제7장 : 여러가지 파일 형식

- 데이터를 콤마로 나눈 형식
- StringTokenizer 클래스를 사용하면 토큰별로 데이터를 읽을 수 있어서 편리

```
String str = "홍길동,한석봉,신사임당";

// 콤마를 기준으로 자를 준비
StringTokenizer st = new StringTokenizer(str, ",");
while (st.hasMoreTokens()) {    // 다음 토큰이 있으면
  String token = st.nextToken();  // 토큰을 얻는다
  System.out.println(token);
}
```

프로퍼티 형식의 파일 읽기

- Properties 클래스를 사용하여 키(key)와 값(value)의 쌍으로 읽고 쓰기가 가능

data.properties 파일의 내용

```
heroName=홍길동
heroHp=100
```

```
Reader fr = new FileReader("data.properties");
Properties prop = new Properties();
prop.load(fr);    // 파일을 읽어들임
String name = prop.getProperty("heroName");
String hp = prop.getProperty("heroHp");
System.out.println("용사의 이름 : " + name);
System.out.println("용사의 HP : " + hp);
fr.close();
```

프로퍼티 형식의 파일 쓰기

- setProperty() 메서드로 데이터를 셋팅하고 store() 메서드로 저장

```
Writer fw = new FileWriter("data.properties");
Properties prop = new Properties();
prop.setProperty("heroName", "한석봉");
prop.setProperty("heroHp", "50");
prop.setProperty("heroMp", "30");
prop.store(fw, "용사의 정보")    // 파일 상단에 코멘트
fr.close();
```

XML 형식

- <> 태그를 활용한 기술 방식
- 포함관계를 기술할 수 있음

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- DOM Parser, SAX Parser 등을 통해 파서를 제작할 필요가 있음

JSON 형식

- 네트워크 통신에서 가장 많이 사용되고 있음
- XML에 비해 적은 용량
- [ ] 로 리스트, { } 로 객체를 표현
- 키(key): 값(value) 형태

```
{
    "이름": "홍길동",
    "나이": 25,
    "성별": "여",
    "주소": "서울특별시 양천구 목동",
    "특기": ["농구", "도술"],
    "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
    "회사": "경기 수원시 팔달구 우만동"
}
```

- JsonObject, JsonArray 클래스를 활용
- Gson 등의 라이브러리 활용을 권장

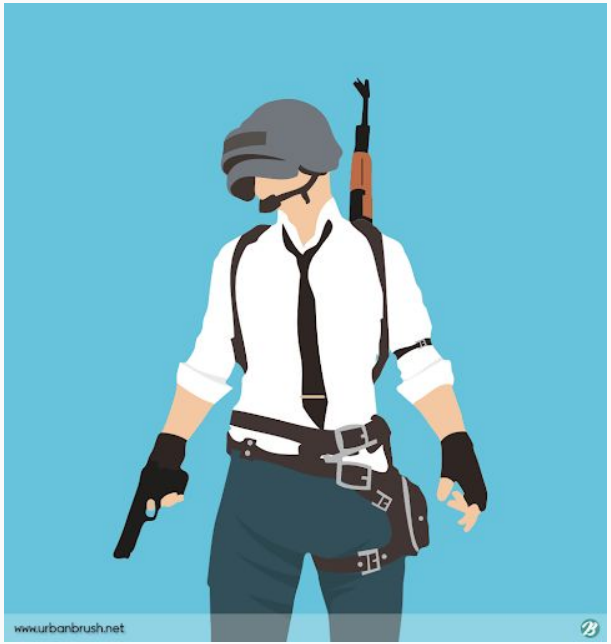
직렬화

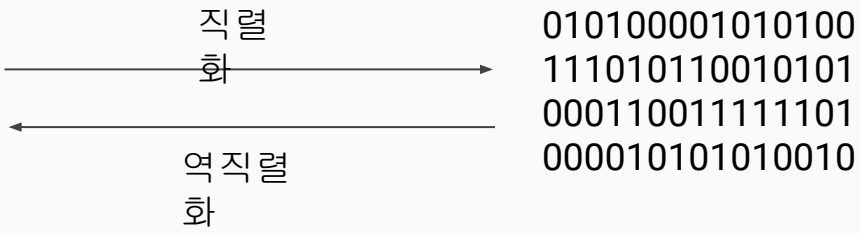
게임의 세이브 파일을 저장하기 위해 적절한 파일 형식은??

정말 그 파일 형식이 편리한가??

직렬화

- 직렬화를 통해 인스턴스를 바이트 배열로 상호 변환할 수 있음
- 직렬화 대상에는 Serializable 인터페이스를 구현해야 함
- 클래스 내부의 필드도 직렬화 대상이라면 모두 직렬화 처리를 해 줘야 함
- 클래스 설계의 변경에 대비하려면 직렬화 버전 UID를 선언한다





직렬화할 클래스


```
class Hero implements Serializable {
  private String name;
  private int hp;
  private Sword sword;

  // 생략
}
```

```
class Sword implements Serializable {

}
```

용사를 직렬화하여 저장


```
Hero hero = new Hero("홍길동", 75, 18);

// 저장
FileOutputStream fos = new FileOutputStream("save.dat");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(hero);
oos.flush();
oos.close();
```

용사를 복원


```
// 로드
FileInputStream fis = new FileInputStream("save.dat");
ObjectInputStream ois = new ObjectInputStream(fis);
Hero hero = (Hero) ois.readObject();
ois.close();
```

시리얼 버전 UID

- 게임이 버전업 되면서 용사 클래스의 구조가 변경되었다면 이전 버전의 세이브파일과 호환되지 않을 것이다
- 시리얼 버전 UID를 선언 해 두면 이런 경우 JVM에서 예외를 발생시켜 준다.
- long 값으로 아무값이나 정하면 된다.

```
class Hero implements Serializable {
  private static final long serialVersionUID = 81923983183821L;
```

연습문제 7-1

다음과 같은 내용이 담긴 파일 gradle.properties 파일이 있습니다.

```
org.gradle.jvmargs=-Xmx1536M
android.enableR8=true
android.useAndroidX=true
android.enableJetifier=true
```

이 파일을 읽어서 android.useAndroidX 값을 출력하시오.

연습문제 7-2

다음과 같은 사원 클래스와 부서 클래스가 있습니다.

```
class Employee {
  String name;
  int age;
}

class Department {
  String name;
  Employee leader;
}
```

총무부 리더 '홍길동(41세)'의 인스턴스를 생성하고 직렬화하여 company.dat 파일에 쓰는 프로그램을 작성하시오.
직렬화를 위해 위의 2개 클래스를 일부 수정해도 됩니다.