# ITEC 3150 Advanced Programming

## Exam 1 Coding

### Problem 1 (25 Points)

#### Description

*Note: this is not the same problem that was provided to the class as an exam sample. If you submit that solution you will receive zero credit for this problem. Build it from scratch.*

Find the *CarsMain.java* class with a *main(...)* method.  In the *main(...)* method, create a *LinkedList* of *Car* objects named *cars* and read in the information from the *carinfo.txt* TEXT file. Create a *Car* objects for each entry and populate the *cars* list with each object.  After all data from the file is read in, print the contents of the *LinkedList* to the console.  Next, sort the *LinkedList* in ascending order, by first model year.

#### Output

Your output should look like this:

Unsorted List:

Car{make='Chevrolet', model='Corvette', firstProductionYear=1953, weight=2886.0, weightUnits='Lb'}

Car{make='Subaru', model='Outback', firstProductionYear=1995, weight=3080.0, weightUnits='Lb'}

Car{make='Ford', model='F-150 Lightning', firstProductionYear=2022, weight=6015.0, weightUnits='Lb'}

Car{make='Pontiac', model='Trans Am', firstProductionYear=1969, weight=3850.0, weightUnits='Lb'}

Car{make='Tesla', model='X', firstProductionYear=2015, weight=4936.0, weightUnits='Lb'}

Car{make='Mercedes', model='G Wagon', firstProductionYear=1979, weight=1775.0, weightUnits='Kg'}

Sorted List:

Car{make='Chevrolet', model='Corvette', firstProductionYear=1953, weight=2886.0, weightUnits='Lb'}

Car{make='Pontiac', model='Trans Am', firstProductionYear=1969, weight=3850.0, weightUnits='Lb'}

Car{make='Mercedes', model='G Wagon', firstProductionYear=1979, weight=1775.0, weightUnits='Kg'}

Car{make='Subaru', model='Outback', firstProductionYear=1995, weight=3080.0, weightUnits='Lb'}

Car{make='Tesla', model='X', firstProductionYear=2015, weight=4936.0, weightUnits='Lb'}

Car{make='Ford', model='F-150 Lightning', firstProductionYear=2022, weight=6015.0, weightUnits='Lb'}

#### Rubric (pts – item)

5 - LinkedList named *cars* created correctly

5 - scans *carinfo.txt* correctly

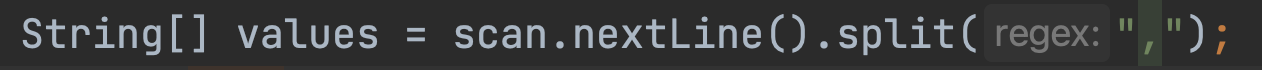
5 - *Car* objects created correctly

5 - LinkedList *cars* sorted correctly

5 - Unsorted and sorted lists printed correctly

#### Restrictions and Hints

* You should use the starter project in the D2L Assignment drop box. It contains the *Car* class, the *CarsMain* class and *carinfo.txt*
* You may not alter the *Car* class
* You should consider using the split method to convert each line read in into a string array of values, like:



* You can safely ignore the junit tests that are provided under the test folder. However, you are welcomed to read them for inspiration!

#### Submission

Include your modified *CarsMain.java* file in a zipped archive of a completed Intellij project. You should submit only one project that contains all problem solutions for the exam.

### Problem 2

#### Description

Write a Java program that creates six instances of the *Queen.java* class. Write them to a binary file named *queens.dat* using *DataOutputStream* library. Create your solution within a *main*(…) method inside of a *QueensMain* class.

Use the data from this table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Year** | | |
| **Name** | **Birth** | **Reign Begins** | **Reign Ends** |
| Mary I | 1516 | 1552 | 1558 |
| Elizabeth I | 1533 | 1558 | 1603 |
| Mary II | 1662 | 1689 | 1694 |
| Anne | 1665 | 1701 | 1714 |
| Victoria | 1819 | 1837 | 1901 |
| Elizabeth II | 1926 | 1952 | 2022 |

The binary data should be written in the same order as what appears above. The data types for the queen’s name should be a text string, the other fields should be primitive integers.

Any checked exceptions need to be caught and handled with a call to the exception’s printStackTrace() method. Do not use the *throws* clause on method signatures.

#### Output

Although we can’t really see the output in a binary file, you could run this code to validate that your binary file has been written correctly:

try {  
 DataInputStream dis = new DataInputStream(new FileInputStream("queens.dat"));  
 while (dis.available() > 0) {  
 String name = dis.readUTF();  
 int birth = dis.readInt();  
 int begin = dis.readInt();  
 int end = dis.readInt();  
 Queen q = new Queen(name, birth, begin, end);  
 System.*out*.println("read and constructed: " + q);  
 }  
} catch (IOException e) {  
 e.printStackTrace();  
}

the resulting output will be:

read and constructed: Queen{name='Mary I', birthYear=1516, beginReignYear=1552, endReignYear=1558}

read and constructed: Queen{name='Elizabeth I', birthYear=1533, beginReignYear=1558, endReignYear=1603}

read and constructed: Queen{name='Mary II', birthYear=1662, beginReignYear=1689, endReignYear=1694}

read and constructed: Queen{name='Anne', birthYear=1665, beginReignYear=1701, endReignYear=1714}

read and constructed: Queen{name='Victoria', birthYear=1819, beginReignYear=1837, endReignYear=1901}

read and constructed: Queen{name='Elizabeth II', birthYear=1926, beginReignYear=1952, endReignYear=2022}

#### Rubric (points – item)

5 - DataOutputStream created correctly

5 - Six queen objects created correctly

5 - Queens written to binary data file in proper order

5 - binary data types used to create binary output meets specification (String/int)

5 - All checked exceptions handled with try/catch blocks and call printStacktrace()

+50 - Challenge met - bonus!! (see challenge below)

#### Restrictions and Hints

* You should use the starter project in the D2L Assignment drop box. It contains the *Queen.java* class
* You may not modify *Queen.java*
* You may want to place each queen object in a list. This way, you can iterate over the list for each entry. This is not required, however
* You may want to copy the data for the queens from the table above and then paste/edit it in your java code
* Use the validation code provided above to confirm that the binary data has been entered into the file in the proper order and with the proper types
* You can safely ignore the junit tests that are provided under the test folder. However, you are welcomed to read them for inspiration!

#### Challenge (extending the queens problem, adds 50 points to overall grade)

Write code to sort the queens in ascending order, where the ordering is established by the length of their reign. That output would look like this:

Mary II reigned for 5 years

Mary I reigned for 6 years

Anne reigned for 13 years

Elizabeth I reigned for 45 years

Victoria reigned for 64 years

Elizabeth II reigned for 70 years

Recall that you may not modify *Queen.java*. Additionally, you may not simply hardcode the output, you must use a programmatic solution such as java’s sort method or Java Streams.

#### Submission

*Include your QueensMain.java file in a zipped archive of a complete Intellij project. You should submit only one project that contains all problem solutions for the exam.*

### Problem 3

#### Description

*Note: this problem differs from the problem that was provided to the class as an exam sample. If you submit that solution you will receive zero credit for the problem. Build it from scratch.*

A previous developer created a binary file using *ObjectOutputStream*. The binary file is believed to have been written with objects of type *TimeCapsule*. Your manager informs that there are an unknown number of *TimeCapsule* objects stored in the *timecapsuleinfo.bin* file. How many objects are there in the file? Can you determine the year and message for each of the TimeCapsule objects? Create your solution as a main(…) method inside of a TimeCapsuleMain class.

#### Output

Your output should look like this:

There are 2 entries in timecapsuleinfo.bin

TimeCapsule{year=1979, message='Welcome Mercedes G Wagon!'}

TimeCapsule{year=2015, message=’Welcome Tesla Model X'}

Your output number and time capsule messages will be different!

#### Rubric (pts – item)

5 - Creates *ObjectInputStream* correctly

5 - Read correct number of TimeCapsule messages

5 - Print the number of messages to the console, as specified above

5 - Reports each of the time capsule messages to the console, as specified above

5 - Closes *ObjectInputStream* correctly

#### Restrictions and Hints

* You should use the starter project in the D2L Assignment drop box. It contains the *TimeCapsule.java* class and *timecapsules.bin* files.
* You may not modify *TimeCapsule.java*
* You must use *ObjectInputStream*
* Using a binary editor to retrieve the values from the provided binary file won’t help
* You can safely ignore the junit tests that are provided under the test folder. However, you are welcomed to read them for inspiration!

#### Submission

*Include your TimeCapsuleMain.java file in a zipped archive of a completed Intellij project. You should submit only one project that contains all problem solutions for the exam.*