

# Java I/O Stream IV - Serialization

## 스트림을 통한 인스턴스의 저장

- `ObjectInputStream`
    - `InputStream in = new FileInputStream("data.bin");`  
`ObjectInputStream ois = new ObjectInputStream(in);`
    - Method  
`public final Object readObject() throws IOException, ClassNotFoundException`
    - Byte 형태로 저장된 인스턴스 데이터를 인스턴스로 복원
  - `ObjectOutputStream`
    - `OutputStream out = new FileOutputStream("data.bin");`  
`ObjectOutputStream oos = new ObjectOutputStream(out);`
    - Method  
`public final void writeObject(Object obj) throws IOException`
    - 인스턴스의 데이터를 Byte 형태로 변환 후 저장
  - 직렬화의 대상이 되는 인스턴스의 클래스는 `java.io.Serializable` 인터페이스를 구현해야 한다.  
`Serializable` 인터페이스의 구현은 '직렬화가 가능한 대상임'을 표시해주는 역할을 한다.
- 

## Serialization (직렬화)

인스턴스의 데이터를 일렬로 늘어선 연속적인 Byte 데이터로 변환하는 과정

## Deserialization (역직렬화)

일렬로 늘어선 연속적인 Byte 데이터를 인스턴스로 복원하는 과정

---

```
public class Circle implements Serializable {
    int xPos;
    int yPos;
    double rad;

    public Circle(int x, int y, double r) {
        xPos = x;
        yPos = y;
        rad = r;
    }

    public void showCircleInfo() {
        System.out.printf("[%d, %d] \n", xPos, yPos);
        System.out.println("rad : "+rad);
    }
}
```

```

}

public class SerializeDemo {
    private static final String FILE_PATH = "D:/Documents/Object.ser";

    public static void main(String[] args)
        throws IOException, ClassNotFoundException
    {
        // Srialize
        OutputStream out = new FileOutputStream(FILE_PATH);
        ObjectOutputStream oos = new ObjectOutputStream(out);

        oos.writeObject(new Circle(1, 1, 2.4));
        oos.writeObject(new Circle(2, 2, 4.8));
        oos.writeObject(new String("String implements Serializable"));

        oos.close();
        out.close();

        // Deserialize
        InputStream in = new FileInputStream(FILE_PATH);
        ObjectInputStream ois = new ObjectInputStream(in);

        // 저장된 순서대로 복원
        Circle cir1 = (Circle)ois.readObject();
        Circle cir2 = (Circle)ois.readObject();
        String str = (String)ois.readObject();

        ois.close();
        in.close();

        cir1.showCircleInfo();
        cir2.showCircleInfo();
        System.out.print(str);
    }
}

```

-----

**줄줄이 사탕처럼 읽어 들어갑니다.**

```

public class Point implements Serializable {
    int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Circle implements Serializable {
    Point p;
    double rad;

    public Circle(int x, int y, double r) {
        p = new Point(x, y);
        rad = r;
    }
}

```

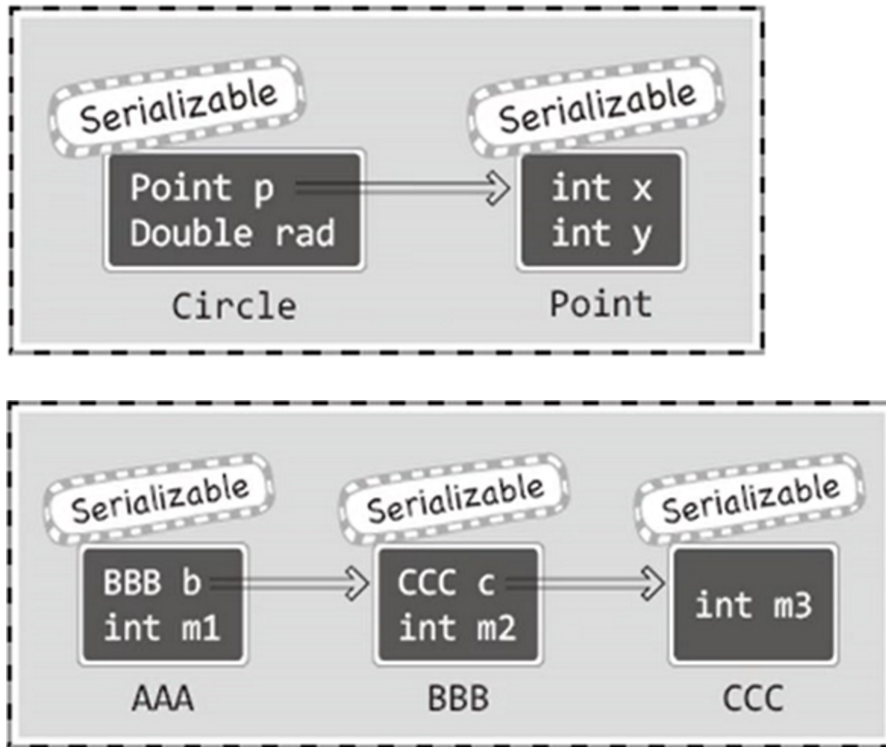
```

    }

    public void showCircleInfo() {
        System.out.printf("[%d, %d] \n", p.x, p.y);
        System.out.println("rad : "+rad);
    }
}

```

- **멤버인 p가 참조하는 인스턴스도 Serialize 인터페이스를 구현하면 (직렬화가 가능하다면) Circle 인스턴스가 직렬화 될 때, 같이 직렬화 된다!**



## 직렬화 대상에서 제외하기! transient


```

public class Circle implements Serializable{
    Point p;
    transient double rad; // default 값으로 직렬화가 진행된다. (0, 0.0, null.....)

    public Circle(int x, int y, double r) {
        p = new Point(x, y);
        rad = r;
    }

    public void showCircleInfo() {
        System.out.printf("[%d, %d] \n", p.x, p.y);
        System.out.println("rad : "+rad);
    }
}

```

 <terminated> SerializeDemo [Java Application] C:\#  
[[1, 1]  
rad : 0.0  
[2, 2]  
rad : 0.0  
String implements Serializable

---