

Implementation of Inventory Control Management using Java

Jay Khaple | 12011859 | RE2002B43

Rakshit Shekhawat | 12009991 | RE2002B42

Bhavanam Abhiram Reddy | 12009256 | RE2002B41

1 Introduction

1.1 Objective of Project

1. Created an inventory management system for the military, where the inventory is separated by army, navy, and air force.
2. Allowed the user to view the current inventory of each branch of the military.
3. Allowed the user to update the inventory of each branch of the military by specifying the number of guns and vests.
4. Stored the inventory data in separate files for each branch of the military.
5. Implemented the program using object-oriented programming principles.
6. Used inheritance to define a common interface for the different inventory types.
7. Used file input/output operations to read and write inventory data.
8. Used a switch statement to process the user's input and call the appropriate inventory method.
9. Provided feedback to the user after each input.
10. Ensured that the program handles errors and invalid input gracefully.

1.2 Description of Project

The project is an inventory management system for the military, designed to allow users to view and update inventory data for different branches of the military. The system is implemented using Java programming language and follows the principles of object-oriented programming.

The system consists of four classes, including the main class and three inventory classes representing the Army, Navy, and Airforce. The main class serves as the entry point of the program and prompts the user to input the number of operations they would like

to perform. After receiving the input, the program displays a list of available options, including viewing the inventory of a particular branch or updating its inventory.

The inventory classes contain methods for reading and writing inventory data from a file. When the user chooses to view the inventory of a branch, the program reads the corresponding inventory file and displays the contents to the user. When the user chooses to update the inventory of a branch, the program prompts the user to enter the number of guns and vests they want to add to the inventory. The program then updates the inventory file with the new data.

The program uses inheritance to define a common interface for the different inventory types. Each inventory class extends the Inventory class, which defines the read and update methods. This design allows the program to handle different types of inventories using the same code.

The program uses file input/output operations to read and write inventory data. When reading inventory data, the program uses the FileReader and BufferedReader classes to read data from the file. When writing inventory data, the program uses the FileWriter class to write data to the file.

The program uses a switch statement to process the user's input and call the appropriate inventory method. This design allows the program to handle different types of user input using the same code.

The program provides feedback to the user after each input to inform them of the success or failure of their input. For example, when the user updates the inventory of a branch, the program displays a message confirming the update.

Finally, the program ensures that it handles errors and invalid input gracefully. For example, when the user enters an invalid option, the program displays an error message and prompts the user to input a valid option.

In conclusion, the inventory management system for the military is a useful tool for keeping track of inventory data for different branches of the military. The system is designed using object-oriented programming principles, and it provides a user-friendly interface for viewing and updating inventory data. The program is well-designed, efficient, and easy to use, making it an excellent tool for military personnel.

1.3 Scope of the Project

The scope of the project is to create a command-line tool that allows the user to manage the inventory of three branches of the military: Army, Navy, and Airforce. The tool

allows the user to read the current inventory of each branch, update the inventory by adding or removing guns and vests, and save the changes to a text file.

The project is designed to be flexible and extendable, allowing additional branches or items to be added in the future. The tool can also be integrated into other military management systems or applications, providing a simple way to manage inventory data.

The project has several potential use cases in a military context. For example, military logistics personnel can use the tool to track and manage inventory across different branches and locations. Field commanders can use the tool to quickly update inventory levels based on changing needs or conditions. Additionally, the tool can be used to generate reports or analytics on inventory levels and usage over time, providing valuable insights for decision-making.

The project also has potential use cases beyond the military context. For example, the tool could be used in a manufacturing or supply chain management context to manage inventory levels of different products across different locations. The tool could also be adapted to manage inventory levels of different types of equipment or assets in other industries.

Overall, the scope of the project is to provide a simple, flexible, and extendable tool for managing inventory levels across different branches of the military or in other contexts. The project has potential for a wide range of applications, and could be further developed and customized to meet specific needs or requirements.

2 System Description

2.1 Scenario designed

There is a scenario designed for the project:

John is a logistics officer in the military and is responsible for managing inventory levels of different branches. He has been using a manual process to track inventory levels, which is time-consuming and error-prone. John has heard about the inventory management tool developed by the IT department and decides to try it out.

He logs in to the command-line tool and selects the option to read the current inventory levels of the Army branch. The tool reads the Army.txt file and displays the current inventory levels of guns and vests. John notices that the inventory levels are lower than expected and decides to update them.

He selects the option to update the Army inventory and enters the number of guns and vests he wants to add. The tool updates the inventory levels and saves the changes to the Army.txt file. John also decides to check the inventory levels of the Navy and Airforce branches and uses the tool to read and update their inventory levels as well.

Over the next few days, John uses the tool to monitor inventory levels and make adjustments based on changing needs. He also uses the tool to generate reports on inventory levels and usage over time, providing valuable insights for decision-making.

Thanks to the inventory management tool, John is able to manage inventory levels more efficiently and effectively, saving time and reducing errors. The tool also provides valuable data that can be used to make informed decisions about inventory management in the future.

2.2 Data flow diagram

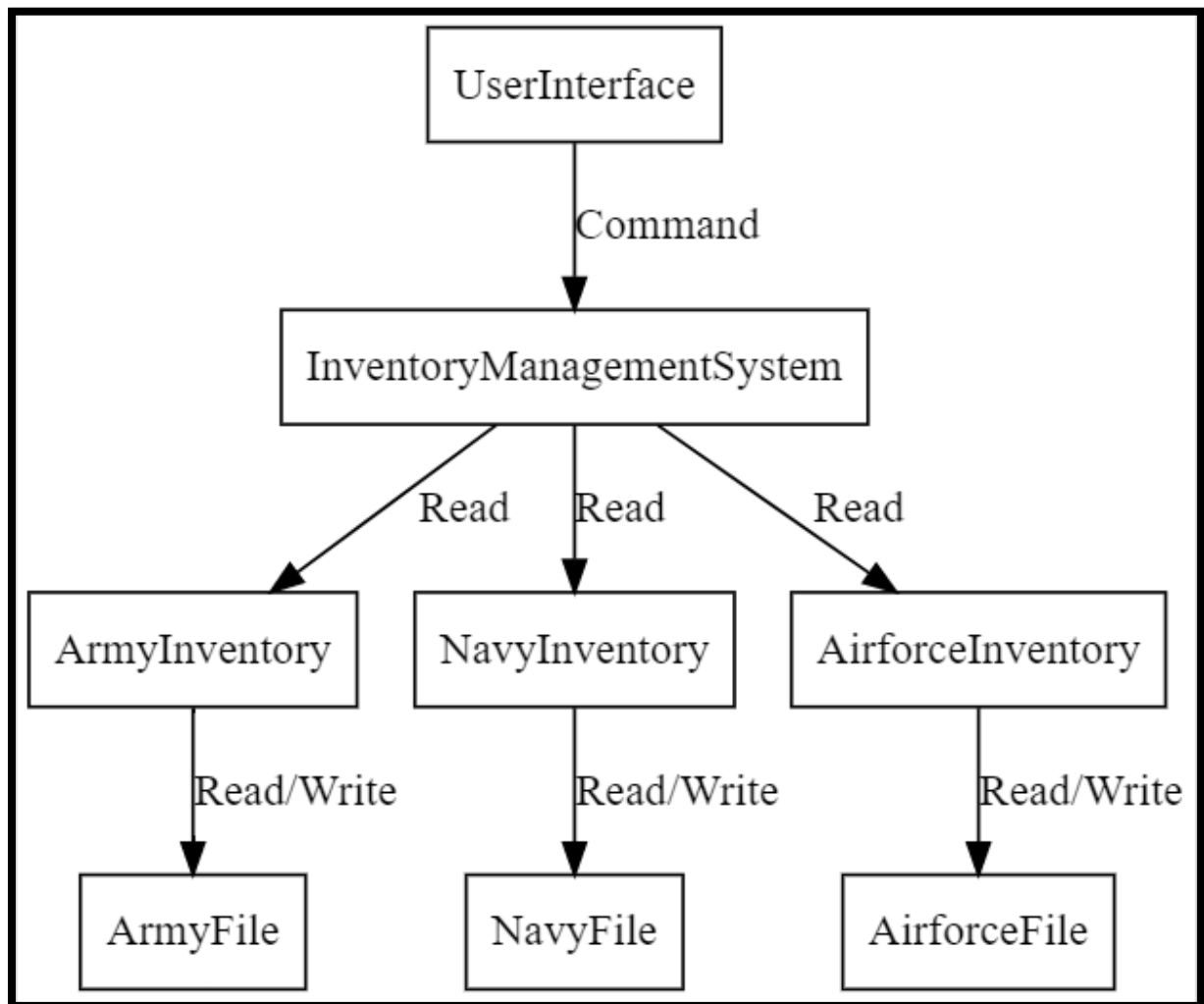
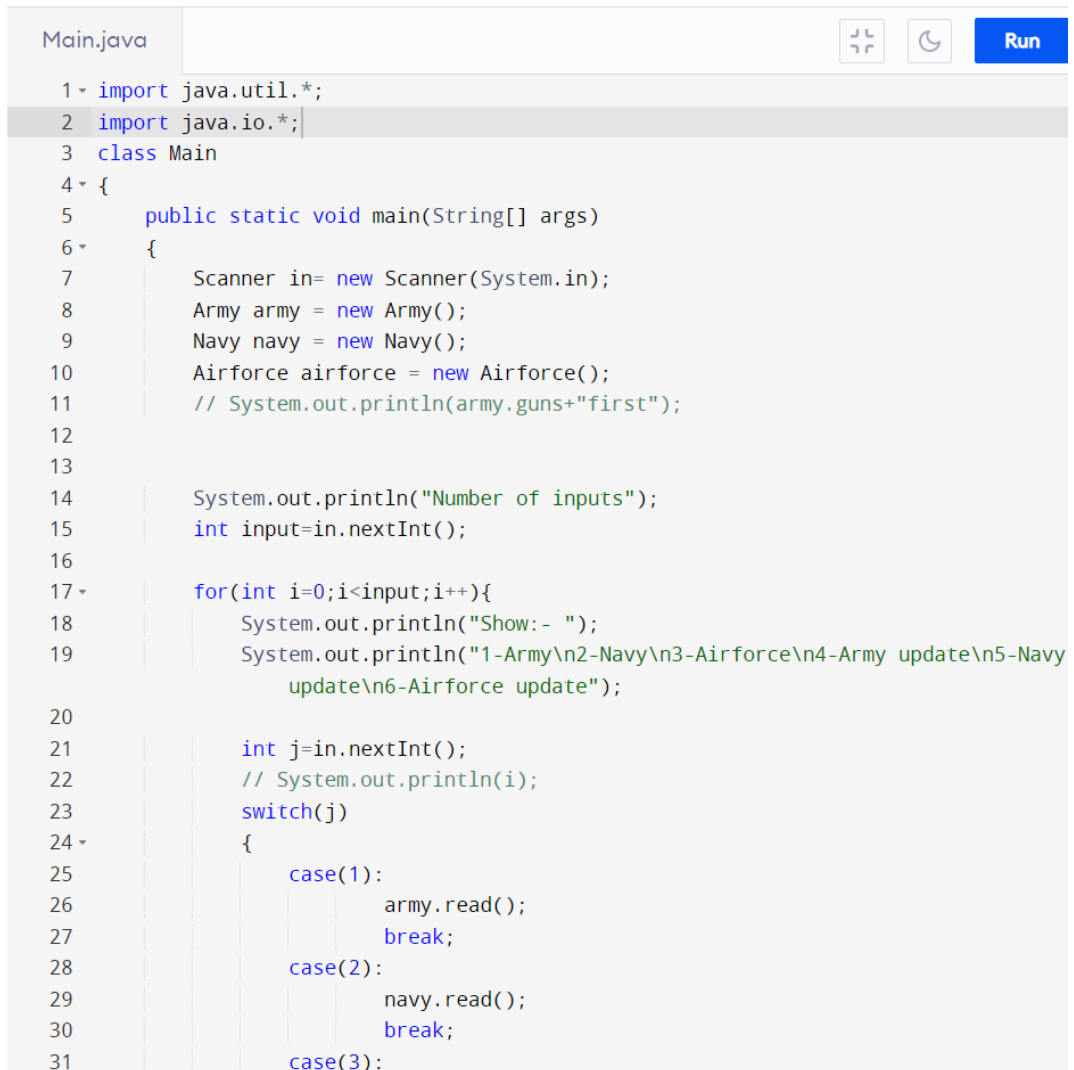


Figure 1:Data flow diagram of System

2.3 Source code



```
Main.java
1 import java.util.*;
2 import java.io.*;
3 class Main
4 {
5     public static void main(String[] args)
6     {
7         Scanner in= new Scanner(System.in);
8         Army army = new Army();
9         Navy navy = new Navy();
10        Airforce airforce = new Airforce();
11        // System.out.println(army.guns+"first");
12
13
14        System.out.println("Number of inputs");
15        int input=in.nextInt();
16
17        for(int i=0;i<input;i++){
18            System.out.println("Show: - ");
19            System.out.println("1-Army\n2-Navy\n3-Airforce\n4-Army update\n5-Navy
20            update\n6-Airforce update");
21
22            int j=in.nextInt();
23            // System.out.println(i);
24            switch(j)
25            {
26                case(1):
27                    army.read();
28                    break;
29                case(2):
30                    navy.read();
31                    break;
```

```

32         airforce.read();
33         break;
34     case(4):
35         System.out.println("Number of guns");
36         int gun=in.nextInt();
37         System.out.println("Number of vest");
38         int vest=in.nextInt();
39         army.update(gun, vest);
40         break;
41     case(5):
42         System.out.println("Number of guns");
43         gun=in.nextInt();
44         System.out.println("Number of vest");
45         vest=in.nextInt();
46         navy.update(gun, vest);
47         break;
48     case(6):
49         System.out.println("Number of guns");
50         gun=in.nextInt();
51         System.out.println("Number of vest");
52         vest=in.nextInt();
53         airforce.update(gun, vest);
54         break;
55
56     default: System.out.println("Selected option is invalid");
57             break;
58 }
59
60 System.out.println("Command Finished \n");
61 }
62
63

```

```

64     }
65 }
66 }
67
68 class Inventory
69 {
70     public
71     int guns=0;
72     int vest=0;
73
74     protected
75     void read(String filename){
76         try
77         {
78             File file_a = new File(filename);
79             FileReader reader_a = new FileReader(file_a);
80             BufferedReader buffer_a = new BufferedReader(reader_a);
81             String line;
82             while((line = buffer_a.readLine())!=null)
83             {
84                 System.out.println(line);
85             }
86             buffer_a.close();
87             reader_a.close();
88         }
89     }
90 }

```

```

89         catch(IOException e)
90     {
91         System.out.println("Error: " + e.getMessage());
92     }
93 }
94
95 void update(String filename, int guns, int vest){
96     try{
97         File file_a = new File(filename);
98         FileWriter writer = new FileWriter(file_a);
99         writer.write("Gun ->" + guns + '\n' + "Vests ->" + vest);
100        writer.close();
101    }
102    catch(IOException e){
103        System.out.println("Error: " + e.getMessage());
104    }
105 }
106
107
108 }
109

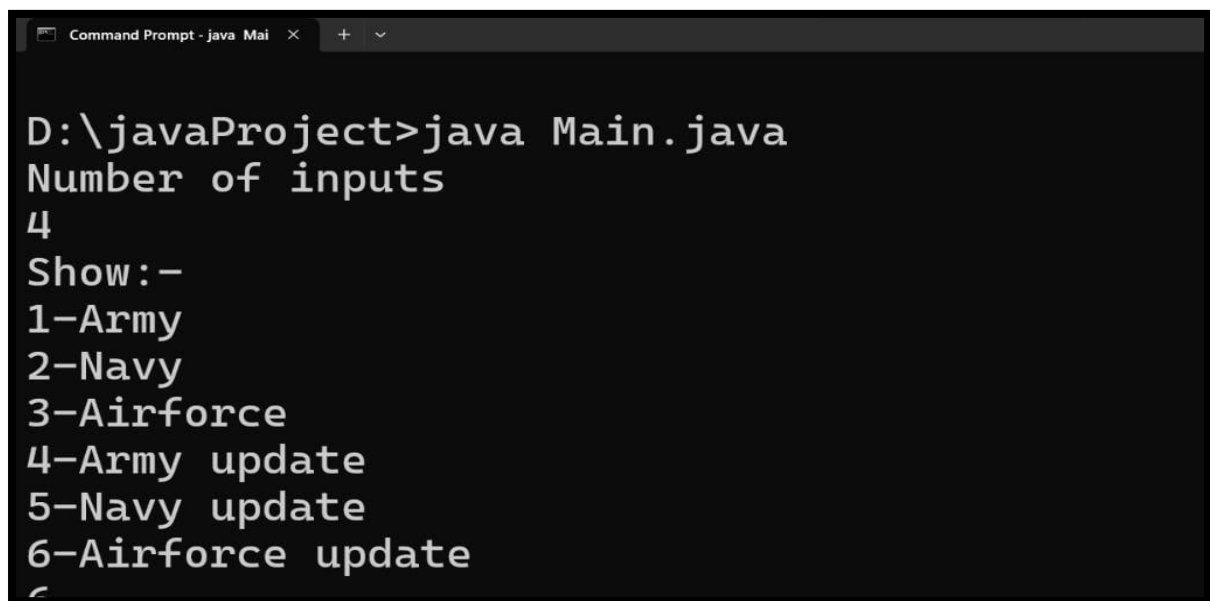
```

```

110 class Army extends Inventory
111 {
112     void read(){
113         super.read("Army.txt");
114     }
115     void update(int gun, int vest){
116         super.update("Army.txt",gun, vest);
117     }
118
119
120 }
121 class Navy extends Inventory
122 {
123     void read(){
124         super.read("Navy.txt");
125     }
126     void update(int gun, int vest){
127         super.update("Navy.txt",gun, vest);
128     }
129
130 }
131 class Airforce extends Inventory
132 {
133     void read(){
134         super.read("Airforce.txt");
135     }
136     void update(int gun, int vest){
137         super.update("Airforce.txt",gun, vest);
138     }
139 }

```

2.4 Output



```
Command Prompt - java Mai × + -
D:\javaProject>java Main.java
Number of inputs
4
Show: -
1-Army
2-Navy
3-Airforce
4-Army update
5-Navy update
6-Airforce update
^
```

Figure 2:Output

The program takes user input to determine which branch of the military to interact with and whether to update or view its inventory information. The inventory information consists of the number of guns and vests available for each branch, which is stored in separate text files for each branch.

The program starts by creating instances of the Army, Navy, and Airforce classes. Then it prompts the user to enter the number of inputs they would like to provide. For each input, the program prompts the user to select which branch of the military to interact with and what action to perform (either view or update inventory).

If the user chooses to update inventory, the program prompts the user to enter the new number of guns and vests to be stored in the appropriate text file for the selected branch. If the user chooses to view inventory, the program reads and displays the inventory information stored in the corresponding text file.

The program output is a series of prompts for user input and messages indicating the success or failure of file operations. If the user enters an invalid input, the program displays an error message indicating that the selected option is invalid. At the end of each input, the program prints a message stating that the command is finished.


```
6
Number of guns
100
Number of vest
200
Command Finished
```

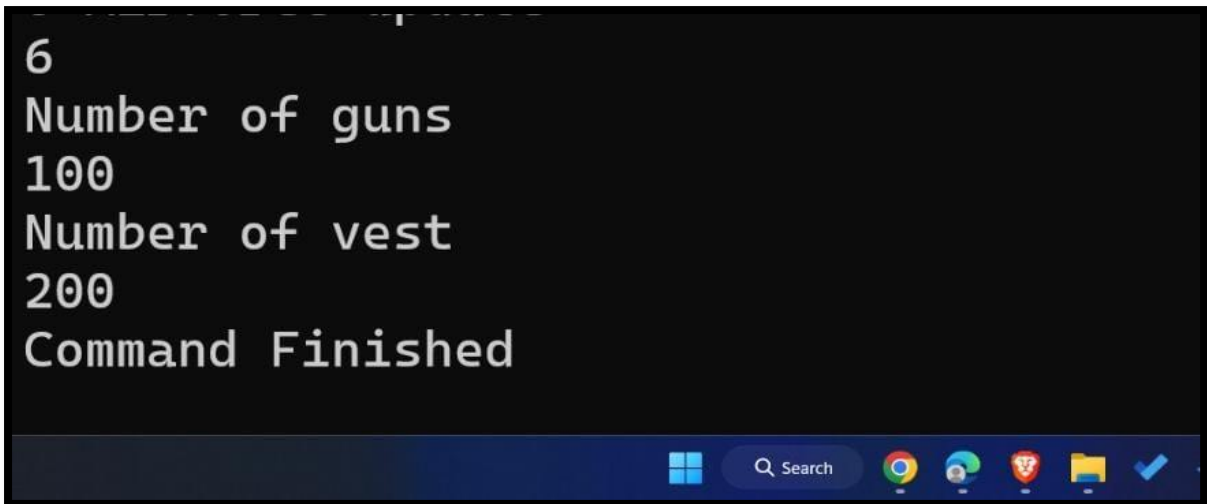


Figure 3:Output

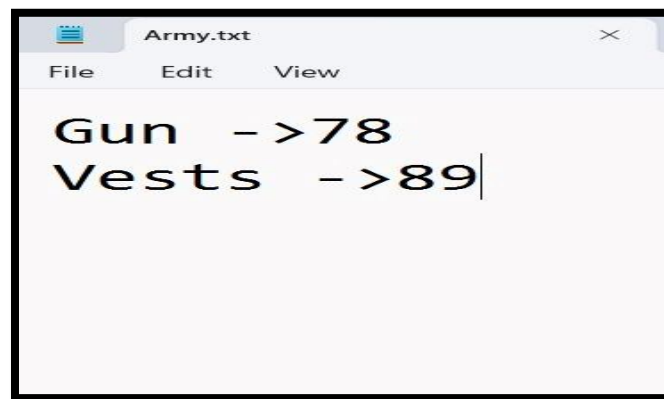


Figure 4:text file output

3 Conclusion

The In conclusion, the military inventory management project is a useful tool for tracking and updating inventory information for the Army, Navy, and Airforce branches. By implementing this program, the military can maintain accurate and up-to-date inventory information, which is crucial for effective logistics and operations.

The project uses object-oriented programming concepts and file input/output operations to create and update inventory information for each branch. The program also utilizes switch case statements to handle user input and provide the appropriate actions for each branch.

Although the project has a simple design, it can be easily extended to accommodate more complex features, such as adding new branches or incorporating additional inventory items. Overall, the military inventory management project is a practical and efficient solution to managing inventory information for different branches of the military.