| Program Overview | | |
| --- | --- | --- |
| Explain how to run your program. | | |
| 1. Open and edit both Menu\_Items.txt and Config.txt to your preferences. Any invalid lines within the Menu\_Items file will be skipped by the program. Any invalid lines within the Config file will cause the program to revert to its  default configuration. (ctrl + t to open the console, this will show the relevant debug messages). 2. Open the BlueJ package (double click). 3. Click compile. 4. Right-click on the shopperGUI class and click run JavaFX application | | Menu\_Items.txt Defaults | Config.txt Defaults | | --- | --- | | Category, Food Name, Price  Sushi, Inari, 2.6  Sushi, Avocado, 1.2  Sushi, Tuna, 1.5  Sushi, Salmon, 1.7  Drinks, Green Tea, 2.5  Drinks, Miso Soup, 2.7  Drinks, Coffee, 4.5  Drinks, Hot Chocolate, 2  Baked goods, Muffin, 3.5  Baked goods, Swiss Roll, 6  Baked goods, Mince Pie, 4.21  Baked goods, Cheese pie, 4 | #To use custom config change useConfig from false, to true.  #syntax errors will revert config to defaults as a failsafe.  useConfig=false  textOutputY=1  textOutputX=4  buttonYOffset=2  initialButtonYOffset=2  largestCategorySize=0  menuItemButtonXOffset=0  menuItemButtonYOffset=0  controlButtonXOffset=0  controlButtonYOffset=1 | | |

| Program Variables | | |
| --- | --- | --- |
| Describe the variables you have used and what they do for your program. | | |
| Variable Name | Type | Purpose/Description |
| totalCost | float | Tracks the users balance.  Starting value of 20f (f explicitly states value is float). |
| basket | Stack<Item> | Holds all the items the user intends to purchase.  Can be edited by the user through the use of the Undo, Clear, and Finish buttons. |
| menuItems | ArrayList<ArrayList<String>> | Stores the text file data from menu.txt in the form of a table [ID, Category, Name, Price].  This is done through the use of a nested arraylist (a number of arraylists that store strings, all stored inside an arraylist). |
| validCategories | Arraylist<String> | Lists all the unique Categories contained within the menuItems arraylist. |
| totalCostDisplay | Label | Shows the users current balance in the GUI. |
| textOutput | textArea | Displays the menu, instructions, and any relevant output.  Relevant output: making edits to the basket stack eg. Using the “Undo”, “Clear”, and “Finish” buttons. |
| pane | GridPane | Main layout container for the GUI. |
| file | File | file name for menu.txt file to be parsed in the loadMenuItems function. |
| scan | Scanner | object used to parse the text file. |
| idNum | Int | Used to auto increment and assign Item IDs to the items in the menuItems table. |
| nextLine | String | String variable containing a parsed text file line. |
| items | String[] | Array of strings containing all data parsed from a text file line split by the commas. |
| row | ArrayList<String> | arraylist containing data from item array |
| category | String | Individual variable from the row arraylist for use in populating row arraylist representing an item’s category. |
| name | String | Individual variable from the row arraylist for use in populating row arraylist representing an item's name. |
| priceStr | String | Individual variable from the row arraylist for use in populating row arraylist representing an item's price. The price is in string format in this case as the arraylist does not accept any other data types. |
| item | object | | Variable Name | Type | Purpose/Description | | --- | --- | --- | | ID | Integer | Represents the item's unique ID. | | name | String | Represents the item’s name. | | price | float | Represents the item’s price. | | category | String | Represents the item’s category | |
| config | Map<String, Integer> | Holds all the configuration values for the GUI content.   | Key | Value | | --- | --- | | textOutputY | Y value for textOutput creation | | textOutputX | X value for textOutput creation | | buttonYOffset | Vertical offset for placing item buttons | | initialButtonYOffset | Initial value of buttonYOffset | | largestCategorySize | Tracks biggest category for alignment | | menuItemButtonXOffset | Manual column shift for item buttons | | menuItemButtonYOffset | Manual Row shift for item buttons | | controlButtonXOffset | Manual column shift for control buttons | | controlButtonYOffset | Manual Row shift for control buttons |   Custom user config can be used, however if the cfg is invalid it will revert to the default config. |
| completedCategories | Arraylist<String> | Keeps track of the button categories for which the buttons have already been populated into the GUI. |
| useConfig | boolean | Determines if the user's config Map is valid or whether the defaults should be used. |
| menuItem | Item | Represents each individual item button for purchase.  These are stored in the basket stack. |
| orderTotal | float | Used in the handleFinish function to keep track of the total. |
| cfgFile | File | file name for the config file for parsing in the loadConfig function. |
| useConfig | boolean | A boolean that indicates whether or not the user's custom config should be used.  If false, the program will revert to default cfg. |
| invalidFound | boolean | Indicates to the user as to whether the user's config was invalid or it was simply disabled. |
| config | Map<String, Integer> | Stores the configuration in a Hashmap that is populated by the loadConfig function |
| line | String | Holds each line from the file parsing in the loadConfig function |
| parts | String[] | An array that represents each individual element of a line split into at least two pieces.  Used in loadConfig. |
| key | String | represents one half of the parts array, used as a keyreference in the hashmap. |
| value | String | represents one half of the parts array, used as a value that is associated with the key it appeared alongside. |
| requiredKeys | List<String> | A list of keys that should always be present in the config Hashmap.  If any aren't present the program will revert to default cfg |

| Testing and Debugging Procedures - Expected | | |
| --- | --- | --- |
| Document expected inputs and outputs of the program.  Add more rows as needed. | | |
| Expected Input | Expected Output | Actual Output (Screenshot or GIF Evidence) |
| Valid Menu\_Items.txt | Populated menuItems array  All Items categorized by category with prices etc. |  |
| Config.txt exists with valid keys and useConfig=true | Program should populate the config hashmap with the users custom config and use it to build the gui |  |
| useConfig=false in file | Program should use default config |  |
| Config.txt is missing entirely | Message: "Debug: Config.txt not found → using defaults."  Reverts to default configuration. |  |
| Config.txt contains invalid syntax, eg. Missing line, malformed line, or invalid integer value. | Messages:  "Debug: Malformed line (missing = or too many = are present): " + line  "Debug: Invalid boolean for useConfig → forcing defaults."  "Debug: Invalid number for " + key + " → forcing defaults."  will revert to default |  |
| Click Undo after buying one item | Item removed from basket and totalCost refunded |  |
| Click Undo when basket is empty | Message:  “Nothing to undo.” |  |
| Click Clear after multiple purchases | Basket cleared with refund given and shown to the user |  |
| Click Clear with empty basket | Message:  “Basket is already empty.” |  |
| Click Finish with Items in basket | Displays the order summary and the total spent by the user |  |
| Click Finish with empty basket | Message:  “Basket is empty. Nothing to checkout.” |  |

| Testing and Debugging Procedures - Boundary/Edge/Invalid | | |
| --- | --- | --- |
| Document the rest. Make sure to note what type it is.. | | |
| Expected Input | Expected Output | Actual Output (Screenshot or GIF Evidence) |
| Menu\_Items.txt not found | Missing Menu\_Items.txt should prevent the GUI from starting  (probably shouldn’t crash outright but oh well) |  |
| Malformed Menu\_Items.txt line | Line with too many/few commas will be excluded from final GUI |  |
| Config.txt not found | Missing Config.txt should prevent the GUI from using the user's custom config. |  |
| Malformed Config.txt line | A malformed Config.txt line should prevent the GUI from using the user's custom config. |  |
| Missing Config Key | If the program is missing a necessary config item, it will revert to default cfg |  |
| Invalid Config value | An invalid value in the Configs parsing will prevent the GUI from using the user's custom config. |  |
| Missing Config value | An invalid value in the Configs parsing will prevent the GUI from using the user's custom config. |  |
| Invalid price (below $0) | Line with negative price will be excluded from final GUI |  |
| empty basket undo | Message communicating that the user cannot undo with nothing to undo  Message:  “Nothing to undo.” |  |
| empty basket clear | Message communicating that the user cannot clear the basket with nothing to clear  Message:  “Basket is already empty. |  |
| empty basket finish | Message communicating that the user cannot finish the transaction with nothing in the basket  Message:  “Basket is empty. Nothing to checkout.” |  |