Project Overview

Develop a Java SE Netbeans project that implements a product processing simulation using the Producer-Consumer parallel design pattern. Create classes for two ProductProducer and four ProductConsumer based on continental US timezones (consult http://en.wikipedia.org/wiki/List_of_time_zones_by_U.S._state): easternConsumer, centralConsumer, mountainConsumer, and pacificConsumer which will be called regions for this exercise.

Create a Product class that encapsulates the following fields: productID, productName, productDescription, weight, and cost.

Create a ProductMessage class that encapsulates a product object, timestamp and region.

Create a Timezone2State map in a utility class referenced by each producer which maps the timezone to a list of states in that zone. If the state has two timezones, choose the larger of the two. Also create an array of continental US states in this class that will be referenced by each producer to randomly select a state during its message construction.

The producers each produce a product for consumption by the regional consumers. They randomly select a state for product distribution. Each producer prepares a ProductMessage by populating the message with its product object, the current timestamp and the region based on a lookup of a state from the list in the map and then pushes the ProductMessage object on its internal queue.

Each consumer consumes only their respective ProductMessage objects from each producer and maintains an internal list of collected products.

Provide the following functionality in the driver:

- begins the simulation by creating and starting the producers and consumer objects
- provides a capability to terminate the simulation by a single keystroke
- displays real-time queue change status per producer
- displays real-time consumption data per consumer
- writes all Product objects per consumer to a file based on region
- displays all Product objects per consumer on completion
- displays the total elapsed time of the simulation

Provide a portable test script that executes the application and writes all display data to a file called mp3out.txt in a portable file system location.

Installation, Compile & Run-Time Requirements

Program was built using a Windows 7 desktop configured with— java version 1.6.0_26 . The source code (project files) was developed using Eclipse IDE for Java Developers / Version: Juno Service Release 1 / Build id: 20120920-0800.

To run, extract the contents of the mp3.zip file to a local folder. Open a windows command prompt, and navigate to the "dist" subdirectory. Type the following:

```
java -jar "..\dist\mp3.jar" ..\data\stateTimeZoneMap.csv ..\output\
```

Alternately, navigate to the test directory and run the test.bat script. This script will capture all STDOUT and STDERR output in the output\mp3out.txt file. After executing the mp3.jar, the script opens the mp1out.txt file in window's Wordpad application.

In order to exit or close the application, press the [Enter] key. This will end the producer / consumer simulation and output the results as described earlier in this document. CSV files listing the products consumed by region are written to the output directory. Each file is named for the time zone data contained. These files have the following columnar layout:

#	Column	Data Type	Expected Value
1	Id	Integer	Randomly generated integer between 1 and 47
2	Product Name	String	Cog / Sprocket + State Name
3	Description	String	Producer's name either "CogsWell Cogs" or "Spacely Sprockets"
4	Weight	Double	Weight attribute of the product
5	Cost	Double	Product's cost
6	TimeStamp	Date	Time stamp when the product was processed. Value is converted to the local time of the region.

When executed, the application begins by parsing the contents of the data\stateTimeZoneMap.csv. This file is used to define the mapping between states and time zones.

Insights and Expected Results

This application delivers the results as described earlier in this document. Errors due to concurrency are negated using the "synchronized" keyword on shared methods. In addition, the code base makes use of Super classes to promote code re-use and simplify maintenance.

Screen Captures

When executed, the application will write its output to the STDOUT in the following format.

```
parsing CSV...
class producer.SpacelySprockets started.
parsing CSV...
class producer.CogsWellCogs started.
class region.EasternRegion started.
class region.MountainRegion started.
class region.CentralRegion started.
class region.PacificRegion started.

Simulation starting

class producer.SpacelySprockets Added IDAHO, MT product message.
class producer.CogsWellCogs Added SOUTHDAKOTA, CST product message.
class region.CentralRegion polled cog @ 11/20/12 11:29 AM
class region.MountainRegion polled sprocket @ 11/20/12 5:29 PM
class producer.SpacelySprockets Added SOUTHCAROLINA, EST product message.
class producer.CogsWellCogs Added OREGON, PT product message.
class region.EasternRegion polled sprocket @ 11/20/12 12:29 PM
class region.PacificRegion polled cog @ 11/20/12 5:29 PM
class producer.SpacelySprockets Added NEWMEXICO, MT product message.
class producer.CogsWellCogs Added WESTVIRGINIA, EST product message.
class region.EasternRegion polled cog @ 11/20/12 12:29 PM
class region.MountainRegion polled sprocket @ 11/20/12 5:29 PM
class producer.SpacelySprockets Added UTAH, MT product message.
class producer.CogsWellCogs Added MISSOURI, CST product message.
class region.CentralRegion polled cog @ 11/20/12 11:29 AM
class region.MountainRegion polled sprocket @ 11/20/12 5:29 PM
class producer.SpacelySprockets Added COLORADO, MT product message.

Simulation ended

class region.EasternRegion processed:

Id, Product Name, Description, Weight, Cost, TimeStamp
36, SOUTHCAROLINA Sprocket, Spacely Sprockets, 1.1, 5.5, 11/20/12 12:29 PM EST
44, WESTVIRGINIA Cog, Cogswell Cogs, 1.1, 5.5, 11/20/12 12:29 PM EST

...

class region.MountainRegion processed:

Id, Product Name, Description, Weight, Cost, TimeStamp
8, IDAHO Sprocket, Spacely Sprockets, 1.1, 5.5, 11/20/12 5:29 PM MT
27, NEWMEXICO Sprocket, Spacely Sprockets, 1.1, 5.5, 11/20/12 5:29 PM MT
40, UTAH Sprocket, Spacely Sprockets, 1.1, 5.5, 11/20/12 5:29 PM MT
3, COLORADO Sprocket, Spacely Sprockets, 1.1, 5.5, 11/20/12 5:29 PM MT

...
```

```
class region.CentralRegion processed:

Id, Product Name, Description, Weight, Cost, TimeStamp
37, SOUTHDAKOTA Cog, Cogswell Cogs, 1.1, 5.5, 11/20/12 11:29 AM CST
21, MISSOURI Cog, Cogswell Cogs, 1.1, 5.5, 11/20/12 11:29 AM CST

...

class region.PacificRegion processed:

Id, Product Name, Description, Weight, Cost, TimeStamp
33, OREGON Cog, Cogswell Cogs, 1.1, 5.5, 11/20/12 5:29 PM PT

...

class region.EasternRegion Total Products:  2689
class region.MountainRegion Total Products:  861
class region.CentralRegion Total Products:  1934
class region.PacificRegion Total Products:  490

Simulation Time: 00:00:01
```

If the data\ stateTimeZoneMap.csv cannot be found, then the application will generate the following error message.

```
Exception in thread "main" java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)
        at org.eclipse.jdt.internal.jarinjarloader.JarRsrcLoader.main(JarRsrcLoader.java:58)
Caused by: java.io.FileNotFoundException: ..\data\stateTimeZoneMap.csv could not be found.
        at main.Program.main(Program.java:31)
        ... 5 more
```

Additional Info

For additional information, please refer to the Javadoc generated html pages published in the \docs sub directory.