UNIVERSITI POLY-TECH MALAYSIA UPTM

COURSE : CC101

Name: 1. **AIMAN HARITH BIN NORIZAL** 2. **HAFIY DANISH NURAKIF BIN RAZIMUHAR** 3. **MUHAMMAD SYAWAL BIN TAJUDIN** 4. **AIMAN NOR HAKIMI BIN OMAR**	ID Number: 1. **AM2408016818** 2. **AM2408016860** 3. **AM2408016794** 4. **AM2408016798**
Lecturer: **MUHAMMAD AMIN BIN MUSTAFA**	Section: **SECTION 2**
Course and Course Code: **SWC3684 DATA STRUCTURE**	Submission Date: **4 / 7 / 2025**
Assignment No. / Title: **ASSIGNMENT**	Date of assignment submission to the lecturer: **4/ 7 / 2025**
Assignment Mark:	Mark deduction:

Penalties:
1. 10% of the original mark will be deducted for every one-week period after the submission date.
2. No work will be accepted after two weeks of the deadline.
3. If you were unable to submit the coursework on time due to extenuating circumstances, **CONSULT** with your lecturer first preferably **THREE (3) DAYS** prior the assignment due date.
4. Extension **SHOULD NOT** exceed one week.
5. I/we agree that marks will be deducted if the work is proven to be plagiarized.

Declaration: I/we the undersigned confirm that I/we have read and agree to abide by these regulations on plagiarism and cheating. I/we confirm that this piece of work is my/our own. I/we consent to appropriate storage of our work for checking to ensure that there is no plagiarism/ academic cheating.

Signature:
Name:

AIMAN HARITH	**MUHAMMAD SYAWAL**
HAFIY DANISH	**AIMAN NOR HAKIMI**

TABLE OF CONTENT

1. Introduction

This project was created as a fun and educational way to explore how data structures work in programming by simulating a pirate mission management system based on the popular anime One Piece. Rather than just writing code for the sake of learning, this project adds a creative twist by turning data handling into a storyline involving pirate crews, dangerous missions, and bounties.

The main idea behind the project is to apply key data structures which is **LinkedList**, **Queue**, and **Stack** to organize and process mission data for different pirate crew members. First, all the crew and their missions are read from a text file and stored using a LinkedList, which makes it easy to manage multiple entries. Each mission contains information like the **mission ID**, **type**, **level of danger**, **how long it takes**, **the date**, and **how much bounty it offers.**

After that, the system assigns each crew member to a mission route (**East Blue**, **South Blue**, or **Grand Line**) based on how many missions they have. This part uses a Queue, which processes data in the same order it arrives known as **FIFO** (First-In, First-Out). It's a fair way to handle routing, just like a line at the post office.

Once the missions are assigned, the system simulates what happens when those **missions are completed**. For this, a Stack is used, which follows the **LIFO** (Last-In, First-Out) concept. That means the last mission added to the stack is the first one to be marked as completed like a pile of papers where you deal with the top one first. To make it more meaningful, the system also keeps track of who completed which **mission**, **including their name**, **ID**, **crew**, and **total bounty earned.**

One of the key highlights of this project is its use of a **Graphical User Interface** (GUI). Instead of printing everything in the console, all output is shown using **JOptionPane** pop-up windows. This makes it **easier to read**, **more visually appealing**, and **gives it the feel of a real application**.

Overall, this project has been a great way to turn something technical like learning data structures into something **fun**, **creative**, and **useful**. It helped reinforce core programming skills in Java, especially **object-oriented programming** (OOP), and showed how important proper data handling is in building systems that make sense, even in a fictional pirate world.

2. IPO Analysis

INPUT	PROGRESS	OUTPUT
- onepiece_combined.txt file containing crew and mission data: • Crew ID, Name, Pirate Crew, Mission ID, Mission Type, Date, Duration, Bounty - User interaction through GUI (JOptionPane) to trigger data processing and display	File Reading: - Use BufferedReader to read data line by line - Split each line with `split("\	- Scrollable pop-up windows showing: • Crew mission details and total bounty • Mission routing by region • Completed missions (latest first) - Clear, interactive, user-friendly GUI display

3. UML Class Diagram & Flowchart

3.1 UML Diagram

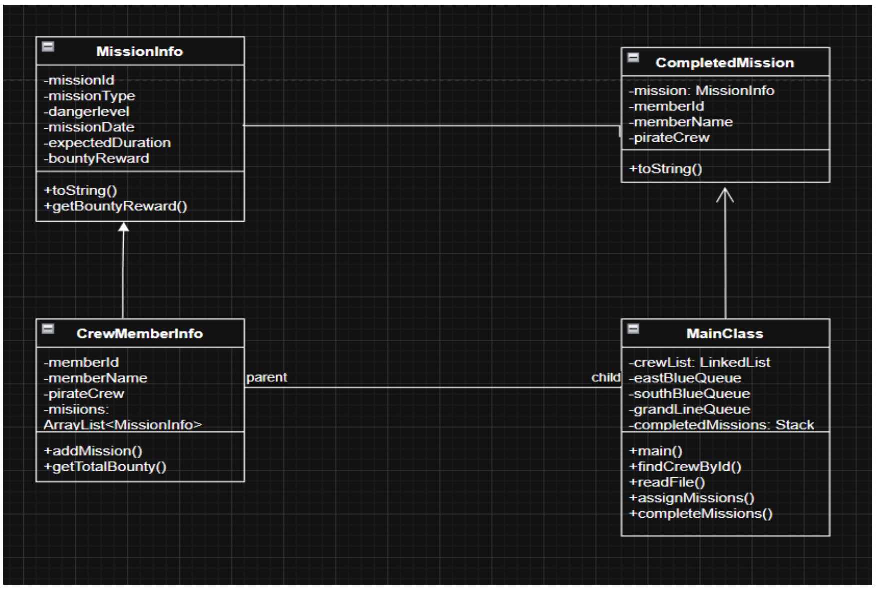
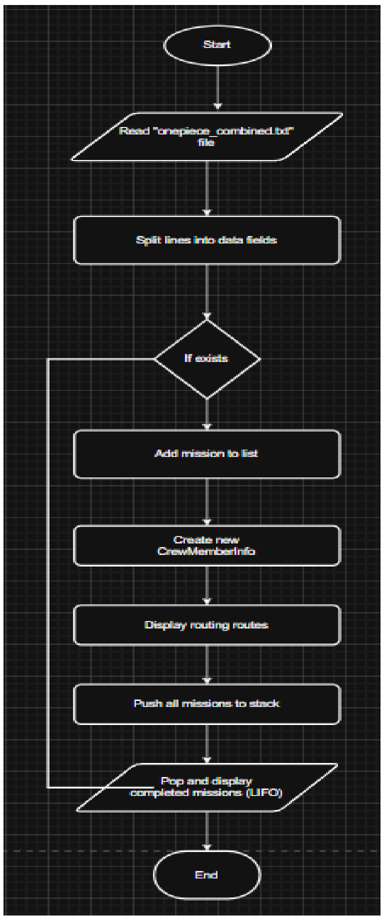


Figure 1: UML Diagram

3.2 Flow chart

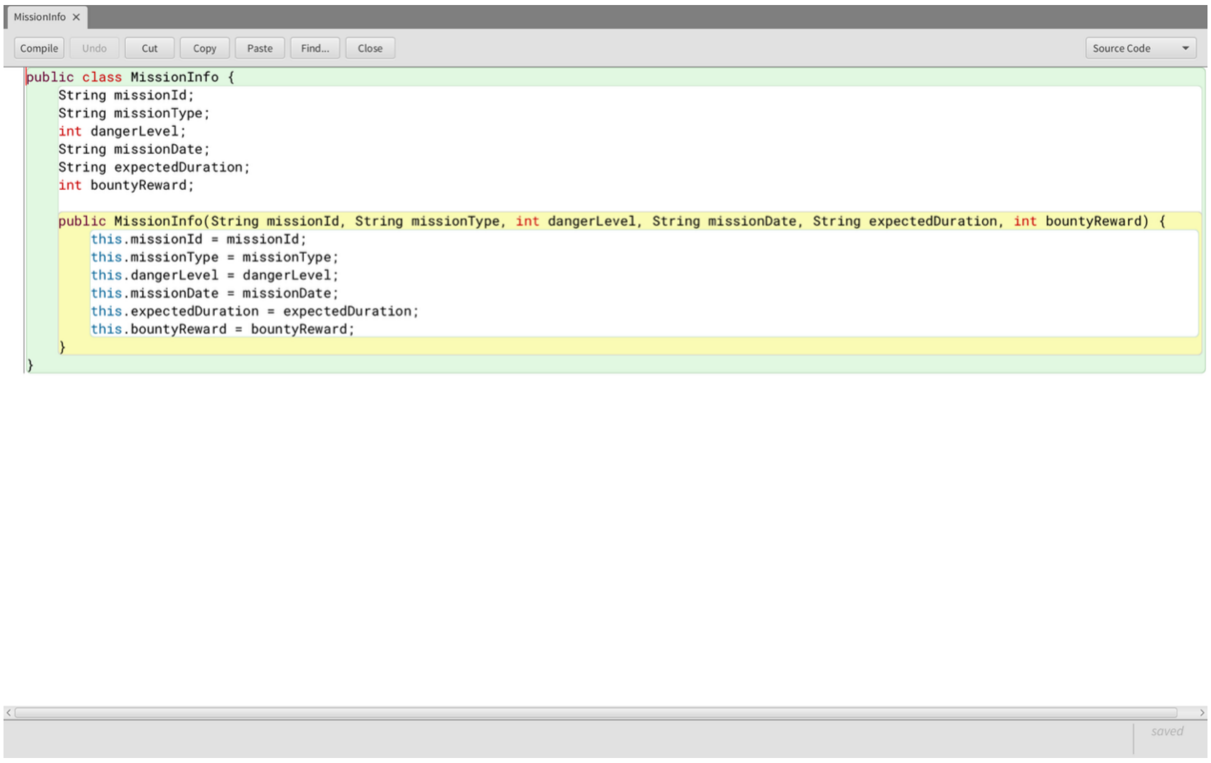


Start
Read "onepiece_combined.txt" file
Split lines into data fields
If exists
Add mission to list
Create new CrewMemberInfo
Display routing routes
Push all missions to stack
Pop and display completed missions (LIFO)
End

Figure 2: Flow chart

4. Coding

4.1 Mission Info class



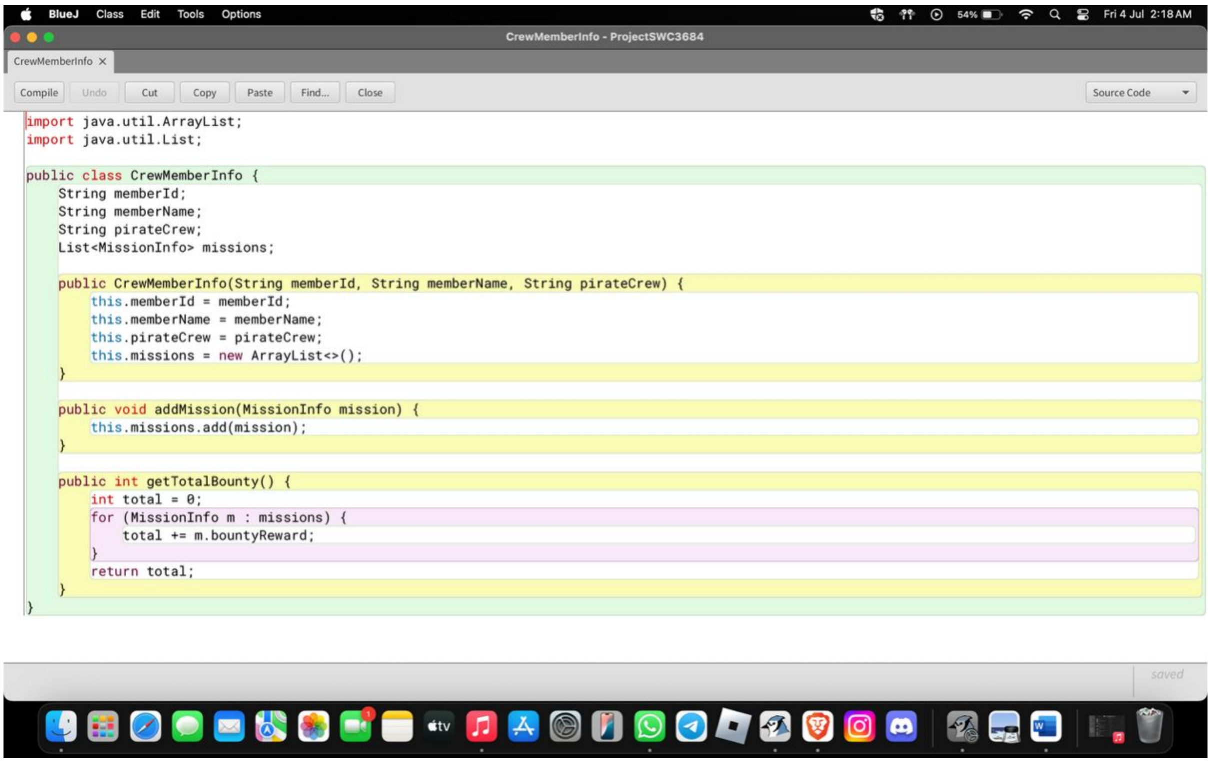
```java
public class MissionInfo {
    String missionId;
    String missionType;
    int dangerLevel;
    String missionDate;
    String expectedDuration;
    int bountyReward;

    public MissionInfo(String missionId, String missionType, int dangerLevel, String missionDate, String expectedDuration, int bountyReward) {
        this.missionId = missionId;
        this.missionType = missionType;
        this.dangerLevel = dangerLevel;
        this.missionDate = missionDate;
        this.expectedDuration = expectedDuration;
        this.bountyReward = bountyReward;
    }
}
```

Figure 3: Mission Info Class

The MissionInfo class is a Java class designed to represent details about a mission. It includes six attributes: missionId, missionType, dangerLevel, missionDate, expectedDuration, and bountyReward. These variables are used to store essential information such as the mission's unique identifier, its category, how dangerous it is, when it is scheduled, how long it is expected to last, and the reward given for completing it. The class has one constructor that takes all these values as parameters and assigns them to the corresponding fields using this keyword. This allows for easy creation of fully initialized mission objects.

However, there are areas where the class can be improved. Firstly, the attributes should be declared as private to follow the principle of encapsulation. This prevents direct access from outside the class and makes the class more secure and easier to maintain. If needed, getter and setter methods can be added to access and modify these values. Additionally, input validation could be implemented in the constructor to ensure data integrity (e.g., checking that dangerLevel and bountyReward are non-negative). Including a toString() method would also be helpful for displaying the mission details in a readable format.

4.2 Crew Member Info Class



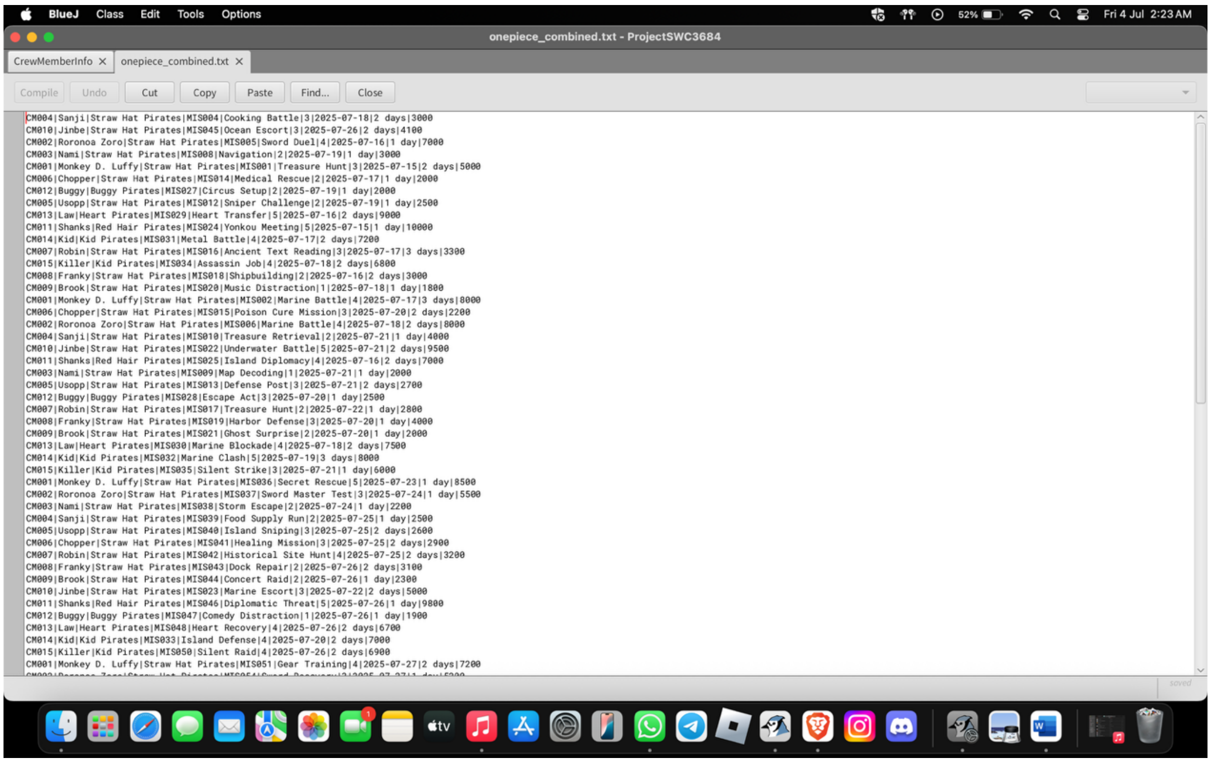
```java
import java.util.ArrayList;
import java.util.List;

public class CrewMemberInfo {
    String memberId;
    String memberName;
    String pirateCrew;
    List<MissionInfo> missions;

    public CrewMemberInfo(String memberId, String memberName, String pirateCrew) {
        this.memberId = memberId;
        this.memberName = memberName;
        this.pirateCrew = pirateCrew;
        this.missions = new ArrayList<>();
    }

    public void addMission(MissionInfo mission) {
        this.missions.add(mission);
    }

    public int getTotalBounty() {
        int total = 0;
        for (MissionInfo m : missions) {
            total += m.bountyReward;
        }
        return total;
    }
}
```

Figure 4: Crew Member Info Class

The CrewMemberInfo class in the image is a Java class designed to manage information about a crew member and the missions they participate in. It includes four main attributes: memberId, memberName, pirateCrew, and missions. The first three attributes are simple String values that store the crew member's ID, name, and the name of their pirate crew. The fourth attribute, missions, is a list of MissionInfo objects (defined in the previous image), allowing each crew member to be associated with multiple missions. The class uses a constructor to initialize these fields and instantiates the missions list as an empty ArrayList.

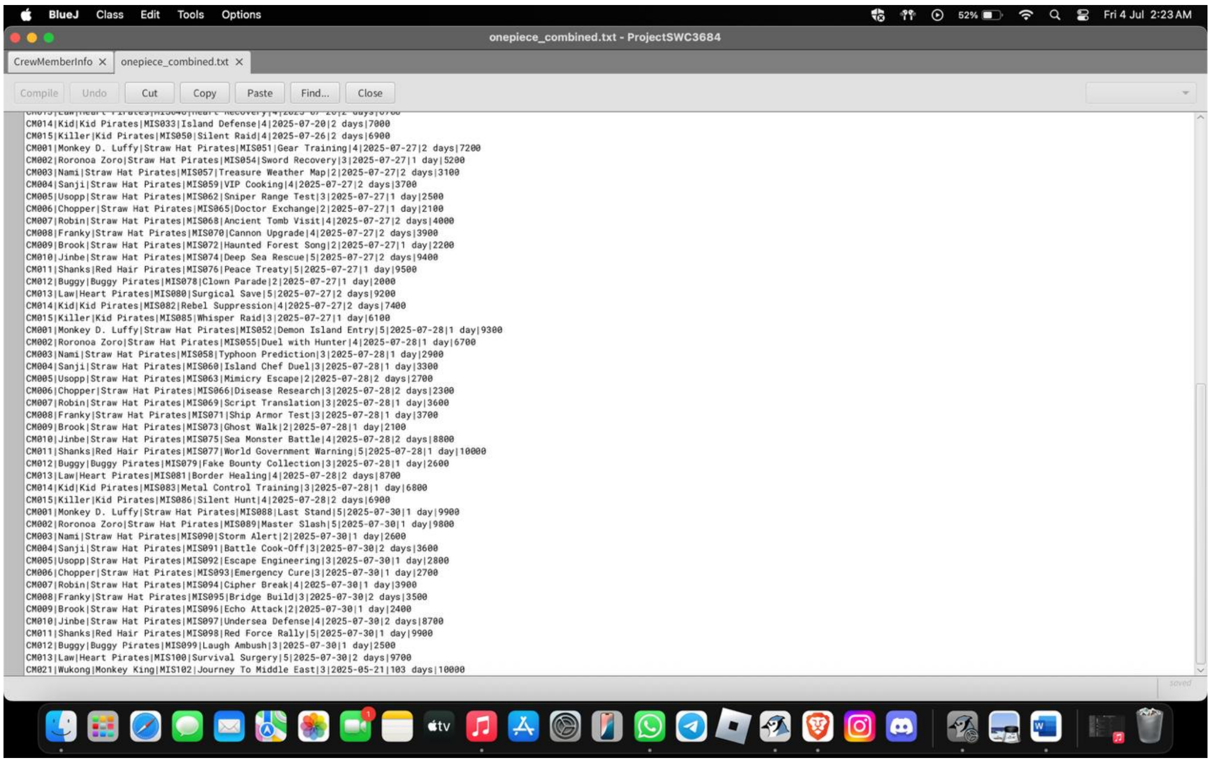
The addMission() method allows missions to be added to the crew member's list dynamically. This helps build a record of all missions assigned to a crew member. The getTotalBounty() method is particularly useful. It calculates the total bounty reward the crew member has earned from all missions by iterating through the missions list and summing up the bountyReward of each mission. However, the method currently accesses the bountyReward field directly (m.bountyReward), which will cause an error if bountyReward is private in the MissionInfo class. To fix this, a getter method like getBountyReward() should be used in the MissionInfo class.

4.3 One piece Combined .txt File



```
CM004|Sanji|Straw Hat Pirates|MIS004|Cooking Battle|3|2025-07-18|2 days|3000
CM010|Jinbe|Straw Hat Pirates|MIS045|Ocean Escort|3|2025-07-26|2 days|4100
CM002|Roronoa Zoro|Straw Hat Pirates|MIS005|Sword Duel|4|2025-07-16|1 day|7000
CM003|Nami|Straw Hat Pirates|MIS008|Navigation|2|2025-07-19|1 day|3000
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS001|Treasure Hunt|3|2025-07-15|2 days|5000
CM006|Chopper|Straw Hat Pirates|MIS014|Medical Rescue|2|2025-07-17|1 day|2000
CM012|Buggy|Buggy Pirates|MIS027|Circus Setup|2|2025-07-19|1 day|2000
CM005|Usopp|Straw Hat Pirates|MIS012|Sniper Challenge|2|2025-07-19|1 day|2500
CM013|Law|Heart Pirates|MIS029|Heart Transfer|5|2025-07-16|2 days|9000
CM011|Shanks|Red Hair Pirates|MIS024|Yonkou Meeting|5|2025-07-15|1 day|10000
CM014|Kid|Kid Pirates|MIS031|Metal Battle|4|2025-07-17|2 days|7200
CM007|Robin|Straw Hat Pirates|MIS016|Ancient Text Reading|3|2025-07-17|3 days|3300
CM015|Killer|Kid Pirates|MIS034|Assassin Job|4|2025-07-18|2 days|6800
CM008|Franky|Straw Hat Pirates|MIS018|Shipbuilding|2|2025-07-16|2 days|3000
CM009|Brook|Straw Hat Pirates|MIS020|Music Distraction|1|2025-07-18|1 day|1800
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS002|Marine Battle|4|2025-07-17|3 days|8000
CM006|Chopper|Straw Hat Pirates|MIS015|Poison Cure Mission|3|2025-07-20|2 days|2200
CM002|Roronoa Zoro|Straw Hat Pirates|MIS006|Marine Battle|4|2025-07-18|2 days|8000
CM004|Sanji|Straw Hat Pirates|MIS010|Treasure Retrieval|2|2025-07-21|1 day|4000
CM010|Jinbe|Straw Hat Pirates|MIS022|Underwater Battle|5|2025-07-21|2 days|9500
CM011|Shanks|Red Hair Pirates|MIS025|Island Diplomacy|4|2025-07-16|2 days|7000
CM003|Nami|Straw Hat Pirates|MIS009|Map Decoding|1|2025-07-21|1 day|2000
CM005|Usopp|Straw Hat Pirates|MIS013|Defense Post|3|2025-07-21|2 days|2700
CM012|Buggy|Buggy Pirates|MIS028|Escape Act|3|2025-07-20|1 day|2500
CM007|Robin|Straw Hat Pirates|MIS017|Treasure Hunt|2|2025-07-22|1 day|2800
CM008|Franky|Straw Hat Pirates|MIS019|Harbor Defense|3|2025-07-20|1 day|4000
CM009|Brook|Straw Hat Pirates|MIS021|Ghost Surprise|2|2025-07-20|1 day|2000
CM013|Law|Heart Pirates|MIS030|Marine Blockade|4|2025-07-18|2 days|7500
CM014|Kid|Kid Pirates|MIS032|Marine Clash|5|2025-07-19|3 days|8000
CM015|Killer|Kid Pirates|MIS035|Silent Strike|3|2025-07-21|1 day|6000
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS036|Secret Rescue|5|2025-07-23|1 day|8500
CM002|Roronoa Zoro|Straw Hat Pirates|MIS037|Sword Master Test|3|2025-07-24|1 day|5500
CM003|Nami|Straw Hat Pirates|MIS038|Storm Escape|2|2025-07-24|1 day|2200
CM004|Sanji|Straw Hat Pirates|MIS039|Food Supply Run|2|2025-07-25|1 day|2500
CM005|Usopp|Straw Hat Pirates|MIS040|Island Sniping|3|2025-07-25|2 days|2600
CM006|Chopper|Straw Hat Pirates|MIS041|Healing Mission|3|2025-07-25|2 days|2900
CM007|Robin|Straw Hat Pirates|MIS042|Historical Site Hunt|4|2025-07-25|2 days|3200
CM008|Franky|Straw Hat Pirates|MIS043|Dock Repair|2|2025-07-26|2 days|3100
CM009|Brook|Straw Hat Pirates|MIS044|Concert Raid|2|2025-07-26|1 day|2300
CM010|Jinbe|Straw Hat Pirates|MIS023|Marine Escort|3|2025-07-22|2 days|5000
CM011|Shanks|Red Hair Pirates|MIS046|Diplomatic Threat|5|2025-07-26|1 day|9800
CM012|Buggy|Buggy Pirates|MIS047|Comedy Distraction|1|2025-07-26|1 day|1900
CM013|Law|Heart Pirates|MIS048|Heart Recovery|4|2025-07-26|2 days|6700
CM014|Kid|Kid Pirates|MIS033|Island Defense|4|2025-07-20|2 days|7000
CM015|Killer|Kid Pirates|MIS050|Silent Raid|4|2025-07-26|2 days|6900
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS051|Gear Training|4|2025-07-27|2 days|7200
```

Figure 5: One piece Combined .txt File



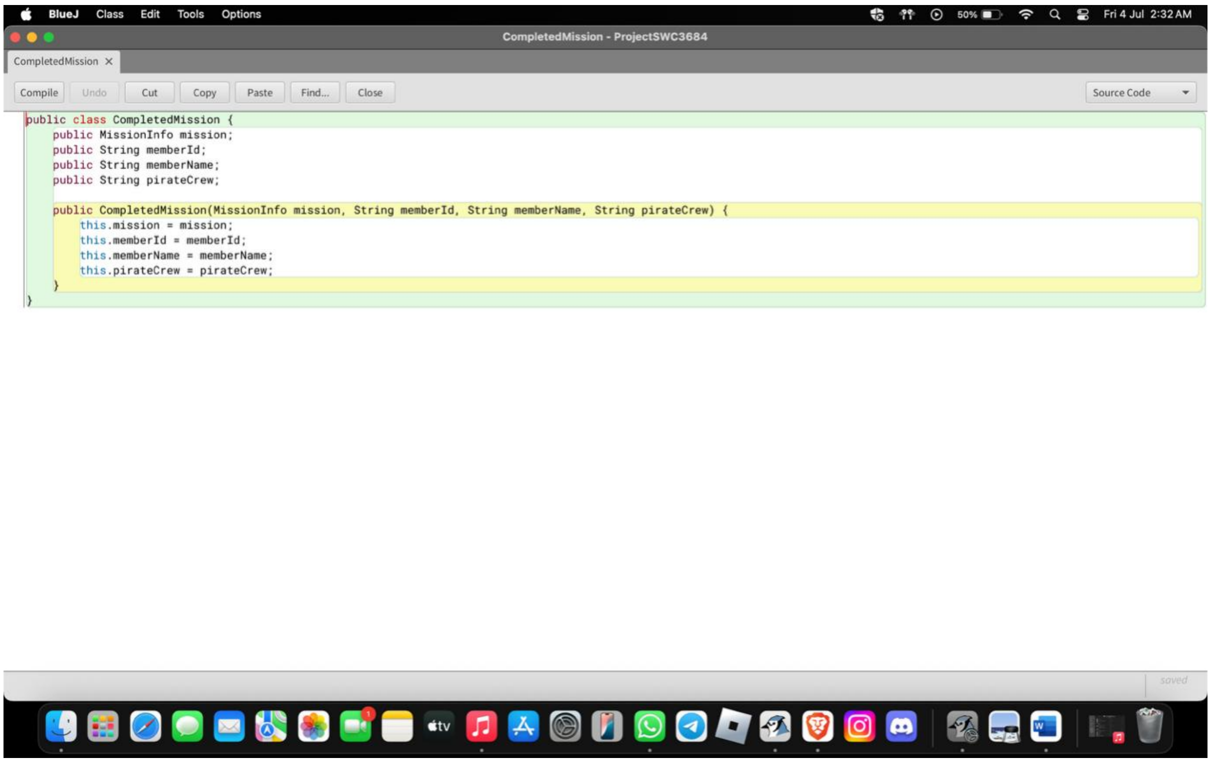
```
CM014|Kid|Kid Pirates|MIS033|Island Defense|4|2025-07-20|2 days|7000
CM015|Killer|Kid Pirates|MIS050|Silent Raid|4|2025-07-26|2 days|6900
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS051|Gear Training|4|2025-07-27|2 days|7200
CM002|Roronoa Zoro|Straw Hat Pirates|MIS054|Sword Recovery|3|2025-07-27|1 day|5200
CM003|Nami|Straw Hat Pirates|MIS057|Treasure Weather Map|2|2025-07-27|2 days|3100
CM004|Sanji|Straw Hat Pirates|MIS059|VIP Cooking|4|2025-07-27|2 days|3700
CM005|Usopp|Straw Hat Pirates|MIS062|Sniper Range Test|3|2025-07-27|1 day|2500
CM006|Chopper|Straw Hat Pirates|MIS065|Doctor Exchange|2|2025-07-27|1 day|2100
CM007|Robin|Straw Hat Pirates|MIS068|Ancient Tomb Visit|4|2025-07-27|2 days|4000
CM008|Franky|Straw Hat Pirates|MIS070|Cannon Upgrade|4|2025-07-27|2 days|3900
CM009|Brook|Straw Hat Pirates|MIS072|Haunted Forest Song|2|2025-07-27|1 day|2200
CM010|Jinbe|Straw Hat Pirates|MIS074|Deep Sea Rescue|5|2025-07-27|2 days|9400
CM011|Shanks|Red Hair Pirates|MIS076|Peace Treaty|5|2025-07-27|1 day|9500
CM012|Buggy|Buggy Pirates|MIS078|Clown Parade|2|2025-07-27|1 day|2000
CM013|Law|Heart Pirates|MIS080|Surgical Save|5|2025-07-27|2 days|9200
CM014|Kid|Kid Pirates|MIS082|Rebel Suppression|4|2025-07-27|2 days|7400
CM015|Killer|Kid Pirates|MIS085|Whisper Raid|3|2025-07-27|1 day|6100
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS052|Demon Island Entry|5|2025-07-28|1 day|9300
CM002|Roronoa Zoro|Straw Hat Pirates|MIS055|Duel with Hunter|4|2025-07-28|1 day|6700
CM003|Nami|Straw Hat Pirates|MIS058|Typhoon Prediction|3|2025-07-28|1 day|2900
CM004|Sanji|Straw Hat Pirates|MIS060|Island Chef Duel|3|2025-07-28|1 day|3300
CM005|Usopp|Straw Hat Pirates|MIS063|Mimicry Escape|2|2025-07-28|2 days|2700
CM006|Chopper|Straw Hat Pirates|MIS066|Disease Research|3|2025-07-28|2 days|2300
CM007|Robin|Straw Hat Pirates|MIS069|Script Translation|3|2025-07-28|1 day|3600
CM008|Franky|Straw Hat Pirates|MIS071|Ship Armor Test|3|2025-07-28|1 day|3700
CM009|Brook|Straw Hat Pirates|MIS073|Ghost Walk|2|2025-07-28|1 day|2100
CM010|Jinbe|Straw Hat Pirates|MIS075|Sea Monster Battle|4|2025-07-28|2 days|8800
CM011|Shanks|Red Hair Pirates|MIS077|World Government Warning|5|2025-07-28|1 day|10000
CM012|Buggy|Buggy Pirates|MIS079|Fake Bounty Collection|3|2025-07-28|1 day|2600
CM013|Law|Heart Pirates|MIS081|Border Healing|4|2025-07-28|2 days|8700
CM014|Kid|Kid Pirates|MIS083|Metal Control Training|3|2025-07-28|1 day|6800
CM015|Killer|Kid Pirates|MIS086|Silent Hunt|4|2025-07-28|2 days|6900
CM001|Monkey D. Luffy|Straw Hat Pirates|MIS088|Last Stand|5|2025-07-30|1 day|9900
CM002|Roronoa Zoro|Straw Hat Pirates|MIS089|Master Slash|5|2025-07-30|1 day|9800
CM003|Nami|Straw Hat Pirates|MIS090|Storm Alert|2|2025-07-30|1 day|2600
CM004|Sanji|Straw Hat Pirates|MIS091|Battle Cook-Off|3|2025-07-30|2 days|3600
CM005|Usopp|Straw Hat Pirates|MIS092|Escape Engineering|3|2025-07-30|1 day|2800
CM006|Chopper|Straw Hat Pirates|MIS093|Emergency Cure|3|2025-07-30|1 day|2700
CM007|Robin|Straw Hat Pirates|MIS094|Cipher Break|4|2025-07-30|1 day|3900
CM008|Franky|Straw Hat Pirates|MIS095|Bridge Build|3|2025-07-30|2 days|3500
CM009|Brook|Straw Hat Pirates|MIS096|Echo Attack|2|2025-07-30|1 day|2400
CM010|Jinbe|Straw Hat Pirates|MIS097|Undersea Defense|4|2025-07-30|2 days|8700
CM011|Shanks|Red Hair Pirates|MIS098|Red Force Rally|5|2025-07-30|1 day|9900
CM012|Buggy|Buggy Pirates|MIS099|Laugh Ambush|3|2025-07-30|1 day|2500
CM013|Law|Heart Pirates|MIS100|Survival Surgery|5|2025-07-30|2 days|9700
CM021|Wukong|Monkey King|MIS102|Journey To Middle East|3|2025-05-21|103 days|10000
```

Figure 6: One piece Combined .txt File

The file onepiece_combined.txt contains structured data representing crew members and their assigned missions in a pirate-themed context, likely inspired by the One-Piece universe. Each line in the file follows a consistent format, with fields separated by the pipe symbol (|). These fields include the crew member's ID, name, pirate crew name, mission ID, mission title, mission date, expected duration, and bounty reward. For example, an entry like CM004|Sanji|Straw Hat Pirates|MIS004|Cooking Battle|2025-07-18|2 days|3000 indicates that crew member Sanji from the Straw Hat Pirates participated in a mission called "Cooking Battle" on July 18, 2025, which lasted 2 days and offered a bounty of 3000.

This file is likely used in a Java application to populate CrewMemberInfo and MissionInfo objects. Each crew member can be represented as an object containing a list of their missions, while each mission is stored with its relevant details. To process the data programmatically, each line would be read and split using the split("\\|") method in Java. A collection such as a HashMap can then be used to store crew members, using their ID as the key. If a crew member already exists in the map, the mission is added to their list; otherwise, a new crew member object is created.

4.4 Completed Mission Class



```java
public class CompletedMission {
    public MissionInfo mission;
    public String memberId;
    public String memberName;
    public String pirateCrew;

    public CompletedMission(MissionInfo mission, String memberId, String memberName, String pirateCrew) {
        this.mission = mission;
        this.memberId = memberId;
        this.memberName = memberName;
        this.pirateCrew = pirateCrew;
    }
}
```

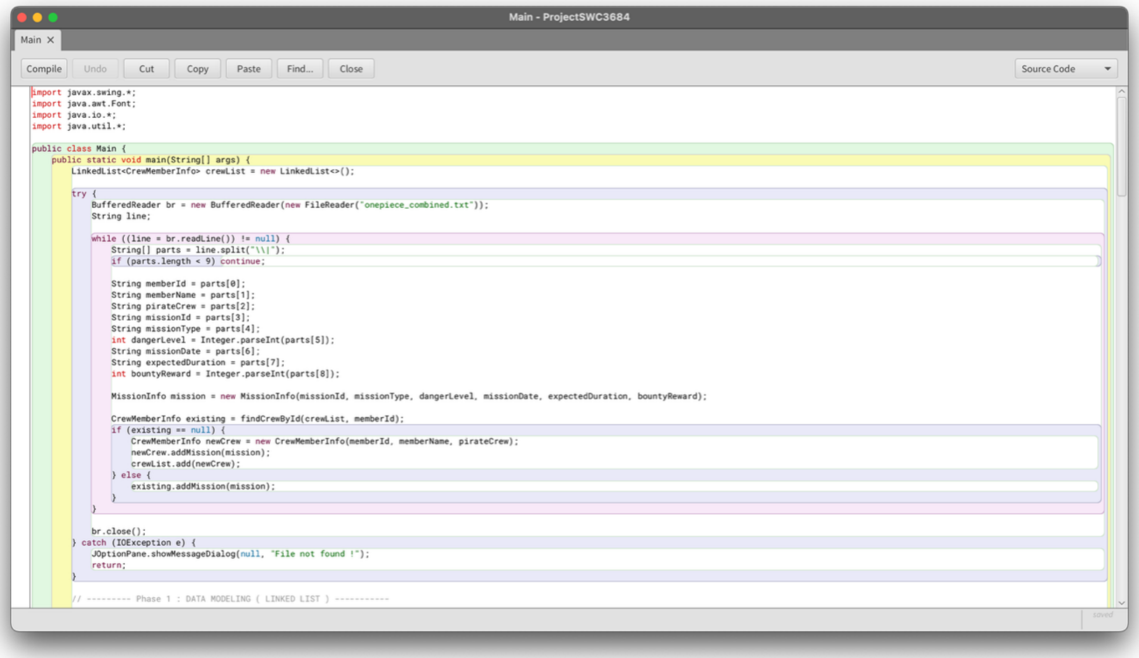
Figure 7: Completed Mission Class

The CompletedMission class shown in the image is a simple Java class designed to represent the completion of a mission by a specific crew member. It contains four public attributes: a MissionInfo object named mission, and three String fields (memberId, memberName, and pirateCrew). These fields collectively store the details of the completed mission and identify the crew member who completed it, along with their pirate crew affiliation.

The class includes one constructor, which takes a MissionInfo object and the three strings as parameters. The constructor initializes all the fields using the this keyword, ensuring that a CompletedMission object is created with complete and accurate information. This class serves as a bridge between mission details and crew member identity, making it useful for generating mission logs, reports, or summaries of completed tasks.

Although functional, the class could be improved by making the fields private to follow proper encapsulation practices. Additionally, getter methods could be added for external access, and a toString() method would make it easier to print or display the object content in a readable format. Overall, this class provides a clean and efficient way to represent mission completions in a pirate-themed mission management system.

4.5 Main Class



```java
import javax.swing.*;
import java.awt.Font;
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        LinkedList<CrewMemberInfo> crewList = new LinkedList<>();

        try {
            BufferedReader br = new BufferedReader(new FileReader("onepiece_combined.txt"));
            String line;

            while ((line = br.readLine()) != null) {
                String[] parts = line.split("\\|");
                if (parts.length < 9) continue;

                String memberId = parts[0];
                String memberName = parts[1];
                String pirateCrew = parts[2];
                String missionId = parts[3];
                String missionType = parts[4];
                int dangerLevel = Integer.parseInt(parts[5]);
                String missionDate = parts[6];
                String expectedDuration = parts[7];
                int bountyReward = Integer.parseInt(parts[8]);

                MissionInfo mission = new MissionInfo(missionId, missionType, dangerLevel, missionDate, expectedDuration, bountyReward);

                CrewMemberInfo existing = findCrewById(crewList, memberId);
                if (existing == null) {
                    CrewMemberInfo newCrew = new CrewMemberInfo(memberId, memberName, pirateCrew);
                    newCrew.addMission(mission);
                    crewList.add(newCrew);
                } else {
                    existing.addMission(mission);
                }
            }

            br.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "File not found !");
            return;
        }

        // --------- Phase 1 : DATA MODELING ( LINKED LIST ) ----------
```

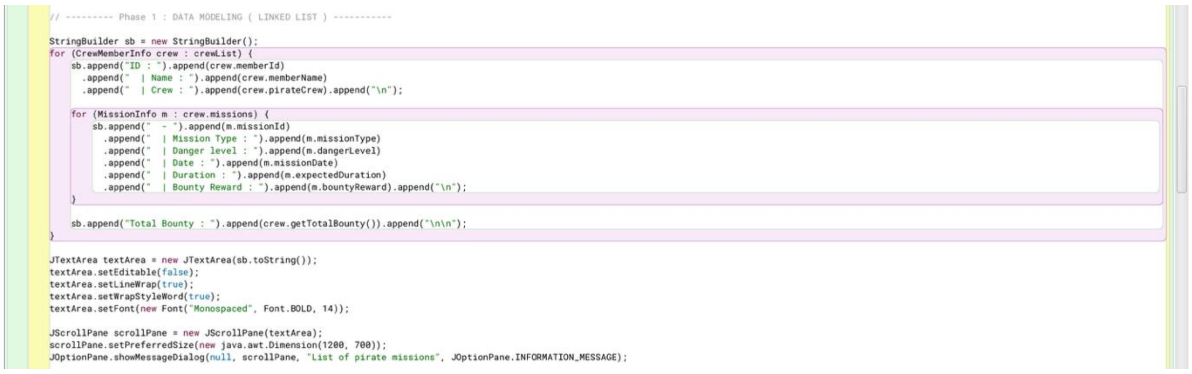
Figure 8: Main class

The Java code in the image is the main part of a program that reads crew member and mission data from the file onepiece_combined.txt and stores it using a LinkedList of CrewMemberInfo objects. The main method begins by initializing an empty LinkedList called crewList to hold all the crew members. It then attempts to read the data file using a BufferedReader. For each line read from the file, the code splits the line into parts using the | delimiter. If the line does not contain exactly 9 parts, it is skipped to prevent errors from malformed entries.

For lines with valid data, the individual parts are extracted into variables, such as memberId, missionId, dangerLevel, and so on. These values are then used to create a MissionInfo object. Before adding the mission to a crew member, the program checks whether the crew member already exists in the list by calling a method named findCrewById (presumably defined elsewhere). If the crew member is not found, a new CrewMemberInfo object is created and the mission is added to them. If the member already exists, the mission is simply added to their existing mission list.

This code effectively populates the data model by linking each crew member to their missions. It ensures no duplicate crew entries are made and keeps their missions organized. It also includes exception handling: if the file cannot be read, a dialog message will be shown to inform the user. Overall, this part of the code is a solid implementation for reading structured data into objects and preparing it for further use, such as calculations, filtering, or GUI display.

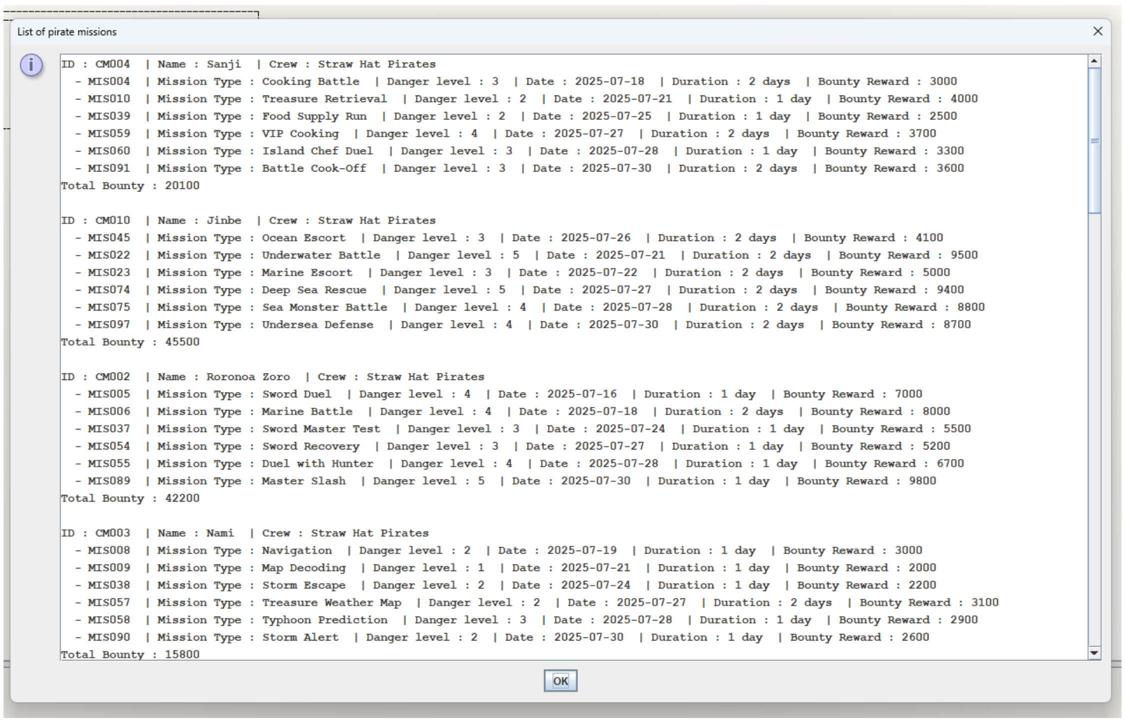
4.5.1 Phase 1 (Linked List)



```java
// --------- Phase 1 : DATA MODELING ( LINKED LIST ) -----------

StringBuilder sb = new StringBuilder();
for (CrewMemberInfo crew : crewList) {
    sb.append("ID : ").append(crew.memberId)
      .append("  | Name : ").append(crew.memberName)
      .append("  | Crew : ").append(crew.pirateCrew).append("\n");

    for (MissionInfo m : crew.missions) {
        sb.append("  - ").append(m.missionId)
          .append("  | Mission Type : ").append(m.missionType)
          .append("  | Danger level : ").append(m.dangerLevel)
          .append("  | Date : ").append(m.missionDate)
          .append("  | Duration : ").append(m.expectedDuration)
          .append("  | Bounty Reward : ").append(m.bountyReward).append("\n");
    }

    sb.append("Total Bounty : ").append(crew.getTotalBounty()).append("\n\n");
}

JTextArea textArea = new JTextArea(sb.toString());
textArea.setEditable(false);
textArea.setLineWrap(true);
textArea.setWrapStyleWord(true);
textArea.setFont(new Font("Monospaced", Font.BOLD, 14));

JScrollPane scrollPane = new JScrollPane(textArea);
scrollPane.setPreferredSize(new java.awt.Dimension(1200, 700));
JOptionPane.showMessageDialog(null, scrollPane, "List of pirate missions", JOptionPane.INFORMATION_MESSAGE);
```

Figure 9: Main class Phase 1 (Linked List)

This Java code creates a formatted list of pirate crew members and their missions, then displays it in a scrollable dialog box. It uses a StringBuilder to combine each crew member's ID, name, crew, and the details of their missions such as mission type, danger level, date, duration, and bounty reward. It also shows the total bounty collected by each crew member. The information is placed into a JTextArea, styled for readability, made scrollable with a JScrollPane, and then shown to the user using a pop-up dialog (JOptionPane).



```
ID : CM004  | Name : Sanji  | Crew : Straw Hat Pirates
  - MIS004  | Mission Type : Cooking Battle  | Danger level : 3  | Date : 2025-07-18  | Duration : 2 days  | Bounty Reward : 3000
  - MIS010  | Mission Type : Treasure Retrieval  | Danger level : 2  | Date : 2025-07-21  | Duration : 1 day  | Bounty Reward : 4000
  - MIS039  | Mission Type : Food Supply Run  | Danger level : 2  | Date : 2025-07-25  | Duration : 1 day  | Bounty Reward : 2500
  - MIS059  | Mission Type : VIP Cooking  | Danger level : 4  | Date : 2025-07-27  | Duration : 2 days  | Bounty Reward : 3700
  - MIS060  | Mission Type : Island Chef Duel  | Danger level : 3  | Date : 2025-07-28  | Duration : 1 day  | Bounty Reward : 3300
  - MIS091  | Mission Type : Battle Cook-Off  | Danger level : 3  | Date : 2025-07-30  | Duration : 2 days  | Bounty Reward : 3600
Total Bounty : 20100

ID : CM010  | Name : Jinbe  | Crew : Straw Hat Pirates
  - MIS045  | Mission Type : Ocean Escort  | Danger level : 3  | Date : 2025-07-26  | Duration : 2 days  | Bounty Reward : 4100
  - MIS022  | Mission Type : Underwater Battle  | Danger level : 5  | Date : 2025-07-21  | Duration : 2 days  | Bounty Reward : 9500
  - MIS023  | Mission Type : Marine Escort  | Danger level : 3  | Date : 2025-07-22  | Duration : 2 days  | Bounty Reward : 5000
  - MIS074  | Mission Type : Deep Sea Rescue  | Danger level : 5  | Date : 2025-07-27  | Duration : 2 days  | Bounty Reward : 9400
  - MIS075  | Mission Type : Sea Monster Battle  | Danger level : 4  | Date : 2025-07-28  | Duration : 2 days  | Bounty Reward : 8800
  - MIS097  | Mission Type : Undersea Defense  | Danger level : 4  | Date : 2025-07-30  | Duration : 2 days  | Bounty Reward : 8700
Total Bounty : 45500

ID : CM002  | Name : Roronoa Zoro  | Crew : Straw Hat Pirates
  - MIS005  | Mission Type : Sword Duel  | Danger level : 4  | Date : 2025-07-16  | Duration : 1 day  | Bounty Reward : 7000
  - MIS006  | Mission Type : Marine Battle  | Danger level : 4  | Date : 2025-07-18  | Duration : 2 days  | Bounty Reward : 8000
  - MIS037  | Mission Type : Sword Master Test  | Danger level : 3  | Date : 2025-07-24  | Duration : 1 day  | Bounty Reward : 5500
  - MIS054  | Mission Type : Sword Recovery  | Danger level : 3  | Date : 2025-07-27  | Duration : 1 day  | Bounty Reward : 5200
  - MIS055  | Mission Type : Duel with Hunter  | Danger level : 4  | Date : 2025-07-28  | Duration : 1 day  | Bounty Reward : 6700
  - MIS089  | Mission Type : Master Slash  | Danger level : 5  | Date : 2025-07-30  | Duration : 1 day  | Bounty Reward : 9800
Total Bounty : 42200

ID : CM003  | Name : Nami  | Crew : Straw Hat Pirates
  - MIS008  | Mission Type : Navigation  | Danger level : 2  | Date : 2025-07-19  | Duration : 1 day  | Bounty Reward : 3000
  - MIS009  | Mission Type : Map Decoding  | Danger level : 1  | Date : 2025-07-21  | Duration : 1 day  | Bounty Reward : 2000
  - MIS038  | Mission Type : Storm Escape  | Danger level : 2  | Date : 2025-07-24  | Duration : 1 day  | Bounty Reward : 2200
  - MIS057  | Mission Type : Treasure Weather Map  | Danger level : 2  | Date : 2025-07-27  | Duration : 2 days  | Bounty Reward : 3100
  - MIS058  | Mission Type : Typhoon Prediction  | Danger level : 3  | Date : 2025-07-28  | Duration : 1 day  | Bounty Reward : 2900
  - MIS090  | Mission Type : Storm Alert  | Danger level : 2  | Date : 2025-07-30  | Duration : 1 day  | Bounty Reward : 2600
Total Bounty : 15800
```

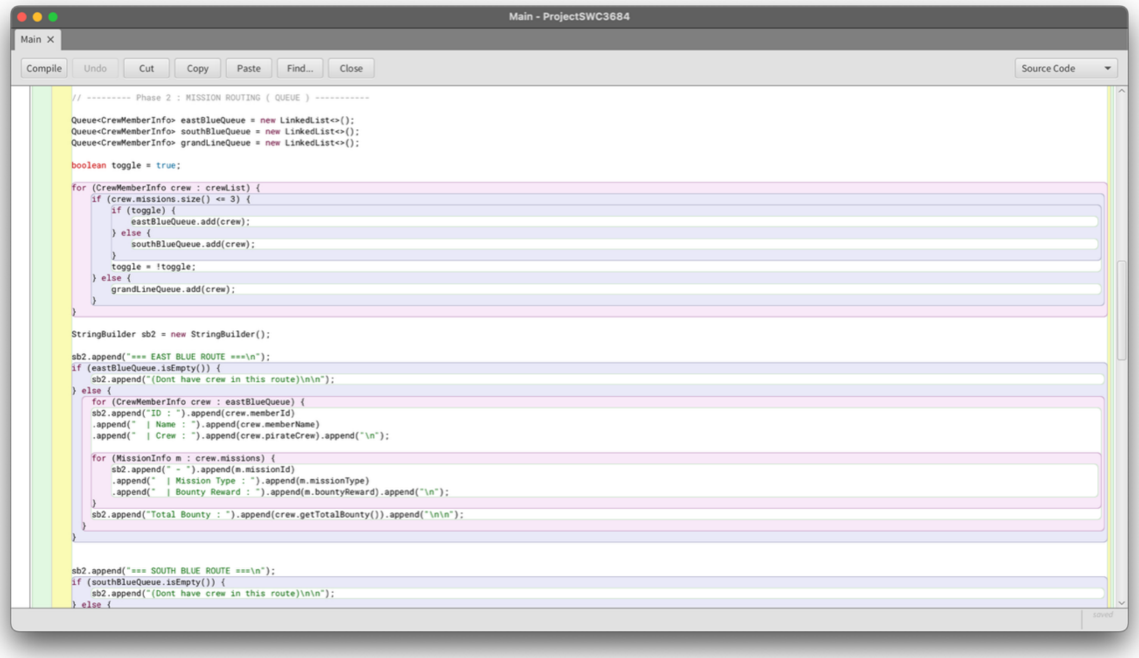
Figure 10: Output Main class Phase 1 (Linked List)

The display above presents a complete list of missions assigned to several pirate crew members from the Straw Hat Pirates. This output is shown using a JOptionPane message dialog as part of the Java GUI.

Each pirate is shown with their ID, name, and crew name. Below each pirate, all assigned missions are listed in detail, including the mission ID, mission type, danger level, mission date, duration, and bounty reward.

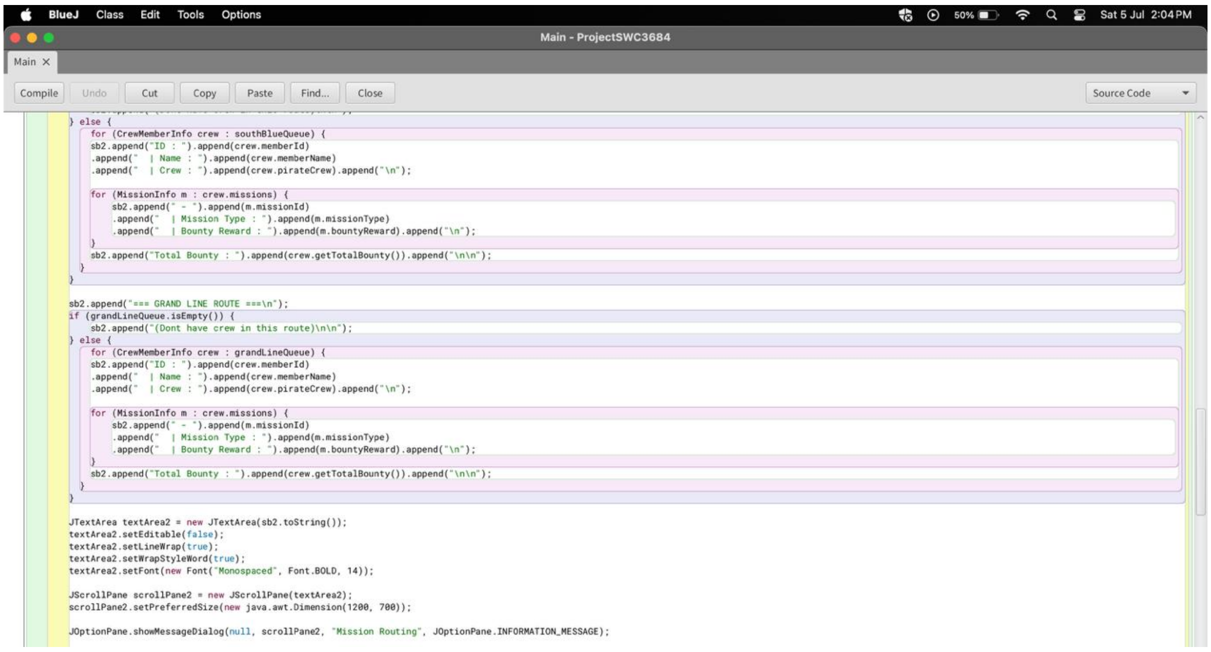
This output demonstrates that the system has successfully loaded and grouped mission data for each crew member from the input file onepiece_combined.txt. It also acts as a verification step before the missions are routed into the corresponding queues (East Blue, South Blue, or Grand Line) based on their count, using a Queue data structure.

4.5.2 Phase 2 (Queue)



```java
// --------- Phase 2 : MISSION ROUTING ( QUEUE ) -----------

Queue<CrewMemberInfo> eastBlueQueue = new LinkedList<>();
Queue<CrewMemberInfo> southBlueQueue = new LinkedList<>();
Queue<CrewMemberInfo> grandLineQueue = new LinkedList<>();

boolean toggle = true;

for (CrewMemberInfo crew : crewList) {
    if (crew.missions.size() <= 3) {
        if (toggle) {
            eastBlueQueue.add(crew);
        } else {
            southBlueQueue.add(crew);
        }
        toggle = !toggle;
    } else {
        grandLineQueue.add(crew);
    }
}

StringBuilder sb2 = new StringBuilder();

sb2.append("=== EAST BLUE ROUTE ===\n");
if (eastBlueQueue.isEmpty()) {
    sb2.append("(Dont have crew in this route)\n\n");
} else {
    for (CrewMemberInfo crew : eastBlueQueue) {
    sb2.append("ID : ").append(crew.memberId)
    .append("  | Name : ").append(crew.memberName)
    .append("  | Crew : ").append(crew.pirateCrew).append("\n");

    for (MissionInfo m : crew.missions) {
        sb2.append(" - ").append(m.missionId)
        .append("  | Mission Type : ").append(m.missionType)
        .append("  | Bounty Reward : ").append(m.bountyReward).append("\n");
    }
    sb2.append("Total Bounty : ").append(crew.getTotalBounty()).append("\n\n");
    }
}


sb2.append("=== SOUTH BLUE ROUTE ===\n");
if (southBlueQueue.isEmpty()) {
    sb2.append("(Dont have crew in this route)\n\n");
} else {
```

Figure 11: Main class Phase 2 (Queue)



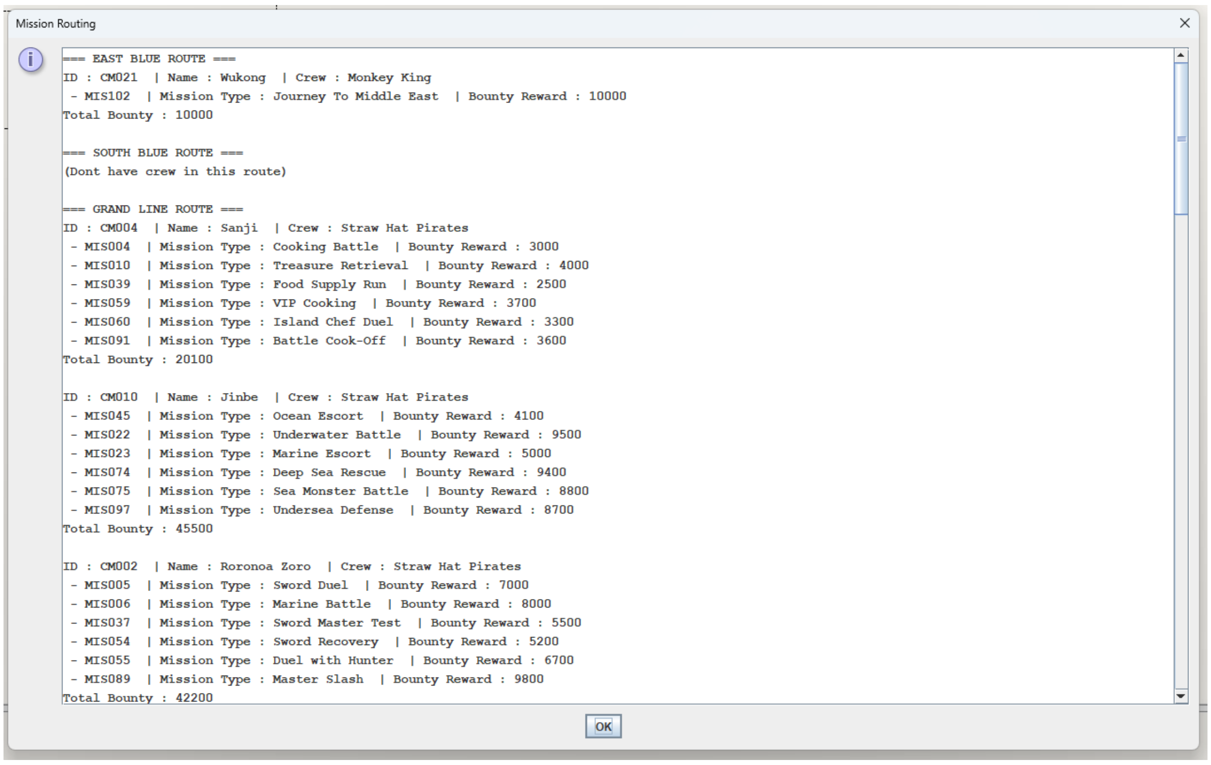
```java
} else {
    for (CrewMemberInfo crew : southBlueQueue) {
    sb2.append("ID : ").append(crew.memberId)
    .append("  | Name : ").append(crew.memberName)
    .append("  | Crew : ").append(crew.pirateCrew).append("\n");

    for (MissionInfo m : crew.missions) {
        sb2.append(" - ").append(m.missionId)
        .append("  | Mission Type : ").append(m.missionType)
        .append("  | Bounty Reward : ").append(m.bountyReward).append("\n");
    }
    sb2.append("Total Bounty : ").append(crew.getTotalBounty()).append("\n\n");
    }
}

sb2.append("=== GRAND LINE ROUTE ===\n");
if (grandLineQueue.isEmpty()) {
    sb2.append("(Dont have crew in this route)\n\n");
} else {
    for (CrewMemberInfo crew : grandLineQueue) {
    sb2.append("ID : ").append(crew.memberId)
    .append("  | Name : ").append(crew.memberName)
    .append("  | Crew : ").append(crew.pirateCrew).append("\n");

    for (MissionInfo m : crew.missions) {
        sb2.append(" - ").append(m.missionId)
        .append("  | Mission Type : ").append(m.missionType)
        .append("  | Bounty Reward : ").append(m.bountyReward).append("\n");
    }
    sb2.append("Total Bounty : ").append(crew.getTotalBounty()).append("\n\n");
    }
}

JTextArea textArea2 = new JTextArea(sb2.toString());
textArea2.setEditable(false);
textArea2.setLineWrap(true);
textArea2.setWrapStyleWord(true);
textArea2.setFont(new Font("Monospaced", Font.BOLD, 14));

JScrollPane scrollPane2 = new JScrollPane(textArea2);
scrollPane2.setPreferredSize(new java.awt.Dimension(1200, 700));

JOptionPane.showMessageDialog(null, scrollPane2, "Mission Routing", JOptionPane.INFORMATION_MESSAGE);
```

Figure 12: Main class Phase 2 (Queue)

This Java code manages the routing of pirate crew members into three mission paths: East Blue, South Blue, and Grand Line, based on the number of missions assigned to each member. Crew members with three or fewer missions are assigned alternately to East or South Blue routes, while those with more than three missions are directed to the Grand Line route.

Once the routing is completed, the program compiles a detailed summary for each route. If a route has no members, it displays a message indicating so. Otherwise, it lists each member’s ID, name, crew, and mission details including mission ID, type, and bounty reward. It also calculates and displays the total bounty for each crew member.

The final output is shown in a scrollable JOptionPane dialog using a monospaced font for better alignment. This helps visualize the distribution and mission details of each pirate in a clear, structured format.



```
=== EAST BLUE ROUTE ===
ID : CM021   | Name : Wukong   | Crew : Monkey King
 - MIS102   | Mission Type : Journey To Middle East   | Bounty Reward : 10000
Total Bounty : 10000

=== SOUTH BLUE ROUTE ===
(Dont have crew in this route)

=== GRAND LINE ROUTE ===
ID : CM004   | Name : Sanji   | Crew : Straw Hat Pirates
 - MIS004   | Mission Type : Cooking Battle   | Bounty Reward : 3000
 - MIS010   | Mission Type : Treasure Retrieval   | Bounty Reward : 4000
 - MIS039   | Mission Type : Food Supply Run   | Bounty Reward : 2500
 - MIS059   | Mission Type : VIP Cooking   | Bounty Reward : 3700
 - MIS060   | Mission Type : Island Chef Duel   | Bounty Reward : 3300
 - MIS091   | Mission Type : Battle Cook-Off   | Bounty Reward : 3600
Total Bounty : 20100

ID : CM010   | Name : Jinbe   | Crew : Straw Hat Pirates
 - MIS045   | Mission Type : Ocean Escort   | Bounty Reward : 4100
 - MIS022   | Mission Type : Underwater Battle   | Bounty Reward : 9500
 - MIS023   | Mission Type : Marine Escort   | Bounty Reward : 5000
 - MIS074   | Mission Type : Deep Sea Rescue   | Bounty Reward : 9400
 - MIS075   | Mission Type : Sea Monster Battle   | Bounty Reward : 8800
 - MIS097   | Mission Type : Undersea Defense   | Bounty Reward : 8700
Total Bounty : 45500

ID : CM002   | Name : Roronoa Zoro   | Crew : Straw Hat Pirates
 - MIS005   | Mission Type : Sword Duel   | Bounty Reward : 7000
 - MIS006   | Mission Type : Marine Battle   | Bounty Reward : 8000
 - MIS037   | Mission Type : Sword Master Test   | Bounty Reward : 5500
 - MIS054   | Mission Type : Sword Recovery   | Bounty Reward : 5200
 - MIS055   | Mission Type : Duel with Hunter   | Bounty Reward : 6700
 - MIS089   | Mission Type : Master Slash   | Bounty Reward : 9800
Total Bounty : 42200
```

Figure 13: Output Main class Phase 2 (Queue)

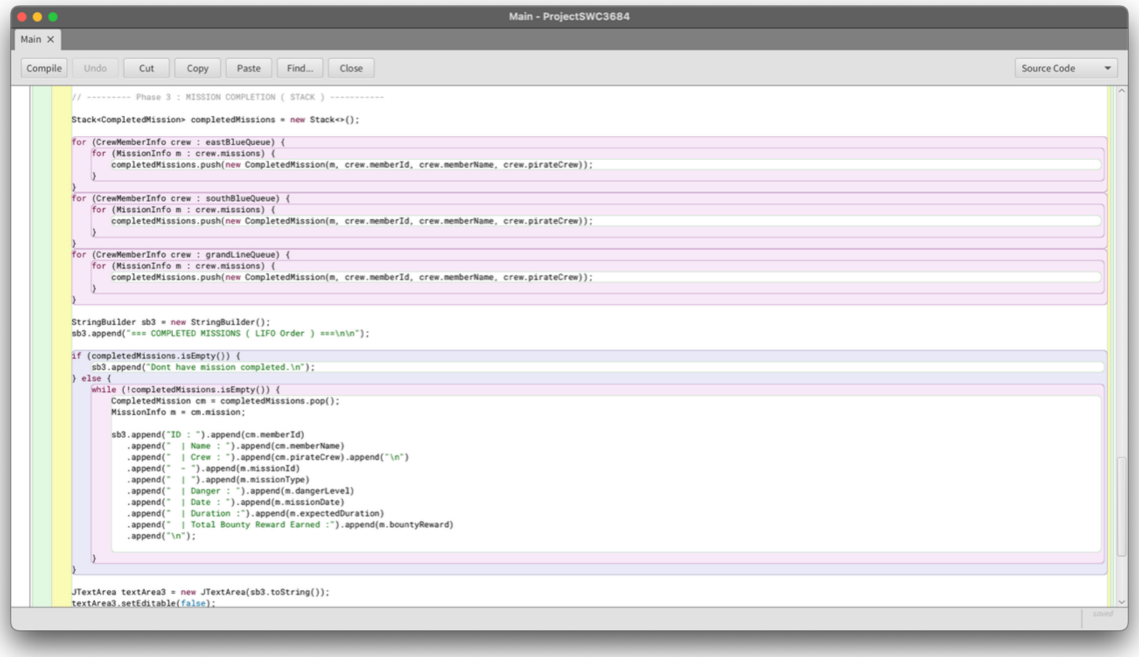
This display shows the result of routing pirate crew members into three mission routes: East Blue, South Blue, and Grand Line, based on the number of missions assigned to each crew member.

Crew members with three or fewer missions are distributed alternately between the East Blue and South Blue routes. Those with more than three missions are directly assigned to the Grand Line route.

In this output, only one crew member, Wukong, is placed under East Blue, while South Blue currently has no crew assigned. The Grand Line route contains multiple members, including Sanji, Jinbe, and Roronoa Zoro, who have been assigned a larger number of missions. For each crew member, the system displays their ID, name, crew name, a detailed list of their missions (including mission ID, type, and bounty reward), and the total bounty accumulated.

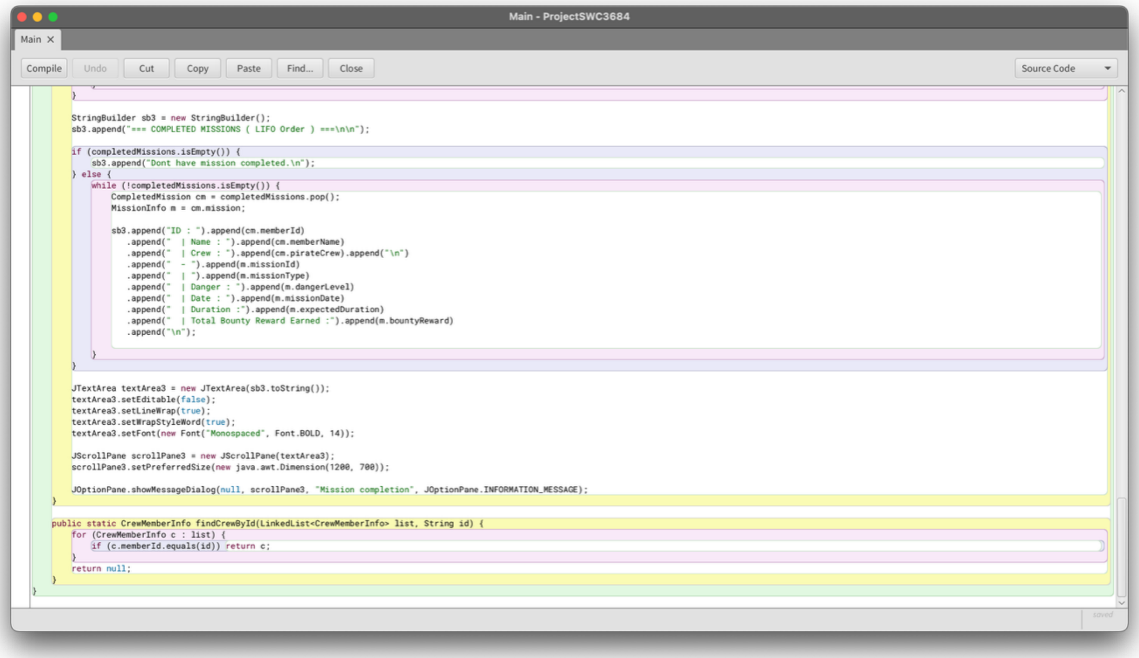
All of this information is displayed in a scrollable JOptionPane window using a monospaced font for neat alignment and improved readability.

4.5.3 Phase 3 (Stack)



```
// --------- Phase 3 : MISSION COMPLETION ( STACK ) -----------

Stack<CompletedMission> completedMissions = new Stack<>();

for (CrewMemberInfo crew : eastBlueQueue) {
    for (MissionInfo m : crew.missions) {
        completedMissions.push(new CompletedMission(m, crew.memberId, crew.memberName, crew.pirateCrew));
    }
}
for (CrewMemberInfo crew : southBlueQueue) {
    for (MissionInfo m : crew.missions) {
        completedMissions.push(new CompletedMission(m, crew.memberId, crew.memberName, crew.pirateCrew));
    }
}
for (CrewMemberInfo crew : grandLineQueue) {
    for (MissionInfo m : crew.missions) {
        completedMissions.push(new CompletedMission(m, crew.memberId, crew.memberName, crew.pirateCrew));
    }
}

StringBuilder sb3 = new StringBuilder();
sb3.append("=== COMPLETED MISSIONS ( LIFO Order ) ===\n\n");

if (completedMissions.isEmpty()) {
    sb3.append("Dont have mission completed.\n");
} else {
    while (!completedMissions.isEmpty()) {
        CompletedMission cm = completedMissions.pop();
        MissionInfo m = cm.mission;

        sb3.append("ID : ").append(cm.memberId)
           .append("  | Name : ").append(cm.memberName)
           .append("  | Crew : ").append(cm.pirateCrew).append("\n")
           .append("  - ").append(m.missionId)
           .append("  | ").append(m.missionType)
           .append("  | Danger : ").append(m.dangerLevel)
           .append("  | Date : ").append(m.missionDate)
           .append("  | Duration :").append(m.expectedDuration)
           .append("  | Total Bounty Reward Earned :").append(m.bountyReward)
           .append("\n");

    }
}

JTextArea textArea3 = new JTextArea(sb3.toString());
textArea3.setEditable(false);
```

Figure 14: Main class Phase 3 (stack)



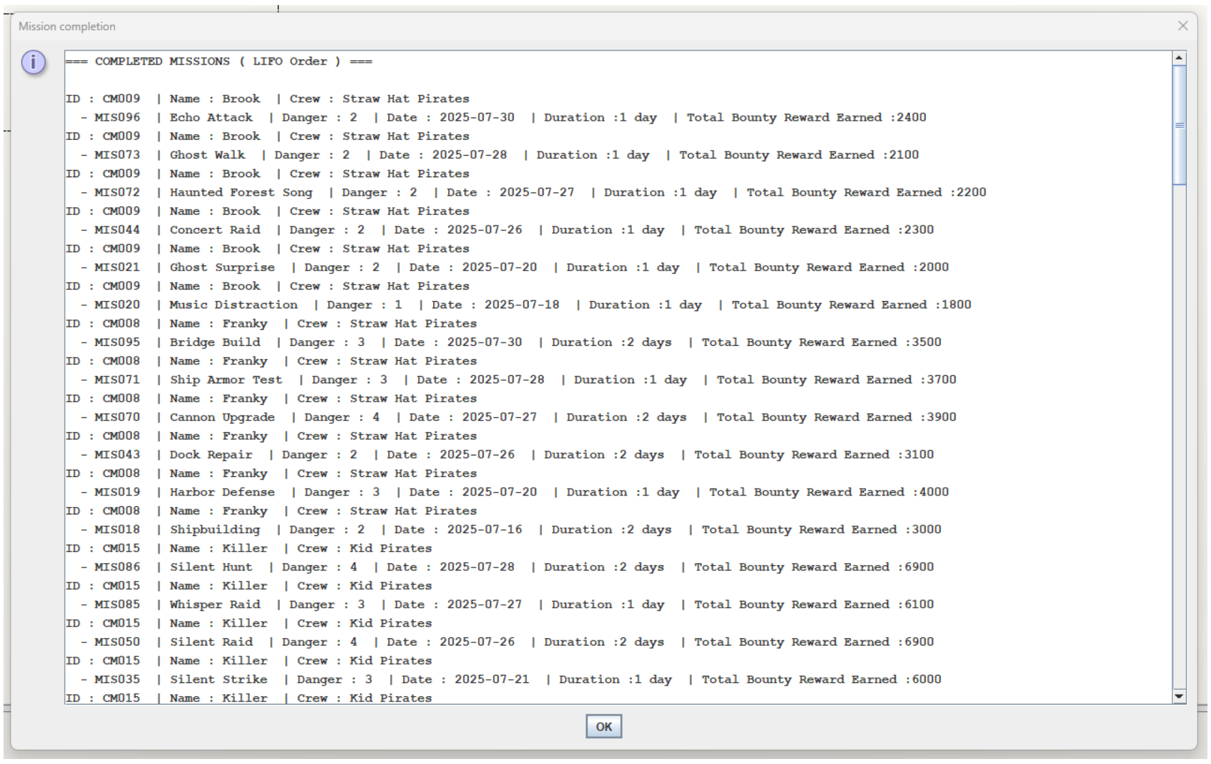
```
        }

        StringBuilder sb3 = new StringBuilder();
        sb3.append("=== COMPLETED MISSIONS ( LIFO Order ) ===\n\n");

        if (completedMissions.isEmpty()) {
            sb3.append("Dont have mission completed.\n");
        } else {
            while (!completedMissions.isEmpty()) {
                CompletedMission cm = completedMissions.pop();
                MissionInfo m = cm.mission;

                sb3.append("ID : ").append(cm.memberId)
                   .append("  | Name : ").append(cm.memberName)
                   .append("  | Crew : ").append(cm.pirateCrew).append("\n")
                   .append("  - ").append(m.missionId)
                   .append("  | ").append(m.missionType)
                   .append("  | Danger : ").append(m.dangerLevel)
                   .append("  | Date : ").append(m.missionDate)
                   .append("  | Duration :").append(m.expectedDuration)
                   .append("  | Total Bounty Reward Earned :").append(m.bountyReward)
                   .append("\n");

            }
        }

        JTextArea textArea3 = new JTextArea(sb3.toString());
        textArea3.setEditable(false);
        textArea3.setLineWrap(true);
        textArea3.setWrapStyleWord(true);
        textArea3.setFont(new Font("Monospaced", Font.BOLD, 14));

        JScrollPane scrollPane3 = new JScrollPane(textArea3);
        scrollPane3.setPreferredSize(new java.awt.Dimension(1200, 700));

        JOptionPane.showMessageDialog(null, scrollPane3, "Mission completion", JOptionPane.INFORMATION_MESSAGE);
    }

    public static CrewMemberInfo findCrewById(LinkedList<CrewMemberInfo> list, String id) {
        for (CrewMemberInfo c : list) {
            if (c.memberId.equals(id)) return c;
        }
        return null;
    }
}
```

Figure 15: Main class Phase 3 (stack)

This portion of the Java code belongs to Phase 3: Mission Completion and it manages how completed pirate missions are processed and displayed using a stack-based structure. The program uses a Stack called completedMissions to store mission details from all three routes: East Blue, South Blue, and Grand Line. It loops through each crew member in these queues and pushes their missions onto the stack. This stack follows the LIFO (Last-In-First-Out) principle, meaning the most recently added missions will appear first when displayed.

After populating the stack, a StringBuilder (sb3) is used to format the output. If the stack is empty, it simply shows a message saying no missions have been completed. If the stack contains data, it enters a loop where it pops each mission off the stack one by one. For each popped mission, it extracts both the mission and crew details including the crew member's ID, name, pirate crew, and all mission specifics like mission type, date, danger level, duration, and bounty reward. Each of these is formatted into a readable multi-line entry.

Finally, the completed mission report is shown in a pop-up window using a JTextArea placed in a JScrollPane, similar to the previous phases. This provides users with a scrollable, clear summary of completed missions with the latest missions shown at the top due to the stack order.



```
=== COMPLETED MISSIONS ( LIFO Order ) ===

ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS096  | Echo Attack  | Danger : 2  | Date : 2025-07-30  | Duration :1 day  | Total Bounty Reward Earned :2400
ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS073  | Ghost Walk  | Danger : 2  | Date : 2025-07-28  | Duration :1 day  | Total Bounty Reward Earned :2100
ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS072  | Haunted Forest Song  | Danger : 2  | Date : 2025-07-27  | Duration :1 day  | Total Bounty Reward Earned :2200
ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS044  | Concert Raid  | Danger : 2  | Date : 2025-07-26  | Duration :1 day  | Total Bounty Reward Earned :2300
ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS021  | Ghost Surprise  | Danger : 2  | Date : 2025-07-20  | Duration :1 day  | Total Bounty Reward Earned :2000
ID : CM009  | Name : Brook  | Crew : Straw Hat Pirates
  - MIS020  | Music Distraction  | Danger : 1  | Date : 2025-07-18  | Duration :1 day  | Total Bounty Reward Earned :1800
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS095  | Bridge Build  | Danger : 3  | Date : 2025-07-30  | Duration :2 days  | Total Bounty Reward Earned :3500
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS071  | Ship Armor Test  | Danger : 3  | Date : 2025-07-28  | Duration :1 day  | Total Bounty Reward Earned :3700
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS070  | Cannon Upgrade  | Danger : 4  | Date : 2025-07-27  | Duration :2 days  | Total Bounty Reward Earned :3900
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS043  | Dock Repair  | Danger : 2  | Date : 2025-07-26  | Duration :2 days  | Total Bounty Reward Earned :3100
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS019  | Harbor Defense  | Danger : 3  | Date : 2025-07-20  | Duration :1 day  | Total Bounty Reward Earned :4000
ID : CM008  | Name : Franky  | Crew : Straw Hat Pirates
  - MIS018  | Shipbuilding  | Danger : 2  | Date : 2025-07-16  | Duration :2 days  | Total Bounty Reward Earned :3000
ID : CM015  | Name : Killer  | Crew : Kid Pirates
  - MIS086  | Silent Hunt  | Danger : 4  | Date : 2025-07-28  | Duration :2 days  | Total Bounty Reward Earned :6900
ID : CM015  | Name : Killer  | Crew : Kid Pirates
  - MIS085  | Whisper Raid  | Danger : 3  | Date : 2025-07-27  | Duration :1 day  | Total Bounty Reward Earned :6100
ID : CM015  | Name : Killer  | Crew : Kid Pirates
  - MIS050  | Silent Raid  | Danger : 4  | Date : 2025-07-26  | Duration :2 days  | Total Bounty Reward Earned :6900
ID : CM015  | Name : Killer  | Crew : Kid Pirates
  - MIS035  | Silent Strike  | Danger : 3  | Date : 2025-07-21  | Duration :1 day  | Total Bounty Reward Earned :6000
ID : CM015  | Name : Killer  | Crew : Kid Pirates
```

Figure 16: Output Main class Phase 3 (Stack)

This display shows the list of crew members who have completed their missions, presented in LIFO (Last-In-First-Out) order using a Stack. Each section includes the crew member’s ID, name, crew, followed by detailed mission information such as mission ID, type, danger level, date, duration, and the total bounty reward earned.

The information is presented in a scrollable JOptionPane using monospaced font, clearly showing the order in which pirates completed their missions starting from the most recent to the earliest.

5. Lessons learned

5.1 Member 1 (Aiman Harith)

Throughout this project, we gained valuable experience in applying key data structures such as LinkedList, Queue, and Stack in the development of a pirate crew mission management system. This hands-on experience allowed us to understand how each data structure serves a specific purpose whether for storing, routing, or processing data and how they work together to create an organized and efficient system.

As the team leader, I was responsible for coordinating task distribution and ensuring that all members completed their parts on time. I also contributed to the development of the mission routing logic using the Queue data structure. This experience helped strengthen my leadership and Java programming skills.

The second team member focused on handling file input by reading and parsing the .txt data into LinkedList objects. From this task, they gained deeper knowledge of file I/O operations and understood the importance of data validation and error handling.

The third member developed the mission completion logic using the Stack structure and designed the output interface using JOptionPane. This helped them understand how the LIFO (Last-In-First-Out) concept works in real applications and enhanced their skills in creating simple yet effective GUI displays.

Lastly, the fourth member was in charge of preparing the documentation and testing the system with sample data. They ensured that the outputs were formatted correctly and that the overall functionality of the system met the project requirements. Through this role, they learned the value of clear documentation and thorough testing in software development.

Overall, this project has not only improved our technical understanding of data structures, but also enhanced our teamwork, problem-solving, and communication skills essential elements for future academic and professional success.

5.2 Member 2 (Hafiy Danish)

Through the development of this Java project, I gained a solid understanding of designing modular classes such as MissionInfo, CrewMemberInfo, and CompletedMission, which helped organize data logically and improve code readability and reusability. By working with LinkedList to manage a dynamic list of crew members and their missions, I learned how to model real-world relationships using composition. I also became more confident in selecting appropriate data structures for different tasks—using a Queue for mission routing (FIFO) and a Stack for mission completion (LIFO)—which simplified logic and improved program efficiency.

In addition, I learned to construct large outputs efficiently using StringBuilder and enhance UI clarity through the use of JTextArea with scrollable panes. Implementing logic such as alternating queue assignment using a toggle flag taught me how to ensure balanced distribution in task handling. The project's modular design also highlighted the importance of scalability and reusability, allowing easy expansion of features.

Working with file input using BufferedReader improved my understanding of reading structured text data, while the use of split() for parsing and try-catch blocks for exception handling gave me confidence in managing file-related errors. I also learned how to validate and update object data without duplication by checking for existing entries before creating new ones. Overall, this project enhanced my skills in data modeling, file processing, object-oriented design, and UI presentation—all essential abilities in real-world Java development.

5.3 Member 3 (Muhammad Syawal)

Throughout the development of this pirate-themed Java project, we deepened our understanding of object-oriented programming, modular design, and the practical use of data structures such as LinkedList, Queue, and Stack. We learned to organize complex data into well-structured classes (MissionInfo, CrewMemberInfo, CompletedMission), which made our program more maintainable and realistic.

We also gained hands-on experience with file handling using BufferedReader, data parsing using the split() method, and implementing input validation. The use of conditional logic helped in fair routing of missions, and applying StringBuilder allowed us to efficiently create structured GUI outputs. Moreover, using JTextArea and JScrollPane improved the user interface, making the data easier to interpret.

One major takeaway was understanding how different data structures are best suited for specific tasks, LinkedList for dynamic storage, Queue for orderly routing, and Stack for managing completed tasks. We also saw how crucial it is to avoid redundancy, handle errors gracefully, and think about scalability. Overall, this project sharpened our problem-solving, coding, and system design skills in a fun and meaningful way.

5.4 Member 4 (Aiman Nor Hakimi)

Through this Java project, I learned how to use important data structures like LinkedList, Queue, and Stack in a real program. Each one had a purpose LinkedList helped store crew and mission data, Queue was used to assign missions fairly, and Stack was used to show mission completion in the correct order. This helped me understand how data structures work and when to use them.

I also improved my skills in reading data from a file using BufferedReader and splitting lines with the split() method. I learned how to handle errors using try-catch blocks to make sure the program runs smoothly even if the file is missing or has wrong data.

Creating separate classes like MissionInfo and CrewMemberInfo taught me how to organize code better. Using JOptionPane and JScrollPane made the output easier to read and more user-friendly. I also learned to use StringBuilder to display large amounts of information without slowing down the program.

Lastly, I understood the importance of checking for duplicates, using simple logic like toggle for fair assignment, and building programs that are clean and easy to update. Overall, this project helped me become more confident in Java and programming in general.

6. References (APA Format)

1.Oracle. (2024). The Java™ Tutorials: Collections. Oracle.

https://docs.oracle.com/javase/tutorial/collections/index.html

2.Liang, Y. D. (2022). Introduction to Java Programming and Data Structures: Comprehensive Version (13th ed.). Pearson.

https://www.pearson.com/en-us/subject-catalog/p/introduction-to-java-programming-and-data-structures-comprehensive-version/P200000006661/9780137455904

3.GeeksforGeeks. (n.d.). Java LinkedList.

https://www.geeksforgeeks.org/linkedlist-in-java/

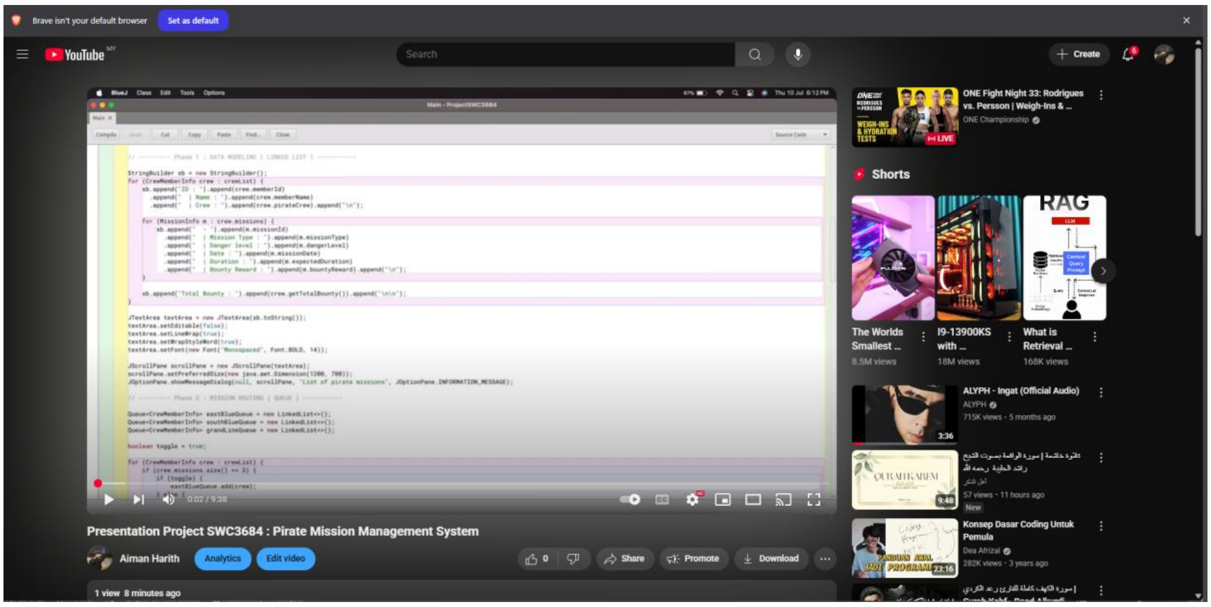
4.Baeldung. (2023). Guide to Java Stack.

https://www.baeldung.com/java-stack

5.Oracle. (2024). How to Use Text Areas (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components).

https://docs.oracle.com/javase/tutorial/uiswing/components/textarea.html

7. Video Presentation Link



```
// ---------- Phase 1 : DATA MODELING ( LINKED LIST ) ------------

StringBuilder sb = new StringBuilder();
for (CrewMemberInfo crew : crewList) {
    sb.append("ID : ").append(crew.memberId)
      .append("  | Name : ").append(crew.memberName)
      .append("  | Crew : ").append(crew.pirateCrew).append("\n");

    for (MissionInfo m : crew.missions) {
        sb.append("  - ").append(m.missionId)
          .append("  | Mission Type : ").append(m.missionType)
          .append("  | Danger level : ").append(m.dangerLevel)
          .append("  | Date : ").append(m.missionDate)
          .append("  | Duration : ").append(m.expectedDuration)
          .append("  | Bounty Reward : ").append(m.bountyReward).append("\n");
    }

    sb.append("Total Bounty : ").append(crew.getTotalBounty()).append("\n\n");
}

JTextArea textArea = new JTextArea(sb.toString());
textArea.setEditable(false);
textArea.setLineWrap(true);
textArea.setWrapStyleWord(true);
textArea.setFont(new Font("Monospaced", Font.BOLD, 14));

JScrollPane scrollPane = new JScrollPane(textArea);
scrollPane.setPreferredSize(new java.awt.Dimension(1200, 700));
JOptionPane.showMessageDialog(null, scrollPane, "List of pirate missions", JOptionPane.INFORMATION_MESSAGE);

// ---------- Phase 2 : MISSION ROUTING ( QUEUE ) ------------

Queue<CrewMemberInfo> eastBlueQueue = new LinkedList<>();
Queue<CrewMemberInfo> southBlueQueue = new LinkedList<>();
Queue<CrewMemberInfo> grandLineQueue = new LinkedList<>();

boolean toggle = true;

for (CrewMemberInfo crew : crewList) {
    if (crew.missions.size() == 3) {
        if (toggle) {
            eastBlueQueue.add(crew);
        } else {
```

Presentation Project SWC3684 : Pirate Mission Management System

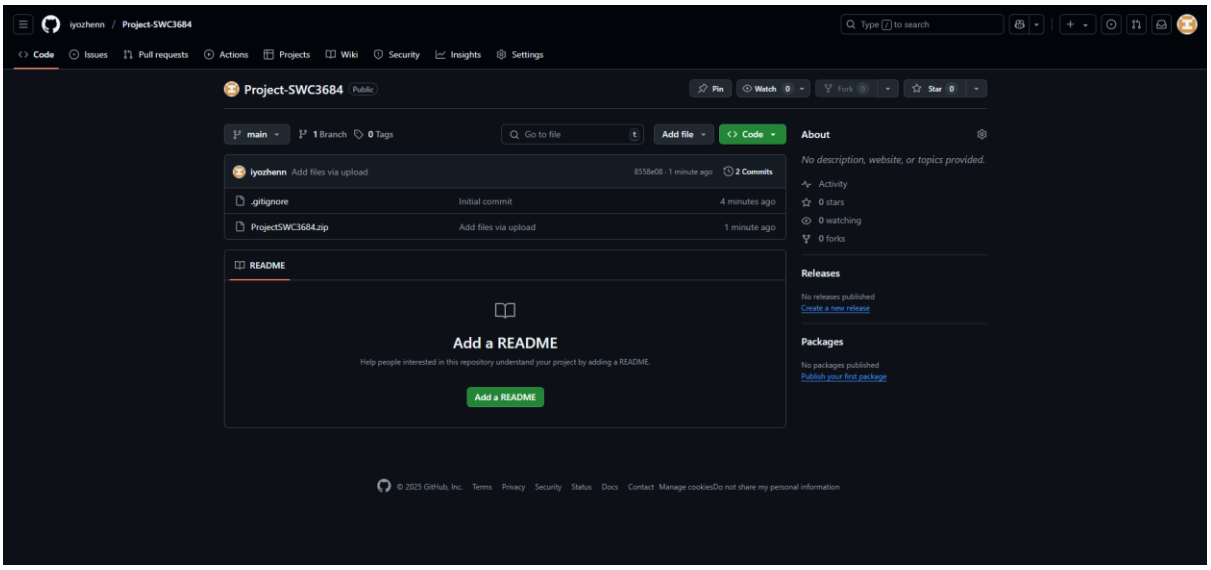
Aiman Harith

1 view 8 minutes ago

Figure 17: Video Presentation Link

Link: https://www.youtube.com/watch?v=IHW7v80kS_o

8. Softcopy GitHub link



iyozhenn Add files via upload 8558e08 · 1 minute ago 2 Commits

.gitignore	Initial commit	4 minutes ago
ProjectSWC3684.zip	Add files via upload	1 minute ago

Help people interested in this repository understand your project by adding a README.

No description, website, or topics provided.
Activity
0 stars
0 watching
0 forks

No releases published
Create a new release

No packages published
Publish your first package

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookiesDo not share my personal information

Figure 18: Softcopy GitHub Link

Link: https://github.com/iyozhenn/Project-SWC3684