

# u58 ● 直列化による転送モデル (EJBにおける分散オブジェクト)

JavaのクラスはSerializable (直列化インターフェース) をimplementsしたクラスを作成することで転送モデルが使えるようになります。

転送モデルとは、実体化したインスタンスを別のアプリケーションに転送できるということです。転送する手段としてファイルやネットワークを経由させることが可能です。

今回は、直列化クラスのインスタンスを一度ファイルに保存して、別のクラスから復元・表示してみたいと思います。

## 直列化クラスの作成

名前と時給と労働時間を設定すると10%の所得税を差し引いた手取り金額を計算するTedoriクラスを作ってみましょう。  
このクラスは直列化インターフェースを実装します。

以下作成場所は `~/libcon/WEB-INF/classes` ですよ！

### クラス図

```
----- //クラス名
Tedori (package:jp.ict.aso.model)
----- //フィールド (プロパティ)
- name:String
- payment:int
- time:double
----- //メソッド
+ Tedori(name:String,payment:int,time:double)
+ getName():String
+ getPayment():int
+ getTime():double
+ getTedori():int
-----
- private
+ public
```

## Tedori.java

```
192.168.56.200 - root@aso: /var/lib/tomcat9/webapps/book/WEB-INF/c VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) 漢字コード(K) ヘルプ(H)
1 package jp.ict.aso.model;
2 import java.io.Serializable;
3 public class Tedori implements Serializable{
4     private String name;
5     private int payment;
6     private double time;
7     public Tedori(String name,int payment,double time) {
8         this.name = name;
9         this.payment = payment;
10        this.time = time;
11    }
12    public String getName() {
13        return name;
14    }
15    public int getPayment() {
16        return payment;
17    }
18    public double getTime() {
19        return time;
20    }
21    public int getTedori() {
22        return (int)((payment*time)-(payment*time)*0.1);
23    }
24 }
25
"Tedori.java" 25L, 543B 25,0-1 All
```

## コンパイル

```
root@aso:classes# javac -d . Tedori.java
```

## インスタンスの保存

Serializable実装クラスを実体化してインスタンス情報をファイルに書き込みます。

(インスタンス情報を直列化したバイナリデータをファイルに書き込みます)

## TedoriWriteMain.java

```
192.168.56.200 - root@aso: /var/lib/tomcat9/webapps/book/WEB-INF/c VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) 漢字コード(K) ヘルプ(H)
1 package jp.ict.aso;
2 import java.io.FileOutputStream;
3 import java.io.ObjectOutputStream;
4 import jp.ict.aso.model.Tedori;
5
6 public class TedoriWriteMain {
7     public static void main(String[] args) throws Exception{
8         Tedori tedori = new Tedori("麻生",1500,3.5);
9         //オブジェクトをファイルに書き込む
10        try(ObjectOutputStream oos = new ObjectOutputStream(
11            new FileOutputStream("tedori.obj"))){
12            oos.writeObject(tedori);
13        }
14    }
15 }
16
"TedoriWriteMain.java" 16L, 480B 16,0-1 All
```

コンパイルと実行 (実行することによりインスタンスが書き込まれます)

```
root@aso:classes# javac -d . TedoriWriteMain.java
```

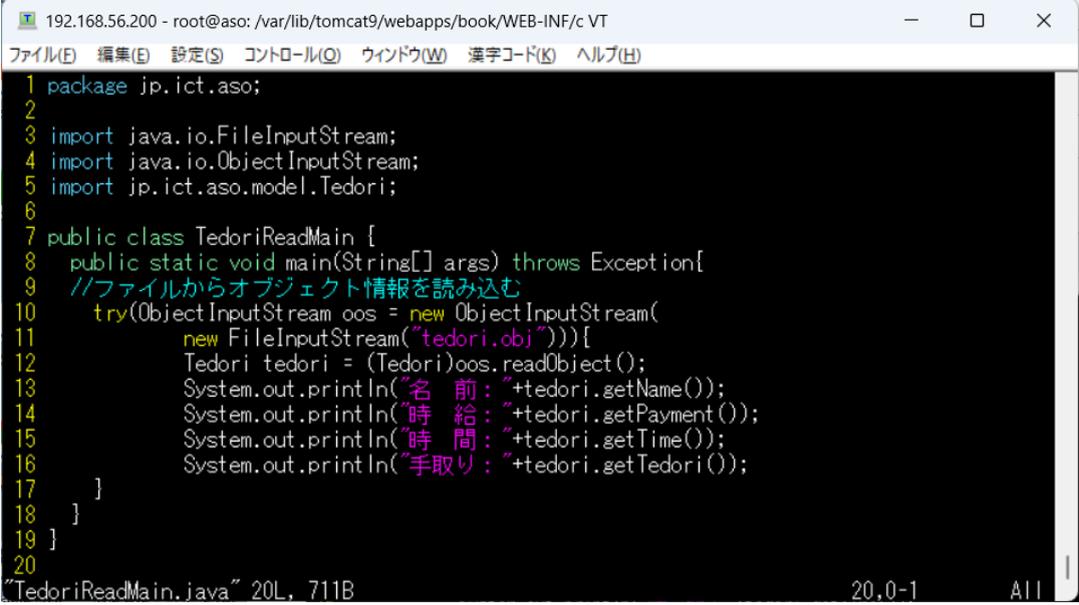
```
root@aso:classes# java jp.ict.aso.TedoriWriteMain
```

# インスタンスの読み出し

逆にバイナリデータからインスタンスを復元してみます。

(ファイルから直列化状態のインスタンスを読み込んで復元してみます)

## TedoriReadMain.java



```
192.168.56.200 - root@aso: /var/lib/tomcat9/webapps/book/WEB-INF/c VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) 漢字コード(K) ヘルプ(H)
1 package jp.ict.aso;
2
3 import java.io.FileInputStream;
4 import java.io.ObjectInputStream;
5 import jp.ict.aso.model.Tedori;
6
7 public class TedoriReadMain {
8     public static void main(String[] args) throws Exception{
9         //ファイルからオブジェクト情報を読み込む
10        try(ObjectInputStream oos = new ObjectInputStream(
11            new FileInputStream("tedori.obj"))){
12            Tedori tedori = (Tedori)oos.readObject();
13            System.out.println("名 前: "+tedori.getName());
14            System.out.println("時 給: "+tedori.getPayment());
15            System.out.println("時 間: "+tedori.getTime());
16            System.out.println("手取り: "+tedori.getTedori());
17        }
18    }
19 }
20
"TedoriReadMain.java" 20L, 711B 20,0-1 All
```

## コンパイルと実行 (実行することによりインスタンスが読み出されます)

```
root@aso:classes# javac -d . TedoriReadMain.java
```

```
root@aso:classes# java jp.ict.aso.TedoriReadMain
```

```
名 前 : 麻生
```

```
時 給 : 1500
```

```
時 間 : 3.5
```

```
手取り : 4725
```

このようにSerializableなクラスは、「バイト直列化」が可能になります。

今回はファイルにインスタンスを保存・復元しましたが、インスタンスの「バイト直列化」が可能ということは、Stream経由でインスタンス情報をネットワークに送受信することも可能になります。

これから行うEnterpriseJavaBeansの分散オブジェクトもこの機能で実現しています。