Sumit Malhotra 6/19/2018 Project WalmartLabs

Design Specifications

WalmartTaskProgram Class (extends JFrame)

-This class is used to create the user interface for the program and adds elements that allow the user to perform functions of reading files, displaying, searching, and sorting.

-The GUI has a JScrollPane and a JTextView to display the file information

Methods

-WalmartTaskProgram()

instantiates all variables being placed into the GUI and creates action listeners for buttons to perform functions

-readFile()

Calls JFileChooser to select file and then creates world object with file, also calls method to create JTree and adds to JFrame

-displayWorld()

a method that is used to display the information from the file in an organized manner

-search()

this method is used to search information from the other classes

-main()

creates a this object to run application

-displayBtn()

action listener method for displayBtn, appends data structure of world to text area

-addBtn()

action listener method for addBtn, add entries from GUI to database

-readBtn()

action listener method for readBtn, calls search method with searchCB and searchTF element values

-sortByNameBtn()

action listener method for sortByNameBtn, appends world sortByName method

return to text area

Thing(implements Comparable)

Sets basic information for other class objects and implements comparable to compare objects Variables
index: int
object index for identification

name: String

object name for identification

parent: int

object parent in data structure hierarchy to match with parent object Methods
Thing(Scanner)
uses file lines to set instance variables

compareTo(Thing o)

compares objects by name

getIndex()

returns index instance

getName()

returns name instance **getParent**()
returns parent instance

Global Class (extends Thing)

This class is used to extract data from the file and places it into designated class objects, it
creates a data structure to hold the objects, methods to get and set data, search, and sort

data

Variables

walmarts HashMap

Stores Walmart objects with index key

Scanner sc

used to read data file

ArrayList<Tasks> tasks

stores tasks from que

ArrayList<Thing> sortedByName

stores Thing objects
Methods
-Global**(File)**
takes a file object and extracts the data that is to be placed in class objects, calls a separate set of methods to place in class objects

-addWalmart(Scanner)

creates new Walmart object and places it in ports data structure in global class

-addDepartment(Scanner)

creates new department object and places it inside department hashmap inside the walmarts hash map with a matching index

-addIndividualTask(Scanner)
adds new task objects

-addTeamTasks(Scanner)
creates new TeamTask object and placed it inside Tasks ArrayList or Department and Que hashmap depending on if its at a deparment, inside the walmarts array list with matching indices for either the dock or port
-addPerson(Scanner)
creates a person object in the Person hashmap in walmarts hash map with the matching index

-addJob(Scanner, JPanel)
creates a job object in the Job hashmap in walmarts hash map with the matching index, starts thread from jobs with task in department
-searchByIndex(int)
searches the walmarts data structure by index and returns string of information of the object

-searchByName(String)
searches the walmarts hash map by name and returns a string of information of the object using helper methods

-searchByType(String)

searches walmarts hash map by type and returns a string of objects with relevant type

-searchByName(String)

searches ports data structure for name

-searchTask(String)

Searches task hashmap for name value, helper method

-searchPerson(String)

Searches task hashmap for name value, helper method

-searchDepartment(String)

Searches department hashmap for name value, helper method

-searchJobs(String)

Searches jobs hashmap within ship object, helper method

-getAllDepartments()

returns all department to string

-getAllTasks()

returns all tasks to string

-getAllPersons()

returns all persons to string

-getAllWalmarts()

returns all walmarts to string

-getAllJobs()

returns all jobs in data structure

-sortByName(sortObject)

adds elements to sortedByName arraylist based on sortObject value and then sorts sortedByName
by name

-createJTree()

Creates J Tree object from walmart data structure and returns it

-toString()

returns walmartss data structure as string

Walmart extends Thing implements runnable
department: Hashmap
stores Department objects at walmart object
que: HashMap
stores Tasks that are at department at walmart object
task**s: HashMap**
stores Task objects at walmart
persons: HashMap
stores Person Objects at walmart
Methods
Walmar**t()**
instantiates instance variables and starts thread
insertDepartment(Department)

inserts department object into department data structure

insertTask(task)

inserts Task object into tasks data structure

insertPerson(Person)

adds person object to persons hash map

getDepartment()

returns dock hash map

getTasks()

returns ship hash map

getPersons()

returns persons hash map

getQue()

returns que hashmap

getInfo()

returns info about Walmart

run()

thread to check if jobs in ship are complete and calls method to update dock

sendResourcesToJob()

Sends person array list to currently running jobs

toString()

returns data structures as string

Department(extends Thing)

This class creates Department object Variables
Task task
stores task in the department Methods

Department(Scanner)

reads file and sets variable from Thing and department

setTask(Task)

sets the Task object

getTask()

returns Tasks instance variable

getTasksObject()

returns Tasks

getInfo()

get department information with tasks

toString()

returns string object with Tasks information

Tasks (extends Things)

Creates tasks object and holds the jobs data structure

Variables

draft: double

stores ship draft

length: double

stores ship length

weight: double

stores ship weight

width: double

stores ship width

jobs: ArrayList

stores ship jobs

Methods

getJobs()

returns jobs hashmap

getInfo()

returns ship info

getShipName()

returns name instance variable

getDraft()

returns draft instance

getLength()

returns length instance

getWeight()

returns weight instance **getWidth()**
return width instance

IndividualTasks(extends Tasks)

stores total rooms

Methods

IndividualTasks(Scanner)

creates instance of object and sets the instances variable and the inherited variables

toString()

returns information from super class as string

getInfo()

return information about passenger ship and super class information as string

TeamTasks (extends Tasks)

Person(extends Thing)

Creates person object
Variable
skill: String
holds the persons skill
resourcePanel: JPanel
panel holding person information and status **status: Color**

holds the persons skill
Methods

Person(Scanner)
takes file line and sets super class instance variable and Person class variables **getSkill()**

returns skill instance

setStatus()

sets color of resourcePanel to indicate whether Person is in job or not

setStatus(boolean st)

set panel color

toString()

returns string of person info

Job extends Thing implements runnable

This class is used to create job objects and hold the job requirements in a data structure, implements runnable to progress jobs, acquires jobs form walmart class to perform jobs in runnable, will not progress until all jobs are acquired
Variables

duration: Double

stores job duration

requirement: ArrayList

stores job requirements

goflag : boolean

controls whether job continues

taskComplete: boolean

boolean for if job is complete

pause: boolean

whether job is paused or not

pm: JProgressBar

shows job completion as a percentage

acquired: ArrayList

list of persons working in job

Methods

Job(Scanner sc)

sets super class variables and local instance variables

Job()

sets gui elements for job and sets instance variables

acquireJobs(ArrayList<Person>)

adds jobs passed in from Walmart to acquired array list

returnResources(ArrayList<Person>)

returns resources stored in acquired array list to person array list in Walmart

resourceCount()

counts if acquired array list has all requirements t**oString()**
returns String of job information

User's Guide (Guide for previous version of program but still applicable to Walmart Program)

This program is meant to run in a java IDE such as eclipse or netbeans. Once the program is in the compiler and ran, the steps below should be followed, when the application appears:

1. The top of the lower panel of the Application will have UI elements to work the program,
   the program needs a file to read, the "Read" button can be pressed to select a file .After the read button is pressed a File Chooser should open displaying a list of local files, the designated file must be navigated to with the port information and then selected

2. Once the file has been selected a message should appear in the text area displaying “Read File button pressed”, **immediately upon selecting a file a tab will be created on the bottom panel of the frame and a J Tree will appear inside** in which the icons can be selected and searched, **on the top of the panel, progress bars with job names, indexes, buttons, and information on resources will loaded to show the state of the job**

3. If the user wants to see the entire file displayed in an organized text of the file uploaded then the user can press the “Display” button, after pressing this button the text area should have all the elements from the file displayed in order and with relevant information relating to name, index, etc..

4. The user can then search the data of the file by user the “Search Target” text field, the Combo box with the search type, and the search button.

5. If the user chooses “Index” from the combo box then the user should enter the index of the item to be searched in the text field “Search Target”, followed by pressing “Search”

6. The user should now see information of the object of interest in the text area

7. If the user chooses “Name” from the combo box then the user should enter the name of the item to be searched in the text field “Search Target”, followed by pressing “Search”, the user should now see information of the object of interest in the text area

8. If the user chooses “Type” from the combo box then the user should enter the type of the item to be searched such as “ship” or “Cargo Ship” in the text field “Search Target”, followed by pressing “Search”, the user should now see information of the objects of

9. If the user would like to sort the data by name, the user can select the object from the the combo box, next to the Sort By Name label, and then press the button labeled Sort By Name to see the sorted list of objects.

10. If the user would like to sort the ships in the port que by a certain attribute the user can select an attribute in the combo box next to Sort Ships in Que, and then press Sort Ships in Que

11. If the user would like to control the job progress, the user can press the cancel button to end the job or the stop button to pause the job for 30 seconds, where after it begins to run again

12. **A resource panel will be created on the left side showing the available workers in the SeaPorts. The panels of each worker are color coded to show that a worker is free(green) or a worker is in a job (red).**

Test Plan (Test for previous version of the program but still applicable to this program)
Expected Output:
The expected output for this program is being able to take in the relevant file from the file chooser and enter them in a data structure. After the data structure is built, the program should be able to search the data

structure given the search elements and then display the information in the text area. The program should be able to display all of the structure using the display button and sort and display data from the sorting elements. Upon selecting the file, the program should also create a new tab on the bottom panel of the frame which updates with a J Tree of the data structure. Also, above the j text area, is a panel showing the job progress with progress bars and controls, **all jobs of the file are added but only the ones able to acquire resources progress.** It it is also important to observe that all the resources appear on the left side of the frame with color indicating availability. Also as jobs progress and acquire resources, the job panels should be tested that they are updating.

To test these features, the files given in the package will be selected for the program to extract and create objects and data structures. The search and the display will be checked for, so that the program can show if all the elements were added into the data structure. The files aSPaa.txt, aSPab.txt, aSPac.txt, aSPad.txt, and aSPae.txt will be used for the program
Test

Test	Expected Output	Actual Output	P/F
Button “Read” is pressed and exited without selecting a file	Message “No file selected”	Message “No file selected”	Pass Test 1
Button “Display” is pressed without file	Message “No File Selected”	Message “No File Selected”	Pass Test 2
Button “Search” is pressed without file	Message “No Entry”	Message “No entry”	Pass Test 3
File aSPaa.txt selected and then display button is pressed	All objects displayed in text area	all objects displayed in text area	Pass Test 4

Search index 30001 after aSPaa.txt file is selected	Displays data on Remora Passenger Ship	Displays data on Remora Passenger Ship	Pass Test 5
Search name Shoetrees after aSPaa.txt file is selected	Displays data on Shoetrees ship	Displays data on Remora Passenger Ship	Pass Test 6
Read and display the 5 files listed	Reads and displays the 5 files listed	Displays data on Remora Passenger Ship	Pass Test 7
selecting file aSPad.txt and searching type with target “job”	Displays all jobs at ports	Displays all jobs at ports	Pass Test 8
read file asPae.txt sort by name: Job	Displays jobs in alphabetic or numeric order from least to greatest	Displays jobs in alphabetic or numeric order from least to greatest	Test 9 Pass
read file asPae.txt sort by name : Person	Displays person in alphabetic or numeric order from least to greatest	Displays person in alphabetic or numeric order from least to greatest	Test 10 Pass
Test	**Expected Output**	**Actual Output**	**P/F**
read file asPad.txt	J Tree loads with data structure and J Panel above the text area loads with job progress	J Tree loads with data structure and J Panel above the text area loads with job progress	Test 11 Pass
read file asPad.txt cancel a job	cancels job at current progress	cancels job at current progress	Test 12 Pass
read file asPad.txt check for all jobs to progress once acquiring resources	all jobs that can acquire resources progress	only a few jobs succeed in acquiring resources and progress	Test 13 Fail

read file asPad.text Opens selected file as new tab for tree	Creates new tab with jtree for file	Creates new tab with jtree for file	Test 14 Pass
Does not open new tab for previously used file	Uses same tab for same files	Uses same tab for same files	Test 15 Pass
Aquired resources are updated for each job	Resources acquired are updated	Resources acquired are updated in job panel	Test 16 Pass
Job cancellation	Jobs are canceled if resources are not acquired within 75 seconds , Updates panel to Cancelled	Jobs are canceled if resources are not acquired within 75 seconds , Updates panel to Cancelled	Test 17 Pass
Resource panel updates used and non used resources	Red when used Green when not used	Not always consistent, works only some of the time	Test 18 Fail