Name: Tayyab Kamran (24)

Roll Number: 22i-2505

Date: 19th September 2024

Weight: **5 Absolutes**

Software Construction and Development Quiz 1

Part 1: Dry Run (3 marks * 5)

```
List<Integer> numbers = new ArrayList<>();
numbers.add(10);
numbers.add(20);
numbers.add(30);

for (int i = 0; i < numbers.size(); i++) {
  numbers.set(i, numbers.get(i) * 2);
}
System.out.println(numbers);
```

20, 40, 60

```
class Animal {
  String name;
  Animal(String name) {
    this.name = name;
  }
}
class Dog extends Animal {
  Dog(String name) {
    super(name);
  }
  void bark() {
    System.out.println(name + " is barking!");
  }
}

Animal animal = new Dog("Buddy");
((Dog) animal).bark();
```

Buddy is barking

```
class GenericBox<T> {
  private T value;
  public void setValue(T value) {
    this.value = value;
  }
  public T getValue() {
    return value;
  }
}

GenericBox<Integer> intBox = new
GenericBox<>();
intBox.setValue(42);
System.out.println(intBox.getValue());
```

```
GenericBox<String> strBox = new
GenericBox<>();

strBox.setValue("Hello");

System.out.println(strBox.getValue());
```

42 ✓

Hello

```
try {
  int[] arr = {1, 2, 3};
  System.out.println(arr[3]);
} catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("Array index out of
bounds.");
} finally {
  System.out.println("Finally block executed.");
}
```

Array index out of bounds.
Finally block executed.

```
class Parent {
  void show() {
    System.out.println("Parent show");
  }
}
class Child extends Parent {
  void show() {
    System.out.println("Child show");
  }
}

Parent obj = new Parent();
Child c = (Child) obj;
c.show();
```

✓

Parent Object cannot be typecasted into a Child Object. After Error remove output will be: Child show.

Part 2: Coding Problem (15 Marks)

Given an array of integers (which may contain duplicates), write a program to generate all the unique subsets of the array. Your program should not output any duplicate subsets, and the subsets should be printed in lexicographically sorted order.

Input: An array of integers, which may contain duplicates.

Output: A list of unique subsets, sorted in lexicographical order.

Example:

[1, 2, 3]

[], [1], [1, 2], [1, 2, 3]

Input:

[1, 2, 2]

Output:

[], [1], [1, 2], [1, 2, 2], [2], [2, 2]

Explanation:

- The input array has duplicates, but the output subsets should be unique.
- The output subsets should be sorted lexicographically (by set length, then elements).
- You must ensure that there are no duplicate subsets in the output.

```
import java.utils.ArrayList;
public class RemoveDuplicate {
    void ~~main~~ {
        ~~ArrayList myList = new ArrayList();~~

    ArrayList sortFunc( ArrayList myList)
    {
        ArrayList * newList = new ArrayList[];
        newList.push([]); ~~System.out.println("[],");~~
        for(int i=0; i < myList.size(); i++){
            for( int j= 0; j < myList.size(); i++) {
            if (i == j) && ~~!~~(newList.contains(myList[i]))
            { ~~myList~~newList.push( myList[i]);
              ~~System.out.println("[" + myList[i] + "]");~~
            } else { continue;}
            newList[i].push (myList[j]);
        }
    }
}
```

newList ←

(7+2)

```
public static void main ( String [] argc){
    ArrayList myList = new ArrayList();
    myList.push(1); myList.push(2); myList.push(2);
Arraylist new = sortFunc( myList);
    for( i=0; i< new.size() ; i++){
} sout ( new[i] );
```

77

Name:
Roll Number:
Section: **BSE-5B** **3rd September, 2024**

Q1. Find Error from the following code and correct it (if any) and write its EXACT output as it will be displayed on compiler

i.

```
public class InputAge {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter your age: ");
        int age = input.nextDouble();
        int age = input.nextInt();
            OR
        Double age = input.nextDouble();

        System.out.println("Your age is " + age);
    }
}
```

Output here:

ii.

```
public class User {
   private String name;
   private int age;
   public void setName(String newName) {
      name = newName;
   }
   public String getName() {
      return name;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int newAge) {
      age=newAge;
   }

}
class Main{
   public static void main(String[] args) {
      User obj = new User();
      obj.age = 25; //age is private so cannot be  changes so create setter
      obj.setAge(25);
      System.out.print("Age: " + obj.getAge());
      System.out.println("Name:  " + obj.getName); //brackets for function call
missing
     System.out.println("Name:  " + obj.getName());
   }
}
```

Output here:

iii.
```
public class ChainingTest {
    public ChainingTest() {
        this(10);
        System.out.println("Default constructor");
    }
    public ChainingTest(int x) {
        System.out.println("Parameterized constructor: " + x);
    }
    public static void main(String[] args) {
        ChainingTest obj = new ChainingTest();
    }
}
```

Output here:

//no error in this one

Parameterized constructor: 10

Default constructor

```
class Parent {
  private  int x = 10;
  Parent() {
    System.out.println("Parent Constructor");
  }
}
 class Child extends Parent { //final class cannot extend also Child class cannot be static
  int y = 20;
  Child() {
    super();  //super() must be the 1st statement in  child's constructor if used
    System.out.println("Child Constructor");
    this();  //this will call infinite chaining for child constructor
    x=100; //cannot access private variable of parent class
  }
  void display() {
    System.out.println("Child display: " + this.y);
    System.out.println("Parent display: " + super.x);
  }
  public static void main(String[] args) {
    Parent obj = new Child();
    obj.display(); //parent object cannot access  child's function
  }
}
```

Output here:

Q2. Write java code to take two strings from user using JOptionPane. Display the frequency of second string in first string. Also display starting index of occurrences. Suppose:

String1 = abcdabcdeabc

String2= abc

Frequency = 3

Indexes = [0, 4, 9]

```
import javax.swing.JOptionPane;
import java.util.ArrayList;

public class StringFrequency {
    public static void main(String[] args) {
        // Taking input from user using JOptionPane
        String str1 = JOptionPane.showInputDialog("Enter the first string:");
        String str2 = JOptionPane.showInputDialog("Enter the second string:");

        // Find frequency and indexes of occurrences
        int frequency = 0;
        ArrayList<Integer> indexes = new ArrayList<>();

        int index = str1.indexOf(str2);
        while (index != -1) {
            frequency++;
            indexes.add(index);
            index = str1.indexOf(str2, index + 1);
        }
```

```
        // Displaying results
        JOptionPane.showMessageDialog(null, "Frequency = " + frequency + "\nIndexes = " +
indexes);
    }
}
```

Name:
Roll Number:
Section: **BSE-5B** **3rd September, 2024**

Write a Java program that reads the contents of a file and continuously takes user input. The user will provide two inputs: a character and an integer value (k). The program should check if the character exists in the file. If the character is found, it will multiply the character's frequency by the input value (k) and update the characters to new frequency count and save the new text in file

The program should keep accepting input from the user until the user enters -1 to stop the process.

[10]

For example:
the text is: hello world

K=3

Character= r

Updated text is: hello worrrld

K=3

Character=r

Updated text is: hello worrrrrrrrld

K=3

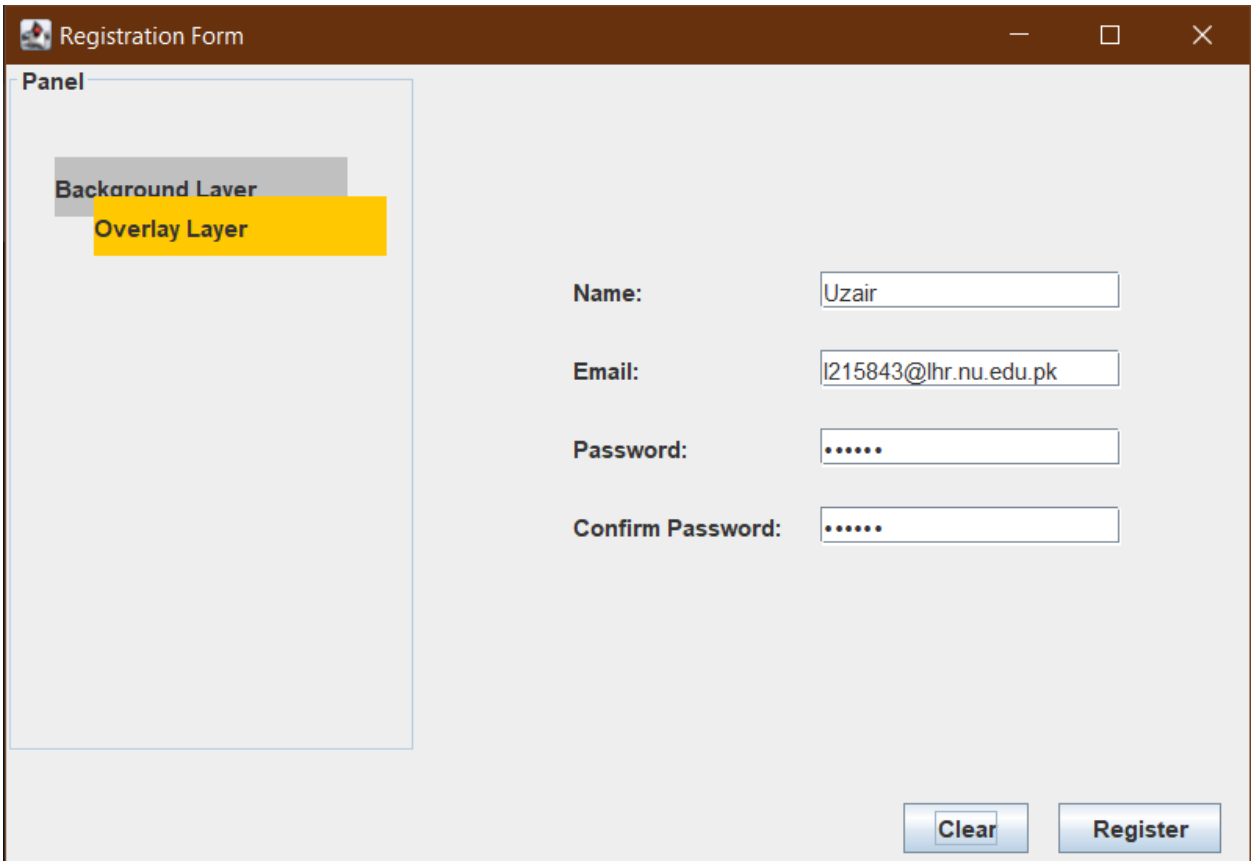
Character=o

Updated text is: hellooo wooorrrrrrrrld

Q1. MCQS **[5]**

1. **Which layout manager is best if you want precise control over the location of each component?**
    a. Flow Layout
    b. Null Layout
    c. Border Layout
    d. Grid Layout
2. **In which layout can components be added to specific positions like NORTH, SOUTH, EAST, WEST, and CENTER?**
    a. Flow Layout
    b. Border Layout
    c. Grid Layout
    d. Null Layout
3. **Which component is used for single-line text input in Java Swing?**
    a. JLabel
    b. JTextField
    c. JTextArea
    d. JPasswordField
4. **Which layout manager arranges components from left to right in a row and wraps them to the next line if there isn't enough space?**
    a. Flow Layout
    b. Border Layout
    c. Grid Layout
    d. Null Layout
5. **Which method is used to add an action listener to a JButton?**
    a. addClickListener()
    b. setActionListener()
    c. addActionListener()
    d. setClickListener()

Q2. Write java code to create the following GUI form also add functionality to buttons. **[10]**



Name:
Roll Number:
Section: **BSE-5B** **28rd October, 2024**

Q1. Write a Java JDBC code to insert a new record into a "students" table with columns *(id, name, section, CGPA)* in the database. The id, name, section and CGPA being your roll number, name, section and CGPA. The username for your database is: "Root" and password is "You are an amazing student".

You can assume that drivers are already there just make a connection and insert a record. Do not hard code the values when inserting data into the table.

Q2.MCQ

1) What does JDBC stand for?

A) Java Database Connectivity

B) Java Direct Connection

C) Java Development Code

D) Java Database Collection

2) Which interface is used to execute SQL queries in JDBC?

A) Statement

B) PreparedStatement

C) Connection

D) ResultSet

3) In JDBC, which method is used to establish a connection with the database?

A) connect()

B) getConnection()

C) open()

D) startConnection()

4) What is the purpose of the PreparedStatement in JDBC?

A) It is used to dynamically create database tables.

B) It precompiles SQL queries to improve performance and security.

C) It is used for executing batch updates.

D) It manages database transactions.

5) Which method is used to execute an SQL SELECT query in JDBC?

A) execute()

B) executeUpdate()

C) executeQuery()

D) executeFetch()

Quiz on Client-Server Socket Programming and Multithreading in Java

Multiple Choice Questions (MCQs)

1. **What is the purpose of the `ServerSocket` class in Java?**
   a) To establish a connection to a remote server
   b) To listen for incoming client connections
   c) To send data to the client
   d) To close the server program
2. **Which of the following methods is used to accept a client connection in a server program?**
   a) `connect()`
   b) `accept()`
   c) `listen()`
   d) `bind()`
3. **What exception must be handled when working with sockets in Java?**
   a) IOException
   b) SocketException
   c) NetworkException
   d) FileNotFoundException
4. **Which method is used to read data from a socket in Java?**
   a) `readLine()`
   b) `read()`
   c) `getData()`
   d) `fetch()`
5. **What is the primary advantage of using multithreading in socket programming?**
   a) Increases CPU usage
   b) Handles multiple clients simultaneously
   c) Reduces server workload
   d) Simplifies debugging

Short Answer Questions

1) Explain the role of the `accept()` method in the `ServerSocket` class.
   a) The accept() method of the ServerSocket class waits for an incoming client connection. When a client attempts to connect to the server, the accept() method accepts the connection and returns a Socket object that can be used to communicate with the client.
2) What is the difference between `Runnable` and `Thread` when creating a multithreaded program?
   a) Runnable is an interface that defines a single method run(). It allows a class to implement multithreading without extending the Thread class, which enables inheritance from other classes.
   b) Thread is a class that directly represents a thread. You can create a thread by extending the Thread class and overriding its run() method.

3) Write the syntax for creating a server socket that listens on port 8080.
   a) ServerSocket serverSocket = new ServerSocket(8080);
4) How does multithreading improve the performance of a socket-based server?
   a) Multithreading allows a server to handle multiple client connections concurrently. Each client can be processed in its own thread, ensuring that the server doesn't become blocked by a single client. This improves responsiveness and scalability.
5) Describe the purpose of the `InputStream` and `OutputStream` classes in socket programming.
   a) InputStream: Used to read data from a socket. It represents an input stream of bytes coming from the client or server.
   b) OutputStream: Used to write data to a socket. It represents an output stream of bytes being sent to the client or server

---

Practical Questions

1. Write a Java program to create a simple server that listens on port 12345 and sends a welcome message to any connected client.
2. Modify the above program to handle multiple clients using multithreading.
3. Write a Java client program that connects to a server, sends a message, and prints the server's response.
