부록 2. JShell 사용법 추가 설명

JShell 에서는 사용자 편의를 위해 몇 가지 설정을 하는 것이 가능하다. 에디터를 설정하는 것은 본문에서 설명했고, 여기서는 "feecback" 설정을 살펴보자.

feedback 설정

JShell 에서 명령문을 실행하면 그 결과를 화면에 보여주는데, 이를 피드백(feedback)이라고 한다. 그런데 이렇게 보여주는 피드백 내용이 너무 많은 경우가 있다. 이를 제거하고 싶으면 "off"로 설정하면 된다. 우선 다음 코드를 실행시켜 피드백에 대해서 좀 더 살펴보자.

```
jshell> /set feedback
|  /set feedback normal
|
|  Available feedback modes:
|     concise
|     normal
|     silent
|     verbose
```

/set feedback 명령은 JShell 에게 현재 피드백 상태가 무엇인지 알려달라고 요청한다. 위에서 볼 수 있는 것처럼 현재는 보통 (normal) 피드백 상태임을 확인할 수 있다. 또한 feedback 설정을 어떻게 바꿀 수 있는지에 대해서도 함께 보인다. 여기서는 concise (간략하게 보이기), normal (보통), silent (최소한의 정보만 보이기), verbose (최대한의 정보를 보이기) 등으로 설정 가능하다. 다음에서는 각 모드로 바꿔보면서 실제 어떻게 결과가 나타나는 지 확인해보도록 하자. 피드백 모드를 설정하는 것은 "/set feedback 모드이름" 형태로 처리한다.

```
jshell> int a = 3;
a ==> 3
jshell> /set feedback concise
jshell> int b = 4;
jshell> /set feedback silent
-> int c = 5;
-> /set feedback verbose
|  Feedback mode: verbose
jshell> int d = 6;
d ==> 6
|  created variable d : int
```

실행 화면에서 볼 수 있듯이 concise 모드로 바뀌면 변수에 값을 넣었다고 표시되던 부분도 사라진다. 그리고 silent 모드로 바뀌면 "jshell>" 이라고 되어 있던 부분마저도 간략하게 "->" 형태로 바뀐다. 가장 정보를 많이 전달하는 verbose 모드에서는 d 라는 이름을 가진 변수가 생성되었고 어떤 자료형으로 만들어졌는지 까지도 보인다. 이렇게 다양한 피드백 모드가 있으므로 본인에게 적절한 형태로 지정해서 사용하면 된다.

변수 보기 (/vars 명령)

JShell 에서는 /vars 명령을 입력하면 현재까지 만들어진 모든 변수들과 값을 확인할 수 있다. 다음 코드를 실행시켜보자.

```
jshell> int a = 3;
a ==> 3
jshell> int b = 4;
b ==> 4
jshell> String s = "abc";
s ==> "abc"
jshell> double d = 3.14;
d ==> 3.14
jshell> d = 3.1415;
d ==> 3.1415
jshell> /vars
|    int a = 3
|    int b = 4
|    String s = "abc"
|    double d = 3.1415
```

/vars 명령은 자바 코드가 아니라 JShell 에 전달되는 명령어로 현재까지 만들어진 변수들과 값을 화면에 보인다.

작성했던 함수 확인하기 (/methods 명령)

JShell 에서 현재까지 만들었던 함수들 목록을 뽑아내기 위해 /methods 명령을 사용할 수 있다. 함수에 대해서는 나중에 다시 학습하기로 하고, 여기서는 다음에 보인 것처럼 앞에서 보았던 add 함수를 다시 입력하고 또 다른 함수를 한 개 작성한 후에 /methods 명령을 실행시켜보자.

```
jshell> int add(int num1, int num2) {
   ...>     int res;
   ...>     res = num1 + num2;
   ...>     return res;
   ...>}
|  created method add(int,int)

jshell> void sayHello() {
   ...>     System.out.println("Hello");
   ...> }
|  created method sayHello()

jshell> /methods
|    int add(int, int)
|    void sayHello()
```

입력했던 코드 살펴보기 (/list 명령)

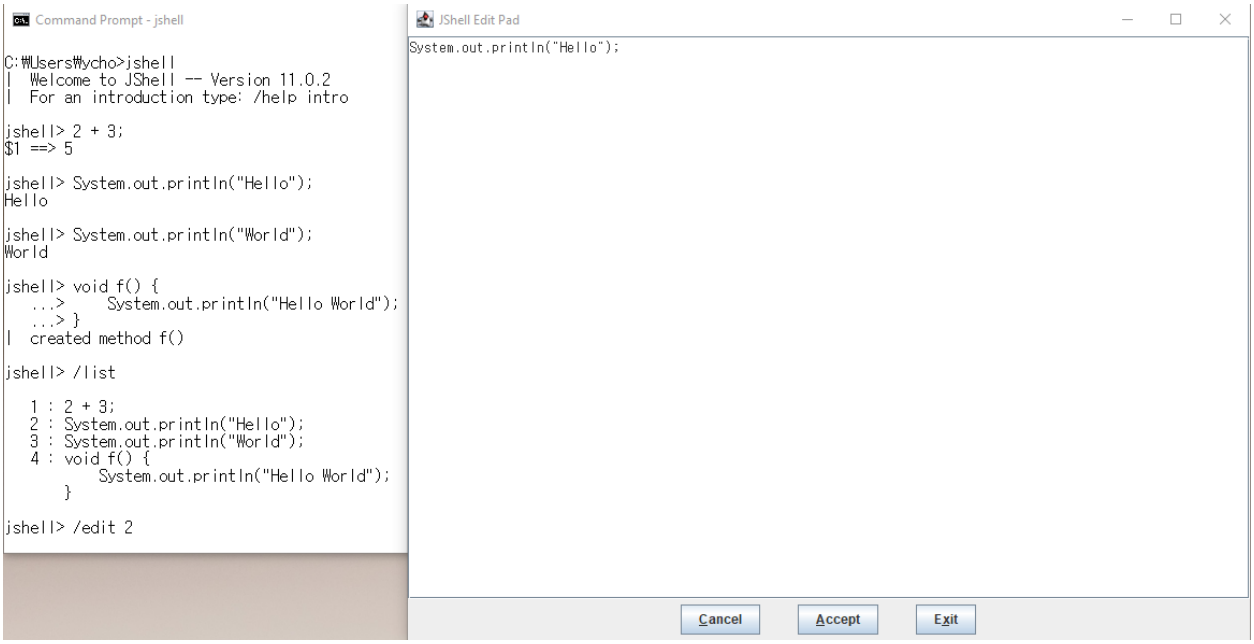
JShell 에서는 /list 명령을 사용하면 입력했던 코드의 목록을 살펴보고 원하는 코드를 수정하는 것이 가능하다. JShell 을 실행하고 다음 코드를 입력해보도록 하자.

```
jshell> 2 + 3;
$1 ==> 5

jshell> System.out.println("Hello");
Hello

jshell> System.out.println("World");
World

jshell> void f() {
   ...>     System.out.println("Hello World");
   ...> }
|  created method f()

jshell>/list

   1 : 2 + 3;
   2 : System.out.println("Hello");
   3 : System.out.println("World");
   4 : void f() {
           System.out.println("Hello World");
       }
```

여기서 보인 것처럼 /list 명령을 사용하면 여태까지 입력했던 코드에 번호가 붙어서 출력된다. 이 번호와 /edit 명령을 이용하면 특정 코드를 수정하는 것이 가능하다. 예를 들어 다음처럼 실행시켜보자.

```
jshell> /edit 2
```

그럼 다음에 보인 화면처럼 2 번에 해당되는 코드가 에디터에 보이고 이를 수정하면 해당 코드에 다시 반영된다. 나중에 함수나 클래스를 학습하게 되고, 그런 함수나 클래스를 수정할 때 유용하게 사용할 수 있다.



준비된 패키지 확인하기 (/imports 명령)

앞에서 〈Shift〉〈Tab〉 + 〈i〉를 사용하면 클래스를 사용할 때 필요한 패키지들을 준비시키는 것이 가능하다고 했다. /imports 명령은 현재까지 사용할 수 있도록 준비된 패키지들을 확인하는 명령이다. 다음 코드를 실행시켜보자.

```
jshell> /imports
|    import java.io.*
|    import java.math.*
|    import java.net.*
|    import java.nio.file.*
|    import java.util.*
|    import java.util.concurrent.*
|    import java.util.prefs.*
|    import java.util.regex.*
|    import java.util.stream.*
```

지금은 이런 패키지들이 어떤 기능들을 제공하는지 알 필요는 없다. 그냥 단순히 이러한 패키지들이 준비되어 있고, 이런 것들을 /imports 명령어를 통해서 확인할 수 있음만 기억하자.

명령어 사용 이력 확인하기 (/history 명령)

JShell 에는 현재까지 사용했던 명령들의 목록을 확인할 수 있는 기능이 제공된다. /history 명령을 사용하면 목록을 확인할 수 있다. 다음처럼 실행시켜보자.

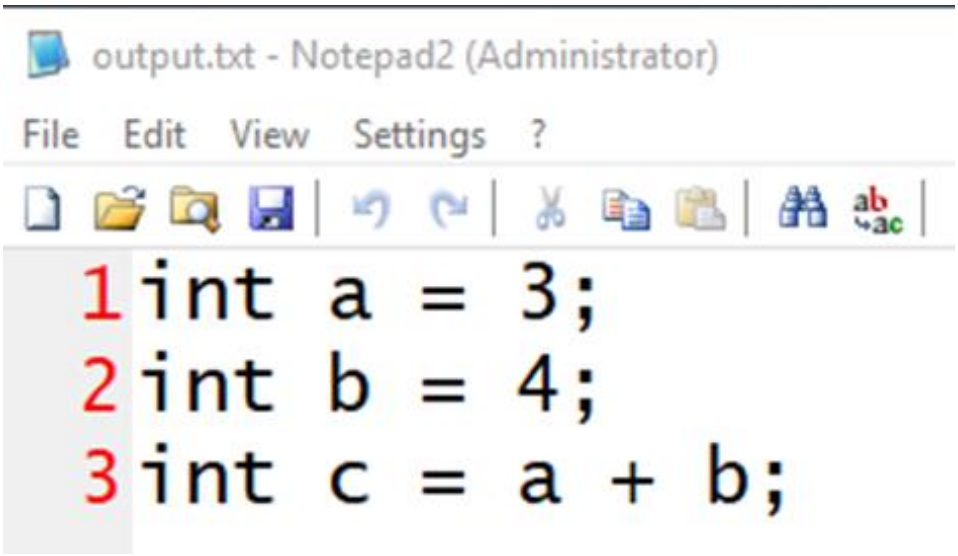
```
jshell> /history
/vars
/imports
/history
```

작성했던 자바 코드 저장하기 (/save 명령)

JShell 에서 코드를 입력하고 실행시킨 후에 현재까지 작성했던 자바 코드를 저장하는 기능이 있다. /save 명령을 사용하면서 파일의 이름을 전달하면 해당 파일에 여태까지 실행했던 자바 코드를 저장한다. 다음 실행 내용을 살펴본다.

```
jshell> int a = 3;
a ==> 3
jshell> int b = 4;
b ==> 4
jshell> int c = a + b;
jshell> /save output.txt
```

output.txt 의 내용



파일에 있는 코드를 실행하기 (/open 명령)

JShell 에서는 /save 명령을 이용해서 저장했거나 다른 도구에서 작성했던 파일에 저장되어 있는 자바 코드를 실행시키는 것도 가능하다. 파일에 저장되어 있는 코드를 실행시키려면 /open 명령과 함께 코드가 있는 파일의 이름을 실행시키면 된다. 다음에 보인 것처럼 JShell 에서 앞에서 저장했던 output.txt 의 코드를 실행시키고 변수의 내용을 살펴보자.

```
jshell> /open output.txt
jshell> /vars
|    int a = 3
|    int b = 4
|    int c = 7
```

output.txt 에는 변수 a 와 b 를 정의하고 각각 3 과 4 를 저장하였다. 그리고 둘의 합을 구해 c 라는 변수에 저장했던 코드가 있었다. 여기서 /open 명령을 이용해서 output.txt 의 내용을 실행시키면 다시 그 코드를 JShell 에서 작성한 것처럼 수행된다. 따라서 /vars 명령을 전달했을 때 변수 a, b, c 의 값을 확인할 수 있다.

클래스 경로(classpath) 지정

자바에서 패키지들을 사용할 때 기본적으로 제공되는 것들이 아닌 경우, 해당 패키지가 어느 디렉토리(폴더)에 있는지 지정해주어야 한다. 이러한 디렉토리를 자바에서는 classpath 라고 하는데 JShell 에서도 이러한 경로를 지정해주는 것이 가능하다. JShell 에서 classpath 를 지정하는 방법은 다음에 보인 것처럼 "/env –class-path <path>"로 처리한다.

```
jshell> /env -class-path C:/Users/ycho/javaclasses
```