**Java Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Michael Bean**

**Date: 10/28/2023**

**Final Replit Program Join Link:**

Complete the following template. Fill out all entries using complete sentences.

## PART 1: Defining Your Problem

|  |
| --- |
| **Task**  State the problem you are planning to solve.  **Requirements**   * Describe the problem you are trying to solve. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Is your problem clearly defined? * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| <write your journal entry response here>  **1. Problem:**  Traditional handwritten grocery lists can be laborious and ineffective in the information age. It can also be difficult to keep track of fluctuating costs and make sure a budget isn't exceeded. The purchasing process can be more economical and well-organized with a digital solution.  **2. Input Data:**  A list of all the groceries you need, like milk, bread, and eggs;  Amounts for each item (like two bottles of milk and one loaf of bread);  An optional preferred brand list; and  A budget cap.  **3. Program Functionality:**  **Item Management**:  Update the list with new grocery items.  Take things off the list.  Modify the item's details (brand, quantity).  **Budget Functionality**:  Calculate the total cost of the shopping list if prices are known (either through user input or database retrieval).  Should the estimated cost surpass the allocated budget, notify the user.  **Categorization**:  Sort products into groups automatically (e.g., dairy, vegetables, bakery). A predetermined list of items and their categories can be used to accomplish this.  **Sorting and Searching**:  Arrange the items in priority, category, or alphabetical order.  The ability to search the list to locate an item quickly.  **Export/Share Feature**:  The ability to share the list digitally with others or export it to a format that can be printed.  **4. Outputs or Results:**  Show a grocery shopping list that is categorized and well-organized.  The total estimated cost of every item on the list.  If the projected cost goes over the user-specified budget, an alert or notification will be sent.  The outcomes of a user's search for a particular item.  Notifications of item additions, deletions, or updates.  The shopping list in an exported file or shared link. |

## PART 2: Working Through Specific Examples

|  |
| --- |
| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.  **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Java) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| <write your journal entry response here>  **Example/Scenario 1: Adding Items to the Shopping List**  **1. Necessary Input Data:**  Name of the item  Quantity of the item  Preferred brand (optional)  **2. Steps to Solve:**   1. Issue a prompt saying, "Enter the name of the grocery item." to the user. 2. Take note of the item name that the user enters. 3. Issue the following prompt to the user: "Enter the quantity for [item name]." 4. Record the amount that the user enters. 5. (Optional) Give the user the prompt, "Enter your preferred brand for [item name], or leave blank if you have no preference." 6. Take note of the user's feedback for the brand, or proceed if none is given. 7. Put the brand, quantity, and item name that you entered into a data structure, such as an array or list. 8. Inform the user that the item they selected, "[item name], has been added to your shopping list."   **3. Specific Result:**  The grocery item can be viewed as a part of the user's shopping list and is stored in the system along with its quantity and brand (if specified).  **Example/Scenario 2: Estimating the Total Cost**  **1. Necessary Input Data:**  List of grocery items with their names, quantities, and brands is the necessary input data.  The cost of every grocery item, which can be retrieved from a database or entered by the user.  **2. Steps to Solve:**   1. Set a variable's initial value to zero, such as totalCost. The total cost of the shopping list will be saved in this way.For each grocery item in the user's shopping list:    1. Retrieve the item's price. Get the price for that specific brand if it is specified.    2. Divide the cost of the item by the quantity.    3. Increase the totalCost by the outcome. 2. Show the user the totalCost along with the following message after every item has been processed: "The estimated total cost of your shopping list is [totalCost]."   **3. Specific Result:**  To assist the user in estimating their possible expenses, the user is given an estimated total cost for each item on their shopping list. |

## PART 3: Generalizing Into Pseudocode

|  |
| --- |
| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.  **Requirements**   * Write pseudocode for the program in English but refer to Java program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.   **Help With Writing Pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it, other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert to code later.   + <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>   + <https://www.wikihow.com/Write-Pseudocode>   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Java? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| <write your journal entry response here>  // Data structures to store information  List<GroceryItem> shoppingList = new ArrayList<>();  Map<String, Double> itemPrices = new HashMap<>(); // This can store prices for items  // Class that represents each grocery item  class GroceryItem {  String name;  String quantity;  String brand;  }  START  // Main program loop  WHILE user does not choose to exit the program:    DISPLAY "Choose an option:"  DISPLAY "1. Add item to list"  DISPLAY "2. Est. total cost"  DISPLAY "3. Exit"    INPUT choice from user  IF choice is "1":  CALL addGroceryItem()    ELSE IF choice is "2":  CALL estimateTotalCost()  ELSE IF choice is "3":  DISPLAY "Thank you for using the Grocery Shopping Assistant."  EXIT the program  END WHILE  // Function to add items to the shopping list  FUNCTION addGroceryItem():  GroceryItem newItem = new GroceryItem()    DISPLAY "Enter the name of the grocery item:"  INPUT item name from user  newItem.name = item name  DISPLAY "Enter the quantity for " + item name + ":"  INPUT item quantity from user  newItem.quantity = item quantity  DISPLAY "Enter your preferred brand for " + item name + ", or leave blank if no preference:"  INPUT item brand from user  newItem.brand = item brand  shoppingList.add(newItem)    DISPLAY item name + " has been added to your shopping list."  END FUNCTION  // Function to estimate total cost of the shopping list  FUNCTION estimateTotalCost():  IF shoppingList is empty:  DISPLAY "Your list is empty."  RETURN  double totalCost = 0  FOR each item in shoppingList:  double itemPrice = itemPrices.get(item.name) // Fetch the price from our prices map  double itemTotal = itemPrice \* item.quantity // Multiply by quantity  totalCost = totalCost + itemTotal  END FOR    DISPLAY "The estimated total cost of your shopping list is: $" + totalCost  END FUNCTION |

## PART 4: Testing Your Program

|  |
| --- |
| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.  **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from Replit. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify which approach was the one that worked.   **Inspiration**  When writing your entry below, ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| <Record your errors and fixes here>  **Test 1: Add a New Item**   * **Action:** Select option 1 in order to insert a new item. Name: "apple", Size: "2.5", Brand: "Red Delicious" * **Anticipated Result:** The application ought to notify users that "apple" has been included in the shopping list. * **Result:** Expected. * **Conclusion:** Based on this test, it appears that the program is operating correctly.   **Test 2: Estimate Cost with No Items**   * **Action:** When no additional items have been added, select option 2 to estimate the cost. * **Anticipated Result:** The program should indicate that the shopping list is empty. * **Result:** Expected. * **Conclusion:** An empty list's error handling in the program is operating as intended.   **Test 3: Input Non-Numeric Value for Quantity**   * **Action:** Select option 1, then enter "two" in lieu of a number when asked about quantity. * **Anticipated Result:** The user ought to be prompted by the program to input a legitimate number. **Result:** As expected. * **Conclusion:** The program successfully handles non-numeric values for the quantity input.   **Error 1:** java.lang.exception in thread "main"NumberFormatException: "two" is the input string.  **Details:** There was a mistake when entering "two" for the quantity.  **Tried Remedies:**  1. At first, input parsing was done without any error handling.  2. Successful Fix: To handle NumberFormatException and ask the user for a valid input, a try-catch block was added around the Double.parseDouble() function.  **Error 2:** java.lang Exception in thread "main"Details of NullPointerException: An error occurred when attempting to calculate the price of an item that was not included in the itemPrices map.  **Tried Remedies:**  1. Made an attempt to retrieve the price without first confirming that the key was present in the map.  2. Effective Fix: Verify that the item is present in the itemPrices map before attempting to fetch a price. If it is not, issue a warning. |

## PART 5: Commenting Your Program

|  |
| --- |
| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.  **Requirements**   * The purpose of the program and each of its parts should be clear to a reader that does not know the Java programming language.   **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer. |
| <Paste your full program code here, including comments>  // Java packages for lists and maps.  import java.util.ArrayList;  import java.util.HashMap;  import java.util.Scanner;  /\*\*  \*Users can manage their grocery shopping list with the help of the GroceryShoppingAssistant program. Users can add products to \*the list, and it gives them an estimate of the total cost based on set prices.  \*/  public class GroceryShoppingAssistant {  // List for storing items user wants to purchase.  private static ArrayList<GroceryItem> shoppingList = new ArrayList<>();  // Mapping to store predefined item prices.  private static HashMap<String, Double> itemPrices = new HashMap<>();  // Object to read user inputs.  private static Scanner scanner = new Scanner(System.in);  public static void main(String[] args) {  // Sample prices for items.  populateSamplePrices();  int choice;  do {  // Displaying main menu options.  System.out.println("\nChoose an option:");  System.out.println("1. Add item to shopping list");  System.out.println("2. Estimate total cost");  System.out.println("3. Exit");  // User's choice.  choice = getIntInput("Enter your choice: ");  // Execute the certain action based on the user's choice.  switch (choice) {  case 1:  addGroceryItem();  break;  case 2:  estimateTotalCost();  break;  case 3:  System.out.println("Thank you for using the Grocery Shopping Assistant.");  break;  default:  System.out.println("Invalid choice. Please try again.");  break;  }  } while (choice != 3);  }  /\*\*  \* Add an item to list.  \*/  public static void addGroceryItem() {  GroceryItem newItem = new GroceryItem();    System.out.print("Enter the name of the grocery item: ");  newItem.name = scanner.nextLine().trim();  newItem.quantity = getDoubleInput("Enter the quantity for " + newItem.name + ": ");  System.out.print("Enter your preferred brand for " + newItem.name + ", or leave blank if no preference: ");  newItem.brand = scanner.nextLine().trim();  shoppingList.add(newItem);  System.out.println(newItem.name + " has been added to your shopping list.");  }  /\*\*  \* Based on the items in the shopping list and their established prices, provide an estimate of the overall cost.  \*/  public static void estimateTotalCost() {  if (shoppingList.isEmpty()) {  System.out.println("Your shopping list is empty.");  return;  }  double totalCost = 0;  for (GroceryItem item : shoppingList) {  Double itemPrice = itemPrices.get(item.name); // Fetch the price from our map.  if (itemPrice == null) {  System.out.println("Warning: No price available for " + item.name + ". Consider updating itemPrices map.");  continue;  }  double itemTotal = itemPrice \* item.quantity;  totalCost += itemTotal;  }  System.out.println("The estimated total cost of your shopping list is: $" + String.format("%.2f", totalCost));  }  /\*\*  \* Valid integer input from the user.  \*/  public static int getIntInput(String prompt) {  int result;  while (true) {  try {  System.out.print(prompt);  result = Integer.parseInt(scanner.nextLine());  break;  } catch (NumberFormatException e) {  System.out.println("Please enter a valid integer.");  }  }  return result;  }  /\*\*  \* Valid double input from the user.  \*/  public static double getDoubleInput(String prompt) {  double result;  while (true) {  try {  System.out.print(prompt);  result = Double.parseDouble(scanner.nextLine());  break;  } catch (NumberFormatException e) {  System.out.println("Please enter a valid number.");  }  }  return result;  }  /\*\*  \* Sample prices into the itemPrices.  \*/  public static void populateSamplePrices() {  itemPrices.put("milk", 2.50);  itemPrices.put("bread", 1.20);  itemPrices.put("egg", 0.20);  // Items and their prices can be added here.  }  }  /\*\*  \* Class to represent a grocery item  \*/  class GroceryItem {  String name;  double quantity;  String brand;  } |

## PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the Replit link to your full program code.  **Requirements**   * The program must work correctly with all the comments included in the program.   **Inspiration**   * Check before submitting your Touchstone that your final version of the program is running successfully. |
| <Provide the link to your program here>  https://replit.com/join/jiaxsfwude-analystbean |