آموزش زبان برنامه نویسی جاوا

Serialization در جاوا

جلسه سی و پنجم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

خب اولین سوالی که در مورد مفهوم Serialization در جاوا برامون پیش میاد اینه که اصلا چی هست و کاربردش چیه؟!!!

Serialization مکانیزمی برای نوشتن و ذخیره یک شی در فایل می باشد.

ما تا اینجا با روش نوشتن و ذخیره داده های کاراکتری،رشته ای،عدد صحیح و... در فایل آشنا شدیم در این اموزش قصد داریم روش ذخیره و نوشتن یک شی در فایل را بررسی کنیم. همچنین در ادامه به روش خواندن یک شی از فایل می پردازیم.

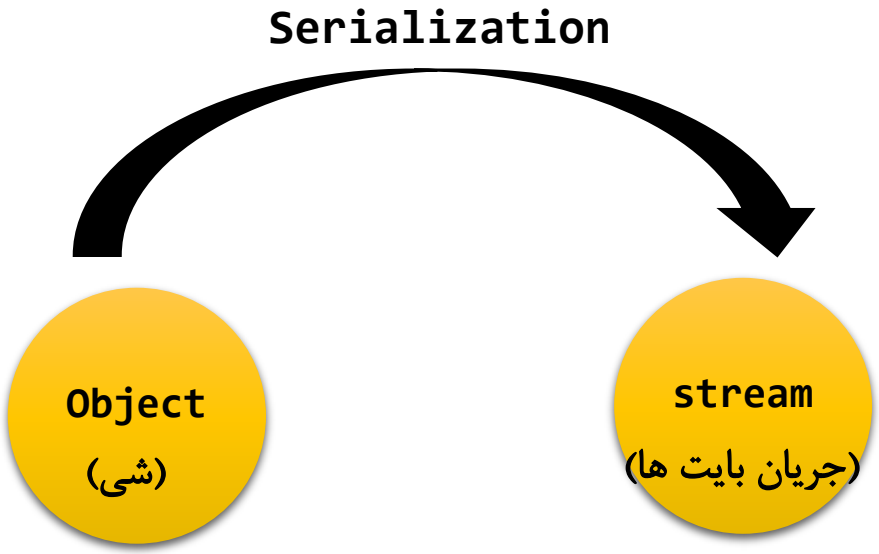
صرفا جهت یادگیری:

عمده استفاده از Serialization در تکنولوژی های Hibernate،RMI، JPA ، EJB وJMS در جاوا می باشد.

این تکنولوژی ها چی هستند و کاربردشون چیه رو کامل فراموش کنید بریم سراغ اصل مطلب یعنی Serialization.

Serialization

در Serialization همان طور که گفتیم یک شی را می توانیم در یک فایل ذخیره کنیم. چگونه ؟! Serialization برای این کار شی ما را به جریانی (stream) از byte ها تبدیل کرده و در فایل ذخیره می کند. تصویر(۱) را ببینید:



تصویر(۱)

ی لحظه یک نفس عمیق بکشید!!!! کامل پشت پرده این که چطور یک شی را در فایل ذخیره می کنیم رو بی خیال شید!!! تنها اینو بدونید که مفهوم Serialization به ما کمک می کند که یک شی از نوع یک کلاس با تمام ویژگی ها و حالت هایی که دارد را مستقیم در یک فایل ذخیره کنیم. ☺

خب در زیر گام هایی که برای ذخیره یک شی در فایل بر می داریم رو بررسی می کنیم:

گام اول:

اینترفیس Serializable:

کلاسی که قصد داریم از آن شی ایجاد کنیم و شی ایجاد شده را در یک فایل ذخیره کنیم را باید به اینترفیس **Serializable** ، implements کنیم. به این کار اصطلاحا **mark** دار (نشانه دار) کردن کلاسی که قصد داریم شی ای از نوع آن را در یک فایل ذخیره کنیم می گویند.

- برای implements کردن اینترفیس Serializable در کلاس خود باید پکیج **java.io.Serializable** را در برنامه خود import کنیم.

```
import java.io.Serializable;
```

- اینترفیس Serializable اینترفیسی است که متد و اعضایی ندارد.

```
public abstract interface java.io.Serializable {
}
```

نتیجه گیری :

در کل هر کلاسی که قصد داریم شی ای از آن در یک فایل ذخیره کنیم را باید به اینترفیس Serializable ، implements کنیم.

پس شی از نوع هر کلاسی را نمی توانیم در یک فایل ذخیره کنیم ، تنها اشیایی را می توانیم در فایل ذخیره کنیم که مجوزهای لازم یعنی پیاده سازی اینترفیس Serializable در کلاس آنها صورت گرفته باشد! ☺ که این کار mark دار کردن کلاس می گویند.

مثال: فرض کنید قصد داریم شی ای از کلاس Student را در یک فایل ذخیره کنیم، برای این کار ابتدا باید کلاس Student را به اینترفیس Serializable ، implements کنیم. یا کلاس Student را به اصلاح mark دار کنیم: ☺

```
package www.javapro.ir;
import java.io.Serializable;

public class Student implements Serializable{

}
```

- خب دیگه اشیایی که از نوع کلاس Student هستند ، مجوز این که در فایل ذخیره شوند را دریافت کرده اند. با این کار اشیای نوع کلاس Student به راحتی می توانند به جریانی (stream) از byte ها برای ذخیره در فایل تبدیل شوند.

گام دوم :

کلاس ObjectOutputStream :

در جلسات قبل برخی از کلاس های کار با فایل را بررسی کردیم ، کارایی اون کلاس ها تنها برای خواندن و نوشتن داده های اولیه نظیر بایت ها، رشته ها و... بود. حالا در این جلسه قصد داریم کلاس ObjectOutputStream که کاربرد آن برای نوشتن و ذخیره اشیا در فایل می باشد را بررسی کنیم.

- برای استفاده از کلاس **ObjectOutputStream** در برنامه خود باید پکیج java.io.ObjectOutputStream را import کنیم.

```
import java.io.ObjectOutputStream;
```

- برای نوشتن و ذخیره یک شی در فایل باید از کلاس **ObjectOutputStream** شی ایجاد کنیم و از طریق شی ایجاد شده به متد هایی که در زمینه ذخیره شی در فایل کاربرد دارد دسرسی پیدا کنیم.
- تنها اشیایی حق دارند در فایل ذخیره شوند که کلاس های آنها به اینترفیس Serializable ، implements شده باشند.

سازنده کلاس ObjectOutputStream :

public ObjectOutputStream(OutputStream out)

- یکی از سازنده های پر کاربردی که هنگام شی سازی از کلاس ObjectOutputStream صدا زده می شود.
- این سازنده به عنوان پارامتر یک شی از نوع کلاس OutputStream دریافت می کند که حاوی آدرس مکانی است که قراره در آن فایل ایجاد شود.

دو متّد مهم کلاس ObjectOutputStream:

1. public final void writeObject(Object obj) throws IOException{}

- این متد برای **ذخیره و نوشتن شی** در فایل استفاده می شود.
- به جای پارامتر obj که از نوع کلاس Object (پدر همه کلاس ها) هستش ، می توان شی مورد نظر خود را قرار دهیم.
- استفاده از این متد ممکنه باعث رخ دادن استثنای مربوطه شود ،پس باید استثنای آن را کنترل کنیم.

2. public void close() throws IOException {}

این متد نیز برای بستن فایل مورد نظر می باشد.

خب تا اینجا با شرایط و عملیاتی که از طریق آنها می توانیم یک شی را در یک فایل ذخیره کنیم آشنا شدیم:

۱. کلاسی که شی آن قراره در فایل ذخیره شود باید به اینترفیس Serializable ، implements شود.

۲. استفاده از کلاس ObjectOutputStream برای ذخیره اشیا در فایل.

مثال: در مثال زیر تمامی مفاهیمی که از ابتدای این جلسه تا اینجا را بررسی کردیم را بهش خواهیم پرداخت،پس اصلا نگران نباشید اگه توضیحات روشن نبود به مثال زیر خوب دقت کنید ☺

```
import java.io.Serializable;
 public class Student implements Serializable{
  int id;
  String name;
  public Student(int id, String name) {
   this.id = id;
   this.name = name;
  }
}
```

- در مثال بالا ما کلاس Student را به اینترفیس Serializable ، implements کرده ایم. با این کار کلاس Student به اصلاح mark دار شده و اشیای ان اجازه خواهند داشت در فایل ذخیره شوند.
- خب کلاس Student سری ویژگی و یک سازنده داره که این ویژگی ها رو مقداردهی می کند.

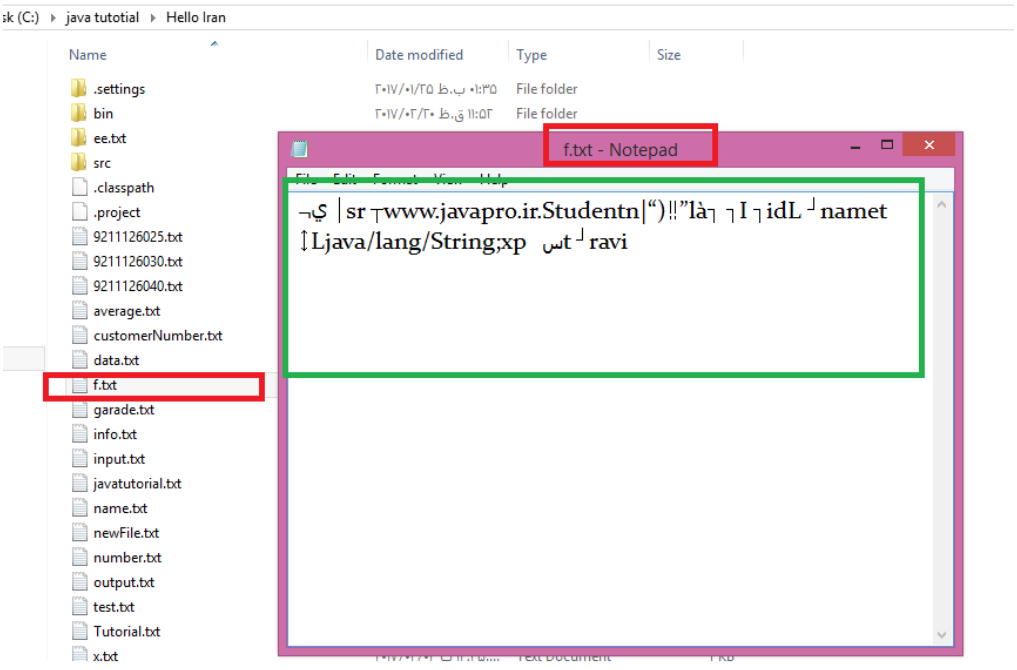
حال در کلاس **Persist** قصد داریم از کلاس **Student** شی بسازیم و شی ان را در یک فایل ذخیره کنیم:

```
    import java.io.*;
    class Persist{
     public static void main(String args[])throws Exception{
      Student s1 =new Student(211,"ravi");

      FileOutputStream fout=new FileOutputStream("f.txt");
      ObjectOutputStream out=new ObjectOutputStream(fout);

      out.writeObject(s1);
      out.close();
      System.out.println("success");
     }
}
```

خروجی:

وقتی کلاس های **Student** و **Persist** را با هم کامپایل و اجرا کنیم خروجی بصورت زیر خواهد بود:

۱. در کامپیوتر پوشه پروژمون یک فایل با نام و فرمت **f.txt** ایجاد می شود که حاوی یک شی از نوع کلاس **Student** می باشد. فایل **f.txt** که در پوشه پروژمون ایجاد شده را در تصویر(۲) مشاهده می کنید:



تصویر(۲)

در تصویر(۲) انگار فایلمون مثل همیشه نیست!!! و یک تفاوت هایی با فایل های قبلی که درونشون داده های متنی و عددی بود داره!! بله فایل هایی که درون آنها یک شی ذخیره می شود ، محتویات انها را نمیشه دید!! و وقتی این گونه فایلها را باز می کنیم با سری اطلاعات ظاهرا درهم بر هم برخورد می کنیم (رنگ سبزه درون تصویر(۲)). این گونه فایل ها را فقط از طریق برنامه و دستورات مربوطه می شود خواند و با چشم بصیرت تنها می شود دید ☺

۲. در خروجی کنسول نیز پیام زیر مبنی بر موفق بودن عملیات ایجاد فایل نمایش داده می شود:

```
success
```

نکته بسیار مهم : وقتی یک شی در یک فایل ذخیره می شود ، تمام ویژگی و حالات شی نیز در فایل حفظ و ذخیره می شود. مثلا در این مثال شی `s1` که از نوع `Student` هستش دو ویژگی `name` و `id` آن که به ترتیب `"ravi"` و `211` می باشد ، هنگام ذخیره سازی شی `s1` در فایل `f.txt` تمام این دو ویژگی در فایل ذخیره و حفظ می شود.

توضیحات:

- کلاس `Student` را که از قبل توضیح داده ایم.از آنجایی که قصد داریم شی از نوع این کلاس را در فایل ذخیره کنیم باید حتما کلاس `Student` به اینترفیس `Serializable` ، `implements` شود.

- کلاس `Persist` در متد `main` ان یک شی با نام `s1` از کلاس `Student` ایجاد کرده و سازنده ان را مقداردهی کرده ایم.

```
Student s1 =new Student(211,"ravi");
```

- برای نوشتن یک شی در فایل باید از کلاس `ObjectOutputStream` شی ایجاد کنیم. هنگام شی سازی از این کلاس ، نیاز هست که پارامتر سازنده آن را مقدار دهی کنیم.همان طور که از قبل گفتیم نوع پارامتر سازنده کلاس `ObjectOutputStream` از نوع کلاس `OutputStream` هستش ، از آنجایی که کلاس `OutputStream` یک کلاس `abstract` (انتزاعی) هستش، نمی توان از ان شی ایجاد کنیم پس تنها راهی که می ماند از یکی از فرزندان کلاس `OutputStream` شی ساخته و جایگزین پارامتر درون سازنده کلاس `OutputStream` کنیم.

- پس همان طور که در زیر می بینید از کلاس FileOutputStream که فرزند کلاس OutputStream هستش شی ای با نام fout ساخته و درون سازنده کلاس ObjectOutputStream ریخته ایم.
- آدرس "f.txt" ، ادرس محل ذخیره فایل در کامپیوتر که در اینجا پوشه پروژمون می باشد.
- f نام فایل و txt فرمت فایل ما می باشد.
- شی ما در این فایل ذخیره می شود.

```
FileOutputStream fout=new FileOutputStream("f.txt");
      ObjectOutputStream out=new ObjectOutputStream(fout);
```

- در این خط شی s1 که از نوع کلاس Student هستش با استفاده از متد writerObject در فایل ذخیره می شود. به همین راحتی! ☺

```
out.writeObject(s1);
```

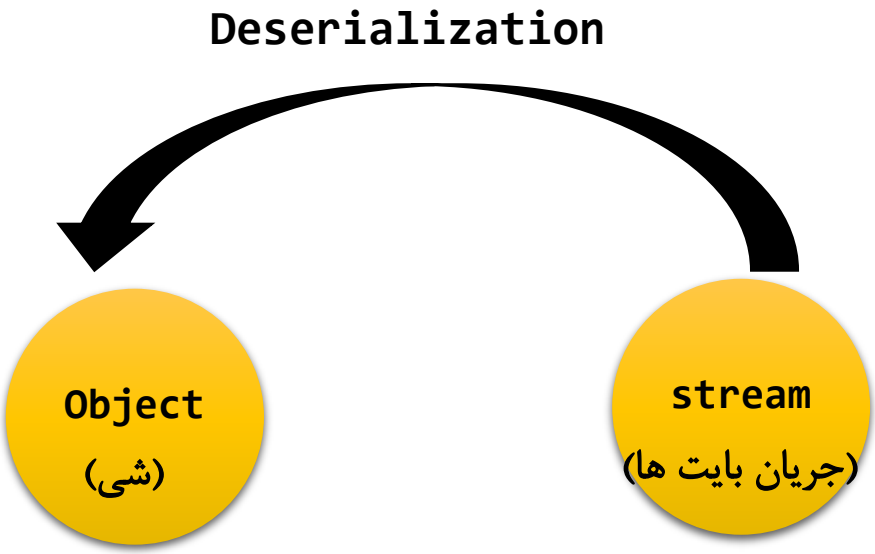
- تنها اشیایی حق دارند در فایل ذخیره شوند که کلاس های آنها اینترفیس Serializable را implemets کرده باشند.

در پایان با استفاده از متد close فایل out را می بندیم و حالا یک پیامم چاپ میکنیم که یعنی عملیات با موفق بوده است ☺

```
      out.close();
      System.out.println("success");
```

Deserialization در جاوا

Deserialization عکس عملیات و مفهوم serialization می باشد. Deserialization جهت بازسازی و خواندن اشیا از فایل می باشد. در Deserialization جریان بایت های ذخیره شده در فایل به شی تبدیل می شود. تصویر(۳)



تصویر(۳)

کلاس ObjectInputStream :

- این کلاس همان طور که از اسمش پیداست برای خواندن شی ذخیره شده در فایل می باشد.
- برای استفاده از کلاس ObjectInputStream باید پکیج java.io.ObjectInputStream را در برنامه خود import کنیم:

```
import java.io.ObjectInputStream;
```

سازنده کلاس ObjectInputStream

```
public ObjectInputStream(InputStream in) throws IOException{}
```

- برای ایجاد یک شی از کلاس **ObjectInputStream** سازنده این کلاس را صدا زده و پارامتر آن را مقداردهی می کنیم.
- پارامتر سازنده کلاس **ObjectInputStream** از نوع کلاس InputStream می باشد.

- از انجایی که کلاس InputStream یک کلاس abstract و انتزاعی می باشد ، نمی توان از آن شی ایجاد کرد، بجای آن می توان اشیای یکی از فرزندان کلاس InputStream را جایگزین پارامتر سازنده کلاس **ObjectInputStream** کنیم.

معرفی متدهای پرکاربرد کلاس ObjectInputStream:

- **public final Object readObject() throws IOException, ClassNotFoundException{}**

❖ این متد برای خواندن اشیا از فایل کاربرد دارد.

❖ هنگام استفاده از این متد احتمال رخ دادن دو استثنای IOException و ClassNotFoundException وجود دارد که باید آنها را کنترل کرد.

- **public void close() throws IOException{}**

❖ این متد برای بستن فایل استفاده می شود. و هنگام استفاده باید استثنای آن کنترل شود.

مثال: در مثال زیر قصد داریم شی Student که در مثال قبل در فایل f.txt ذخیره کردیم را بخوانیم، و ویژگی های آن شی یعنی name و id آن را در خروجی کنسول چاپ کنیم:

```
package www.javapro.ir;
import java.io.*;
class Depersist{
 public static void main(String args[])throws Exception{

  ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
  Student s=(Student)in.readObject();
  System.out.println(s.id+" "+s.name);

  in.close();
 }
}
```

خروجی:

```
211 ravi
```

- همان طور که مشاهده می کنید شی Student که در مثال قبل با ویژگی های name=”ravi” و id=211 را از فایل خواندیم و در خروجی کنسول چاپ کرده ایم.
- پس هر به همراه شی تمام ویژگی ها و حالات شی نیز حفظ و در فایل ذخیره می شود.

توضیحات:

```
ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
```

- در این خط کد یک شی از کلاس ObjectInputStream ایجاد کرده ایم.
- درون سازنده کلاس ObjectInputStream مستقیما یک شی از کلاس FileInputStream ایجاد کرده و پارامتر سازنده کلاس FileInputStream را آدرس محلی که فایل در آن قرار دارد را گذاشتیم.
- از آنجایی که FileInputStream فرزند کلاس InputStream است ، اجازه داریم شی ای از نوع کلاس FileInputStream جای پارامتر سازنده کلاس ObjectInputStream قرار دهیم.

```
Student s=(Student)in.readObject();
```

- در این خط کد با استفاده از متد readObject شی درون فایل f.txt را می خوانیم.
- متد readObject هنگامی که فایل را می خواند یک مقدار از نوع کلاس Object که پدر همه کلاس های ما می باشد را به ما پس می دهد، و شی برگردانده شده از فایل مشخص نیست دقیقا مربوط به شی کدام کلاس می باشد، به همین خاطر ما همیشه کنار متد readObject یه پرانتز باز و بسته می کنیم و درونش نوع کلاسی که شی از نوع ان می باشد را قرار می دهیم. با این کار شی ای که از فایل خوانده شده اگر از نوع کلاس درون پرانتز باشد ، به کلاس درون پرانتز تبدیل می شود.و ما می توانیم ازش استفاده کنیم که در اینجا این شی را درون یک متغیر مثل s که از نوع کلاس Student هستش ریخته ایم.

نکته بسیار بسیار مهم!!! : هنگام خواندن یک شی از فایل با استفاده از متد readObject الزامی است که در کنار این متد و درون یک پرانتز نوع شی را مشخص کنیم:

```
in.readObject()(مشخص کردن نوع شی)
```

- in در اینجا شی ای از نوع کلاس ObjectInputStream می باشد.

❖ همیشه هنگام خواندن شی از فایل ، نوع شی که از نوع یک کلاس می باشد را درون یک پرانتز در کنار متد readObject آورده می شود. که به عمل cast کردن در جاوا میگن که فعلا فراموشش کنید در جلسه ای مجزا به cast کردن می پردازیم.

Serialization همراه با ارثْ بری در جاوا :

اگر یک کلاس اینترفیس serializable را implements کرد و به اصلاح mark دار شد، تمام کلاس های فرزند (زیرکلاس های) ان نیز serializable (mark دار) خواهند بود.یعنی تمامی اشیای کلاس های فرزند آن می توانند در فایل ذخیره شوند.

مثال:

```
package example.javalike;
import java.io.Serializable;
class Person implements Serializable{
 int id;
 String name;
 Person(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

```
package example.javalike;
class Student extends Person{
 String course;
 int fee;
 public Student(int id, String name, String course, int fee) {
  super(id,name);
  this.course=course;
  this.fee=fee;
 }
}
```

❖ در مثال بالا کلاس Person اینترفیس Serializable را implements کرده است.

❖ حال کلاس Student کلاس Person را به ارث برده است.با این کار از انجایی که کلاس Student فرزند کلاس Person هستش ،کلاس Student نیز Serializable و mark دار شده و اشیای ان می تواند در فایل ذخیره شوند.

مثال بعد تغییر یافته مثال قبل می باشد:

مثال:

```
package example.javalike;
import java.io.Serializable;
class Person implements Serializable{
 int id;
 String name;
 Person(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

```
package example.javalike;
class Student extends Person{
 String course;
 int fee;
 public Student(int id, String name, String course, int fee) {
  super(id,name);
  this.course=course;
  this.fee=fee;
}
}
```

```
package example.javalike;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Test {

     public static void main(String[] args) {
          Student s2 = new Student(123, "jafar", "A", 3000);
```

```
            try {
                  FileOutputStream fout = new
FileOutputStream("objectStudent.txt");
                  ObjectOutputStream out = new ObjectOutputStream(fout);
                  out.writeObject(s2);

            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

      }

}
```

خروجی:

خروجی این برنامه یک فایل با نام و فرمت objectStudent.txt هستش، حاوی شی از نوع کلاس Student می باشد.

- با وجود این که کلاس Student اینترفیس Serializable را implements نکرده اما چون کلاس پدر آن یعنی Person اینترفیس Serializable را implements کرده ، اشیای آن می توانند در فایل ذخیره شوند.
- دو کلاس Person و Student که تغییری نکرده و در مثال قبل توضیح داده ایم. در کلاس Test نیز از کلاس Student شی ساخته و درون فایل ذخیره کرده ایم.

نتیجه گیری : اگر یک کلاس اینترفیس Serializable را implements کند، اشیای تمام فرزندان این کلاس می توانند در فایل ذخیره شوند.

Serialization همراه با ویژگی های static یک کلاس:

- اگر ویژگی از یک کلاس از نوع `static` باشد،`Serialization` برای ان صادق نیست!! (نمی توان از آن سریال سازی کرد) یعنی دیگه نمی توان شی ای از آن را در یک فایل ذخیره کرد، زیرا یک متغیر `static` تنها بخشی از یک کلاس است نه یک شی.
- در کل اشیای کلاسی که دارای ویژگی های `static` باشد را نمی توان در یک فایل ذخیره کرد.

با مفهوم `Serialization` در جاوا آشنا شدیم ،نیاز به حل کردن مثال و تمرین بیشتر داریم که سعی میکنم مثالهای بیشتری طراحی و در سایت و کانال قرار دهم،امیدوارم روشن توضیح داده باشم.

پیروز و موفق باشید
