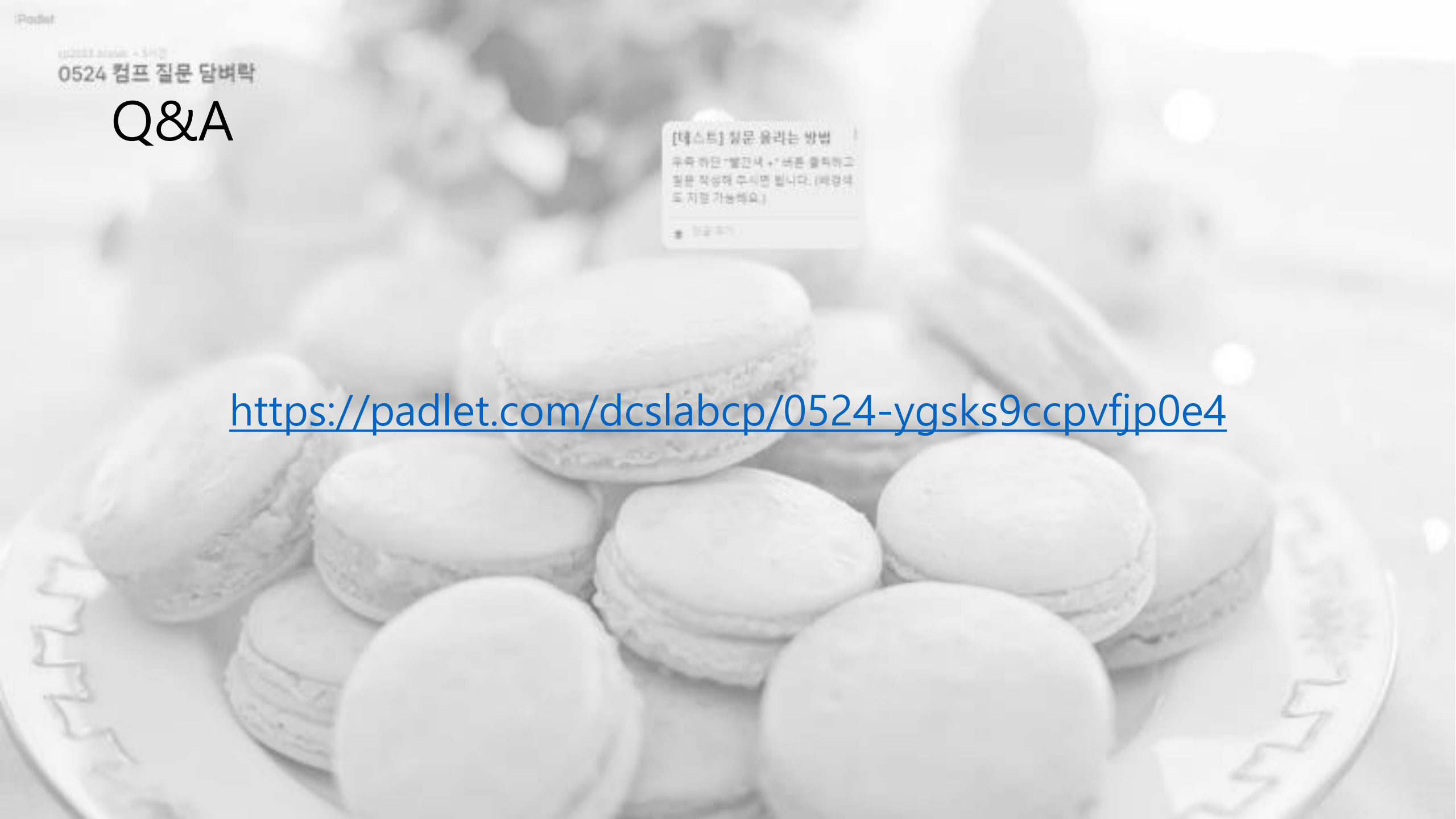
Lab 11



Q&A

https://padlet.com/dcslabcp/0524-ygsks9ccpvfjp0e4

오늘 수업 내용

- String class
- Regular expression
- File I/O
- Nested class
- Introduction of Project 2
  (with CP2023_Project2.pdf)

String class

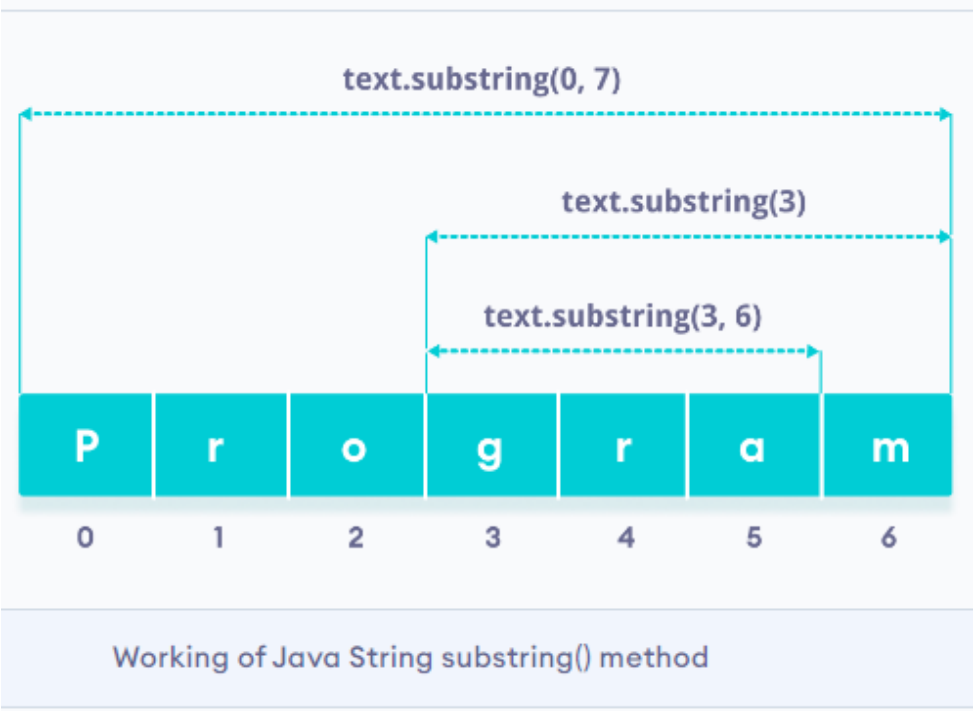
- 개발 상황에서 문자열을 효과적으로 다뤄야 하는 수요는 곧잘 발생
- 이에 주요 String 메소드를 어떤 상황에서 어떻게 활용하는지 연습 진행

- String 주요 메소드 리스트
  - substring(), indexOf(), split()
  - length(), toLowerCase(), trim(), equals(), replace(), matches()

- 추천 학습자료
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html
  - https://www.w3schools.com/java/java_ref_string.asp
  - https://www.programiz.com/java-programming/library/string

substring() method

- 문자열에서 부분 문자열을 얻고자 할 때 활용
- 파라미터로 startIndex 한 개 또는 startIndex, endIndex 두 개 명시 가능



Working of Java String substring() method

```java
public class Main {
    public static void main(String[] args) {
        String str = "Program";
        System.out.println(str.substring(3));  // gram
        System.out.println(str.substring(3, 5));  // gr
    }
}
```

substring() method

5분

연습문제

- tomato 문자열에서 앞의 to와 뒤의 to를 각각 별도 변수에 저장 및 출력해보자

substring() method

연습문제 정답

- tomato 문자열에서 앞의 to와 뒤의 to를 각각 별도 변수에 저장 및 출력해보자

```
public class Main {
    public static void main(String[] args) {
        String str = "tomato";
        String str1 = str.substring(0, 2);
        String str2 = str.substring(4, 6);
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str1 == str2);
    }
}
```

```
to
to
```

substring() method

연습문제 정답

- tomato 문자열에서 앞의 to와 뒤의 to를 각각 별도 변수에 저장 및 출력해보자

```java
public class Main {
    public static void main(String[] args) {
        String str = "tomato";
        String str1 = str.substring(0, 2);
        String str2 = str.substring(4, 6);
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str1 == str2);
    }
}
```

```
to
to
false
```

객체는 주소값 비교!

=> str1.equals(str2)

Java 언어 내 Primitive types

(Stack에 저장, 값 비교 가능)

- byte
- short
- int
- long
- float
- double
- char
- boolean

indexOf() method

- 문자열 내 특정 문자열이 포함되어 있을 때 해당 위치(index)를 얻을 때 활용
- 특정 문자열이 2개 이상 있는 경우, 가장 처음 발견되는 것의 위치를 리턴
- 특정 문자열이 포함되어 있지 않으면 -1을 리턴

```java
public class Main {
    public static void main(String[] args) {
        String str = "Hello World!";
        int result = str.indexOf('l');
        System.out.println(result);  // 2
    }
}
```

indexOf() method

5분

연습문제

- 앞의 예제 코드에서 가장 뒤에 위치한 I의 index를 얻도록 코드를 수정해보자 (answer: 9)

indexOf() method

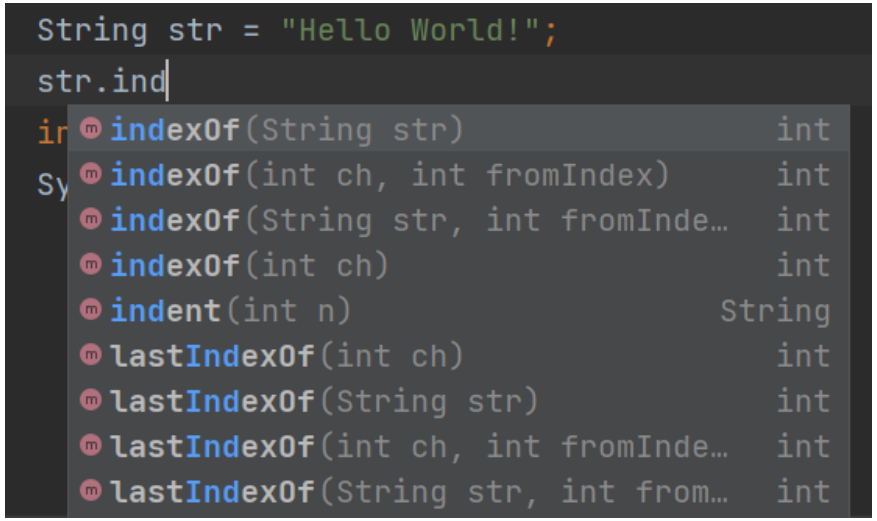
연습문제 정답

- 앞의 예제 코드에서 가장 뒤에 위치한 l의 index를 얻도록 수정해보자 (answer: 9)

```java
public class Main {
    public static void main(String[] args) {
        String str = "Hello World!";
        int result = str.lastIndexOf("l");
        System.out.println(result);
    }
}
```

```
9
```

- Java Docs를 가까이하자
- IntelliJ IDE 기능 적극 활용



split() method

- 특정 문자열을 구분자(delimiter)로 해서 전체 문자열을 분할할 때 활용
- 배열 타입으로 리턴 받으며, 구분자로 regular expression 명시 가능

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        String ip_addr = "147.46.100.100";
        String[] result = ip_addr.split("\\.");
        System.out.println("result = " + Arrays.toString(result));
    }
}
```

```
result = [147, 46, 100, 100]
```

split() method

5분

연습문제

- 아래 샘플 문자열에 대한 ArrayList 출력 결과가 나오는 코드를 작성해보자

- 샘플 문자열:
```
String str = "IP address ?147.46.100.100";
```

- 출력 결과:
```
[IP address, 147, 46, 100, 100]
```

split() method

연습문제 정답

```java
public class Main {
    public static void main(String[] args) {
        String str = "IP address ?147.46.100.100";
        // String[] result = str.split("[?.]");
        String[] result = str.split("\s\\?|\\.");
        ArrayList list = new ArrayList(Arrays.asList(result));
        System.out.println(list);
    }
}
```

```
[IP address, 147, 46, 100, 100]
```

Other methods in String class

- str.**length**() : 문자열의 길이를 정수로 리턴
- str.**toLowerCase**() : 문자열의 모든 문자를 소문자로 변경
- str.**trim**() : 문자열 시작과 끝의 공백 삭제
- str.**equals**(str2) : 두 문자열이 동일한지 검사하여 true/false 리턴
- str.**replace**(oldstr, newstr): 문자열 내 oldstr을 newstr로 변경 (* replaceAll()은 regex 지원)
- str.**matches**(regex) : 문자열이 주어진 regex와 일치 여부 검사 및 true/false 리턴

```java
public class Main {
    public static void main(String[] args) {
        String str1 = "  Namsan Tower ";
        String str2 = "namsan tower";
        System.out.println(str1.trim().toLowerCase().equals(str2));
        System.out.println(str1.replace("\s", "").matches("[a-zA-Z]*"));
        System.out.println(str1);
```

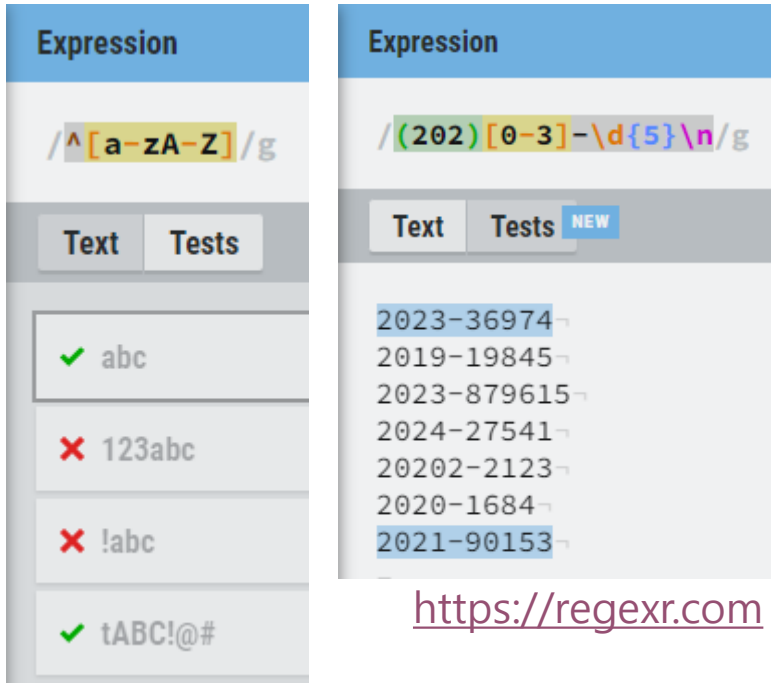
```
true
true
  Namsan Tower
```

Regular Expression

- 프로그래밍에서 문자열의 일정한 패턴을 표현하는 일종의 형식 언어
- 활용 예시
  1. 특정 문자열과 패턴과의 일치 여부 검증 (예시: 주민등록번호, 이메일, 전화번호, 우편번호)

     => 2020년 또는 그 이후에 입학한 학생의 학번을 정규표현식을 통해 검증

     => 영문자로 시작하는 문자열 여부를 정규표현식을 통해 검증
  2. 긴 문자열에서 특정 패턴을 만족하는 부분 문자열 탐색/치환



https://regexr.com

- 참고 사이트
  - https://namu.wiki/w/정규%20표현식
  - https://bit.ly/3os13hA

Regular Expression

- 정규 표현식 예제
  - 2019년 또는 그 전에 입학한 학생의 학번 필터링

```
String str = "2019-12121";  // 2019년 또는 그 전에 입학한 학생의 학번
System.out.println(str.matches("(20)[0-1]\\d{1}-\\d{5}"));
```

  - 이메일 필터링
    (Pattern 객체 활용)

```
import java.util.regex.Pattern;

public class Main {
    public static boolean isEmail(String str) {
        return Pattern.matches("[0-9a-zA-Z]+(.[_a-z0-9-]+)*@(?:\\w+\\.)+\\w+$", str);
    }
    public static void main(String[] args) {
        System.out.println(isEmail("test@snu.ac.kr"));
    }
}
```

Regular Expression

5분

연습문제

- 휴대폰 번호 패턴과 매치하면 true를 리턴하는 메소드 만들기

  * String.matches(regex) 메소드 활용

```
public class Main {
    public static void main(String[] args) {
        String str = "010-1234-5678";
        System.out.println(str.matches("                              "));
    }
}
```

Regular Expression

연습문제 정답

- 휴대폰 번호 패턴과 매치하면 true를 리턴하는 메소드 만들기

  * String.matches(regex) 메소드 활용

```
public class Main {
    public static void main(String[] args) {
        String str = "010-1234-5678";
        System.out.println(str.matches("^01[016789]-\\d{4}-\\d{4}$"));
    }
}
```

File I/O

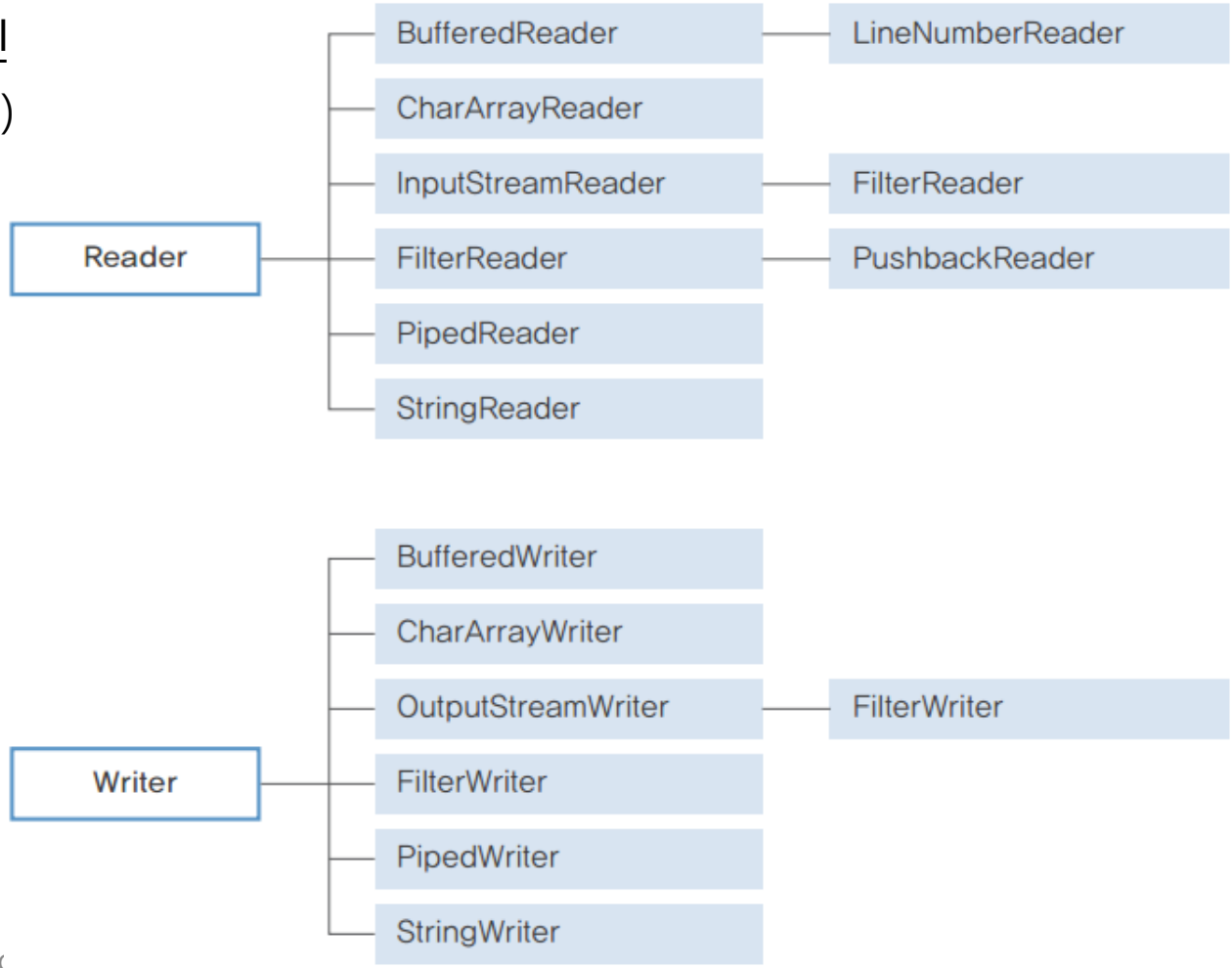
* 오랫동안 Java의 약점이었던 I/O 성능

- 주요 개념
  - Reader: 입력(Input) 스트림을 받는 모듈 (프로그램 관점)
  - Writer: 출력(Output) 스트림을 보내는 모듈 (프로그램 관점)
  - Stream (스트림): 연속된 데이터의 단방향 흐름을 추상화 (키보드/모니터 입출력, 파일, 네트워크 등)

File I/O

- 문자 스트림
  - 데이터를 2바이트 단위인 유니코드로 송수신
    (바이너리 데이터는 바이트 스트림으로 지원)
  - 스트림의 효율을 높이기 위해 버퍼 사용

* Reader, Writer는 Abstract Class

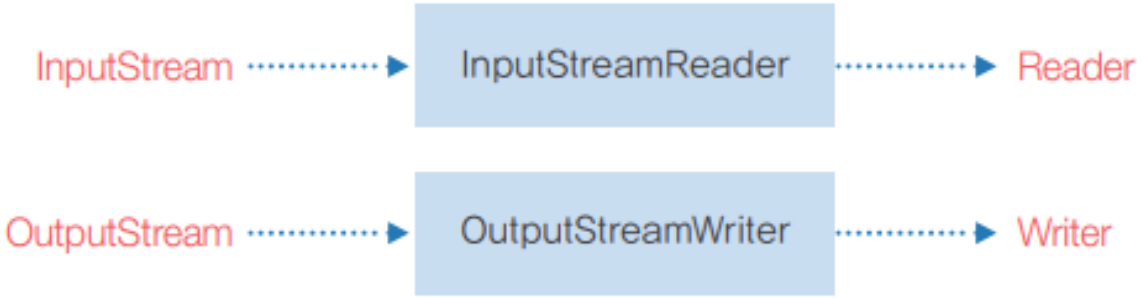


- BufferedReader 주요 메소드

메서드	설명
Stream<String> lines( )	읽은 행을 스트림으로 반환한다.
String readLine( )	한 행을 읽어 문자열로 반환한다.

File I/O

- InputSteramReader, OutputStreamWriter 클래스
  (바이트 스트림을 문자 기반의 Reader, Writer로 변환)




```
InputStreamReader(InputStream in)
InputStreamReader(InputStream in, Charset cs)
OutputStreamWriter(OutputStream out)
OutputStreamWriter(OutputStream out, Charset cs)
```

- PrintWriter 클래스
  (PrintStream과 같이 다양한 데이터 값을 편리하게 표현 가능 (Writer의 Subclass)

```
PrintWriter(File file)
PrintWriter(String filename)
PrintWriter(OutputStream out)
PrintWriter(OutputStream out, boolean autoFlush)
PrintWriter(Writer out)
PrintWriter(Writer out, boolean autoFlush)
```

File I/O

- Index.html 파일의 복사본 만들기

```java
import java.io.*;

public class Main {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader("src/index.html"));
            PrintWriter writer = new PrintWriter(new FileWriter("src/index2.html"));
            // reader.lines().forEach(x -> writer.println(x));
            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line + "\n");
            }
        } catch (IOException e) {
        }
    }
}
```

File I/O

10분

연습문제

- String 배열에 가수나 걸그룹 이름 5개를 넣어두고
  이를 한 줄씩 db/actors.txt 파일에 쓰는 코드를 작성해보자

File I/O

8분

연습문제

- String 배열에 가수나 걸그룹 이름 5개를 넣어두고
  이를 한 줄씩 db/actors.txt 파일에 쓰는 코드를 작성해보자

Hint

- 코드로 정의된 문자열을 읽어 들이면 되기에, BufferedReader는 필요하지 않음 => writer.println(singer);

File I/O

5분

연습문제

- String 배열에 가수나 걸그룹 이름 5개를 넣어두고
  이를 한 줄씩 db/actors.txt 파일에 쓰는 코드를 작성해보자

Hint

- 코드로 정의된 문자열을 읽어 들이면 되기에, BufferedReader는 필요하지 않음 => writer.println(singer);
- 우선 src 디렉토리에 문자열 쓰기를 반영하려면 writer.flush(); 실행 필요 (보통은 객체 소멸 시 자동 실행)

File I/O

3분

연습문제

- String 배열에 가수나 걸그룹 이름 5개를 넣어두고
  이를 한 줄씩 db/actors.txt 파일에 쓰는 코드를 작성해보자

Hint

단계적으로 문제를 해결해보자!

- 코드로 정의된 문자열을 읽어 들이면 되기에, BufferedReader는 필요하지 않음 => writer.println(singer);
- 우선 src 디렉토리에 문자열 쓰기를 반영하려면 writer.flush(); 실행 필요 (보통은 객체 소멸 시 자동 실행)
- db 디렉토리가 존재하지 않으면 파일 쓰기가 진행되지 않음

  => File directory = new File("db"); 객체 생성 후, directory.exists(); 조건 검사 후 false 이면 directory.mkdir(); 실행하여 디렉토리 생성

File I/O

연습문제 정답

```
import java.io.*;

public class Main {
    public static void main(String[] args) {
        try {
            File directory = new File("db");
            if (!directory.exists()) {
                directory.mkdir();
            }
            String[] singers = {"아이즈원", "오마이걸", "걸스데이", "르세라핌", "이영지"};
            PrintWriter writer = new PrintWriter(new FileWriter("db/singer.db"));
            for (String singer : singers) {
                writer.println(singer);
            }
            writer.close();
        } catch (IOException e) {
        }
    }
}
```

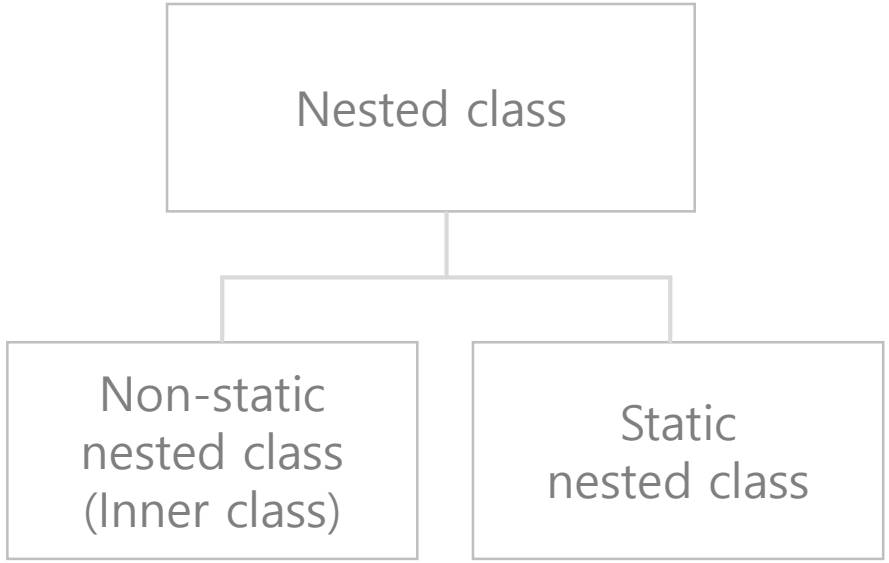
```
아이즈원
오마이걸
걸스데이
르세라핌
이영지
```

Nested class

- Class 내에 또 다른 Class를 정의했을 때, 내부에 위치한 Class를 Nested Class라고 부름
- 간편하게 Class를 활용하고자 할 때 등 여러 용도로 활용

```java
class MotherBoard {
    class USB{
        int usb2 = 1, usb3 = 2;
        int getTotalPorts(){
            return usb2 + usb3;
        }
    }
}
public class Main {
    public static void main(String[] args) {
        MotherBoard m = new MotherBoard();
        MotherBoard.USB u = m.new USB();
        System.out.println("Total Ports = " + u.getTotalPorts());
    }
}
```

Inner class



Nested class

- Static Nested Class는 Inner Class와 달리 상위 Class에 접근할 수 없음
- 특징: 상위 Class와 분리, 상위 Class 내에 여러 하위 Class 모음을 통한 Grouping

Static nested class

```
class MotherBoard {
    public String model = "ABC";
    static class USB{
        int usb2 = 1, usb3 = 2;
        int getTotalPorts() {
            return usb2 + usb3;
        }
    }
}
public class Main {
    public static void main(String[] args) {
        MotherBoard.USB u = new MotherBoard.USB();
        System.out.println("Total Ports = " + u.getTotalPorts());
    }
}
```

Nested class

- Static Nested Class 사례 이해해보기

  (IntelliJ IDE에서 직접 각 클래스/객체 코드 내용을 확인하며 이해)

Static nested class

```
public class SimpleHttpServer {
    public static void main(String[] args) {
        Thread thread = new Thread(new ClientHandler(clientSocket));
    }
    private static class ClientHandler implements Runnable {
        @Override
        public void run() {
            // ...
        }
    }
}
```

Project 2 제공 코드 (SimpleHttpServer.java)

How to install an external library

- IntelliJ에서 GSON 외부 라이브러리 설치 및 활용
  - 공식 사이트에서 jar 파일 다운로드 (예시: GSON library)
    https://search.maven.org/artifact/com.google.code.gson/gson/2.10.1/jar
  - IntelliJ에 라이브러리 설치 방식 참고 문서
    https://velog.io/@kimmjieun/Intellij-외부-라이브러리-추가하기


```java
import com.google.gson.Gson;
import com.google.gson.JsonObject;

public class Main {
    public static void main(String[] args) {
        Gson gson = new Gson();
        // Json key, value 추가
        JsonObject jsonObject = new JsonObject();
        jsonObject.addProperty("name", "anna");
        jsonObject.addProperty("id", 1);
        // JsonObject를 Json 문자열로 변환
        String jsonStr = gson.toJson(jsonObject);
        // 생성된 Json 문자열 출력
        System.out.println(jsonStr); // {"name":"anna","id":1}
    }
}
```