Enthuware Mobile Test Studio

Home | Certifications | **Test** | Review | Progress | Notes

Standard Tests - Last Day Test (Unique) : 2019-10-24 19:49

Q 38 of 86 Mark I/O Fundamentals - Serialization enthuware.ocpjp.v8.2.1703

Consider the following class:

```
public class Student implements Serializable{
    public static final long serialVersionUID = 1;
    public String name;
    public String grade;
    public String toString(){ return "["+name+", "+grade+"]"; }
}
```

An object of this class was created as follows and was serialized to a file `c:\temp\bob.ser`:

```
        Student s = new Student();
        s.name = "bob";
        s.grade = "10";
```

After some time the Student class was changed as follows:

```
public class Student implements Serializable{
    public static final long serialVersionUID = 1;
    public String id="S111";
    public String name;
    public String grade;
    public int age=15;
    public String toString(){ return "["+id+", "+name+", "+grade+", "+age+"]"; }
}
```

Now, the serialized file is read back as follows:

```
   FileInputStream fis = new FileInputStream("c:\\temp\\bob.ser");
   ObjectInputStream is = new ObjectInputStream(fis);
   s = (Student) is.readObject();
   is.close();
   System.out.println("Loaded "+s);
```

What will it print?

Answered Incorrectly You had to select 1 option(s)

- It will throw an exception while deserializing the file.
- **Loaded [null, bob, 10, 0]**

  Since the serialVersionUID of the serialized class and the new class are same, the file will be deserialized without any issue. The new fields will be initialized to their Java defaults (because constructors and initializers are not invoked during deserialization). So the values for id and and age will remain null and 0 respectively.
- Loaded [S111, bob, 10, 15]
- **It will have unpredicable values for id and age.**

Previous Next Evaluate Finish Review

NOTE: Some candidates have reported getting a question on `serialVersionUID`. So you should understand the following points:

1. When a file is deserialized into an object, the class's constructor and instance initializers are not called. So the fields

for which no value is available in the serialized file, are initialized to their default values (i.e. number fields to `0`, boolean to `false`, and references to `null`).

2. `serialVersionUID` denotes the version number of the class. If you don't specify `serialVersionUID` for a class that implements Serializable, Java compiler automatically adds this field. It computes a value based on the attributes of the class such as the fields and interfaces, and assigns that value to `serialVersionUID`.

It is used during deserialization to verify that the sender and receiver of a serialized object have loaded classes for that object that are compatible with respect to serialization. If the receiver has loaded a class for the object that has a different `serialVersionUID` than that of the corresponding sender's class, then deserialization will result in an `InvalidClassException`.

3. If the `serialVersionUID` for the serialized object and the actual class is same then the deserialization completes successfully but any additional fields that are not present in the serialized file are initialized to their default value (as per point 1. above). Any fields that are missing in the class but are present in the serialized file are ignored.

Add/Edit Note