Lab 04 Computer Science 1027B Summer 2025

Introduction

The purpose of these labs is to provide practice problems for you to get used to the concepts being covered in class. It is very important that you do these labs on your own without relying on available AI tools. Questions on exams are often very similar to exercises presented in these labs, so understanding them and solving these problems gives you a huge advantage for the exams. Make sure that you complete both the code as well as provide written answers to the blue questions within the boxes.

When you have concluded this lab, you should submit the following files to OWL Brightspace:

- GameCharacter.java
- DoublyLinkedList.java
- written_answers.txt

Topics Covered

- Input/Output
- Serialization
- LinkedLists

Exercise 01

You are back working with the game company from lab 02, but now they want to add some additional features and allow for serialization and deserialization. Write a class, **GameCharacter** that initially has four private instance variables (you'll need to add another one when you implement serialization and deserialization methods later):

- String name
- int level
- int health
- ArrayList<String> inventory

As well as a constructor that takes a String 'name', an int 'level', an int 'health', and an ArrayList<String> 'inventory' and initializes the values for each of the aforementioned instance variables.

Additionally, you should write an overridden toString method that returns a String containing the information associated with an instance of GameCharacter. For example, creating an instance of GameCharacter using the constructor you just wrote and calling toString on that object might yield:

GameCharacter{name='Link', level=1, health=3, inventory=[Branch]}

Note that the String doesn't have to be formatted exactly the same, but it should contain the same vital information (name, level, health, and inventory).

Great! Now let's get to serialization and deserialization.

Let's start with serialization. Write a **public static** method, **saveToFile** (with a void return type), that takes a GameCharacter object 'character' and a String 'filename'. This method should serialize the GameCharacter object to the specified file, ensuring that the object's state is saved so that it can be restored later.. Use 'ObjectOutputStream' and 'FileOutputStream' for this purpose, handling any potential 'IOException' that might occur.

Next, write a **public static** method, **loadFromFile** (with a GameCharacter return type), that takes a String 'filename' as a parameter. This method should deserialize the 'GameCharacter' object from the specified file, effectively restoring the object's state as it was when it was serialized. Use 'ObjectInputStream' and 'FileInputStream' for this purpose, handling any

potential 'IOException' and 'ClassNotFoundException' that might occur. If successful, this method should return the deserialized 'GameCharacter' object.

Q 01.1

What are the benefits of serialization? Is it always necessary?

Now, create a file called Main.java and paste in the following code:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        // create a new game character
        ArrayList<String> inventory = new ArrayList<>();
        inventory.add("Branch");

        GameCharacter character = new GameCharacter("Link", 1, 3, inventory);

        // display the original character details
        System.out.println("Original Character: " + character);

        // save the character to a file
        String filename = "gameCharacter.ser";
        GameCharacter.saveToFile(character, filename);
        System.out.println("Character saved to file: " + filename);

        // load the character from the file
        GameCharacter loadedCharacter = GameCharacter.loadFromFile(filename);

        // display the loaded character details
        System.out.println("Loaded Character: " + loadedCharacter);
    }
}
```

Q 01.2

What is the output after compiling and executing the above code?

Exercise 02

The year is 1999. You are working on a top-secret project to take down the head of an illegal operation focused on changing the dollar menu at a local food chain to the 'two-dollar menu' which is unacceptable.

You sneak into the operation's hub early in the morning, and unfortunately, their computer is locked. You need the documents on that machine, and you need to go undetected. You sit down and find a floppy disk in the desk drawer (again, it's 1999). After checking the contents on the floppy disk, you find that there is just one file labeled "ToP_SecREt_PassWorD.txt". Thinking that it seemed suspicious, you open it and read its contents (check them out in the provided file).

You scratch your head, wondering what this could possibly mean. You know that the password has to be less than 20 characters long, so the contents of the file cannot be the password itself. You remember that you noticed a sticky note on the monitor that read:

Okay, so clearly this guy is a huge fan of the alphabet. Or, maybe, it was a hint. You think to yourself "what if the password is beneath this mess of non-alphabetical characters in the textfile?"

Using what you know about FileInputStreams, write some code that takes the ToP_SecREt_PassWorD.txt and collects and concatenates the alphabetical characters.

Hint: Is there a method in the Character class that could help us with determining whether a character is alphabetical or not?

Q 02.1

What is the password?

Once you've gotten the password, you log into the computer at the hub, freeze the price of every item on the dollar menu, and change the password so that the head of this operation cannot log back in to change them. Everyone in town celebrates your programming knowledge and names you mayor. What a day.

Exercise 03

In the lectures, I demonstrated how to code a SinglyLinkedList and a DoublyLinkedList class. Now, we will be adding some additional functionality to the DoublyLinkedList class.

We want to write a method, removeValue that removes the first node with a specified value within a doubly linked list where the nodes contain int values.

I have provided the DoublyLinkedList.java and Node.java file with this lab. You must add the removeValue method to the DoubleLinkedList class.

Think about what cases your might run into (the specified node is at the front of the DoublyLinkedList, it's in the middle, it's at the end, or it doesn't exist at all). We must handle all those cases!

Consider the following template for the removeValue(int data) method:

```java
public void removeValue(int data) {
    Node currentNode = head;

    // if the list is empty
    if (currentNode == null) {
        // you should then exit the method using return
    }

    // if the node to be removed is the head
    if (currentNode.data == data) {
        // set head to be currentNode.next
        // If head != null, you should probably set head.prev to something, but what?
        // you should then exit the method using return
    }

    // traverse the list to find the node to remove
    while (currentNode != null && currentNode.data != data) {
        // move to the next node (what should we set currentNode to?)
    }

    // if the node was not found
    if (currentNode == null) {
        // you should then exit the method using return
    }

    // if the node to be removed is in the middle or end
    if (currentNode.next != null) {
        // set the next node's prev pointer to currentNode's prev node
    }

    if (currentNode.prev != null) {
        // set the previous node's next pointer to currentNode's next node
    }
}
```

Q 03.1

Did you know that linked lists are common topics for interviews at tech companies? If you were having a student in to interview them for a programming position at your tech company, what linked list method would you get them to write for you to demonstrate their knowledge?