제1장 : 문자열 조작

후반전 목차

- 문자열 조작
- 컬렉션
- 인스턴스 기본 조작
- 제네릭, 열거형, 이너클래스
- 람다식과 함수
- 파일 조작
- 여러가지 파일 형식
- 네트워크 통신
- 데이터베이스 접속
- 스레드를 사용한 병렬처리

“Hello" + “ Java”

=> “Hello Java”

```
String s1 = "Hello";
String s2 = " Java"

System.out.print(s1 + s2);
```

“HELLO”

=> “HE”

```
String s1 = "HELLO";

System.out.println(s1.substring(0, 2));
```

“HELLO”

=> “HEXXO”

```
String s1 = "HELLO";

System.out.println(s1.replace("LL", "XX"));
```

“1,2,3”

=> “1”, “2”, “3”

```
String s1 = "1,2,3";

String splited[] = s1.split(",");

for (String s : splited) {
  System.out.println(s);
}
```

“HELLO”

=> “hello”

```
String s1 = "HELLO";

System.out.println(s1.toLowerCase());
```

“HELLO”

=> E 는 2번째 글자

```
String s1 = "HELLO";

System.out.println(s1.indexOf('E'));
```

문자열 처리 (내용 비교)


```
String s1 = "Java";
String s2 = "java";

if (s1.equals(s2)) {
  System.out.println("s1과 s2는 같다");
}

if (s1.equalsIgnoreCase(s2)) {
  System.out.println("s1과 s2는 대소문자 무시하면 같다");
}
```

문자열 처리 (길이)


```
String s1 = "Java";

System.out.println(s1.length());  // 4

System.out.println(s1.isEmpty());  // false
```

length() : 길이
isEmpty() : 길이가 0인지

문자열 처리 (검색)


```
String s1 = "Java and JavaScript";

System.out.println(s1.contains("Java"));  // true
System.out.println(s1.endsWith("Java"));  // false
System.out.println(s1.indexOf("Java"));  // 0
System.out.println(s1.lastIndexOf("Java"));  // 9
```

contains() : 포함 관계
endsWith() : 끝나는 단어가 맞는지
indexOf() : 단어가 몇 번째에 있는지
lastIndexOf() : 뒤에서 몇 번째에 단어가 있는지

```
String s1 = "Java and JavaScript";

System.out.println(s1.toLowerCase());    // 소문자로
System.out.println(s1.toUpperCase());    // 대문자로
System.out.println(s1.trim());  // 좌우 공백 제거
System.out.println(s1.replace("and", "or"));  // 교체
```

문자열 결합 방법

1) + 연산
2) StringBuilder
3) StringBuffer

append() 메서드로 결합한 결과를 내부 메모리(버퍼)에 담아 두고 toString()으로 결과를 얻음

```
StringBuilder sb = new StringBuilder();
for (int i = 0; i < 10000; i++) {
  sb.append("Java");  // 결합
}
String s = sb.toString();   // 완성된 결과
```

메서드 체인

append() 처럼 자기 자신의 참조를 리턴하여 연속해서 메서드를 호출하는 기법

```
@Override
public StringBuilder append(Object obj)
```

```
StringBuilder sb = new StringBuilder();
sb.append("Hello").append("Java").append("World");
```

String 인스턴스는 불변 객체 (immutable)

퀴즈 1

i 번째 글자가 모음인지 알려주는 isVowel() 메서드를 완성하시오

```java
public class Word {
    private String letters;

    public Word(String letters) {
        this.letters = letters;
    }

    /**
     * i번째 글자가 모음인지?
     */
    public boolean isVowel(int i) {
        return letters.substring(i, i + 1).equals("a");
    }
```

퀴즈 2

i 번째 글자가 자음인지 알려주는 isConsonant() 메서드를 완성하시오

```
public class Word {
    private String letters;

    public Word(String letters) {
        this.letters = letters;
    }

    /**
     * i번째 글자가 모음인지?
     */
    public boolean isVowel(int i) {
        return letters.substring(i, i + 1).equals("a");
    }

    /**
     * i번째 글자가 자음인지?
     */
    public boolean isConsonant(int i) {
        return false;
    }
}
```

정규표현식 (Regular Expression)

https://regexr.com/

정규 표현식

위키백과, 우리 모두의 백과사전.

정규 표현식(正規表現式, 영어: regular expression, 간단히 regexp[1] 또는 regex, rational expression)[2][3] 또는 **정규식**(正規式)은 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어이다. 정규 표현식은 많은 텍스트 편집기와 프로그래밍 언어에서 문자열의 검색과 치환을 위해 지원하고 있으며, 특히 펄과 Tcl은 언어 자체에 강력한 정규 표현식을 구현하고 있다.

일반적인 조건에 따른 처리

첫 글자가 영어로 시작하는 A~Z, 0~9 사이의 8자리 문자

```
boolean isValidPlayerName(String name) {
  if (name.length != 8) {
    return false;
  }
  char first = name.charAt(0);
  if (!(first >= 'A' && first <= 'Z')) {
    return false;
  }
  for (int i = 1; i < 8; i++) {
    char c = name.charAt(i);
    if (!((c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))) {
      return false;
    }
  }
  return true;
}
```

정규표현식 활용 예

첫 글자가 영어로 시작하는 A~Z, 0~9 사이의 8자리 문자

```
boolean isValidPlayerName(String name) {
  return name.matches("[A-Z][A-Z0-9]{7}");
}
```

1) 문자 : 그 문자가 아니면 false

```
String s = "Java";
s.matches("Java");  // true
s.matches("JavaJava");  // false
s.matches("java");  // false
```

2) 점(.) : 임의의 한 문자

```
"Java".matches("J.va"); // true
```

3) * : 앞 문자 0회 이상 반복

```
"Jaaaaava".matches("Ja*va");     // true
"abcd한글011".matches(".*");     // true

"Java".matches("Ja.*");    // Ja로 시작하는 문자열
"Java".matches(".*va");    // va로 끝나는 문자열
```

4) { } : 지정횟수만큼 반복

패턴	의미
{n}	앞 문자가 n번 반복
{n,}	앞 문자 n번 이상 반복
{n,m}	앞 문자 n번 이상 m번 이하 반복
?	앞 문자 0번 또는 1번
+	앞 문자 1번 이상 반복

5) [ ] : 어느 한 문자

예) UR[LIN] : 첫번째 문자는 U, 두 번째 문자는 R, 세 번째 문자는 'L, I, N' 중 하나

6) [-] : 지정 범위 중 한 문자

```
"url".matches("[a-z]{3}");  // true
```

7) ^ : 맨 앞,
$: 맨 뒤

예) “^j.*p$”
=> 맨 앞 글자가 j, 맨 뒷 문자가 p 인 문자열

split() 메서드 : 문자열 분기

```
class Main {
  public static void main(String[] args) {
    String s = "abc,def:ghi";
    String[] words = s.split("[,:]");
    for (String word : words) {
      System.out.println(word + "->");
    }
  }
}
```

replaceAll() 메서드 : 문자열 교체

```
class Main {
  public static void main(String[] args) {
    String s = "abc,def:ghi";
    String[] words = s.replaceAll("[beh]", "X");
    System.out.println(words);  // aXc,dXf:gXi
  }
}
```

String.format("%s 입문 %d일 코스 %s", "Java", 14, "후반전");

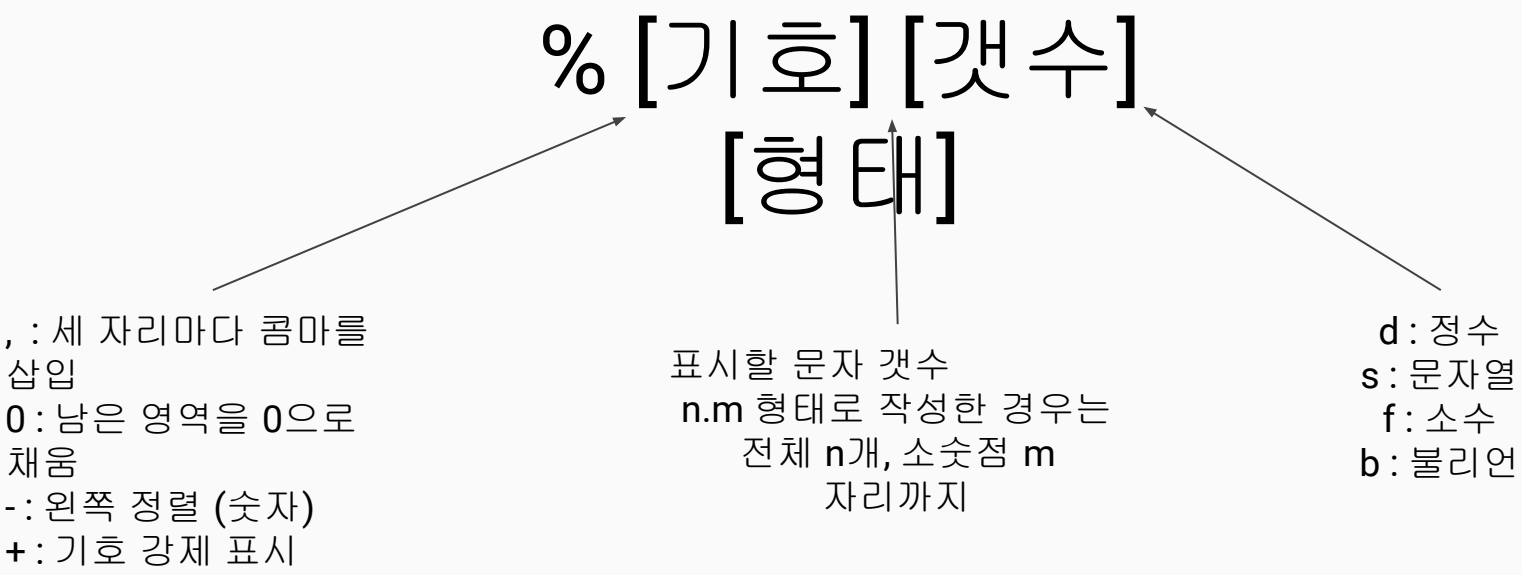
=> Java 입문 14일 코스 후반전

```
public static String format(String format,
                            Object... args)
```

Returns a formatted string using the specified format string and arguments.

The locale always used is the one returned by `Locale.getDefault()`.

문자열 포맷




```
System.out.printf("제품번호%s-%02d", "ABC", 3);    // 제품번호ABC-03
```

연습문제 1-1

1부터 100까지의 정수를 콤마로 연결한 다음과 같은 문자열 str 을 생성하는 코드를 작성하시오

1,2,3,4,5 … 98,99,100,

완성된 문자열 str 을 콤마를 구분자로 하여 String 배열 a 에 대입하시오.

연습문제 1-2

폴더명과 파일명을 매개변수로 받아서 연결한 파일 주소를 리턴하는 메서드를 작성하시오.
폴더명의 경우 마지막에 \ 기호가 붙어있는 경우도 있고 아닌 경우도 있으니 두 경우 모두 처리할 수 있도록 하시오.

예) 폴더명 - C:\dev 또는 C:\dev\, 파일명 - abc.txt 일 경우 => C:\dev\abc.txt

선언 예) String getPath(String path, String fileName)

사용 예) getPath(“C:\dev”, “abc.txt”)

연습문제 1-3

다음 조건에 맞는 정규표현식을 작성하시오.

1) 모든 문자열
2) 최초 첫번째 문자는 A, 두 번째 문자는 숫자, 세 번째 문자는 숫자이거나 아무것도 없거나
3) 최초 첫번째 문자는 U, 2~4번째 문자는 영어 대문자