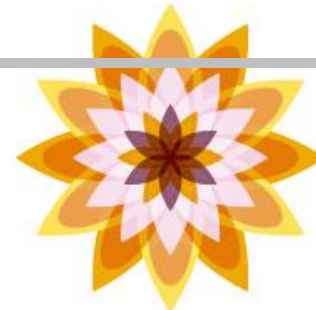
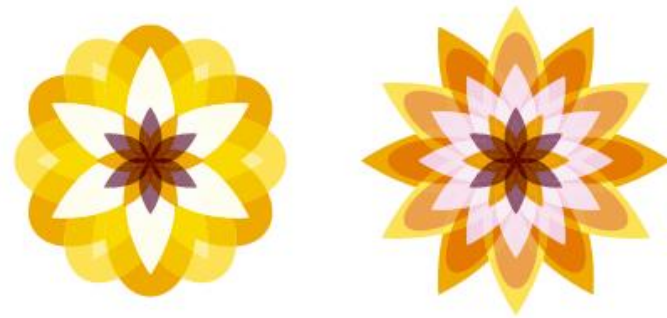


Chapter 02

가위, 바위, 보 게임



1. 이번에 만들 코드

- 우리가 이번에 만들 게임은 컴퓨터와 가위, 바위, 보를 하는 게임이다.
- 앞장에서 배운 난수를 만드는 방법을 이용해서 컴퓨터가 먼저 가위, 바위, 보 중 하나를 미리 정하고, 플레이어(사람)가 선택한 가위, 바위, 보 중 하나와 비교하여 승부(사람 승, 컴퓨터 승, 무승부)를 알려주는 프로그램이다.
- 이 프로그램을 통해서 우리는 자바의 중요한 클래스 중 하나인 String 클래스와 레퍼런스 데이터형, 키보드에서 글자를 입력받는 방법, if 문 등의 제어문 사용법을 익힐 수 있다.

2. String 클래스

■ 문자열의 표현과 객체 생성

- 문자열은 반드시 큰따옴표(String a = "문자열") 안에 표기해야 한다.
- 큰따옴표 안에는 String a = ""와 같이 아무런 문자열이 오지 않아도 상관없다.
- 하지만 큰따옴표는 절대 생략할 수 없다.
- String 클래스의 객체를 생성하는 데는 크게 2가지 방법이 있다.
- 첫 번째 방법은 new 키워드를 사용하는 방법으로 생성자의 입력 매개 변수로 저장할 문자열을 입력한다.
- 아직 클래스를 배우기 전이므로 이런 모양이 다소 어색해 보일 수 있지만, 대부분의 클래스가 이 방법으로 객체를 생성한다.
- 생성자의 개념은 아직 배우지 않았지만, 클래스명과 동일하면서 뒤에 소괄호가 있는 형태다.
- 두 번째 방법은 간단히 문자열 리터럴, 즉 문자열 값만 입력하는 방법이다.

```
String str1 = new String("Hello");  
String str2 = "Hello";
```

2. String 클래스

■ 문자열의 표현과 객체 생성

- 첫 번째 방법을 사용하든, 두 번째 방법을 사용하든 메모리에 저장되는 방식은 동일하다.
- String은 참조 자료형이므로 그림과 같이 실제 데이터인 String 객체는 힙 메모리에 위치하고, 참조 변수는 힙 메모리의 실제 객체 위치를 가리키게 될 것이다.
- 하지만 이 2가지 방법 사이에는 결정적인 차이가 1개 있다.
- 그 차이는 곧 설명하기로 하고, 여기서는 2가지의 String 객체를 생성하는 방법만 기억하자.



2. String 클래스

■ 문자열의 표현과 객체 생성

- 실습. String 객체를 생성하는 2가지 방법

CreateStringObject.java

```
1 package sec02_string.EX01_CreateStringObject;
2
3 /*String 객체 생성의 두 가지 방법*/
4 public class CreateStringObject {
5     public static void main(String[] args) {
6         // #1. String 객체 생성 1
7         String str1 = new String("안녕");
8         System.out.println(str1);
9
10        // #2. String 객체 생성 2
11        String str2 = "안녕하세요";
12        System.out.println(str2);
13    }
14 }
```

2. String 클래스

■ String 클래스의 2가지 특징

- String 클래스도 당연히 클래스이므로 다른 클래스들의 특징을 모두 지니고 있다.
- 하지만 워낙 자주 사용되는 클래스이다 보니 다른 클래스에는 없는 2개의 특징이 있다.
- 첫 번째 특징은 한 번 정의된 문자열은 변경할 수 없다는 것이다.
- 만일 문자열의 내용을 변경하면 자바가상 머신은 기존의 문자열을 수정하는 것이 아니라 새로운 문자열을 포함하고 있는 객체를 생성해 사용하고, 기존의 객체는 버린다.
- 두 번째 특징은 문자열 리터럴을 바로 입력해 객체를 생성할 때 같은 문자열끼리 객체를 공유한다는 것이다.
- 이는 메모리의 효율성 때문이다.
- 그럼 각각의 특징을 자세히 살펴보자.

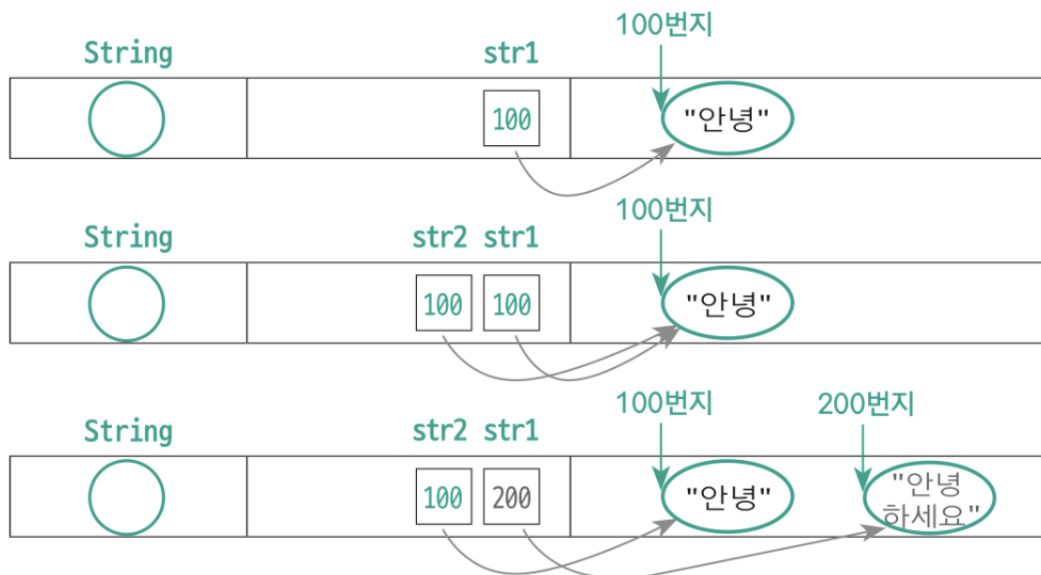
2. String 클래스

■ String 클래스의 2가지 특징

■ 특징 ① 객체 안의 값을 변경하면 새로운 객체를 생성

- String 객체는 내부에 포함된 문자열을 변경할 수 없다.
- 다음 예를 살펴보자.

```
String str1 = new String("안녕");  
String str2 = str1;  
str1 = "안녕하세요";  
System.out.println(str1);           // 안녕하세요  
System.out.println(str2);           // 안녕
```



2. String 클래스

■ String 클래스의 2가지 특징

- 특징 ① 객체 안의 값을 변경하면 새로운 객체를 생성
 - 실습. String 객체의 문자열 수정 및 다른 참조 자료형과의 비교

ModificationOfStringData.java

```
1 package sec02_string.EX02_ModificationOfStringData;
2
3 public class ModificationOfStringData {
4     public static void main(String[] args) {
5         // 문자열 수정 (객체내의 내용변경 불가 --> 새로운 객체 생성)
6         String str1 = new String("안녕");
7         String str2 = str1;
8
9         str1="안녕하세요";
10        System.out.println(str1); //안녕하세요
11        System.out.println(str2); //안녕
12    }
13 }
```

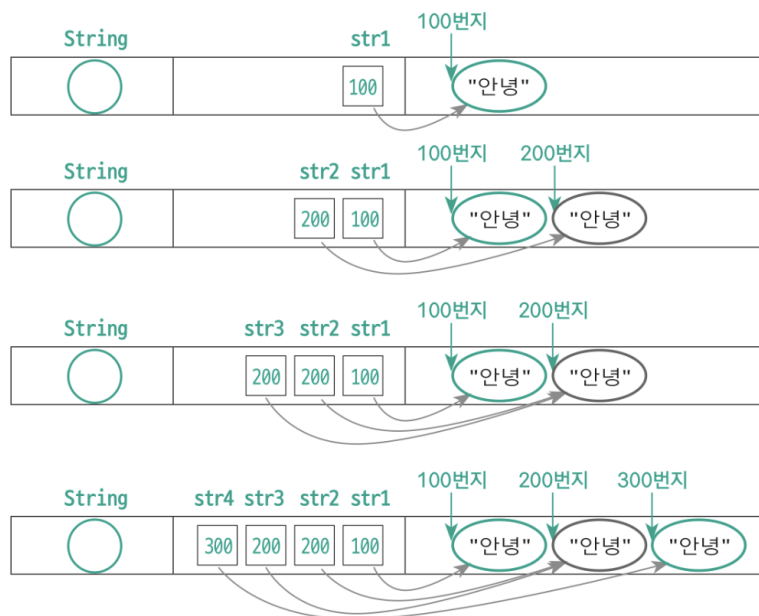

2. String 클래스

■ String 클래스의 2가지 특징

■ 특징 ② 리터럴을 바로 입력한 데이터는 문자열이 같을 때 하나의 객체를 공유

- 두 번째 방법인 문자열 리터럴만 입력해 String 객체를 생성하면 하나의 문자열을 여러 객체가 공유할 수 있다.
- 이는 다른 클래스에 없는 특징으로, 특정 문자열의 객체를 여러 개 만들어 사용할 때 메모리 효율성을 증가시키기 위한 것이다.
- 다음처럼 4개의 String 객체를 생성해보자.

```
String str1 = new String("안녕");  
String str2 = "안녕";  
String str3 = "안녕";  
String str4 = new String("안녕");
```



2. String 클래스

■ String 클래스의 2가지 특징

- 특징 ② 리터럴을 바로 입력한 데이터는 문자열이 같을 때 하나의 객체를 공유

- 실습. 문자열 리터럴에 따른 생성 문자열 객체의 공유

SharingStringObject.java

```
1 package sec02_string.EX03_SharingStringObject;
2
3 /*문자열 리터럴에 의한 생성 문자열 객체의 공유*/
4
5 public class SharingStringObject {
6     public static void main(String[] args) {
7         // #1. 문자열 객체 공유 (리터럴로 객체를 생성한 경우) new 키워드로 객체 생
8         성한 경우 (별도의 객체 생성) 공유X
9         String str1 = new String("안녕");
10        String str2 = "안녕";
11        String str3 = "안녕";
12        String str4 = new String("안녕");
```

2. String 클래스

■ String 클래스의 2가지 특징

- 특징 ② 리터럴을 바로 입력한 데이터는 문자열이 같을 때 하나의 객체를 공유

- 실습. 문자열 리터럴에 따른 생성 문자열 객체의 공유

SharingStringObject.java

```
13      //@stack 메모리 값 비교 (==)
14      System.out.println(str1==str2); //false
15      System.out.println(str2==str3); //true
16      System.out.println(str3==str4); //false
17      System.out.println(str4==str1); //false
18  }
19 }
```

2. String 클래스

■ String 객체의 '+' 연산

- String 객체는 + 연산을 이용해 문자열을 연결할 수 있다.
- 당연히 '더하기'의 의미가 아니라 '연결하기'의 의미다.
- String 객체의 + 연산은 크게 2가지 형태가 있다.
- 첫 번째는 '문자열 + 문자열'의 형태인데, 이때는 그대로 문자열을 연결한 결과가 리턴된다.
- 두 번째는 '문자열 + 기본 자료형' 또는 '기본 자료형 + 문자열'일 때인데, 이때는 기본 자료형이 먼저 문자열로 변환되고, 이후 '문자열 + 문자열'의 형태로 연결된 값이 리턴된다.

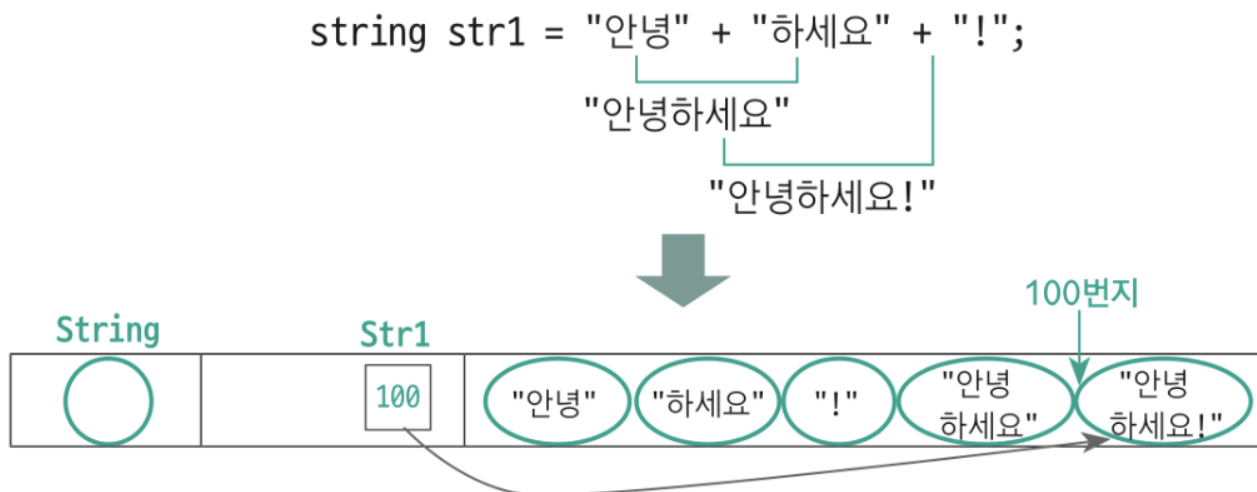
2. String 클래스

■ String 객체의 '+' 연산

▪ 유형 ① '문자열 + 문자열' 연산

```
String str1 = "안녕" + "하세요" + "!";  
System.out.println(str1);    // 안녕하세요!
```

```
String str2 = "반갑";  
str2 += "습니다";  
str2 += "!";  
System.out.println(str2);    // 반갑습니다!  
System.out.println();
```



2. String 클래스

■ String 객체의 '+' 연산

- 유형 ② '문자열 + 기본 자료형' 또는 '기본 자료형 + 문자열' 연산
 - 모든 연산은 동일한 자료형끼리만 가능하다.
 - 따라서 기본 자료형과 문자열을 연산하려면 먼저 기본 자료형을 문자열로 바꾸고, 이어서 '문자열 + 문자열' 연산을 수행한다.

```
System.out.println(1 + "안녕");           // 1안녕
System.out.println(1 + "안녕" + 2);        // 1안녕2
System.out.println("안녕" + 1 + 2);        // 안녕12
System.out.println(1 + 2 + "안녕");        // 3안녕
```

2. String 클래스

■ String 객체의 '+' 연산

- 유형 ② '문자열 + 기본 자료형' 또는 '기본 자료형 + 문자열' 연산

- 실습. 문자열의 '+' 연산자

PlusOperationOfString.java

```
1 package sec02_string.EX04_PlusOperationOfString;
2
3 /*문자열의 '+' 연산자(plus operation)*/
4
5 public class PlusOperationOfString {
6     public static void main(String[] args) {
7         // #1. 문자열 + 문자열
8         String str1 = "안녕" + "하세요" + "!";
9         System.out.println(str1); //안녕하세요!
10
11         String str2 = "반갑";
12         str2 += "습니다";
13         str2 += "!";
```

2. String 클래스

■ String 객체의 '+' 연산

- 유형 ② '문자열 + 기본 자료형' 또는 '기본 자료형 + 문자열' 연산

- 실습. 문자열의 '+' 연산자

PlusOperationOfString.java

```
14      System.out.println(str2);          //반갑습니다!
15      System.out.println();
16
17      // #2. 문자열 + 기본자료형      // 기본자료형 + 문자열
18      String str3 = "안녕"+1;          //안녕1;
19      String str4 = "안녕"+String.valueOf(1); //안녕1
20      String str5 = "안녕"+"1";        //안녕1
21
22      System.out.println(str3);          //안녕1
23      System.out.println(str4);          //안녕1
24      System.out.println(str5);          //안녕1
25      System.out.println();
26
```


2. String 클래스

■ String 객체의 '+' 연산

- 유형 ② '문자열 + 기본 자료형' 또는 '기본 자료형 + 문자열' 연산

- 실습. 문자열의 '+' 연산자

PlusOperationOfString.java

```
27      // #3. 문자열과 기본자료형 혼용
28      System.out.println(1+"안녕");      // 1안녕
29      System.out.println(1+"안녕"+2);    // 1안녕2
30      System.out.println("안녕"+1+2);    // 안녕12
31      System.out.println(1+2+"안녕");    // 3안녕
32  }
33 }
```

2. String 클래스

■ 다음 표는 String 클래스 내의 대표적인 메서드들을 정리한 것이다.

메서드	설명
int length()	문자열의 길이를 반환
boolean equals(String str)	저장된 문자열과 str 문자열이 같은지 비교
boolean equalsIgnoreCase(String str)	대소문자 구분없이, 저장된 문자열과 str 문자열이 같은지 비교
String substring(int begin)	begin 위치부터 마지막까지의 문자열을 반환
String concat(String str)	저장된 문자열과 str 문자열을 결합
String replace(char old, char new)	문자열내의 old 문자를 new 문자로 변경
String toLowerCase()	문자열을 소문자로 변경
String toUpperCase()	문자열을 대문자로 변경
char charAt(int index)	index 위치의 문자를 반환
int indexOf(int ch)	저장된 첫번째 ch 문자의 위치를 반환
int lastIndexOf(int ch)	저장된 마지막 ch 문자의 위치를 반환
String trim()	문자열 끝의 공백문자를 제거

2. String 클래스

■ 실습. String 클래스의 주요 메서드 1

MethodsOfString_1.java

```
1 package sec02_string.EX05_MethodsOfString_1;
2
3 public class MethodsOfString_1 {
4     public static void main(String[] args) {
5         // #1. 문자열 길이 (length())
6         String str1 = "Hello Java!";
7         String str2 = "안녕하세요! 반갑습니다.";
8         System.out.println(str1.length());           // 11
9         System.out.println(str2.length());           // 13
10        System.out.println();
11
12        // #2. 문자열 검색 (charAt(), indexOf(), lastIndexOf())
13        // @charAt()
14        System.out.println(str1.charAt(1));           // e
15        System.out.println(str2.charAt(1));           // 녕
```

2. String 클래스

■ 실습. String 클래스의 주요 메서드 1

MethodsOfString_1.java

```
16      System.out.println();
17
18      //@indexOf(), lastIndexOf()
19      System.out.println(str1.indexOf('a'));      //7
20      System.out.println(str1.lastIndexOf('a'));  //9
21      System.out.println(str1.indexOf('a', 8));   //9
22      System.out.println(str1.lastIndexOf('a', 8)); //7
23      System.out.println(str1.indexOf("Java"));  //6
24      System.out.println(str1.lastIndexOf("Java")); //6
25      System.out.println(str2.indexOf("하세요")); //2
26      System.out.println(str2.lastIndexOf("하세요")); //2
27      System.out.println(str1.indexOf("Bye"));    //-1
28      System.out.println(str2.lastIndexOf("고맙습니다.")); //-1
29      System.out.println();
30
```

2. String 클래스

■ 실습. String 클래스의 주요 메서드 1

MethodsOfString_1.java

```
31      // #3. 문자열 변환 및 연결 (String.valueOf(), concat())
32      // @String.valueOf(기본자료형) 기본자료형 -> String
33      String str3 = String.valueOf(2.3);
34      String str4 = String.valueOf(false);
35      System.out.println(str3);
36      System.out.println(str4);
37
38      // @concat()
39      String str5 = str3.concat(str4);
40      System.out.println(str5);
41
42      // String.valueOf() + concat() => +
43      String str6 = "안녕" + 3; // 안녕3
44      String str7 = "안녕".concat(String.valueOf(3)); // 안녕3
45  }
```

2. String 클래스

■ 실습. String 클래스의 주요 메서드 2

MethodsOfString_2.java

```
1 package sec02_string.EX06_MethodsOfString_2;
2
3 public class MethodsOfString_2 {
4     public static void main(String[] args) {
5         //#4. 문자열 수정
6         //@toLowerCase(), toUpperCase()
7         String str1 = "Java Study";
8         System.out.println(str1.toLowerCase());    //java study
9         System.out.println(str1.toUpperCase());    //JAVA STUDY
10
11        //@ replace(.)
12        System.out.println(str1.replace("Study", "공부"));
13
14        //@ substring(.)
15        System.out.println(str1.substring(0,5)); //Java
```

2. String 클래스

■ 실습. String 클래스의 주요 메서드 2

MethodsOfString_2.java

```
16
17     //@ trim()
18     System.out.println("  abc  ".trim());
19     System.out.println();
20
21     //@ #5. 문자열의 내용 비교 (equals(), equalsIgnoreCase())
22     String str2 = new String("Java");
23     String str3 = new String("Java");
24     String str4 = new String("java");
25
26     //@ stack 메모리 비교 (==)
27     System.out.println(str2 == str3); //false
28     System.out.println(str3 == str4); //false
29     System.out.println(str4 == str2); //false
30
```

2. String 클래스

■ 실습. String 클래스의 주요 메서드 2

MethodsOfString_2.java

```
31      //@ equals(), equalsIgnoreCase() : 내용비교
32      System.out.println(str2.equals(str3)); //true
33      System.out.println(str3.equals(str4)); //false
34      System.out.println(str3.equalsIgnoreCase(str4)); //true
35  }
36 }
```


3. 키보드에서 입력받기

■ 실습 키보드 입력

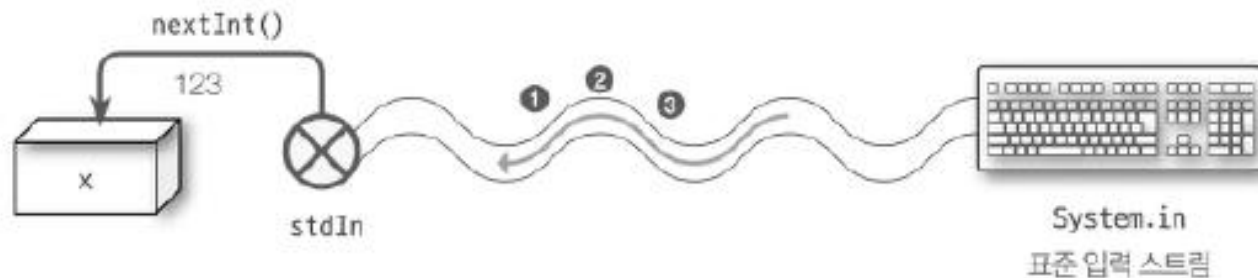
ScanInteger.java

```
1 // 키보드에서 입력한 정수값 표시
2 // 프로그램의 시작 부분(클래스 선언 이전)에 작성한다.
3 import java.util.Scanner;
4
5 class ScanInteger{
6     public static void main(String[] args){
7         // main 메서드의 시작 부분(값을 입력받는 코드 이전에)에 작성한다.
8         Scanner stdIn = new Scanner(System.in);
9
10        System.out.print("정수값: ");
11        // 키보드를 통해 입력한 정수값을 받는다.
12        int x = stdIn.nextInt();    // 입력받은 정수값을 x에 저장한다.
13        System.out.println(x + "를 입력했습니다.");
14    }
15 }
```

3. 키보드에서 입력받기

- 키보드로 값을 입력하려면 3행, 8행, 12행 순서를 따른다 (정해진 작성법이라고 생각하면 된다.
- 12행에서 System.in은 키보드와 연결된 표준 입력 스트림(standard input stream(STDIN))이다. (
- 키보드로 입력하는 정숫값은 int형으로 표현할 수 있는 범위(-2,147,483,648 ~ 2,147,483,647)여야 한다.
- 또한, 알파벳, 기호, 문자등을 입력해서는 안된다.

입력된 문자를 정수로 추출



3. 키보드에서 입력받기

- 키보드와 연결된 표준 입력 스트림인 `System.in`에서 문자나 숫자를 꺼내는 '추출 장치'가 `stdin`이다.
- `stdin`은 다른 이름으로 변경할 수 있다(변경하려면 프로그램 내의 모든 `stdin`을 해당 이름으로 변경해야 한다).
- 변수 `x`의 초기값은 12행에서 입력한 값을 사용하므로 변수 `x`는 키보드에서 입력한 정수값으로 초기화된다.
- 그런데 이 선언이 `main` 메서드의 시작부분이 아닌 중간에 있는 것에 주목하자. 변수는 필요할 때 해당 위치에 선언하는 것이 원칙이다.
- 다음과 같이 일단 변수를 선언하고 뒤에 `stdin.nextInt()` 값을 변수에 대입할 수도 있다(단, 처리가 중복된다).

```
int x;                // 변수 선언
x = stdin.nextInt();  // 변수에 값을 대입
```

3. 키보드에서 입력받기

■ next 메서드

- 키보드를 통해 값을 입력할 때는 변수의 자료형에 맞추어 메서드를 구분해서 사용한다.
- 표는 각 자료형에 맞는 메서드를 보여준다.

메서드	자료형	입력받을 수 있는 값
nextBoolean()	boolean	true 또는 false
nextByte()	byte	-128 ~ +127
nextShort()	short	-32768 ~ +32767
nextInt()	int	-2147483648 ~ +2147483646
nextLong()	long	-9223372036854775808 ~ +9223372036854775807
nextFloat()	float	$\pm 3.40282347E+38 \sim \pm 1.40239846E-45$
nextDouble()	double	$1.79769313486231507E+378 \sim \pm 4.94065645841246544E-324$
next()	String	문자열(스페이스, 줄 바꿈 등으로 구분 가능)
nextLine()	String	한 줄의 문자열

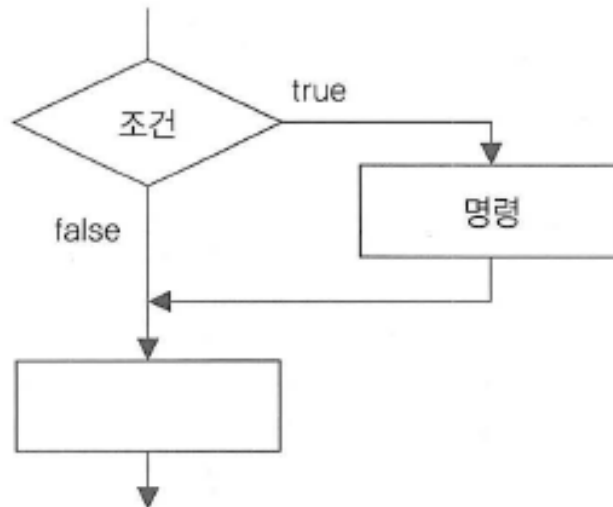
3. 키보드에서 입력받기

- 다음 예제는 키보드로 입력한 성과 이름으로 인사하는 프로그램이다.

```
1  import java.util.Scanner;
2
3  public class Greeting {
4      public static void main(String[] args) {
5          Scanner stdIn = new Scanner(System.in);
6
7          System.out.print("성: ");
8          String lastName = stdIn.next();
9          System.out.print("이름: ");
10         String firstName = stdIn.next();
11         System.out.println("안녕하세요. " + lastName + firstName + " 씨.");
12     }
13 }
```

4. if문

- 프롤로그나 리스프와 같은 특수한 언어가 아닌 이상, 우리가 알고 있는 대부분의 컴퓨터 언어들은 왼쪽에서 오른쪽으로, 위에서 아래로 프로그램을 실행한다.
- 우리가 지금까지 만든 모든 프로그램들도 이 원칙을 지켜서 순서대로 실행되었다.
- 그런데 어떤 때는 이 순서를 바꾸고 싶을 때가 있다.
- 예를 들어 입력된 값이 홀수이면 ㉠작업을 하고, 짝수이면 ㉡작업을 하고 싶을 때처럼.



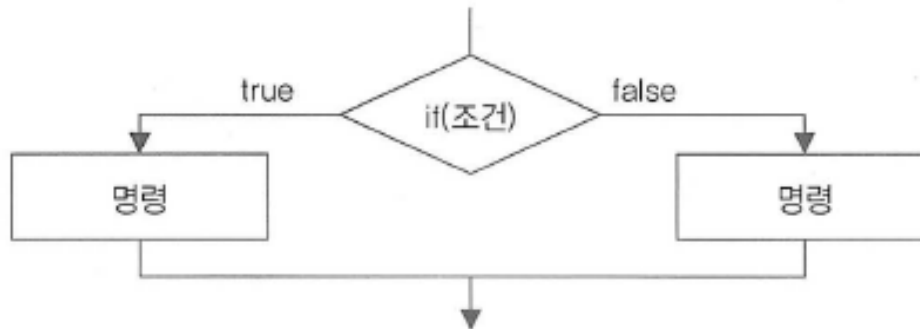
4. if문

- 이렇게 조건에 따라 처리할 명령을 달리하고 싶을 때 사용하는 것이 분기문이다.
- 자바에서는 if문과 switch문의 두 가지 분기문을 준비해두고 있다.
- if문은 true나 false로 결정지을 수 있는 단순한 조건에서 많이 사용되고, switch 문은 다양한 값을 가지는 조건에 많이 사용된다.
- 따라서 if문의 조건에는 boolean형 변수가 사용되고, switch문의 조건에는 int형의 변수를 사용한다.

4. if문

- if문은 다음과 같은 형식을 가진다.
- if문과 else문 사이의 명령어들은 조건이 true일 때 실행되는 명령들이고, else문 다음의 명령어들은 조건이 false일 때 실행되는 명령들이다.
- 만약 false일 때 실행시키고 싶은 명령이 없을 때는 else 이하 부분은 생략할 수 있다.

```
if (조건) {  
    명령어;    → 조건이 true일 때 실행될 명령  
}else{  
    명령어;    → 조건이 false일 때 실행될 명령  
}
```



4. if문

- 주의할 점은 if문의 조건부분에는 반드시 true나 false가 들어가야 한다는 점이다
- 바꿔 말하면 boolean형 변수 또는 boolean형 값을 돌려주는 연산이 들어가야만 한다.
- 만약 int형이나 char형 등을 사용하면 에러가 발생한다.
- 다음은 boolean형 변수인 condition의 값을 출력하는 프로그램이다.
- Condition이 true이면 참, false이면 거짓을 출력한다.
- 이 예제에서는 if문 앞에서 condition에 true 값을 저장하기 때문에, 참이라고 나온다.

4. if문

```
1 public class IfTest {
2     public static void main(String[] args) {
3         boolean condition = true;
4
5         if(condition) {
6             // 조건(condition 변수)이 참이면 실행되는 부분
7             System.out.println("조건은 참!");
8         }else {
9             // 조건(condition 변수)이 거짓이면 실행되는 부분
10            System.out.println("조건은 거짓!");
11        }
12    }
13 }
```

4. if문

- if 문에서 중괄호({ 와 })를 생략할 수 있을까?
- 만일 IfTest 예제에서 if문의 중괄호를 생략하면 어떻게 될까?
 - 자바에서는 처리하고 싶은 여러 개의 명령어들을 중괄호({ 와 })로 묶어서 하나의 명령어처럼 다루는데, 이것을 블록이라고 한다.
 - 원칙적으로 if문을 포함한 자바의 제어문들(if, switch, while, do-while, for 등)은 1개의 명령어만 처리할 수 있기 때문에 하나 이상의 명령어를 처리하고 싶을 때는 반드시 블록으로 만들어야 한다.
 - 예를 들어, 다음처럼 if문을 만들면, ①만 조건(condition 변수)에 따라 출력되거나(true일 때) 출력되지 않고(false일 때), ②는 if문과 상관없이 항상 실행된다.

```
if (condition)
```

```
    System.out.println("①");    // if문의 영향을 받는 명령
```

```
    System.out.println("②");    // if문과는 상관없는 명령
```

4. if문

- if 문에서 중괄호({ 와 })를 생략할 수 있을까?
- 만일 IfTest 예제에서 if문의 중괄호를 생략하면 어떻게 될까?
 - ②가 if문의 영향을 받는 것처럼 보이는 이유는 단지 ②를 출력하는 명령 (System.out.println ("②") ;)을 ①을 출력하는 명령 (System.out.println("①");)처럼 들여쓰기를 하여 줄을 맞췄기 때문이다.
 - 줄 맞추기는 프로그램의 가독성을 높이기 때문에 (사람에게는) 매우 중요하지만, 자바 컴파일러는 전혀 고려하지 않기 때문에, ②는 if문의 영향을 받지 않는 것이다.
 - 이 프로그램에서 ①과 ②가 모두 if문의 조건에 따라 출력되도록 하려면 다음처럼 고치면 된다.

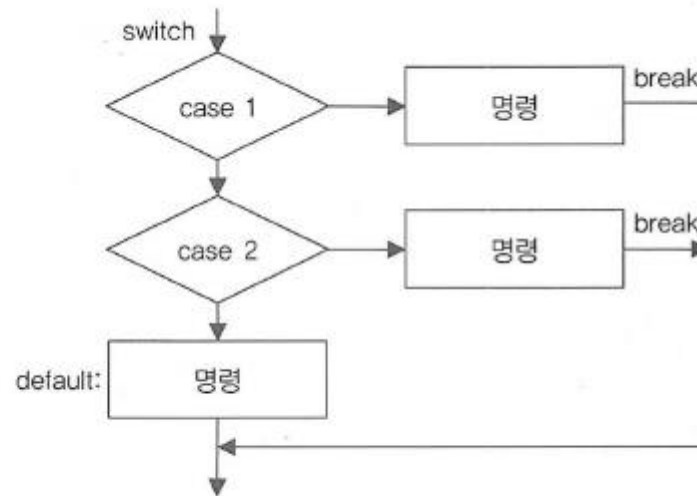
```
if (condition){  
    System.out.println("①");  
    System.out.println("②");  
}
```

- IfTest 예제에서는 if문과 else문의 영향을 받는 명령어가 각각 1개뿐이기 때문에 중괄호를 생략해도 결과는 같다.
- 그러나 if문 또는 else문의 영향을 받는 명령어가 2개 이상이 될 때는 반드시 중괄호({ 와 })를 사용하여야 한다.
- 이 점은 자바의 다른 제어문인 switch, for, while, do-while 등에도 마찬가지이다.

5. Switch 문

- if문은 편리하지만 true와 false의 2가지 조건 밖에 없기 때문에 다양한 조건에 따라 일을 처리하고 싶을 때는 불편하다.
- Switch문은 표현식의 결과로 나온 값에 따라 일을 처리할 때 편리하다.
- Switch문은 다음과 같은 형식을 갖는데, 표현식의 결과에 해당하는 case 다음의 명령어가 실행되고, 해당되는 case가 없으면 default 다음의 명령어가 실행된다.

```
switch(표현식) {  
    case 결과1:  
        명령어;  
        break;  
    case 결과2:  
        명령어;  
        break;  
    default:  
        명령어;  
        break;  
}
```



5. Switch 문

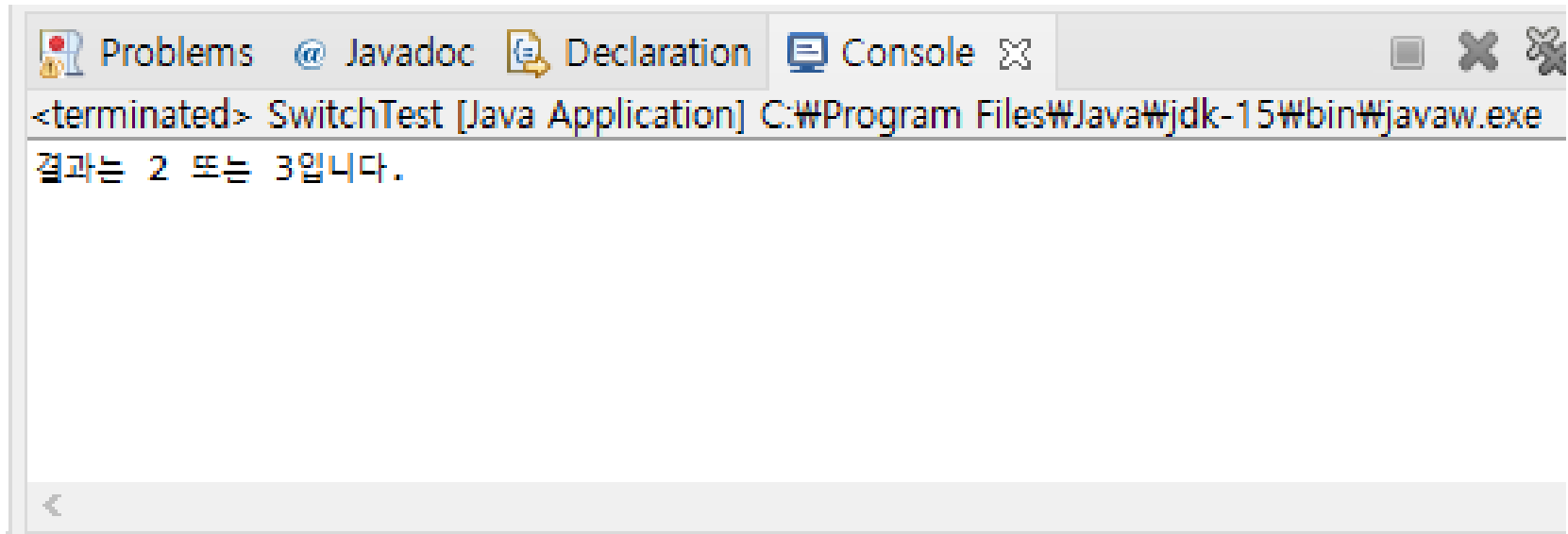
- Switch문에서 주의할 점은 case 뒤에 실행할 명령어들을 쓰고 break를 이용하여 switch문을 끝내야 한다는 점이다.
- break를 생략하면 다음 case 뒤의 명령들까지 이어서 실행된다.
- 여러 개의 조건에 따르는 명령어를 만들고 싶을 때는 일부러 break를 생략할 수도 있다.
- 다음의 예제에서 num값이 2일 때는 case 2: 뒤에 break가 없기 때문에 case 3: 다음에서 break를 만날 때까지 실행하게 된다.

5. Switch 문

```
1 public class SwitchTest {
2     public static void main(String[] args) {
3         int num = 2;
4
5         switch(num){
6             case 1:
7                 System.out.println("결과는 1입니다.");
8                 break;
9             case 2:    // break가 없어서 다음 명령까지 연이어 실행
10            case 3:
11                System.out.println("결과는 2 또는 3입니다.");
12                break;
13            default: // 해당되는 case가 없을 때 실행되는 부분
14                System.out.println("결과는 1, 2, 3이 아닙니다.");
15                break;
16        }
17    }
18 }
```

5. Switch 문

- 이 예제는 num 변수에 2를 저장했기 때문에 case 2: 부터 break를 만날 때까지 모두 실행된다.
- 따라서, num 변수에 3을 저장해도 결과는 같다.



The screenshot shows a console window from an IDE. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text reads: '<terminated> SwitchTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe' followed by the Korean text '결과는 2 또는 3입니다.' (The result is 2 or 3).

```
<terminated> SwitchTest [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe  
결과는 2 또는 3입니다.
```


5. Switch 문

- switch문을 사용하면 표현식의 결과 값에 따라 다양한 처리를 할 수 있어서 편리하지만, if문을 여러 개 사용하면 비슷한 결과를 얻을 수 있을 것 같다.
- switch문을 if문으로 나타낼 수 있을까?
 - 모든 switch문은 여러 개의 if문으로 바꿀 수 있다.
 - 특히 여러 개의 조건을 처리하는 if문을 다중 if문이라고 한다.
 - 예를 들어 위 예제의 switch문은 다음과 같은 다중 if문으로 바꿀 수 있다.
 - 그러나 모든 if문을 switch문으로 바꿀 수 있는 것은 아니다.

```
...
if (num == 1) {
    System.out.println("결과는 1입니다.");
}else if ((num == 2) || (num == 3)) {
    System.out.println ("결과는 2 또는 3 입니다.");
}else{
    System.out.println("결과는 1, 2, 3이 아닙니다.");
}
...
```

6. 가위, 바위, 보 게임 만들기

- 이제 지금까지 배운 것들을 종합해서 가위, 바위, 보 게임을 만들어보자.
- 우선 컴퓨터가 가위, 바위, 보 중 하나를 골라야 하는데, 이 부분은 Random 클래스를 이용해서 0 ~ 2 중 하나를 구해서 해결한다.
- 0은 가위, 1은 바위, 2는 보라고 생각하자.
- 게임을 하는 플레이어에게 가위, 바위, 보 중 하나를 고르라는 메시지를 내보내고, 앞에서 배운 대로 키보드로부터 문자를 읽어 플레이어의 선택을 알아낸다.
- 이 때 플레이어가 a를 입력하면 가위, b를 입력하면 바위, c를 입력하면 보라고 정하자.
- 컴퓨터가 정한 0 ~ 2 중 하나와 플레이어가 정한 a ~ c 중 하나의 상관관계를 if 문으로 구해서 승부의 결과를 출력한다.

6. 가위, 바위, 보 게임 만들기

■ 다음은 전체 게임의 소스코드 이다.

```
1 import java.util.*;
2 import java.util.Scanner;
3
4 public class GameJave2_03 {
5     public static void main(String[] args) throws IOException {
6         // 0~2 사이의 난수를 구한다.
7         Random r = new Random();
8         // 난수를 구해서 3으로 나눈 나머지
9         int computer = Math.abs(r.nextInt() % 3);
10        Scanner stdIn = new Scanner(System.in);
11
12        String user;
13        System.out.print("가위, 바위, 보 중 하나를 선택하세요. (가위=a, 바위=b, 보
=c): ");
```

6. 가위, 바위, 보 게임 만들기

```
14      // 키보드로부터 한 줄을 입력받음
15      user = stdIn.nextLine();
16      if(user.equals("a")) {      // 가위를 선택한 경우
17          if(computer == 0) System.out.println("무승부 (컴퓨터:가위, 사람:가위)");
18          if(computer == 1) System.out.println("컴퓨터 승! (컴퓨터:바위, 사람:가
위)");
19          if(computer == 2) System.out.println("사람 승! (컴퓨터:보, 사람:가위)");
20      }else if(user.equals("b")) {      // 바위를 선택한 경우
21          if(computer == 0) System.out.println("사람 승! (컴퓨터:가위, 사람:바위)");
22          if(computer == 1) System.out.println("무승부 (컴퓨터:바위, 사람: 바위)");
23          if(computer == 2) System.out.println("컴퓨터 승! (컴퓨터:보, 사람:바위)");
24      }else if(user.equals("c")) {      // 보를 선택한 경우
25          if(computer == 0) System.out.println("컴퓨터 승! (컴퓨터:가위, 사람:보)");
26          if(computer == 1) System.out.println("사람 승! (컴퓨터:바위, 사람:보)");
27          if(computer == 2) System.out.println("무승부 (컴퓨터:보, 사람:보)");
28      }
29  }
30 }
```



Thank You
