Instructions for Exercises

What you'll need

You'll need a few tools installed on your system:

- JAVA JDK 1.6
- Maven (3.x)
- Eclipse (very useful but not required)
- GIT

Building the project

I created a public github project here:

- https://github.com/jbroch/net.jbroch.questions

You can clone that repo to your local system by doing:

- git clone https://github.com/jbroch/net.jbroch.questions.git

After cloning, you can build the project using maven:

- mvn clean package (this builds everything and runs the unit tests)

If you want to use eclipse as your editor (recommended) make sure to create your eclipse project file:

- mvn eclipse:clean eclipse:eclipse

Then you can import the project into eclipse.
Note: The maven build runs both check style and find bugs against the source. The build fails if you violate any of the rules specified by either pluggin.

Setting up Eclipse (optional)

- **Install Eclipse plugins**

- Select **Spring Source Tool Suite > Preferences > Install / Update > Available Software Sites**
- Press **Import...**
- Browse to **<tree root>/dev/eclipse/UpdateSites.xml** and press **Open** and **OK**
- Select **Help > Install New Software...**
- In the **Work with** drop down select **– All Available Sites --**
- Check the box to **Hide items that are already installed**
- Uncheck the box next to **Group items by category**
- Select and **install the following plugins**:

Name	Description	Update Site	Home Page
AnyEdit Tools	Editor enhancements (e.g. convert tabs / spaces, case changes)	http://andrei.gmxhome.de/eclipse/	http://andrei.gmxhome.de/anyedit/
Checkstyle configuration plugin for M2Eclipse	Configures Eclipse Checkstyle plugin from the POM Checkstyle plugin configuration	http://bimargulies.github.com/m2e-code-quality/site/1.0/	https://github.com/bimargulies/m2e-code-quality
Eclipse Checkstyle Plug-in	Integrates the Checkstyle tool into Eclipse	http://eclipse-cs.sf.net/update/	http://eclipse-cs.sourceforge.net/
Findbugs	Integrates the Findbugs tool into Eclipse	http://findbugs.cs.umd.edu/eclipse	http://findbugs.sourceforge.net/
QuickREx	Regular expression development tool	http://www.bastian-bergerhoff.com/eclipse/features	http://sourceforge.net/projects/quickrex/
TestNG	Provides "green bar" support for TestNG framework	http://beust.com/eclipse	http://testng.org/doc/eclipse.html

- Click **Next, accept licenses** and press **Finish**. After installation is complete, restart when prompted.

- **Load global Eclipse preferences**
  - Select **File > Import > General > Preferences** and press **Next**
  - Browse to **`<tree root>/dev/eclipse/GlobalPrefs.epf`**, ensure that **Import All** is checked, and press **Finish**
  - Restart Eclipse: **File > Restart**

- **Set the Maven installation**
  - Select **Preferences > Maven > Installations**
  - Press **Add...**
  - Navigate to **`<MAVEN INSTALL DIRECTORY>`**
  - Press **Open**
  - Press **OK**

- **Import the git project**
  - Ensure **Projects > Build Automatically** is **checked**
  - Select **File > Import > General > Existing Projects into Workspace**, browse to your **`<tree root>`** directory, press **OK**. You should see the **polaris** project listed with a check next to it. Press **Finish**.
  - Wait a few minutes until Eclipse is finished updating dependencies and building the workspace.
  - You should not see any errors or warnings but if you do, select **Project > Clean > Clean All Projects** and press **OK**. You may notice that the project is badged with the Maven (**M**) and Java (**J**) icons but not the Sping (**S**) icon. The project does have the Spring nature but the Java nature's badge is shown because its nature is listed before the Spring nature in the project file.

- **Share the project with Git**
  - Right click over the polaris project in the **Package Explorer** and select **Team > Share Project...**
  - Select **Git** and press **Next**
  - Check the box **"Use or create repository in parent folder of project"** and press **Finish**

Exercise 1

In the "STACK" package (net.jbroch.project.stack), I've included an interface (Stack.java) and an empty implementation of a stack (see StackImpl.java). Please try the following:

- Part 1
  - Complete the stack implementation
  - Add appropriate test cases to the StackImplTest.java class and make sure that these tests pass.
- Part 2

    - Extend the Stack interface and StackImpl to support a stack which takes an object of arbitrary type
    - Make sure your stack is type-safe
    - Update test cases as appropriate and make sure they pass
- Part 3
    - Make sure the implementation of Stack is thread-safe
    - Update test cases as appropriate and make sure they pass

---

Exercise 2

The "THREADING" package (net.jbroch.project.threading) is about building a thread-safe producer/consumer.
In this problem, the producer and consumer share a single queue object. I've provided an implementation of the Queue (Queue.java). The queue is not itself thread-safe.
I've also provided skeleton implementations of the producer and the consumer. ***The exercise is to implement both the producer and consumer in a thread-safe way so that the producer can add elements to the queue as fast as possible and the consumer can remove elements from the queue as fast as possible.***
I've added one test case in ProducerConsumerTest.java to exercise my non-thread-safe version of these classes. Once you implement thread-safe versions of the Producer and the Consumer, please construct unit tests that simultaneously run both the producer and the consumer.

Exercise 3

No code here; just schema design.
Assume my system has two object types: device and user.
Device consists of the following attributes:

- Device Model
- Device Serial Number

User consists of the following attributes:

- UID
- First Name
- Last Name

Now each device has 1 owner. A single user can own multiple devices.

Please design a SQL schema that meets these requirements.
Please write a query that displays the UID for each user and the number of devices associated (owned) by that user.