기본 API 클래스

.실제 코딩할 때 많이 사용되는 method를 소개한다.

Object 클래스

객체가 생성될 때 무조건 Object를 상속하게 되어 있다. 그래서 Object class의 모든 method를 사용할 수 있다.

```
public class ObjectEx1 {
   public static void main(String[] args) {
            String s1 = "Hello";
            String s2 = new String("Hello");
            System.out.println(s1 == s2);
            System.out.println(s1.equals(s2)); // 문자열을 비교

            MyClass mc1 = new MyClass(10);
            MyClass mc2 = new MyClass(10);

            System.out.println(mc1 == mc2);
            System.out.println(mc1.equals(mc2));   // 객체의 주소를 비교해 버리므로 객체가 가지고 있는
값을 비교 하고 싶은때 equals를 재정의 해야 한다.(아래처럼)

            System.out.println(mc1.toString());
            System.out.println(mc2.toString());
    }
}

class MyClass {
      private int val;
      MyClass(int val) {
            this.val = val;
       }

       @Override
       public boolean equals(Object obj) {
            if(obj != null && obj instanceof MyClass) {
                        if(val == ((MyClass)obj).val)
                                    return true;
                        else
                                    return false;
            } else {
                        return false;
            }
        }
        // ctrl + space 치면 override 할 메서드들이 나열된다
        @Override
        public String toString() {
            return "val=" + val;
        }
    }
}
```

아래 코드 실습하세요. 객체를 매개변수로 넘기는 예제

```
class Rect {
	int width;
	int height;
	public Rect(int width, int height) {
		this.width = width;
		this.height = height;
	}
	public boolean equals(Rect p) {
		if (width*height == p.width*p.height) // 사각형 면적 비교
			return true;
		else
			return false;
	}
}
public class EqualsEx {
	public static void main(String[] args) {
		Rect a = new Rect(2,3);
		Rect b = new Rect(3,2);
		Rect c = new Rect(3,4);
		if(a.equals(b))		System.out.println("a is equal to b");
		if(a.equals(c))		System.out.println("a is equal to c");
		if(b.equals(c))		System.out.println("b is equal to c");

	}
}
```

```
a is equal to b
```

String 클래스

문자열 변환(valueOf()) – 정수, 실수 등을 문자열로 변환 방법

```
static String valueOf(boolean b)
static String valueOf(char c)
static String valueOf(int i)
static String valueOf(long l)
static String valueOf(double d)
static String valueOf(float f)
```

```
public class StringEx2 {
	public static void main(String[] args) {
		int val = 50;
		String s1 = String.valueOf(val);   // 정수를 문자열로 변환
		String s2 = val + "";     // 꼼수를 써도 된다

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

```
// String 에서 제공하는 method 활용 예제 – 문자열 쪼개기
public class SplitEx1 {
	public static void main(String[] args) {
		String fruits = "사과, 배, 귤, 바나나, 딸기, 키위, 메론";
		String[] words = fruits.split(",");
```

```
                for(String f : words) {
                        System.out.println(f);
                }
        }
}
```

```
// 문자열을 byte단위로 만들기
import java.io.UnsupportedEncodingException;

public class StringByte {
        public static void main(String[] args) throws UnsupportedEncodingException {
                String msg = "안녕하세요?";
                byte[] bytes = msg.getBytes("UTF-8");   //msg.getBytes() 에서 ctrl + 1을 쳐서 선택

                String msg2 = new String(bytes, "UTF-8");
                System.out.println(msg2);
        }
}
```

```
public class StringEx1 {
        public static void main(String[] args) {
                String filePath = "C:\\Program Files\\MyApp\\test.txt";
                String fileName = filePath.substring(filePath.lastIndexOf("\\") + 1);
                System.out.println("파일명: " + fileName);
        }
}
```

StringTokenizer

```
import java.util.StringTokenizer;

public class StringTokenizerEx1 {
        public static void main(String[] args) {
                String fruits = "사과, 배, 귤, 바나나, 딸기, 키위, 메론";
                StringTokenizer tokenizer =
                                new StringTokenizer(fruits, ",");

                while(tokenizer.hasMoreTokens()) {
                        System.out.println(tokenizer.nextToken());
                }
        }
}
```

Pattern (정규식 표현) – 어떤 문자열이 규칙에 맞는지 확인할 경우 사용

아래 예제에서는 전화번호가 010, 170으로 시작하고 앞자리가 3자리 혹은 4자리만 허용되고 뒤자리는 4자리만 허용된다.

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

public class PatternEx1 {
        public static void main(String[] args) {
```

```
                String pattern = "(010|070)-\\d{3,4}-\\d{4}";
                String telNo = "010-523-1456";

                if(Pattern.matches(pattern, telNo)) {
                        System.out.println(telNo + "는 유효한 전화번호 입니다.");
                } else {
                        System.out.println(telNo + "는 잘못된 전화번호입니다.");
                }
        }
}
```

Wrapper 클래스

기본 타입(byte, char, short, int, long, float, double, boolean) 값을 내부에 두고 포장하는 객체

```
public class WrapperEx1 {
        public static void main(String[] args) {
                Emp e1 = new Emp();
                e1.bonus = 1000000;
                int bonus = e1.bonus;   // 대입가능
                System.out.println(e1.bonus);
        }
}

class Emp {
        String empNo;
        String name;
        //int bonus;   0 값이 들어가는데 0이 의미 있는 값이 될 수도 있다. 보너스가
없다는 의미로
        Integer bonus;
}
```

Wrapper 클래스 중에서 Character 클래스 안에는 isDisigit(i), isUppercase(i) 등의 메서드를 사용할 수 있는 장점이 있다.

아래 코드 실습하세요.

```
public class WrapperClassEx {
        public static void main(String[] args) {
            Integer i = new Integer(10);
            char c = '4';
            Double d = new Double(3.1234566);
            System.out.println(Character.toLowerCase('A')); // 대문자 A를 소문자로 변환
            if (Character.isDigit(c)) // 문자 c가 숫자를 나타내면 true
                System.out.println(Character.getNumericValue(c)); // 문자 c를 숫자로 변환하여 출력
                System.out.println(Integer.parseInt("-123")); // 문자열 “-123”을 정수로 변환하여 출력
                System.out.println(Integer.toBinaryString(28)); // 28의 2진수 표현을 나타내는 문자열 출력
                System.out.println(Integer.bitCount(28)); // 28의 2진수에서 1의 개수출력
                System.out.println(Integer.toHexString(28)); // 28의 16진수 표현을 나타내는 문자열 출력
                System.out.println(i.doubleValue()); // i값(=10)을 double로 변환하여 출력
                System.out.println(d.toString()); // d값(=3.1234566)을 문자열로 변환하여 출력
        }
}
```

```
a
4
-123
11100
3
1c
10.0
3.1234566
```

StringBuffer 클래스를 사용했을 경우와 사용하지 않을 경우 문자열을 연속적으로 연결했을 경우의 성능 비교 코드

```
// 문자열을 이어주는 코드인데 어느 정도 시간이 걸리는지 체크해 보자
public class TimeTest {
	public static void main(String[] args) {
		String s1 = "";

		long from = System.currentTimeMillis();
		for(int i=0; i<100000; i++) {
			s1 += "ABCD";
		}
		long to = System.currentTimeMillis();

		System.out.println("s1 문자열의 길이: " + s1.length());
		System.out.println("소요시간: " + (to - from)+ "ms");

		// ------------------------------------------------
		//StringBuilder sb = new StringBuilder();
		StringBuffer sb = new StringBuffer();	// multi thread의 동기화 기능이 제공된다.

		from = System.currentTimeMillis();

		for(int i=0; i<100000; i++) {
			sb.append("ABCD");
		}

		to = System.currentTimeMillis();

		System.out.println("sb 문자열의 길이: " + sb.length());
		System.out.println("소요시간: " + (to - from)+ "ms"); // 시간이 단출된다. sql 장문의 구문에 효율성이 좋다
	}
}
```

Date 클래스

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateEx1 {
	public static void main(String[] args) {
		// 현재 시스템 클럭값을 담는 객체 생성
		Date now = new Date();	// ctrl + 1 을 눌러 import 선책한다
		System.out.println(now);

		// 특정 포맷의 문자열로 변환
```

```
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String formattedNow = sdf.format(now);
                System.out.println(formattedNow);
        }
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class DateEx1 {
        public static void main(String[] args) throws ParseException {
                // 현재 시스템 클럭값을 담는 객체 생성
                Date now = new Date();     // ctrl + 1 을 눌러 import 선택한다
                System.out.println(now);

                // 특정 포맷의 문자열로 변환
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String formattedNow = sdf.format(now);
                System.out.println(formattedNow);

                // 문자열을 Date 객체로 변환
                String inputDateStr = "2016-05-05";
                SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
                Date inputDate = format.parse(inputDateStr);
                System.out.println(inputDate);

                // 특정일로부터 100일후의 날짜를 구함
                Calendar c1 = new GregorianCalendar();
                c1.setTime(inputDate);
                c1.add(Calendar.DAY_OF_YEAR, 100);
                Date dateAfter100 = c1.getTime();
                System.out.println(inputDateStr + "로부터 100일 후의 날짜: " +
format.format(dateAfter100));

                // 터어난 이후 몇일 살았나?
                Date birthday = format.parse("1990-05-05"); // 태어난 날 입력
                long diffMsec = now.getTime() - birthday.getTime(); // ms 단위이다.
                long diffDay = diffMsec / (1000 * 60 * 60 * 24);
                System.out.println("지금까지 내가 산 날수:" + diffDay);

                //자바8에서 제공하는 LocalDate 를 이용 (format을 사용할 필요가 없다) - 100일 이후
날짜
                LocalDate localDate = LocalDate.now();
                LocalDate futureDate = localDate.plus(100, ChronoUnit.DAYS);
                System.out.println(futureDate);
        }
}
```

수고했습니다. 위에 소개된 클래스들은 실제 코딩시 많이 사용되고 있기 때문에 사용방법을 익혀 두도록 합시다. 이제 String, StringBuffer, StringTokenizer 등의 클래스를 이용하여 아래 문제를 코

딩해 봅시다.

문제: 1660, 1714, 1751, 1752, 1754