2023-03-22

표준 입출력

다양한 곳에 다양한 형태로 존재하는 데이터를 입출력하기 위해서는 일관된 처리방법이 있어야한다.

Java에서는 Stream을 제공하여 데이터를 읽거나 기록할 수 있도록 하는데 가장 기본적으로 많이 사용되는 스트림이 표준 입출력 스트림이다.

- System.in과 System.out
    - console을 통한 데이터 입출력

표준입력으로부터 문자 단위로 하나의 문자값만 읽을 때는 System.in으로 InputStreamReader객체를 추가 생성한 뒤 read()메서드를 호출한다.

```
InputStreamReader isr = new InputStreamReader(System.in);
int inputchar = isr.read();
```

표준입력에서 한 행을 읽을 때는 BufferedReader 객체를 추가생성하여 readLine()메서드를 호출한다.

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String inputline = br.readLine();
```

객체 직렬화

Java 프로그램이 수행되는 동안에는 힙에 메모리공간이 할당되어 객체의 내용이 보관되고 있으나, 이 내용을 지속적으로 보관하고자 할 때 객체 직렬화를 사용

- 직렬화 가능한 객체를 구현하기 위해서는 erializable 인터페이스, Externalizable 인터페이스

  두 인터페이스를 상속해야하고있는 클래스의 객체만 직렬화가 가능하다.

  NotSerializableException 발생

- 부모가 Serializable을 상속하고 있으면 자식도 그대로 적용된다. 하지만 자식만 상속하는 경우 자식에게서만 직렬화가 일어난다.

- non-static, non-transient 멤버 변수들만 직렬화 대상이 된다

  직렬화를 할 때 모든 인스턴스형 멤버변수들의 값은 직렬화가 되지만, static변수(클래스 변수)는 객체의 상태 정보가 아니므로 직렬화 대상에서 제외된다.

  하지만 인스턴스형 변수이더라도 직렬화 시 제외시키고 싶은 변수가 있다면 **transient** 키워드를 사용하면 된다.

- 직렬화의 대상이 되는 멤버 변수가 참조형일 경우 참조하는 객체 역시 직렬화가 가능한 객체여야한다. 아닐시 NotSerializableException 발생

ObjectOutputStream을 사용한 직렬화

- 기본형 타입의 데이터와 객체를 직렬화하여 출력할 수 있는 보조스트림 클래스
- writeInt(), writeDouble(), writeChar(),writeObject() 등의 메서드 사용하여 출력

직렬화한 결과를 파일에 출력하고자 할 때 바이트 스트림으로 동작하는 FileOutputStream 클래스의 객체를 생성하여 파일 출력모드로 오픈해야한다.

```
FileOutputStream fos = new FileOutputStream("test1.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeInt(100);
oos.writeDouble(3.14);
oos.writeObject(new Date());
```

ObjectInputStream을 사용한 역 직렬화

- ObjectInputStream은 ObjectOu tputStream에 의해 출력된 기본형 타입의 데이터와 객체를 역으로 직렬화하여 입력받을 수 있도록 지원하는 보조스트림 클래스
- readInt(), readDouble(), readChar(), readObject() 등의 메서드를 사용하여 출력

파일에 출력된 직렬화 결과를 읽으려면 우선 바이트 스트림으로 동작하는 FileInputStream 클래스의 객체를 생성하여 파일을 입력모드로 오픈한다.

```
FileInputStream fis = new FileInputStream("test1.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
int value1 = oos.readInt();
double value2 = oos.readDouble();
Date value3 = (Date)oos.readObject();
```

URL(Uniform Resource Locator) 프로그래밍

URL : 어떠한 자원의 위치를 알리는 단일화된 형식의 문자열/주소

URL클래스를 이용하는 프로그래밍 (java.net)

I/O 프로그래밍, Network 프로그래밍

HTTP URL - 웹사이트의 주소 문자열

JDBC URL - 접속할 DB서버와 JDBC 드라이버 정보를 정해진 규격으로 작성한 문자열

java.net.URL 클래스 : 웹 서버에 접속하여 컨텐트를 요청하는 프로그램을 개발할 때

```
객체생성
openStream(): InputStream - GET방식
```

Enum

한정된 값으로 이루어진 Enum 타입

- 요일, 계절처럼 한정된 값을 갖는 타입(상수-상수는 한글로도 가능하다.)
- 먼저 열거타입 이름으로 소스파일을 생성한 뒤 한정된 값을 코드로 정의
- 열거타입 이름은 캐멀타입으로 지어주는 것이 관례이다.

어떠한 값만을 표현하고 싶다면 Enum을 사용하면된다.

```
enum Season {
  SPRING, SUMMER, FALL, WINTER
}
public class EnumTest1 {
  public static void main(String args[]) {
    Season data1 = Season.FALL;
    if (data1 == Season.FALL)
      System.out.println("당신이 좋아하는 계절은 가을!!");
////////////////////////////////////////////////////////////////////////
    switch (data1) {
    case SPRING:
      System.out.println("대저토마토");
      break;
    case SUMMER:
      System.out.println("복숭아");
      break;
    case FALL:
      System.out.println("홍로");
      break;
    case WINTER:
      System.out.println("레드향");
      break;
    }
////////////////////////////////////////////////////////////////////////
// Enum은 values가 자동으로 만들어진다.
// Season타입의 배열을 return한다
    for (Season v : Season.values())
```

```
      System.out.print(v + " ");
    System.out.println();

    Season data2 = Season.valueOf("SUMMER");
    System.out.println(data2);
  }
}
```

일반적으로 enum안에 일반메서드를 생성하지는 않는다.

만약 enum의 상수값 역시 바꿀 수 있으나 일반적으로 바꾸지않는다.

```
enum SeasonInit{
   SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울");
  // 상수값을 변경하기 위해서 필요한 코드
  private final String name;
   SeasonInit(String name){
      this.name = name;
   }
   String returnName(){
      return name;
   }
}
```