Triton recruitment Alentaris

Java – Technical interview answers

07/05/2013

Stéphane Robert

Introduction

All my source codes are available here on GitHub :

https://github.com/srobert72/ForTestOnly/

You'll find a Java project directory « Triton » and this present document.

Java project is based on JDK 1.6.0.35, Maven3 and Eclipse Juno.

To compile :

```
cd Triton

mvn compile
```

To launch all JUnit Tests :

```
cd Triton

mvn test
```

Exercice 1 : Unit testing

Asked class is implemented here :

main/java : com.triton.DiskFreeSpaceUtils.java

We can see it in action in TestCase here :

test/java : com.triton.ExerciceDisk.java

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceDisk
```

This implementation uses only JDK but we could use Jakarta IOUtils that is often used as dependencies in many projects.

Exercice 2 : Spring

XML Spring fragment explanations :

This fragment will instanciate a Bean of Java Class « com.stw.core.person.Person » and its ID is «myTestBean ». To instanciate it, the constructor called is « createNew() » by passing 2 arguments « firstName » and « phone ».

We can assume this constructor signature is something like :

```
public static Person createNew(String firstName, String phone) ;
```

Value setted to « firstName » is a variable maybe read from a Properties file.

Value setted to « phone » is another Bean with ID « phone ».

This Bean must be instanciate after another Bean named « dataSource » is instanciated.

This Bean has scope « prototype » which means each time this Bean is used as reference a new instance will be created. It is not a Singleton.

Exercice 3 : Fib

Fibonacci has been implemented by an abstract class with 2 sub classes :

```
main/java : com.triton.AbstractFibonacci.java

main/java : com.triton.FibonacciRecurcive.java

main/java : com.triton.FibonacciNonRecurcive.java
```

We can see it in action in TestCase here :

```
test/java : com.triton.ExerciceFib.java
```

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceFib
```

Advantages of using non-recursive function is mainly performance. It drastically reduces heap space and avoid problems like : **java.lang.OutOfMemoryError: Java heap space**. Because each call to a function in recursive mode is stored before application could give complete result.

Memory usage and performances should be improved using non-recursive function.

To mesure this improvement in our particular Fibonacci example, I wrote the 2 implementations and compare them in JUnit TestCase.

Exercice 4 : Incrementing an integer

Incrementing an integer using 3 different thread safe ways has benn implemented in a class :

main/java : com.triton.IncrementalValue.java

It contains 3 different integers and 3 ways to increment them :

- Using a synchronized function
- Using a volatile integer
- Using an AtomicInteger (java.util.concurrent.atomic.AtomicInteger)

We can see it in action in TestCase here :

test/java : com.triton.ExerciceIncrement.java

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceIncrement
```

Exercice 5 : StwAndo

StwAndo has been implemented in a TestCase here :

test/java : com.triton.ExerciceStwAndo.java

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceStwAndo
```

Exercice 6 : Algorithm

NumberToWords has been implemented by an abstract class with 1 sub class :

main/java : com.triton.AbstractNumberToWords.java

main/java : com.triton.EnglishNumberToWords.java

It is a first step if we want to implement other languages.

We can see it in action in TestCase here :

test/java : com.triton.ExerciceNumberToWords.java

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceNumberToWords
```

Exercice 7 : Java

To implement asked functionnality I wrote a class that is a wrapper over a java.util.Map. It is as generic as possible in the way you provide your own Map<K,V> (where K and V can be any type of classes) in wrapper's constructor.

This technique uses no external libray. But we could also implement it by AOP. In this case, there would be less code to write, but it would need more dependencies.

MapWithStat has been implemented by 1 classe :

```
main/java : com.triton.MapWithStat.java
```

It's a first step if we want to implement other languages.

We can see it in action in TestCase here :

```
test/java : com.triton.ExerciceMap.java
```

We can see it in action in Maven command line :

```
mvn test -Dtest=ExerciceMap
```

Exercice 8 : Code sample

Difficult to answer. There are many good and bad examples but I don't know how to choose the perfect one and if it exists. Depending on quality tools used and how it is configured, there will be alwyas someone to tell something could be better.

In few words I think SpringFramework codes are a good example of great code. Using Interface the code is easy to maintain and easy to customize.