API

API

API

API(Application Programming Interface)

프로그래밍을 할 때 어떠한 특정 기능을 처리할 수 있도록 만들어 놓은 클래스나 메소드의 집합

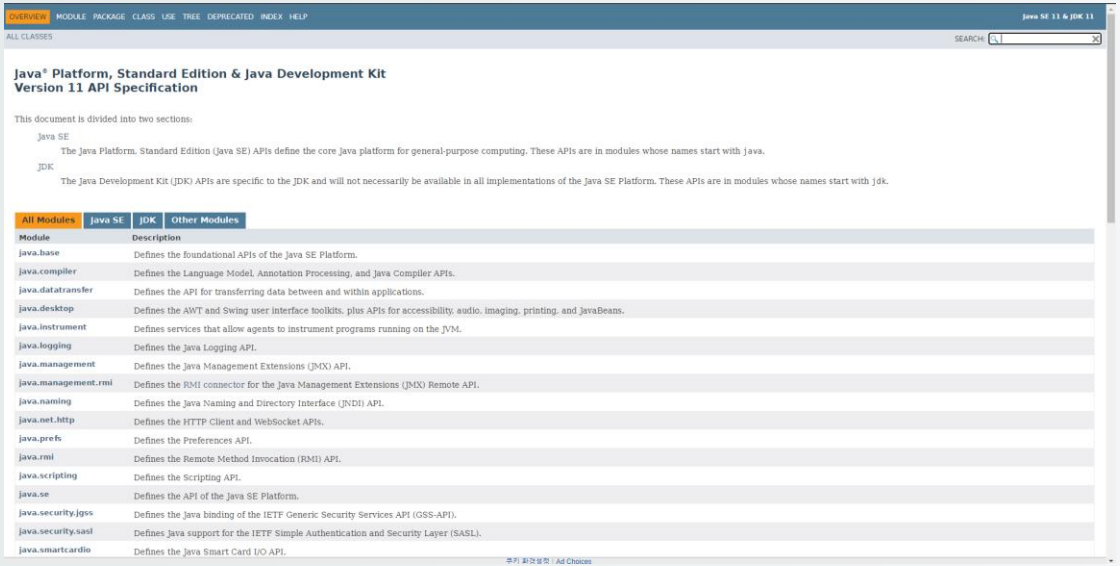
JAVA API

자바 언어를 사용하여 기능 구현을 할 수 있도록 미리 여러가지 기능을 정의 해 놓은 API

※ 실제 JAVA는 다양한 API를 가지고있고, 그러한 API를 기반으로 프로그래밍

API

https://docs.oracle.com/en/java/javase/17/docs/api/index.html



기본 API

String 관련 API

String 클래스

- 문자열 값을 수정 못하는 immutable(불변) 성질을 가짐
- 수정 시 수정된 문자열이 새로 할당되어 새 주소를 저장
- 문자열을 자주 수정하는 경우 메모리 낭비가 심함

StringBuffer 클래스

- 문자열 값을 수정 할 수 있는 mutable(가변) 성질을 가짐
- 수정 시 수정된 기존 문자열이 수정
- 문자열을 자주 수정하는 경우 String보다 유용
- Thread Safe 기능 제공(성능저하 요인)

StringBuilder 클래스

- StringBuffer클래스와 동일하지만, Thread Safe기능이 제공되지 않음

String 관련 API

StringBuffer의 메소드

- capacity() : 실제 할당된 공간의 크기 확인
- length() : 실제 문자열 길이 확인
- reverse() : 저장된 문자열을 거꾸로 재배치(가나다 → 다나가)
- insert(): 문자열의 특정 위치에 문자열 추가
- append() : 문자열 끝에 문자열을 추가
- delete() : 문자열의 일부분을 삭제
- replace() : 문자열의 일부분을 다른 문자열로 변경

String 관련 API

StringTokenizer 클래스

- 문자열을 분석하여 토큰으로 분리시켜주는 기능의 클래스
- 생성시 전달받은 문자열을 구분자로 나누어 각 토큰에 저장
- 파일에 저장된 텍스트를 처리하는 경우 유용

StringTokenizer 클래스의 메소드

- countTokens() : 꺼내지 않고 남아있는 토큰의 수
- hasMoreTokens() : 남아 있는 토큰이 있는지 확인(true,false)
- nextToken() : 토큰을 하나씩 꺼내옴

Math Class

Math 클래스

- 수학에서 자주 사용하는 상수들과 메소드들을 구현해 놓은 클래스
- Math 클래스의 메소드는 모두 static method로 객체를 생성하지 않고 바로 사용가능

Math 클래스 사용 예

```
System.out.println(Math.abs(-10));          //절대 값 리턴
System.out.println(Math.ceil(10.1));        //주어진 값 올림
System.out.println(Math.floor(10.9));       //주어진 값 버림
System.out.println(Math.round(10.5));       //주어진 값 반올림
System.out.println(Math.random());          //0.0 ~ 1.0 범위의 임의의 값 추출
System.out.println(Math.max(1,2));          //비교 후 큰 값 리턴
System.out.println(Math.min(1,2));          //비교 후 작은 값 리턴
```

시간관련 Class

Date 클래스

- 시스템으로부터 현재 날짜, 시간정보를 가져와서 다룰 수 있게 만들어진 클래스로, 생성자 2개만 사용 가능하고 나머지는 deprecated
- Calendar 또는 GregorianCalendar 클래스 사용을 권장

Date 클래스의 생성자

```
Date date1 = new Date()
//시스템으로 부터 현재 날짜, 시간 정보를 가져와 기본값으로 사용
Date date2 = new Date(123456789L);
//long형 정수 값을 가지고 날짜 시간을 계산(ms)
//1970년 1월 1일 09시 00분 00초를 기준으로 기준시간에서
//생성자의 매개변수로 입력한 정수값까지의 시간이 흐른 시점의
//날짜와 시간정보 저장
```

※ deprecated : 잘 사용되지 않는 기능으로 없어질 수 있는 기능

시간관련 Class

SimpleDateFormat 클래스

- Date의 날짜, 시간 정보를 원하는 format으로 출력하도록 기능을 제공하는 클래스

SimpleDateFormat 클래스 사용 예

```
Date today = new Date();
SimpleDateFormat sdf1 = new SimpleDateFormat(“yyyy-MM-dd”);
SimpleDateFormat sdf2 = new SimpleDateFormat(“yyyy-MM-dd-HH-mm-ss”);
String date1 = sdf1.format(today);
String date2 = sdf2.format (today);
```

시간관련 Class

Calendar 클래스

- 현재 시간과 관련 있는 클래스
- Calendar클래스는 추상클래스로, 생성자의 접근제어 지시자가 protected로 new 생성자()를 통해 객체를 생성 할 수 없고, getInstance() 메소드를 이용하여 객체를 생성
- Calendar.getInstance()메소드는 GregorianCalendar 객체를 리턴하며, 해당 객체는 Calendar클래스를 상속하여 작성된 자식클래스로, 년,월, 일,시,분,초 정보를 필드(변수)를 통해 다룰 수 있다.

Wrapper Class

Wrapper Class

Wrapper Class

- 기본자료형을 객체화 해주는 클래스
- 기본 자료형 데이터를 포장하여 표현하기 때문에 Wrapper라고 부름
- 객체지향 프로그래밍의 경우 기본자료형을 객체로 처리해야 하는 상황이 존재하는데 그러한 상황에서 사용하는 클래스

Wrapper Class

Wrapper Class의 종류

기본자료형	Wrapper Class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Wrapper Class

Wrapper Class의 종류

Wrapper Class	Wrapper Class로 기본자료형 객체 생성
Boolean	Boolean bool = new Boolean(true);
Character	Character ch = new Character('A');
Byte	Byte bNum = new Byte((byte)1);
Short	Short sNum = new Short((short)2);
Integer	Integer iNum = new Integer(4);
Long	Long lNum = new Long(8);
Float	Float fNum = new Float(0.4f);
Double	Double dNum = new Double(0.8);

Wrapper Class

Wrapper Class의 boxing / unboxing

- Boxing : Wrapper 클래스에 기본자료형을 넣는 것
  Integer num = new Integer(10);
  //정수 10을 Integer클래스로 객체화

- Unboxing : Wrapper 클래스로 만들어진 객체에서 기본자료형을 빼는 것
  int n = num.intValue();
  //Integer객체 num에 들어있는 정수 값을 꺼내서 정수형 변수에 저장

Wrapper Class

Wrapper Class의 auto boxing / auto unboxing

- Auto Boxing
  Integer num = 10;
  //정수 10을 Auto Boxing하여 Integer클래스로 객체화

- Auto Unboxing
  int n = num;
  //Integer객체 num에 들어있는 정수 값을 Auto Unboxing하여 정수형 변수에 저장

Wrapper Class

Wrapper Class를 이용한 문자열 처리

- Wrapper 클래스를 이용하면 문자열 → 기본자료형, 기본자료형 → 문자열 변환 가능

문자열 → 기본 자료형

```
String data = "999";
int num = Integer.parseInt(data);
```

문자형을 제외한 다른 Wrapper 클래스는 사용법 동일
String data = "11.1";
double d = Double.parseDouble(data);
문자형의 경우 charAt(인덱스) 사용

기본 자료형 → 문자열

```
int num = 999;
String data = Integer.valueOf(num).toString();
//또는
String data1 = String.valueOf(num);
```

문자형을 제외한 다른 래퍼클래스는 사용법 동일
double d = 11.1;
String data = Double.valueOf(d).toString();

문자형은
Char ch = 'a';
String data = ch.toString();