



Java and C# in Depth

Carlo A. Furia, Marco Piccioni, Bertrand Meyer

Exercise Session – Week 9



Today, pitfalls and surprises w.r.t. persistence

- JDBC
- LINQ
- Serialization (Java and C#)
- OODBMS db4o

Quiz 1: scrolling a *ResultSet* (JDBC)



How do you assess the following code snippet that iterates through a *ResultSet*?

```
ResultSet rs = stmt.executeQuery("SELECT...");
while(rs.next())
{
    String firstColumnInfo = rs.getString(0);
    String secondColumnInfo = rs.getString(1);
    System.out.println("Fetched info:" + firstColumnInfo + ";" +
        "secondColumnInfo");
}
...
```

Quiz 1: Solution



An exception is thrown: rows in a *ResultSet* start from 1

...

```
ResultSet rs = stmt.executeQuery("SELECT...");
```

```
while(rs.next())
```

```
{
```

```
    String firstColumnInfo = rs.getString(0);
```

```
    String secondColumnInfo = rs.getString(1);
```

```
    System.out.println("Fetched info:" + firstColumnInfo + ";" +  
        "secondColumnInfo");
```

```
}
```

...

Quiz 2: What's the problem here? (JDBC)

```
public String getPassword(String name) throws
    ApplicationException{
    try {
        con = //get connection here;
        stmt = con.createStatement();
        rs = stmt.executeQuery("Query here");
        while (rs.next()) { password=rs.getString(1); }
        rs.close(); stmt.close(); con.close();
    } catch (SQLException ex) {
        throw new ApplicationException ("Couldn't run query [" +
            sql + "]", ex);
    }
    return password;
}
```

Quiz 2: Solution



```
public String getPassword(String name) throws ApplicationException {
    try {
        //as before, but "con" not local anymore
        rs.close(); stmt.close();
    } catch (SQLException ex) {
        //as before...
    } finally {
        try {
            if (con != null) { con.close(); }
        } catch (SQLException ex) {
            throw new ApplicationException ("Failed to close
            connection", ex);
        }
    }
    return password;
}
```

Quiz 3: What is printed? (LINQ)



```
List<int> numbers = new List<int>() { 1, 2 };  
IEnumerable<int> sequence =  
    (from n in numbers select n * 10);  
foreach (int n in sequence) Console.Write(n + "|");
```

10|20|

```
numbers.Add (3);  
foreach (int n in sequence) Console.Write(n + "|");
```

10|20|30|

LINQ queries are evaluated lazily!

You can "freeze" the result of a query by calling:

```
IEnumerable<int> sequence = (from n in numbers select n *  
    10).ToList();
```

Quiz 4: What is printed? (LINQ)



```
IEnumerable<char> query = "Not what you might expect!";  
foreach (char vowel in "aeiou")  
    query = (from c in query where c != vowel select c);  
foreach (char c in query) Console.Write(c);
```

"Not what yo might expect!"

When the query is executed *vowel* has value *u* (we delete *u* multiple times, 1 in this case)

```
foreach (char vowel in "aeiou") {  
    char temp = vowel;  
    query = (from c in query where c != temp select c);  
}  
foreach (char c in query) Console.Write(c);
```

"Nt wht y mght xpct!"

Quiz 5: Serialization (Java)



```
class Student implements Serializable {  
    private String name;  
    private int birthYear;  
    transient private int age = 19;  
    ...  
}
```

Exception handling omitted

```
Student student = new Student ("B. Meyer", 1950);  
ObjectOutputStream out = new ObjectOutputStream(...);  
out.writeObject(student );  
out.close();  
...
```

```
ObjectInputStream in = new ObjectInputStream(...); // the same file  
student = (Student) in.readObject();  
in.close();  
System.out.println(student);
```

B. Meyer 1950 (age 0)

Initializers (as well as constructors)
are omitted during deserialization

Quiz 5: How to make it work?



```
class Student implements Serializable {  
    private String name;  
    private int birthYear;  
    transient private int age;
```

```
    ...
```

```
// The following method is not necessary in this example
```

```
    private void writeObject(ObjectOutputStream out) throws  
        IOException {  
        out.defaultWriteObject(); }  
}
```

```
    private void readObject(ObjectInputStream in) throws IOException,  
        ClassNotFoundException {  
        in.defaultReadObject(); // Reads previously serialized fields  
        calculateAge(); // Calculates age from birthYear and current year  
        and assigns it to the age attribute  
    }  
}
```

Quiz 5: Serialization (C#)



```
[Serializable] class Student {  
    public string name;  
    public int birthYear;  
    [NonSerialized] public int age = 19;  
}
```

```
Student student = new Student() { name = "B. Meyer",  
    birthYear = 1950, age = 63 };
```

```
IFormatter formatter = new BinaryFormatter();  
using (FileStream fs = File.Create("my.bin"))  
    formatter.Serialize(fs, student);
```

```
using (FileStream fs = File.OpenRead("my.bin")) {  
    student = (Student)formatter.Deserialize(fs);  
    Console.WriteLine(student);  
}
```

B. Meyer 1950 (age 0)

Initializers (as well as constructors)
are omitted during deserialization

Quiz 5: How to make it work?



```
[Serializable] class Student {  
    public string name;  
    public int birthYear;  
    [NonSerialized] public int age;  
  
    [OnDeserialized]  
    private void ComputeAge(StreamingContext context)  
    {  
        age = DateTime.Now.Year - birthYear;  
    }  
}
```

Quiz 5: XML serialization (C#)



```
public class Student {  
    public string name;  
    public int birthYear;  
    [XmlIgnore] public int age = 19;  
}
```

B. Meyer 1950 (age 19)

XML deserialization invokes the default constructor and initializers

```
Student student = new Student() { name = "B. Meyer", birthYear =  
    1950, age = 63 };
```

```
XmlSerializer xs = new XmlSerializer(typeof(Student));  
using (Stream s = File.Create("my.xml"))  
    xs.Serialize(s, student);
```

```
using (Stream s = File.OpenRead("my.xml"))  
    student = (Student)xs.Deserialize(s);  
Console.WriteLine(student);
```

Quiz 6: Schema evolution (Java)



Suppose we added a method to class *Student*:

```
class Student implements Serializable {  
    public void subscribe(Course c) { ... }  
    ... // The rest as before  
}
```

What happens if we try to deserialize a student from a file, created on previous slides?

InvalidClassException

If *serialVersionUID* is not defined explicitly, even a small change to the class code leads to incompatibility

Quiz 7: db4o updates



```
class StudyTrack {  
    private int code;  
    private String name;  
    ...  
}
```

```
class Student {  
    private StudyTrack track;  
    private String name;  
    ...  
}
```

```
ObjectContainer db=Db4o.openFile(...);
```

Exception handling omitted

```
StudyTrack se = new StudyTrack (117, "Software Engineering");
```

```
StudyTrack is = new StudyTrack (118, "Information Security");
```

```
db.store(new Student(se, "Sheldon"));
```

```
db.store(new Student(is, "Penny"));
```

```
db.close();
```

```
// see next slide...
```

Quiz 7: What is printed?



```
List<Student> result=db.query(new Predicate<Student>() {  
    public boolean match(Student s){  
        return s.getName().equals("Sheldon");  
    }  
});
```

```
Student found=result.get(0);  
found.getTrack().setCode(666);  
db.store(found);
```

// In another session:

```
List<Student> result = db.query (... // the same predicate)  
Student found=result.get(0);  
System.out.println (found.getTrack().getCode());
```

117

As a default, child objects are updated automatically only until depth 1. To activate the whole object structure would be too expensive

Quiz 7 Solution



```
Db4o.configure().objectClass("Student").  
    cascadeOnUpdate(true); // before opening a db
```

```
ObjectContainer db=Db4o.openFile(...);  
List<Student> result=db.query(new Predicate<Student>() {  
    public boolean match(Student s){  
        return s.getName().equals("Angela Merkel");  
    }  
});  
Student found=result.get(0);  
found.getTrack().setCode(666);  
db.store(found);
```

Now the track is updated, regardless of the depth

Quiz 8: db4o transactions



```
List<Student> result=db.query(new Predicate<Student>() {  
    public boolean match(Student s){  
        return s.getName().equals("Sheldon");  
    }  
});  
Student found=result.get(0);  
found.prependTitle("Dr."); // prepends title to the name  
db.store(found);  
... // Oops, she didn't pay the exam fee!  
db.rollback();  
System.out.println(found.getName());
```

Dr. Angela Merkel

"Live" (memory) objects are not rolled back!
Call *db.ext().refresh* to refresh memory explicitly