CSU Long Beach
CECS 277
Object Oriented Application Development

Syllabus
Schedule
Grading
Project ArrayList
Project Inheritance
Project File I/O
Project Generics
Project Regular Expressions
Project Collections
Project Gui
Project Threads
Project Binary I/O
Code
Lecture Notes
Using Eclipse
Textbook Resources
Java API Documentation
Java Tutorial

Mimi Opkins Home
CECS 323 Home
CECS 493 Home

Project Collections

Collections Assignment

For this assignment, you are required to finish two parts. The third part is extra credit.

Part One

- This program will work with maps. Use the following file. It contains the words that start with Q not followed by U. here it is.
- Create a second file that contains each of the letters in the alphabet and their scrabble point value. You can find the file at Scrabble Site.
- Read this file into a map data structure.
- For each of the words in the file, find their point value and print it out.

Part Two

- Insert all words from a large file such as the novel Alice in Wonderland, into a hash set and a tree set.
- Time the results. Which data structure is more efficient.
- **Important: You must write two programs for this part.**

Part Three

- A Scavenger Hunt is a party game that people play in teams. They are given a list of items to find throughout the surrounding neighborhood. Our challenge is to find the best data structure to store the results of the hunt.
- Create a list of 100 items that you might have people find on the hunt. These should be items that they have a chance of finding through out the neighboorhood. For example, yesterday's newspaper, a shoelace, a paperbag,.... Load the list anyway you want (console input, flatfile, etc.)
    - You will run the program two times, once using a set of ArrayLists, and once using a set of LinkedLists. Our pupose is to find the more efficient data structure to use for a scavenger hunt.
    - Ask the user how many teams will play the game. Create this number of teams. For each team load all of the items from the list. Shuffle the list after loading the items each time. Find the total time it takes to add the items to all of the teams.
    - Ask the user what position in the list to be used for retrieving and inserting elements. Retrieve that element from each of the teams. Find the total time it takes.
    - Next insert a new element at that position in each of the lists. Find the total time it takes.
    - Use the Random class to generate a number between 0 and the 100. Find the element in the scavenger hunt list at that position. Retrieve that element from each of the lists. Find the total time it takes.
    - Display the time in milliseconds for each of these operations.
    - Run the program several times with different input values and see how it affects the output.
- Write a brief analysis of your findings along with your recommendations.
- Make sure you use appropriate exception handling. This would be related to how you initially set up the list of items for the scanvenger hunt and how you have the user enter the valus for the number of teams and the element to work worth.
- The following code snippet will help with the timing

```
Date today = new Date();
long time1, time2;
```

```
time1=System.currentTimeMillis();
.
.
.
.
time2=System.currentTimeMillis();
System.out.println("Time for the operation is: " + (time2-time1));
```

- If the time difference is too small, try:

```
long startTime = System.nanoTime();
// ... the code being measured ...
long estimatedTime = System.nanoTime() - startTime;
```

- For all programs, catch and handle the Exceptions appropriately and validate the input data where needed.

Grading Criteria

You will be graded on the following components:

- Does the program do what is required
- Is it properly documented
- Is it fully tested
- Is it properly designed

Latest Update: Wednesday, 09-Nov-2016 11:03:06 PST

mimi.opkins@csulb.edu

[Top of Page]