**Java/SQL Practical Test Instructions**

**Checking out the code**

**Prerequisites**

* Install JDK 8 (<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>)
* Install MySQL (<https://dev.mysql.com/downloads/mysql/5.7.html>)
* Install Git <https://git-scm.com/download/win>
* (Optional) Install IntelliJ Community Edition (<https://www.jetbrains.com/idea/download>)

**Checking out the project**

From the command line (assuming Git is installed), change to your desired directory and type:

git clone <https://bitbucket.org/costcutter/junior-java-test>

**Contents**

The project will be downloaded into a folder called ‘junior-java-test’. Within this there are two files and two folders:

* Instructions.docx – this file!
* pom.xml – Maven dependency file for the project, you shouldn’t need to modify this
* src – contains the Java project
* sql – contains database setup script

**Database Setup**

* Open a command prompt and change directory to the location of practicaltest.sql
* From the command line, type mysql -u<username> -p<password> to log in to MySQL
* Then type \. practicaltest.sql
* Note that some of the data is created using random functions, therefore repeated executions of the setup script will generate different results.

**The Java Project**

The application is structured into two main source packages:

* com.cc.practicaltest.config - contains only one class, AppConfig, which configures the database connection.
  + Please replace the various connection parameters as per your local database setup.
* com.cc.practicaltest.app – contains several classes
  + PracticalTestService – contains business logic for the application. Already contains example code.
  + PracticalTestRepository – contains SQL queries and logic relating to the underlying database. Already contains example code.
  + Order – POJO representing a row in the orders table. For example purposes only.

There is also one test package:

* com.cc.practicaltest.app – contains unit tests for application
  + PracticalTestServiceTest – single class intended to contain all automated tests. Class is already set up with a single example test and mock object infrastructure.

The application uses the following frameworks (links to documentation provided):

* Maven for dependency management (you shouldn’t need to make any changes to how this works).
* Spring Framework for dependency injection (see <https://spring.io/projects/spring-framework> and <https://docs.spring.io/spring/docs/current/javadoc-api>)
* SLF4J for logging (see <https://www.slf4j.org/docs.html>)
* Mockito for test object mocking (see <http://site.mockito.org/>)

**Instructions**

* Study the created database and template Java project.
* When run, you will notice that the application currently writes to the standard output a count of orders since 2017-01-01. This is for example purposes only.
* Extend the application to retrieve all uninvoiced orders placed in the last 28 days. An uninvoiced order is one for which there is no corresponding entry in the invoices table.
  + The list should be printed to the standard output, one order per line with the following information:
    - Order number
    - Order date (formatted dd/mm/yyyy)
    - Number of vehicles ordered
    - Forename and surname of customer
    - Total left to pay
  + Answer can be implemented using any combination of SQL and Java code.
* (Optional) Wherever possible, write unit tests to cover your Java code.