**[ 14 ] API-I**

목표 : Java API를 이해한다.

글자열 관련 API와 날짜관련 API를 이해한다

1. JDK 라이브러리(자바 표준 라이브러리)에 대하여

지금까지 무심코 사용해 왔던 String클래스와 System클래스도 모두 이 이 라이브러리에 속한 클래스입니다. 이 라이브러리는 JDK안에 포함되어 있는데 자바 가상기계가 자바 프로그램을 실행할 때 이 라이브러리의 클래스와 인터페이스들을 자동으로 읽어 들이기 때문에 우리가 지금까지 그 존재를 인식하지 않고도 사용할 수 있었던 겁니다.

자 그럼 컴퓨터의 어느 위치에 있는지 보자(C:\Program Files\Java\jdk1.8.0\_271\jre\lib\rt.jar 혹은 C:\Program Files\Java\jre1.8.0\_271\lib\rt.jar). JRE System Library밑의 rt.jar밑의 java.lang밑의 System.class와 String.class를 확인.

※JDK에서 많은 패키지 제공

1. **java.lang** 자바프로그램의 기본적인 기능을 제공. 명시적으로 지정하지 않아도 모든 자바 프로그램에 포함되는 패키지 java.lang.String나 java.lang.Exception은 모두 java.lang.을 생략 가능
2. **java.util** 유용한 유틸리티 클래스를 제공
3. **java.io** 입출력 기능을 제공하는 패키지
4. **java.awt** 그래픽 유저인터페이스(GUI)를 구축하기 위한 다양한 컴포넌트를 제공하는 패키지
5. **java.awt.event** awt컴포넌트들의 이벤트를 제어하는 패키지

이러한 라이브러리 안의 클래스나 인터페이스를 API라 부르고 http://java.sum.com를 통해 API 규격서를 다운로드 받아 로컬디스크에 저장해 두고 수시로 봅니다.

(<http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html>)

(<http://docs.oracle.com/javase/8/docs/api>)

Java Source attr-External location: c:\Program Files/Java/jdk1.8.xx/src.zip과 연결하면 API 소스를 볼 수 있다

1. JAVA 필수 API

JAVA는 개발자들이 편리하게 이용할 수 있는 풍부한 클래스들이 많습니다.

이러한 클래스들을 이용해서 개발자들은 깊은 지식이 없이, 그냥 사용만 하면 되고, 이러한 기능들을 정의해 둔 클래스들을 API(Application Programming Interface)라고 합니다. 그냥 쉽게 말하자면, API란, 누군가가 만들어 놓은 기능들이라고 생각하면 됩니다.

우리는 지금 JAVA를 배우고 있으니, JAVA API에서 제공하는 이해하기 쉬운 클래스들을 익혀 나가면 됩니다. 그리고 그 중에서도 자주 쓰이는 API만 익히고, 나머지는 프로젝트를 하면서 조금씩 익혀나가면 됩니다. 참고로 JAVA에는 3,000개가 넘는 클래스가 있습니다. 모두 알 수 없고, 그때 그때 조금씩 익혀나가면 됩니다.

1. 문자열에 관련된 대표적 클래스 : String

3-1. String의 이해

※String은 객체 자료형입니다. 우리가 문자열을 사용하면서 아래와 같이 사용하였습니다.

String str = “HelloWorld~~”;

뭔가 이상한 점이 느껴 지시나요?

String은 분명 대문자로 시작하므로, 기초데이터가 아닌 객체데이터입니다. 하지만 뒤에 생성자(new)를 사용하지 않고 기초데이터를 만들 때처럼 초기화 하고 있습니다.

int i = 10;

String의 경우만 특이하게 객체데이터이지만, 기초데이터처럼 사용할 수 있게 하였습니다.

원래는 아래처럼 사용하였어야 했겠죠.

String string = new String(“HelloWorld~~”);

두 경우 모두 사용할 수 있습니다.

String string = “HelloWorld”;

String string = new String(“HelloWorld~~”);

그럼 위의 두 경우의 차이를 알아보자.

**public** **static** **void** main(String[] args) {

String str1 = "Java";

String str2 = "Java";

String str3 = **new** String("Java");

String str4 = **new** String("Java");

// new로 생성하지 않고 ""를 사용한 문자열 상수는 자동으로 힙영역에 String 객체가 생성됨.

// 이미 존재하는 문자열 상수를 다시 사용한다면 이미 생성된 문자열 상수를 공유한다.

System.***out***.println("str1의 해쉬코드 : "+str1.hashCode());

System.***out***.println("str2의 해쉬코드 : "+str3.hashCode());

System.***out***.println("str3의 해쉬코드 : "+str3.hashCode());

System.***out***.println("str4의 해쉬코드 : "+str4.hashCode());

System.***out***.println(str1==str2? "(str1과 str2은 같은 객체)":"(str1과 str2은 다른 객체)");

System.***out***.println(str1.equals(str2)? "(1과2은 같은 스트링)":"(1과2은 같은 스트링)");

System.***out***.println(str1==str3? "(str1과 str3은 같은 객체)":"(str1과 str3은 다른 객체)");

System.***out***.println(str1.equals(str3)? "(1과3은 같은 스트링)":"(1과3은 같은 스트링)");

System.***out***.println(str3==str4? "(str3과 str4은 같은 객체)":"(str3과 str4은 다른 객체)");

System.***out***.println(str3.equals(str4)? "(3과4은 같은 스트링)":"(3과4은 같은 스트링)");

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

String fn;

**do** {

System.***out***.print("I:input, U:update, X:exit");

fn = sc.next();

**if**(fn.equalsIgnoreCase("X")) {

**break**;

}

}**while**(**true**);

sc.close();

}

※String의 주요 기능들(메소드)

String concat(String str) : 저장된 문자열과 str문자열을 결합

String substring(int begin) : 시작위치부터 마지막까지의 문자열을 반환

int length() : 문자열 길이를 반환

String toUpperCase() : 대문자로 반환

String toLowerCase() : 소문자로 반환

char charAt(int index) : index 위치의 문자를 반환

int indexOf(char ch) : 첫번째 ch문자의 위치를 반환

int lastIndexOf(char ch) : 마지막 ch문자의 위치를 반환

boolean equals(String str) : 지정된문자열과 str문자열이 같은지 비교

boolean equalslgnoreCase(String str) : 대소문자구분없이 지정된문자열과 str문자열이 같은지 비교

String trim() : 문자열 앞뒤 공백제거

String replace(char old, char new) : 문자열 내의 old문자를 new문자로 반환

String repalceAll(String old, String new) : 문자열 내의 old문자열을 new로 반환

<예제>

**public** **static** **void** main(String[] args) {

String str1 = "abcXabc";

String str2 = **new** String("abcXabc");

String str3 = " Java ";

System.***out***.println(str1.concat(str2)); //abcXabcabcXabc

System.***out***.println(str1.substring(3)); //Xabc

System.***out***.println(str1.substring(3, 5));//3번째부터5번째앞까지 Xa

System.***out***.println(str1.length() ); // 7(글자갯수)

System.***out***.println(str1.toUpperCase()); // ABCXABC (대문자로)

System.***out***.println(str1.toLowerCase()); // abcxabc (소문자로)

System.***out***.println(str2.charAt(3)); // X (3번째 문자)

System.***out***.println(str2.indexOf('b')); // 1(처음으로 나오는 'b'의 위치)

System.***out***.println(str2.lastIndexOf('b'));// 5(마지막 'b'의 위치)

System.***out***.println(str2.indexOf('@')); // '@'가 없으면 -1

String str4 = "abcXabcXabc";

System.***out***.println(str4.lastIndexOf('b'));//9

System.***out***.println(str4.indexOf('b')); // 1(0번째부터 처음 나오는 'b'의 위치)

System.***out***.println(str4.indexOf('b', 2));//5(2번째부터 처음나오는 'b'의 위치)

System.***out***.println(str1.equals("abcXabc")); //true(같은문자인지)

System.***out***.println(str1.equals("ABCXabc")); //false

System.***out***.println(str1.equalsIgnoreCase("ABCXabc"));//true(대소문자구분없이)

System.***out***.println(str3.trim()); // 앞뒤공백제거 "java"

System.***out***.println(str1.replace('a', 'A')); // a를 A로 바꿔

System.***out***.println(str1.replace("ab", "A"));

System.***out***.println(str1.replaceAll("a", "많이바꿔")); //문자열 교체

System.***out***.println();

System.***out***.println("str1="+str1);

System.***out***.println("str2="+str2);

System.***out***.println("str3="+str3);

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

String fn;

**do** {

System.***out***.print("I:input, U:update, X:exit");

fn = sc.next();

**if**(fn.equalsIgnoreCase("X")) {

**break**;

}

}**while**(**true**);

sc.close();

}

<주민번호를 입력받아 여자인지 남자인지 출력 예제>

<문자열 뒤집기나 짝수번째 문자열만 출력 예제>

출력 : 정상 문자열 : 2002 WorldCup Korea

문자열 뒤집기 : aeroK puCdlroW 2002

짝수번째 문자열 : 0 2 W r d u o e

**public** **static** **void** main(String[] args) {

String str = "2002 WorldCup Korea";

}

<전화번호를 입력받아 다음을 출력하는 로직을 구현>

String tel;

while(){

// 번호(031-234-5678)입력 하면 (sc.next())

"입력한 전화번호 : 031-234-5678

짝수번째 문자열 : 0 1 2 4 5 7

문자를 꺼꾸로 : 8765-432-130

전화번호 맨앞자리는: 031

전화번호 뒷자리는: 5678"을 sysout

// x를 입력하면 프로그램 끝

}

<전화번호 뒷자리로 검색하고 있으면 전화번호 전체를 출력하기, 없는 번호이면 없는 번호라고 출력하기>

**public** **static** **void** main(String[] args) {

String[] tels = {"010-9999-9999","010-8888-8888"};

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("검색하고자 회원의 전화번호 뒷자리는 ?");

}

<친구 핸드폰번호 뒷자리로 친구검색하고 해당 친구 정보 출력하기>

요구사항 : 친구정보 : name, phone, birthday

main()에서 친구정보를 배열로 저장 후 검색하고자 하는 친구의 전화번호 뒷자리로 친구를

검색하여 친구정보를 모두 출력하는 프로그램을 구현하시오(단, 배열에 저장되어 있지

않는 전화번호라면 검색되지 않는 번호라고 출력한다)

제출파일 : Friend.java

TestMain.java

실행화면.jpg

3-2. String의 문제점

※ String은 메모리를 과소비 해요

Stirng클래스는 많이 쓰이고, 좋은 기능(메소드)를 많이 가지고 있습니다. 이렇게 좋은 클래스가 치명적인 단점이 있습니다.

바로 메모리를 과소비 하는 것입니다. String객체의 경우 처음 초기화된 데이터에 변화가 생기면 기존 것을 재활용하지 않고 기존의 것은 버리고 새로운 메모리를 이용합니다.

위의 메소드는 String 객체가 가지고 있는 문자열의 내용을 바꾸는 것이 아니라 바뀐 내용을 갖는 새로운 String객체를 생성해 냅니다. 그렇기 때문에 문자열 조작을 많이 하는 프로그램에서 이런 메소드를 많이 사용하면 String 객체가 너무 많이 생기기 때문에 효율적이지 않습니다. 많은 객체를 만들게 되면 메모리도 많이 쓰게 되고 프로그램의 성능도 떨어지기 때문입니다.

3-3. StringBuffer와 StringBuilder의 등장

문자열 변수의 잦은 내용 변경이 불가피할 시에는, Stirng 클래스의 새로운 메모리를 생성하는 문제로 인해 속도가 느려 지는 현상이 있습니다.

그럴 때는 문자열 조작에 적합하게 만들어진 다른 클래스를 사용하는 것이 좋습니다.

문자열 조작에 적합하게 만들어진 클래스로는 StringBuilder클래스와 StringBuffer클래스가 있습니다. 이 클래스들은 모두 객체 내부에 있는 버퍼(buffer, 데이터를 임시로 저장하는 메모리)에 문자열의 내용을 저장해 두고 그 안에서 추가, 수정, 삭제 작업을 합니다. 그렇게 때문에 이 클래스들은 이용하면 새로운 객체를 만들지 않고도 문자열 조작을 할 수 있습니다.

그래서 속도적인 측면에서 더욱 개선된 StringBuffer와 StringBuilder가 등장합니다.

※ StringBuilder 주요 기능(메소드)

append(String str) : 문자열 str 추가

insert(int index, String str) : 특정 index에 문자열 str 추가

delete(int start, int end) : index위치 start부터 end앞 까지 삭제

deleteCharAt(int index) : index위치의 특정 문자 하나 삭제

int capacity() : 문자열 크기 반환

ensureCapacity(int size) : 버퍼의 크기를 size만큼 늘리는 메소드

trimToSize() : 과도한 버퍼 크기를 적당하게 줄이는 메소드

※StringBuffer와 StringBuilder는 약간의 차이가 있을 뿐 같다고 생각하시면 됩니다

StringBuffer와 StringBuilder는 동일하다고 생각 하면 됩니다. 단, StringBuffer가 StringBuilder보다 먼저 세상에 등장 한 클래스 입니다. 기능은 같고, 단지 StringBuilder가 속도 면에서 좀더 빠릅니다

**public** **static** **void** main(String[] args) {

StringBuilder strBuilder = **new** StringBuilder("abc");

strBuilder.append("def");

System.***out***.println(strBuilder); //abcdef

strBuilder.insert(3, "AAA");

System.***out***.println(strBuilder); //abcAAAdef

strBuilder.delete(3, 5); //3번째부터 5앞에까지 삭제

System.***out***.println(strBuilder); //abcAdef

strBuilder.deleteCharAt(5);

System.***out***.println(strBuilder); //abcAdf

**int** buffersize = strBuilder.capacity();

System.***out***.println("빌더 크기 "+buffersize);

strBuilder.ensureCapacity(100);

buffersize = strBuilder.capacity();

System.***out***.println("빌더 크기 "+buffersize);

**for**(**int** i=0 ; i<100 ; i++) {

strBuilder.append("abcdefghijklmnopqrstuvwxyz");

buffersize = strBuilder.capacity();

System.***out***.println("빌더의 가용크기 "+buffersize);

}//for

}//main

※개발 테스트에 많이 쓰이는 System.currentTimeMillis()

System.currentTimeMillis()는 1970년도부터 현재까지의 밀리세컨(1/1,000초) 단위로 표시합니다. 거의 속도 테스트 용도로 쓰입니다

<예제>

**public** **static** **void** main(String[] args) {

System.***out***.println(System.*currentTimeMillis*());

String str = "A";

**long** startMimiis = System.*currentTimeMillis*();

**for**(**int** i=0 ; i<50000 ; i++) {

str = str.concat("a"); //str = str+"a";

}

**long** endMimiis = System.*currentTimeMillis*();

System.***out***.println("String 경과 시간 : "+(endMimiis-startMimiis)+"밀리초");

StringBuffer strBuf = **new** StringBuffer("A");

startMimiis = System.*currentTimeMillis*();

**for**(**int** i=0 ; i<50000 ; i++) {

strBuf.append("a");

}

endMimiis = System.*currentTimeMillis*();

System.***out***.println("StringBuffer 경과 시간 : "+(endMimiis-startMimiis)+"밀리초");

StringBuilder strBui = **new** StringBuilder("A");

startMimiis = System.*currentTimeMillis*();

**for**(**int** i=0 ; i<50000 ; i++) {

strBui.append("a");

}

endMimiis = System.*currentTimeMillis*();

System.***out***.println("StringBuilder 경과 시간 : "+(endMimiis-startMimiis)+"밀리초");

}

※ StringTokenizer 주요 기능(메소드) : 문자열 분할

"정해인 유준상 강동원 김윤석 하정우"문자열을 space를 기준으로 문자열을 분할한다.

"2019/12/09" 문자열을 “/”를 기준으로 문자열을 분할한다.

**public** **static** **void** main(String[] args) {

String str1 = "정해인 유준상 강동원 김윤석 하정우";

String str2 = "2019/12/09";

System.***out***.println(str1);

System.***out***.println(str2);

StringTokenizer tokenizer1 = **new** StringTokenizer(str1);

StringTokenizer tokenizer2 = **new** StringTokenizer(str2,"/");

System.***out***.println("tokenizer1의 문자열들의 수 : "+tokenizer1.countTokens());

System.***out***.println("tokenizer2의 문자열들의 수 : "+tokenizer2.countTokens());

**while**(tokenizer1.hasMoreTokens()) {

System.***out***.println(tokenizer1.nextToken());

}

**while**(tokenizer2.hasMoreTokens()) {

System.***out***.println(tokenizer2.nextToken());

}

}

4. 날짜(Calendar와 GregorianCalendar) API

※날짜와 시간을 표현할 때 많이 쓰이는 Calendar클래스는 싱글톤클래스

※날짜와 시간을 표현할 때 많이 쓰이는 GregorianCalendar클래스는 일반클래스

<예제>

**public** **static** **void** main(String[] args) {

Calendar calendar = Calendar.*getInstance*(); // Calendar는 싱글톤임. 생성자가 protected.

// 시스템으로부터 현재 시간 정보를 갖는 객체 생성

System.***out***.println(calendar);

System.***out***.println(System.*currentTimeMillis*());

**int** year = calendar.get(Calendar.***YEAR***);

**int** month = calendar.get(Calendar.***MONTH***) + 1; // 0~11월이라 현실이랑 맞춰 줌

**int** day = calendar.get(Calendar.***DAY\_OF\_MONTH***);

**int** week = calendar.get(Calendar.***DAY\_OF\_WEEK***);//1:일 2:월 3:화 4:수 5:목 6:금 7:토

**int** hour24 = calendar.get(Calendar.***HOUR\_OF\_DAY***); // 24시

**int** hour12 = calendar.get(Calendar.***HOUR***); // 12시

**int** ampm = calendar.get(Calendar.***AM\_PM***);

**int** minute = calendar.get(Calendar.***MINUTE***);

**int** second = calendar.get(Calendar.***SECOND***);

**int** millisec = calendar.get(Calendar.***MILLISECOND***);

// DAY\_OF\_WEEK 월,화,수..

// DAY\_OF\_YEAR 그해 날짜

// WEEK\_OF\_MONTH 그달의 몇째주인지

// DATE나 DAY\_OF\_MONTH 그달의 날짜

// WEEK\_OF\_YEAR 그 해에서의 몇째주인지

// MONTH 달

System.***out***.printf("%d년 %d월 %d일 ", year, month, day);

**switch** (week) {

**case** 1:

System.***out***.println("일요일");**break**;

**case** 2:

System.***out***.println("월요일");**break**;

**case** 3:

System.***out***.println("화요일");**break**;

**case** 4:

System.***out***.println("수요일");**break**;

**case** 5:

System.***out***.println("목요일");**break**;

**case** 6:

System.***out***.println("금요일");**break**;

**case** 7:

System.***out***.println("토요일");**break**;

}

System.***out***.print((ampm == 0) ? "오전 " : "오후 ");

System.***out***.printf("%d시 %d분 %d초 %d\n", hour12, minute, second, millisec);

System.***out***.printf("%d시 %d분 %d초 %d\n", hour24, minute, second, millisec);

System.***out***.println("70년1월1일부터 milli초 : "+calendar.getTimeInMillis());

}

**public** **static** **void** main(String[] args) {

Friend[] friends = {**new** Friend("홍길동","010-9999-9999","12-09"),

**new** Friend("김길동","010-8888-8888","11-01"),

**new** Friend("이선동","010-6666-6666","12-09")};

**int** idx;

Calendar calendar = Calendar.*getInstance*();

**int** month = calendar.get(Calendar.***MONTH***)+1;

**int** day = calendar.get(Calendar.***DAY\_OF\_MONTH***);

String today = month+"-"+(day<10? "0"+day : day);

System.***out***.println("오늘은 "+today);

**boolean** output = **false**;

**for**(idx=0 ; idx<friends.length ; idx++) {

String birth = friends[idx].getBirth();

**if**(birth.equals(today)){

System.***out***.println(friends[idx].infoString());

output = **true**;

}//if

}//for

**if**(!output) {

System.***out***.println("오늘이 생일인 사람이 없습니다.");

}

}

<예제>

**public** **static** **void** main(String[] args) {

GregorianCalendar gc = **new** GregorianCalendar();

System.***out***.println(gc);

System.***out***.println(System.*currentTimeMillis*());

**int** year = gc.get(Calendar.***YEAR***);

**int** month = gc.get(Calendar.***MONTH***) + 1; // 0~11월이라 현실이랑 맞춰 줌

**int** day = gc.get(Calendar.***DAY\_OF\_MONTH***);

**int** week = gc.get(Calendar.***DAY\_OF\_WEEK***); // 1:일 2:월 3:화 4:수 5:목 6:금 7:토

**int** hour24 = gc.get(Calendar.***HOUR\_OF\_DAY***); // 24시

**int** hour12 = gc.get(Calendar.***HOUR***); // 12시

**int** ampm = gc.get(Calendar.***AM\_PM***);

**int** minute = gc.get(Calendar.***MINUTE***);

**int** second = gc.get(Calendar.***SECOND***);

**int** millisec = gc.get(Calendar.***MILLISECOND***);

// DAY\_OF\_WEEK 월,화,수..

// DAY\_OF\_YEAR 그해 날짜

// WEEK\_OF\_MONTH 그달의 몇째주인지

// DATE나 DAY\_OF\_MONTH 그달의 날짜

// WEEK\_OF\_YEAR 그 해에서의 몇째주인지

// MONTH 달의 이름

System.***out***.printf("%d년 %d월 %d일 ", year, month, day);

**switch** (week) {

**case** 1:

System.***out***.println("일요일");**break**;

**case** 2:

System.***out***.println("월요일");**break**;

**case** 3:

System.***out***.println("화요일");**break**;

**case** 4:

System.***out***.println("수요일");**break**;

**case** 5:

System.***out***.println("목요일");**break**;

**case** 6:

System.***out***.println("금요일");**break**;

**case** 7:

System.***out***.println("토요일");**break**;

}

System.***out***.print((ampm == 0) ? "오전 " : "오후 ");

System.***out***.printf("%d시 %d분 %d초 %d\n", hour12, minute, second, millisec);

System.***out***.printf("%d시 %d분 %d초 %d\n", hour24, minute, second, millisec);

System.***out***.println("70년1월1일부터 milli초 : "+gc.getTimeInMillis());

GregorianCalendar thatGc = **new** GregorianCalendar(2019,11,9,22,12,12);

year = thatGc.get(Calendar.***YEAR***);

month = thatGc.get(Calendar.***MONTH***) + 1; // 0~11월이라 현실이랑 맞춰 줌

day = thatGc.get(Calendar.***DAY\_OF\_MONTH***);

week = thatGc.get(Calendar.***DAY\_OF\_WEEK***); // 1:일 2:월 3:화 4:수 5:목 6:금 7:토

hour24 = thatGc.get(Calendar.***HOUR\_OF\_DAY***); // 24시

hour12 = thatGc.get(Calendar.***HOUR***); // 12시

ampm = thatGc.get(Calendar.***AM\_PM***);

minute = thatGc.get(Calendar.***MINUTE***);

second = thatGc.get(Calendar.***SECOND***);

millisec = thatGc.get(Calendar.***MILLISECOND***);

System.***out***.printf("%d년 %d월 %d일 ", year, month, day);

**switch** (week) {

**case** 1:

System.***out***.println("일요일");**break**;

**case** 2:

System.***out***.println("월요일");**break**;

**case** 3:

System.***out***.println("화요일");**break**;

**case** 4:

System.***out***.println("수요일");**break**;

**case** 5:

System.***out***.println("목요일");**break**;

**case** 6:

System.***out***.println("금요일");**break**;

**case** 7:

System.***out***.println("토요일");**break**;

}

System.***out***.print((ampm == 0) ? "오전 " : "오후 ");

System.***out***.printf("%d시 %d분 %d초 %d\n", hour12, minute, second, millisec);

System.***out***.printf("%d시 %d분 %d초 %d\n", hour24, minute, second, millisec);

System.***out***.println("70년1월1일부터 milli초 : "+thatGc.getTimeInMillis());

}

**public** **static** **void** main(String[] args) {

Date date = **new** Date();

System.***out***.println(date);

GregorianCalendar gc = **new** GregorianCalendar();

Calendar cal = Calendar.*getInstance*();

**int** year= date.~~getYear~~() + 1900;

**int** month = date.~~getMonth~~()+1;

**int** day = date.~~getDate~~();

**int** hour = date.~~getHours~~();

**int** minute = date.~~getMinutes~~();

**int** second = date.~~getSeconds~~();

System.***out***.printf("%d년 %d월 %d일 %d시 %d분 %d초\n",

year, month, day, hour, minute, second);

System.***out***.printf("%TY년 %Tm월 %Td일 %Ta요일 %TH시 %TM분 %TS초\n",

date, date, date, date, date, date, date);

System.***out***.printf("%tY년 %tm월 %td일 %ta요일 %tH시 %tM분 %tS초\n",

date, date, date, date, date, date, date);

System.***out***.printf("%1$tY년 %1$tm월 %1$td일 %1$ta요일 %1$tH시 %1$tM분 %1$tS초\n",

date);

System.***out***.println("=============== gc =================");

System.***out***.printf("%TY년 %Tm월 %Td일 %Ta요일 %TH시 %TM분 %TS초\n",

gc, gc, gc, gc, gc, gc, gc);

System.***out***.printf("%tY년 %tm월 %td일 %ta요일 %tH시 %tM분 %tS초\n",

gc, gc, gc, gc, gc, gc, gc);

System.***out***.printf("%1$tY년 %1$tm월 %1$td일 %1$ta요일 %1$tH시 %1$tM분 %1$tS초\n",

gc);

System.***out***.println("=============== cal =================");

System.***out***.printf("%TY년 %Tm월 %Td일 %Ta요일 %TH시 %TM분 %TS초\n",

cal, cal, cal, cal, cal, cal, cal);

System.***out***.printf("%tY년 %tm월 %td일 %ta요일 %tH시 %tM분 %tS초\n",

cal, cal, cal, cal, cal, cal, cal);

System.***out***.printf("%1$tY년 %1$tm월 %1$td일 %1$ta요일 %1$tH시 %1$tM분 %1$tS초\n",

cal);

}

<예제> DateFormat

**public** **static** **void** main(String[] args) {

GregorianCalendar gc = **new** GregorianCalendar();

SimpleDateFormat dateFormat

= **new** SimpleDateFormat("yyyy년 MM월 dd일 aa hh시 mm분 ss초");

// y년 M월

// d일

// D 월구분없는일(1~365)

// E 요일

// a오전/오후

// W월의 몇번째 주

// w 년의 몇번째주

// H시(0~23)

// h시(1~12)

// m 분

// s초

// S밀리세컨드(1/1000)

String outString = dateFormat.format(gc.getTime());

System.***out***.println(outString);

}

<예제>

**public** **static** **void** main(String[] args) {

Calendar cal = Calendar.*getInstance*();

SimpleDateFormat dateFormat

= **new** SimpleDateFormat("yyyy년 MM월 dd일 aa hh시 mm분 ss초");

String outString = dateFormat.format(cal.getTime());

System.***out***.println(outString);

}

<예제>

**public** **static** **void** main(String[] args) {

Date date = **new** Date();

SimpleDateFormat dateFormat

= **new** SimpleDateFormat("yyyy년 MM월 dd일 aa hh시 mm분 ss초");

String outString = dateFormat.format(date);

System.***out***.println(outString);

}

public static void main(String[] args) {

Date today = new Date();

SimpleDateFormat sdf1, sdf2, sdf3, sdf4;

SimpleDateFormat sdf5, sdf6, sdf7, sdf8, sdf9;

sdf1 = new SimpleDateFormat("yyyy-MM-dd");

sdf2 = new SimpleDateFormat("yy년 MMM dd일 E요일");

sdf3 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");

sdf4 = new SimpleDateFormat("yyy-MM-dd hh:mm:ss a");

sdf5 = new SimpleDateFormat("오늘은 올 해의 D번째 날입니다");

sdf6 = new SimpleDateFormat("오늘은 이 달의 d번째 날입니다");

sdf7 = new SimpleDateFormat("오늘은 올 해의 w번째 주입니다");

sdf8 = new SimpleDateFormat("오늘은 이달의 W번째 주입니다");

sdf9 = new SimpleDateFormat("오늘은 이달의 F번째 E요일입니다");

System.out.println(sdf1.format(today));

System.out.println(sdf2.format(today));

System.out.println(sdf3.format(today));

System.out.println(sdf4.format(today));

System.out.println(sdf5.format(today));

System.out.println(sdf6.format(today));

System.out.println(sdf7.format(today));

System.out.println(sdf8.format(today));

System.out.println(sdf9.format(today));

}

**public** **static** **void** main(String[] args) {

Friend[] friends = {**new** Friend("홍길동","010-9999-9999","12-08"),

**new** Friend("김길동","010-8888-8888","11-01"),

**new** Friend("이선동","010-6666-6666","12-08")};

**int** idx;

Calendar calendar = Calendar.*getInstance*();

SimpleDateFormat sdf = **new** SimpleDateFormat("MM-dd");

String today = sdf.format(calendar.getTime());

**boolean** output = **false**;

**for**(idx=0 ; idx<friends.length ; idx++) {

String birth = friends[idx].getBirth();

**if**(birth.equals(today)){

System.***out***.println(friends[idx].infoString());

output = **true**;

}//if

}//for

**if**(!output) {

System.***out***.println("오늘이 생일인 사람이 없습니다.");

}

}

**public** **static** **void** main(String[] args) {

// 지금

Date today1 = **new** Date();

GregorianCalendar today2 = **new** GregorianCalendar();

Calendar today3 = Calendar.*getInstance*();

System.***out***.printf("오늘 : %1$TY년 %1$tm월 %1$td일",today1);

System.***out***.printf("오늘 : %1$TY년 %1$tm월 %1$td일",today2);

System.***out***.printf("오늘 : %1$TY년 %1$tm월 %1$td일",today3);

// 1970.1.1부터 지금까지 밀리세컨

System.***out***.println(today2.getTimeInMillis());

System.***out***.println(System.*currentTimeMillis*());

// 특정시점 (1990,0,11)

Date thatDay1 = **new** ~~Date~~(90,0,11);

Date thatDay2 = **new** Date(**new** GregorianCalendar(1990,0,11).getTimeInMillis());

GregorianCalendar thatDay3 = **new** GregorianCalendar(1990, 0, 11);

System.***out***.printf("1990.1.11 : %1$TY년 %1$tm월 %1$td일",thatDay1);

System.***out***.printf("1990.1.11 : %1$TY년 %1$tm월 %1$td일",thatDay2);

System.***out***.printf("1990.1.11 : %1$TY년 %1$tm월 %1$td일",thatDay3);

}

Sawon 클래스와 main함수가 들어있는 부분을 구현하여 보세요

Sawon클래스 내용

데이터 : 사번, 이름, 부서(COMPUTER, PLANNING, DESIGN, ACCOUNTING, HUMANRESOURCES), 입사일.

생성자 : 사번, 이름, 부서, 입사일은 객체생성 당일로 합니다

메소드: infoString()의 실행결과는 다음과 같습니다

[사번]200121 [이름]홍길동 [부서]COMPUTER [입사일]2019년2월2일

main()함수에서 Sawon 객체 만들어 화면에 print해 보세요. 오늘이 입사한 사원도 출력합니다.