 10이므로 배열이 꽉 차게 된다.)



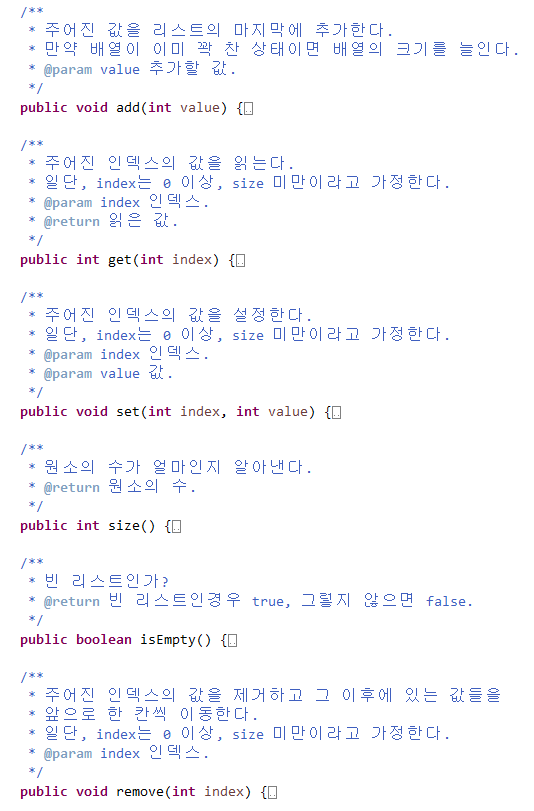
* 리스트를 출력하여 제대로 add가 되었는지 확인한다.
* 리스트의 1번 방에 숫자 하나(22)를 추가한다.



* 리스트를 출력하여 제대로 삽입이 되었는지 확인한다. 특히 배열의 크기를 늘이고 11개의 숫자를 모두 저장했는지 확인한다.



* + 1. 지금까지 구현한 기능이 제대로 작동하면 아래 기능들을 추가합니다. 파라미터를 하나만 갖는 아래 add 메소드는 위에서 이미 구현한 파라미터 두 개 짜리 add메소드를 이용(호출)하여 손쉽게 구현할 수 있습니다. 나머지 메소드들도 간단히 구현할 수 있습니다. size와 isEmpty 메소드를 구현할 때는 size 상태변수를 이용합니다.

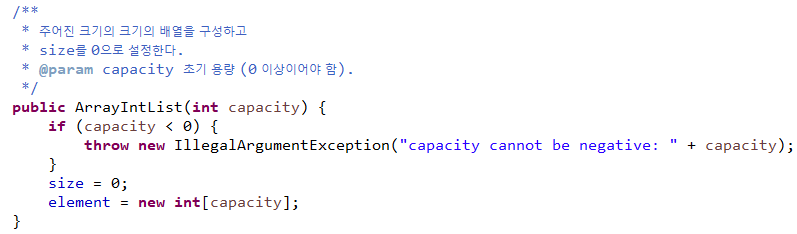


* + 1. 위 코드를 완성한 후에는 UseList1.main에 위 메소드들을 아래와 같이 테스트하는 코드를 추가하여 ArrayIntList의 기능을 테스트하세요.
* ArrayIntList 객체를 하나 생성한다.
* 리스트에 10개의 숫자를 add한다. (초기 배열 크기가 10이므로 배열이 꽉 차게 된다.)
* 리스트를 출력하여 제대로 add가 되었는지 확인한다.
* 리스트에 한 개의 숫자를 추가로 add한다. Add할 위치는 적절히 정한다.
* 리스트를 출력하여 제대로 삽입이 되었는지 확인한다. 특히 배열의 크기를 늘여 11개의 숫자를 모두 저장했는지 확인한다.
* set 메소드를 적절히 호출하고 제대로 수행됐는지 리스트를 출력하여 확인한다.
* size 메소드가 잘 작동하는지 테스트한다.
* remove 메소드를 이용하여 원소를 모두 삭제하고 제대로 삭제됐는지 리스트를 출력하여 확인한다.
* isEmpty 메소드가 잘 작동하는지 테스트한다.

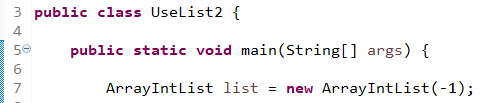
4.2 w6.arratlist1 패키지를 복사하여 w6.arraylist2 패키지를 만들고 그 아래에 프로그램을 작성하세요. 위 4.1번 문제에서 작성한 ArrayIntList에 아래 기능을 추가하세요.

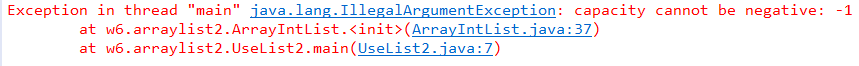
* + 1. Capacity 파라미터를 갖는 두 번째 생성자를 갖도록 하세요.

기존 생성자는 DEFAULT\_CAPCITY, 즉 10의 크기를 갖는 리스트를 생성하는 것이었습니다. 새로 추가되는 생성자를 사용하면 임의의 초기 크기를 갖는 리스트를 만들 수 있습니다. 이 생성자는 아래 그림처럼 작성합니다.



만약 capacity가 음수라면, 위 그림의 끝부분에 있는 new int[capacity] 문장을 실행할 때 에러가 발생합니다. 큰 프로그램 개발이 끝나고 현장에 배포한 후 프로그램이 그냥 죽어 버리면 어디에서 어떤 에러가 나서 죽었는지 알기 어려우므로 그냥 죽는 대신 적절한 예외(Exception)를 던지도록 합니다. 위 코드에서는 capacity로 음수가 주어진 경우 IllegalArgumentException을 던지도록 했습니다. **프로그램 실행 중 예외가 던져지면 그 다음 문장으로 넘어가는 대신 예외처리기로 넘어 갑니다. 프로그램 내에 예외처리기를 작성해 넣지 않은 경우에는 프로그램이 종료되고 시스템에 의해 예외가 처리됩니다.** 시스템이 예외를 처리한다는 말은 프로그램 시작 이후 어떤 경로를 거쳐 예외가 발생하게 됐는지(Call stack)와 예외의 내용을 화면에 출력해 준다는 의미입니다. 아래 화면은 그 예입니다. main 메소드를 갖는 UseList2.java라는 이름의 파일을 만들고 아래 그림과 같은 간단한 코드를 넣어 실행한 것입니다.





위 출력은

1. 프로그램 시작 이후UseList2.java 파일의 7번 줄에서 ArrayIntList.java 파일의 37번 줄로 넘어갔고,
2. 그 줄을 실행하는 중에 IllegalArgumentException이 던져졌으며,
3. IllegalArgumentException 속에는 "capacity cannot be negative: -1"이라는 정보가 들어 있다

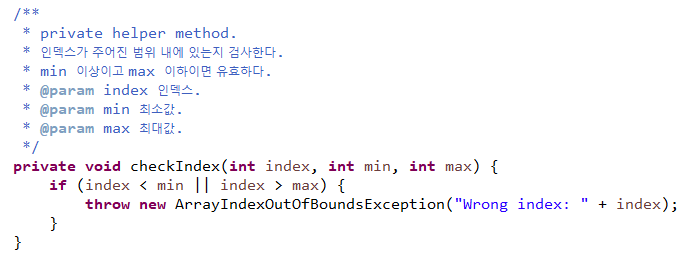
는 점을 알려 줍니다.

* + 1. 리스트를 탐색하는 데 사용하는 아래 두 메소드를 추가로 구현하고 두 메소드 각각이 제대로 작동하는지 UseList2.main에 적당한 코드를 넣어 테스트하세요. 아래 메소드의 형식은 [api 문서](doc\w6\arraylist2\ArrayIntList.html)를 참고하세요.

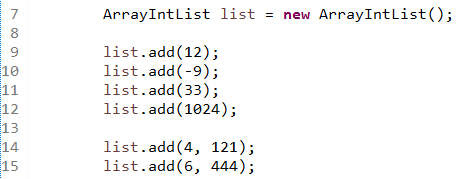
indexOf

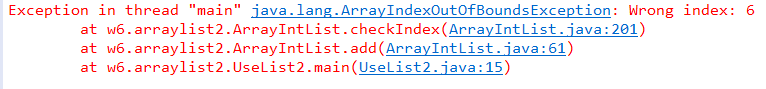
contains

* + 1. 지금까지 구현한 메소드들 중 몇 가지는 index를 파라미터로 갖습니다. 그런데 인덱스는 일정 범위 내에 있어야 합니다. 그렇지 않으면 실행 중 에러가 발생합니다. 따라서 index를 파라미터로 갖는 메소드들은 본 기능을 수행하기 전에 index가 올바른 범위 내에 있는지 점검해야 합니다. Index가 올바른지 점검하는 기능은 여러 메소드들에서 공통적으로 필요하므로 이 기능을 아래 그림처럼 별도의 private 메소드로 구현합니다. 이 메소드는 Index가 올바른 범위에 있지 않으면 ArrayIndexOutOfBoundsException를 던집니다.



지금까지 구현한 메소드들 중 파라미터로 주어진 index가 올바른지 점검할 필요가 있는 메소드들(index를 파라미터로 갖는 메소드들: add, get, set, remove)에 점검 기능을 추가하세요. 점검할 때는 위 checkIndex 메소드를 이용하세요. 또, 이 메소드들에 올바르지 않은 인덱스를 주면서 호출하여 예외던지기가 제대로 되는지 확인하세요. 예를 들면 아래와 같습니다.





위 그림의 실행 결과는 아래와 같은 의미입니다.

1. 프로그램이 시작된 후 UseList2.java의 15번 줄에서 ArrayIntList.java의 61번 줄로 넘어갔고,
2. 거기서 다시 201번 줄로 넘어갔고,
3. 201번 줄을 실행하는 과정에서 ArrayIndexOutOfBoundsException이 던져졌습니다.
4. ArrayIndexOutOfBoundsException에는 "Wrong index: 6"라는 정보가 들어 있습니다.
   * 1. UseList2의 main에 올바르지 않은 인덱스 사용으로 인해 던져진 ArrayIndexOutOfBoundsException를 잡아 Exception에 getMessage를 호출해 얻은 메시지를 출력하는 아래 코드를 작성하세요. 16번 줄에서 발생하는 예외를 18번 줄에서 잡아 19번 줄에서 처리합니다. 그 이후에는 23번 줄로 프로그램 실행이 이어집니다.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

출력은 아래와 같아야 합니다.

Text

Description automatically generated

이 문제에서는 ArrayIntList의 기능을 테스트하는 코드를 스스로 작성하도록 했습니다. 테스트를 철저하게 할 수 있는 능력도 개발자에게 중요합니다.

완성된 ArrayIntList 클래스를 w6p4\_ArrayIntList 문제의 답으로 제출하세요. ArrayIntList 클래스만 제출하세요. 패키지 선언문, import 문장은 넣지 마세요. 강온실의 문제 자체에 java.util.Arrays는 이미 import되어 있습니다.

끝.