Data Structure and Algorithm Practicum
Midterm Exam

Name
Dicha Zelianivan Arkana

NIM
2241720002

Class
1i

Department
Information Technology

Study Program
D4 Informatics Engineering

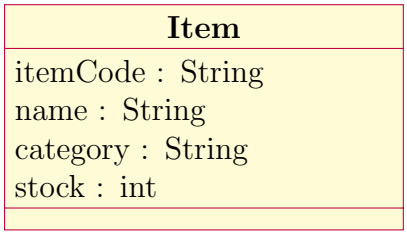
1 Features

These are the feature of the program

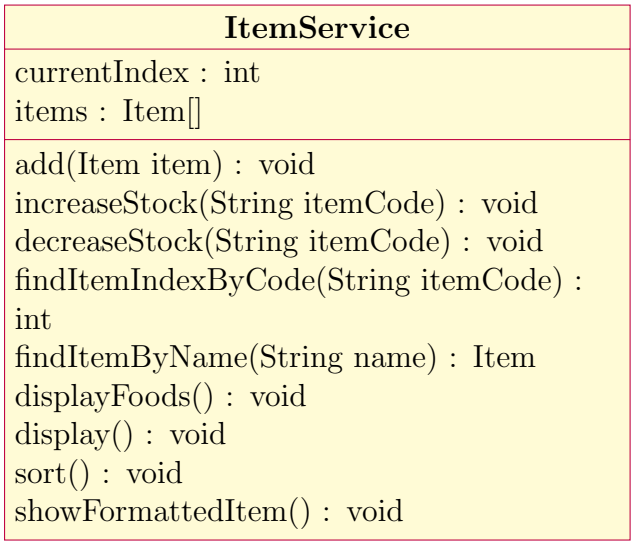
1. Input/add Item data
2. Display all Item data
3. Sort Item data based on the stock values in ascending mode
4. Display Items data classified as food that have no stock
5. Search Item data based on the name keyword
6. Add the stock for certain Item
7. Decrease the stock for certain Item

2 Class Diagram

- `Item.java`



- `ItemService.java`



- `Main.java`

Main
input : Scanner
main(String[] args) : void showMenu(Item item) : void getNumberInput(String prompt, int min, int max) : int getStringInput(String prompt) : String

3 Code

1. **Item.java**

```java
public class Item {
    String itemCode;
    String name;
    String category;
    int stock;

    public Item(String itemCode, String name, String category, int stock) {
        this.itemCode = itemCode;
        this.name = name;
        this.category = category;
        this.stock = stock;
    }
}
```

2. **ItemService.java**

```java
public class ItemService {
    private int currentIndex = -1;
    private final Item[] items = new Item[100];

    /**
     * Adds a new item to the storage
     * @param item The item you want to add
     */
    public void add(Item item) {
        if (currentIndex > items.length - 1) {
            System.out.println(
                "The storage is already full, " +
                "please remove an item before adding a new one.");
            return;
        }

        currentIndex++;
        items[currentIndex] = item;
    }

    /**
     * Adds the stock for certain item using the itemCode
     * @param itemCode The code of the item
     * @param stock The amount of stock
     */
```

```
26      public void increaseStock(String itemCode, int stock) {
27          int itemIndex = findItemIndexByCode(itemCode);
28          if (itemIndex == -1) {
29              System.out.println("There is no item with an id of: " + itemCode);
30              return;
31          }
32
33          items[itemIndex].stock += stock;
34      }
35
36      /**
37       * Adds the stock for certain item using the itemCode
38       * @param itemCode The code of the item
39       * @param stock The amount of stock
40       */
41      public void decreaseStock(String itemCode, int stock) {
42          int itemIndex = findItemIndexByCode(itemCode);
43          if (itemIndex == -1) {
44              System.out.println("There is no item with an id of: " + itemCode);
45              return;
46          }
47
48          items[itemIndex].stock -= stock;
49      }
50
51      /**
52       * Finds an item index based on its item code
53       * @param itemCode The code of the item you want to find
54       * @return The item index if found, -1 if not found
55       */
56      public int findItemIndexByCode(String itemCode) {
57          for (int i = 0; i < currentIndex; i++) {
58              if (items[i].itemCode.equals(itemCode)) {
59                  return i;
60              }
61          }
62          return -1;
63      }
64
65      /**
66       * Finds an item based on its name. This method is case insensitive
67       * @param name The name of the item
68       * @return The matching item, null if not found
69       */
70      public Item findItemByName(String name) {
```

```
71          for (int i = 0; i < currentIndex; i++) {
72              if (items[i].name.equalsIgnoreCase(name)) {
73                  return items[i];
74              }
75          }
76          return null;
77      }
78
79      /**
80       * Displays every items that has a category of 'food'. Case insensitive
81       */
82       public void displayFoodsWithNoStock() {
83          for (int i = 0; i < currentIndex; i++) {
84              Item currentItem = items[i];
85              if (!currentItem.category.equalsIgnoreCase("food")
86                      && currentItem.stock > 0) continue;
87              showFormattedItem(currentItem);
88          }
89      }
90
91      /**
92       * Display all items
93       */
94      public void display() {
95          for (int i = 0; i < currentIndex; i++) {
96              Item currentItem = items[i];
97              showFormattedItem(currentItem);
98          }
99      }
100
101     /**
102      * Sorts the items using bubble sort algorithm based on its stock (ascending)
103      */
104     public void sort() {
105         for (int i = 0; i < currentIndex - 1; i++) {
106             for (int j = 1; j < currentIndex - i; j++) {
107                 if (items[j].stock < items[j - 1].stock) {
108                     Item tmp = items[j];
109                     items[j] = items[j - 1];
110                     items[j - 1] = tmp;
111                 }
112             }
113         }
114     }
115
```

```
116     /**
117      * Shows the item using predefined format to stdout
118      * @param item The item you want to show
119      */
120     public void showFormattedItem(Item item) {
121         System.out.println("------------------");
122         System.out.println("Item Code\t: " + item.itemCode);
123         System.out.println("Name\t\t: " + item.name);
124         System.out.println("Category\t: " + item.category);
125         System.out.println("Stock\t\t: " + item.stock);
126         System.out.println("------------------");
127     }
128 }
```

3. **Main.java**

```
1  public class Main {
2      private static Scanner input = new Scanner(System.in);
3
4      public static void main(String[] args) {
5          Item[] items = {
6                  new Item("16030927", "Indomilk", "drink", 100),
7                  new Item("16100617", "Sprite", "drink", 70),
8                  new Item("16240401", "Yakult", "drink", 500),
9                  new Item("16270525", "Indomie", "food", 250),
10                 new Item("16971204", "Oreo", "food", 320),
11                 new Item("16100727", "Chocochips", "food", 120),
12                 new Item("16460329", "Ballpoint", "stationary", 75),
13                 new Item("16320421", "Pencil", "stationary", 110),
14                 new Item("16180729", "Book", "stationary", 57),
15         };
16         ItemService itemService = new ItemService();
17
18         // seed with initial data
19         for (Item item : items) {
20             itemService.add(item);
21         }
22
23         // loop indefinitely until the user wants to stop
24         while (true) {
25             showMenu();
26             int chosenMenu = getNumberInput("Choose Menu: ", 1, 7);
27             switch (chosenMenu) {
28                 case 1: {
29                     String itemCode = getStringInput("Insert the item code: ");
```

```
30                      String name = getStringInput("Insert the name: ");
31                      String category = getStringInput("Insert the category: ");
32                      int stock = getNumberInput("Insert the initial stock: ", 1, 1000);
33                      itemService.add(new Item(itemCode, name, category, stock));
34                      System.out.println("New item has been successfully added");
35                      break;
36                  }
37                  case 2: {
38                      itemService.display();
39                      break;
40                  }
41                  case 3: {
42                      itemService.sort();
43                      System.out.println(
44                              "Items has been sorted based on" +
45                                      " its stock value (ascending)"
46                      );
47                      break;
48                  }
49                  case 4: {
50                      itemService.displayFoodsWithNoStock();
51                      break;
52                  }
53                  case 5: {
54                      String name = getStringInput("Insert the food name: ");
55                      Item item = itemService.findItemByName(name);
56                      if (item == null) {
57                          System.out.println(
58                                  "The item was not found. " +
59                                  "Try with other name instead.");
60                          break;
61                      }
62                      itemService.showFormattedItem(item);
63                      break;
64                  }
65                  case 6: {
66                      String itemCode = getStringInput("Insert the item code: ");
67                      int stock = getNumberInput(
68                              "Insert the stock you want to add: ",
69                              1, 1000
70                      );
71                      itemService.increaseStock(itemCode, stock);
72                      System.out.println(
73                              "Stock has been added to an item with an ID of: "
74                              + itemCode
```

```
75                      );
76                      break;
77                  }
78                  case 7: {
79                      String itemCode = getStringInput("Insert the item code: ");
80                      int stock = getNumberInput(
81                              "Insert the stock you want to add: ",
82                              1, 1000
83                      );
84                      itemService.decreaseStock(itemCode, stock);
85                      System.out.println(
86                              "Stock has been added to an item with an ID of: "
87                              + itemCode
88                      );
89                      break;
90                  }
91                  default: {
92                      System.out.println("Incorrect choice, please try again!");
93                  }
94              }
95
96              // break from the loop if the user no longer wants to continue
97              String shouldContinueAnswer = getStringInput(
98                      "Do you want to continue (y/n)? "
99              );
100             if (!shouldContinueAnswer.equalsIgnoreCase("y")) break;
101         }
102     }
103
104     /**
105      * Shows the available menu for the program
106      */
107     private static void showMenu() {
108         String[] menu = {
109                 "Add Item",
110                 "Display All Items",
111                 "Sort Items by Stock (Ascending)",
112                 "Display All Foods",
113                 "Search Item by Name",
114                 "Add Stock for Item",
115                 "Decrease Stock for Item",
116         };
117         System.out.println("== Stock Management Program == ");
118         for (int i = 0; i < menu.length; i++) {
119             System.out.printf("%d. %s\n", i + 1, menu[i]);
```

```
120             }
121         }
122 
123         /**
124          * Shortcut to get a user input in form of int
125          * @param prompt The prompt to show to the user
126          * @param min The minimum valid amount
127          * @param max The maximum valid amount
128          * @return The answer in integer within the specified bound
129          */
130         private static int getNumberInput(String prompt, int min, int max) {
131             while (true) {
132                 System.out.print(prompt);
133                 int answer = input.nextInt();
134                 // consumes the next newline
135                 input.nextLine();
136                 if (answer >= min && answer <= max) return answer;
137                 System.out.printf("The input can only be between %d and %d.", min, max);
138             }
139         }
140 
141         /**
142          * Shortcut to get a user input in form of String
143          * @param prompt The prompt to show to the user
144          * @return The answer as a string
145          */
146         private static String getStringInput(String prompt) {
147             System.out.print(prompt);
148             return input.nextLine();
149         }
150     }
```