Enthuware Mobile Test Studio

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

Standard Tests - Test 5 : 2019-10-16 07:18

Q 16 of 80 □ Mark I/O Fundamentals - Serialization enthuware.ocpjp.v8.2.1701

What will the following program print when compiled and run?

```
class Boo implements Serializable {
    transient int ti = 10;
    static int si = 20;
}

public class TestClass
{
    public static void main(String[] args) throws Exception
    {

        Boo boo = new Boo();
        boo.si++;
        System.out.println(boo.ti+" "+boo.si);
        FileOutputStream fos = new FileOutputStream("c:\\temp\\boo.ser");
        ObjectOutputStream os = new ObjectOutputStream(fos);
        os.writeObject(boo);
        os.close();

        FileInputStream fis = new FileInputStream("c:\\temp\\boo.ser");
        ObjectInputStream is = new ObjectInputStream(fis);
        boo = (Boo) is.readObject();
        is.close();
        System.out.println(boo.ti+" "+boo.si);
    }
}
```

Answered Incorrectly You had to select 1 option(s)

○ It will not compile.

○ It will throw an exception at run time.

○ 10 21
10 21

○ 10 21
10 20

● **10 21**
0 20

○ **10 21**
0 21

Remember that transient fields and static fields are never serialized. Constructor, instance blocks, and field initialization of the class being deserialized are also not invoked. So, when boo is deserialized, the value of `ti` is set to 0.

The class `Boo` is loaded as soon as the code refers to the class (here, it happens at `Boo boo = new Boo();`), and so the static int `si` is initialized to the value given in the class code i.e. 20 and then it is incremented to 21 because of `boo.si++;`. This part has nothing to do with serialization. So when you deserialize an instance of `Boo, Boo.si` is not affected and is not reset to 20.

Therefore, if you run the program again with just the deserialization part, you will see that si is 20 and not 21.

Previous Next Evaluate Finish Review

Add/Edit Note