

Data Access Layer (DAL)

Sebastian Alejandro Velasco
Object Oriented Programming



Agenda

Serialization



Writing and
Reading Files

Serialization

Saving objects' state

Serialization and **plain Text** are two possible options to **save application states**.

Use serialization to write and read Java objects

If your data will be used by **only the Java program that generated it**, use **serialization**



Serialization save the **entire object state**

Objects must implement *Serializable* Interface

```
import java.io.Serializable;

public class Carreer implements Serializable {

    private int id;
    private int startYear;
    private String name;

    // ...
}
```

```
import java.io.Serializable;

public class Person implements Serializable {

    private Carreer carreer;
    private String name;
    private int age;

    // ...
}
```

```
// ...
public class SerializationSaveExample {

    public static void main(String[] args) {

        FileOutputStream fileStream = null;

        try {

            // If the file "MySaveFile.cbj" does not exist, it will be created
            // FileOutputStream knows how to connect (and create) a file
            fileStream = new FileOutputStream("MySaveFile.cbj");

            // Two test objects
            Person jack = new Person(new Career(5, 2005, "Soccer Player"), "Jack", 20);
            Person sunny = new Person(new Career(079, 1909, "Singer"), "Sunny", 40);

            // ObjectOutputStream let us write objects
            ObjectOutputStream os = new ObjectOutputStream(fileStream);

            // Serialize and write the object to file
            os.writeObject(jack);
            os.writeObject(sunny);

            // Close the stream
            os.close();

            System.out.println(jack);
            System.out.println(sunny);

            // ...

```

Serializable objects
can be written in a file

Serializable objects
can be loaded from a
file previously created

```
public class SerializationLoadExample {  
  
    public static void main(String[] args) {  
  
        FileInputStream fileStream = null;  
  
        try {  
            // If the file "MySaveFile.obj" does not exist, an exception will be thrown  
            fileStream = new FileInputStream("MySaveFile.obj");  
  
            // ObjectOutputStream let us load objects  
            ObjectInputStream os = new ObjectInputStream(fileStream);  
  
            // Load first two object from the file  
            Object firstPerson = os.readObject();  
            Object secondPerson = os.readObject();  
  
            // Cast from objects to Person  
            Person oldJack = (Person) firstPerson;  
            Person oldSunny = (Person) secondPerson;  
  
            System.out.println(oldJack);  
            System.out.println(oldSunny);  
  
            // ..  
        }  
    }  
}
```

Output - SerializationExample (run)

```
run:  
Person:  
    Name: Jack  
    Age: 20  
    Career:  
        Id: 5  
        Start Year: 2006  
        Name: Soccer Player  
  
Person:  
    Name: Sunny  
    Age: 40  
    Career:  
        Id: 879  
        Start Year: 1985  
        Name: Singer  
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
import java.io.Serializable;

public class Person implements Serializable {

    private Career career;
    private String name;
    private int age;

    // ...
}
```

Output - SerializationExample (run)

```
run:
Person:
  Name: Jack
  Age: 20
  Career:
    Id: 5
    Start Year: 2008
    Name: Soccer Player

Person:
  Name: Sunny
  Age: 40
  Career:
    Id: 879
    Start Year: 1989
    Name: Singer

BUILD SUCCESSFUL (total time: 0 seconds)
```

Object references
are stored within
the object

Writing and Reading Files

Use text plain file to write and read text strings

If your data will be used by other program write a **text plain file**



```

public class PlainTextSaveExample {

    public static void main(String[] args) {

        FileWriter writer = null;

        try {

            // Two test objects
            Person jack = new Person(new Carrear(5, 2005, "Soccer Player"), "Jack", 20);
            Person sunny = new Person(new Carrear(879, 1989, "Singer"), "Sunny", 40);

            // If the file "MyPlainText.csv" does not exist, it will be created
            // FileWriter knows how to connect (and create) a file
            writer = new FileWriter("MyPlainText.csv");

            // Write a string to the file
            writer.write(jack.toString());
            writer.write(sunny.toString());

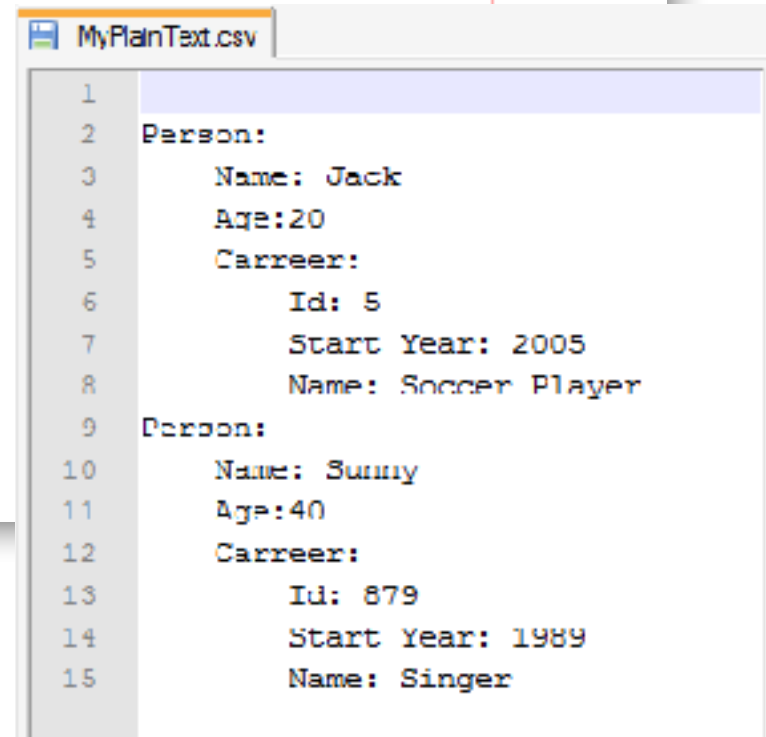
            // Close the file
            writer.close();

            System.out.println(jack);
            System.out.println(sunny);

            // ...

```

Text String can be stored in plain text files



The screenshot shows a text editor window titled 'MyPlainText.csv'. The content is as follows:

```


1
2 Person:
3     Name: Jack
4     Age:20
5     Carrear:
6         Id: 5
7         Start Year: 2005
8         Name: Soccer Player
9 Person:
10    Name: Sunny
11    Age:40
12    Carrear:
13        Id: 879
14        Start Year: 1989
15        Name: Singer

```

The generated files are human readable

Text String can be stored in plain text files

```
// Write a string to the file
writer.write(jack.getName() + ","
            + jack.getAge() + ","
            + jack.getCarreer().getId() + ","
            + jack.getCarreer().getName() + ","
            + jack.getCarreer().getStartYear() + "\n");
writer.write(sunny.getName() + ","
            + sunny.getAge() + ","
            + sunny.getCarreer().getId() + ","
            + sunny.getCarreer().getName() + ","
            + sunny.getCarreer().getStartYear() + "\n");
```



| MyPlainText.csv | |
|-----------------|------------------------------|
| 1 | Jack,20,5,Soccer Player,2005 |
| 2 | Sunny,40,879,Singer,1989 |
| 3 | |

We can adjust the output string to satisfy the requirements

Text String can be read from plain text files previously created

```
public class PlainTextLoadExample {  
  
    public static void main(String[] args) {  
  
        try {  
  
            // If the file "MyPlainText.csv" does not exist, it will be created  
            File myFile = new File("MyPlainText.csv");  
  
            // FileReader knows how to connect to a file  
            FileReader fileReader = new FileReader(myFile);  
            BufferedReader reader = new BufferedReader(fileReader);  
  
            String line = null;  
  
            // Read the file line by line  
            while ((line = reader.readLine()) != null) {  
                System.out.println(line);  
            }  
            reader.close();  
  
            // ...  
        }  
    }  
}
```

Output - SerializationExample (run)

```
run:  
Jack,20,5,Soccer Player,2005  
Sunny,40,879,Singer,1989  
BUILD SUCCESSFUL (total time: 0 seconds)
```

References

Trail: Internationalization <http://download.oracle.com/javase/tutorial/i18n/index.html>

[Sierra] K. Sierra and B. Bates, *Head First Java*, 2nd Edition, O'Reilly Media, 2005.