February 18, 2013

Spring 2013 MP2

Develop a *Java* SE *Netbeans* project using the sample data files contained in *data_files.zip* as an input feed into an application that provides data analytics. The data was obtained from the United States Naval Observatory (USNO) at: http://aa.usno.navy.mil/data/docs/RS_OneYear.php

Create an abstract class called *Record* and a subclass called *DaylightRecord* employing instance variables for each of the fields in the data files. Provide default (*noarg*) and (*full-arg*) constructors where required. Provide accessors (gets) and mutators (sets) for all of the instance variables/fields. Provide *toString()* methods to format object state.

Provide comparator classes implementing *java.util.Comparator* or *java.lang.Comparable* for comparing various fields in the application's data analysis requirements. Locate the *Record*, *DaylightRecord* and comparators in a package called *domain*.

Provide the following functionality in the driver:

- Read input data into *DaylightRecord* objects.
- Collect *DaylightRecord* objects into an *ArrayList* called *daylightRecords*.
- Encapsulate a *java.util.Date* object and the *ArrayList* object into a serializable class called *PersistentObject.*
- Create an instance of *PersistentObject* with the current timestamp and the *ArrayList* object.
- Serialize the persistent object to a file called *daylight-record.ser* and folder *data* relative to your project path.
- Make the application sleep for 10 seconds.
- Deserialize the persisted object into a date object and an *ArrayList* object called d*eserializedDaylightRecords*.
- Display the time difference between serialization and deserialization.
- Display the following data analytics:
    - Shortest day => winter solstice
    - Equal day and night => vernal equinox (Spring)
    - Longest day => summer solstice
    - Equal day and night => autumnal equinox (Fall)
- Write all displayed data to a text file called *daylight-records.txt* in your project folder called *output.*
- Minimize the *main()* method by calling external utility classes.

Provide a portable test script that executes the application and writes all display data to a file called *mp2out.txt* in a portable file system location (i.e. project top-level)

Comment your code and provide a detailed *README* (PDF only) file that includes:

- project description/abstract
- installation, compile and run-time requirements
- insights and expected results

- screen captures demonstrating all application capabilities
- generate project *Javadocs* API and move from the *dist* folder to a *docs* folder

Submit a compressed file called mp2.zip (zip only) of all project code and documentation by 03/03/13, 23:59/CST. Late mini-projects will lose points. (**50 points**)