**1). How can you suggest enhancement to this code snippet.**

**public** **class** ArrArgs {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

**if** (args.length == 0) {

System.***out***.println("No arguments provided.");

**return**;

}

**try** {

**int** k = 0;

**while** (k < args.length) {

System.***out***.println("Value of input is " + k + " and argument is " + args[k]);

k++;

}

} **catch** (ArrayIndexOutOfBoundsException errorOc) {

System.***err***.println("Error occurred: " + errorOc.toString());

}

}

}

1. Add a condition to terminate the loop: The current implementation uses an infinite loop with a catch block to handle the ArrayIndexOutOfBoundsException and print an error message. Instead, you can modify the loop condition to terminate when all arguments have been processed. For example, you can change the loop to while (k < args.length).

2. Provide a user-friendly message for missing arguments: If no command-line arguments are provided, the current implementation will throw an ArrayIndexOutOfBoundsException. You can add a check before entering the loop to ensure that there are arguments present. If not, you can display a user-friendly message indicating that no arguments were provided.

3. Separate the printing logic from exception handling: In the current implementation, the printing of values and handling of exceptions are combined in the same block. It is generally good practice to separate these concerns. You can move the printing logic outside the try-catch block, and if an exception occurs, it can be caught and handled separately.

1. **What is the o/p if java security manager is installed and below program is ran**.

The code you provided reads two integers from an input file ("input.txt"), calculates their sum, and writes the sum to an output file ("output.txt"). If a security manager is installed and properly configured, it may restrict certain actions performed by the code, such as reading from or writing to files.

The specific output you will get when running this program with a security manager depends on the security policy in place. It could vary based on factors such as the permissions granted to the code and the restrictions imposed by the security manager.

If the security manager has not granted the necessary permissions to read from or write to files, you may encounter a SecurityException or an AccessControlException when attempting those operations. In such cases, the catch block will be executed, and the program will print the error message "Error" followed by the exception details.

To ensure the program can read from and write to files successfully, make sure the necessary permissions are granted in the security policy configuration.

**3) Write a program to occur out of memory error using StringBuffer(while doing append) with**

**storing millions of records and how to resolve it with GC and without GC?**

Note: Don’t explicitly set perm size. System should take by default.

To create a Java program that triggers an OutOfMemoryError while using StringBuffer's append method to store millions of records, you can follow this example:

import java.util.ArrayList;

import java.util.List;

public class OutOfMemoryExample {

public static void main(String[] args) {

List<String> records = new ArrayList<>();

try {

while (true) {

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append("Record ");

records.add(stringBuffer.toString());

}

} catch (OutOfMemoryError error) {

System.out.println("Out of memory error occurred!");

error.printStackTrace();

}

}

}

In this program, an ArrayList named records is used to store millions of records. Inside the try block, a StringBuffer is created in each iteration, and the text "Record " is appended to it. The resulting string is then added to the records list. This process continues until an OutOfMemoryError is thrown, indicating that the JVM has exhausted its memory resources.

To resolve this issue, you can employ garbage collection (GC) and/or increase the available memory to the JVM.

1. Using GC: In most cases, the JVM will automatically perform garbage collection when it detects that memory is running low. However, if you want to trigger garbage collection explicitly, you can call System.gc() within the catch block, right before printing the error message. This request suggests to the JVM that it should attempt to free up memory before throwing the OutOfMemoryError.

catch (OutOfMemoryError error) {

System.out.println("Out of memory error occurred!");

System.gc(); // Request garbage collection

error.printStackTrace();

}

1. Increasing available memory: If you want to allocate more memory to the JVM, you can specify the maximum heap size using the -Xmx flag when running the Java program. For example, to set the maximum heap size to 2 GB, you can use the following command:

java -Xmx2g OutOfMemoryExample

By increasing the available memory, you provide the JVM with more space to store the records, reducing the likelihood of encountering an OutOfMemoryError.

Please note that while GC and increasing the memory can mitigate OutOfMemoryErrors in certain scenarios, they may not always be sufficient. If the program requires an excessive amount of memory or there are memory leaks, alternative approaches such as optimizing data structures or employing pagination strategies may be necessary.

**4) Write an implementation to read list of items to be added into the restaurant menulist.**

**The items can be sorted naturally based on categories and also explicitly able to sort based on**

**price/name.**

Here's an implementation in Java that allows you to read a list of items to be added to a restaurant menu list. The items can be sorted naturally based on categories and can also be explicitly sorted based on price or name:

import java.util.\*;

class MenuItem implements Comparable<MenuItem> {

private String name;

private String category;

private double price;

public MenuItem(String name, String category, double price) {

this.name = name;

this.category = category;

this.price = price;

}

public String getName() {

return name;

}

public String getCategory() {

return category;

}

public double getPrice() {

return price;

}

@Override

public int compareTo(MenuItem other) {

// Compare based on category, then price, then name

int categoryComparison = this.category.compareTo(other.category);

if (categoryComparison != 0) {

return categoryComparison;

}

int priceComparison = Double.compare(this.price, other.price);

if (priceComparison != 0) {

return priceComparison;

}

return this.name.compareTo(other.name);

}

@Override

public String toString() {

return name + " (" + category + ") - $" + price;

}

}

public class RestaurantMenu {

public static void main(String[] args) {

List<MenuItem> menu = readMenuItems();

// Sort menu items naturally based on categories

Collections.sort(menu);

System.out.println("Menu items sorted by category:");

printMenu(menu);

// Sort menu items by price

Collections.sort(menu, Comparator.comparingDouble(MenuItem::getPrice));

System.out.println("\nMenu items sorted by price:");

printMenu(menu);

// Sort menu items by name

Collections.sort(menu, Comparator.comparing(MenuItem::getName));

System.out.println("\nMenu items sorted by name:");

printMenu(menu);

}

private static List<MenuItem> readMenuItems() {

List<MenuItem> menu = new ArrayList<>();

// Read menu items from input (example)

menu.add(new MenuItem("Hamburger", "Main Course", 9.99));

menu.add(new MenuItem("Caesar Salad", "Appetizer", 6.99));

menu.add(new MenuItem("Cheesecake", "Dessert", 4.99));

menu.add(new MenuItem("Pizza", "Main Course", 12.99));

menu.add(new MenuItem("French Fries", "Side Dish", 3.49));

menu.add(new MenuItem("Ice Cream", "Dessert", 3.99));

menu.add(new MenuItem("Bruschetta", "Appetizer", 7.99));

menu.add(new MenuItem("Steak", "Main Course", 18.99));

return menu;

}

private static void printMenu(List<MenuItem> menu) {

for (MenuItem item : menu) {

System.out.println(item);

}

}

}

**5) Design a project to display hello world.**

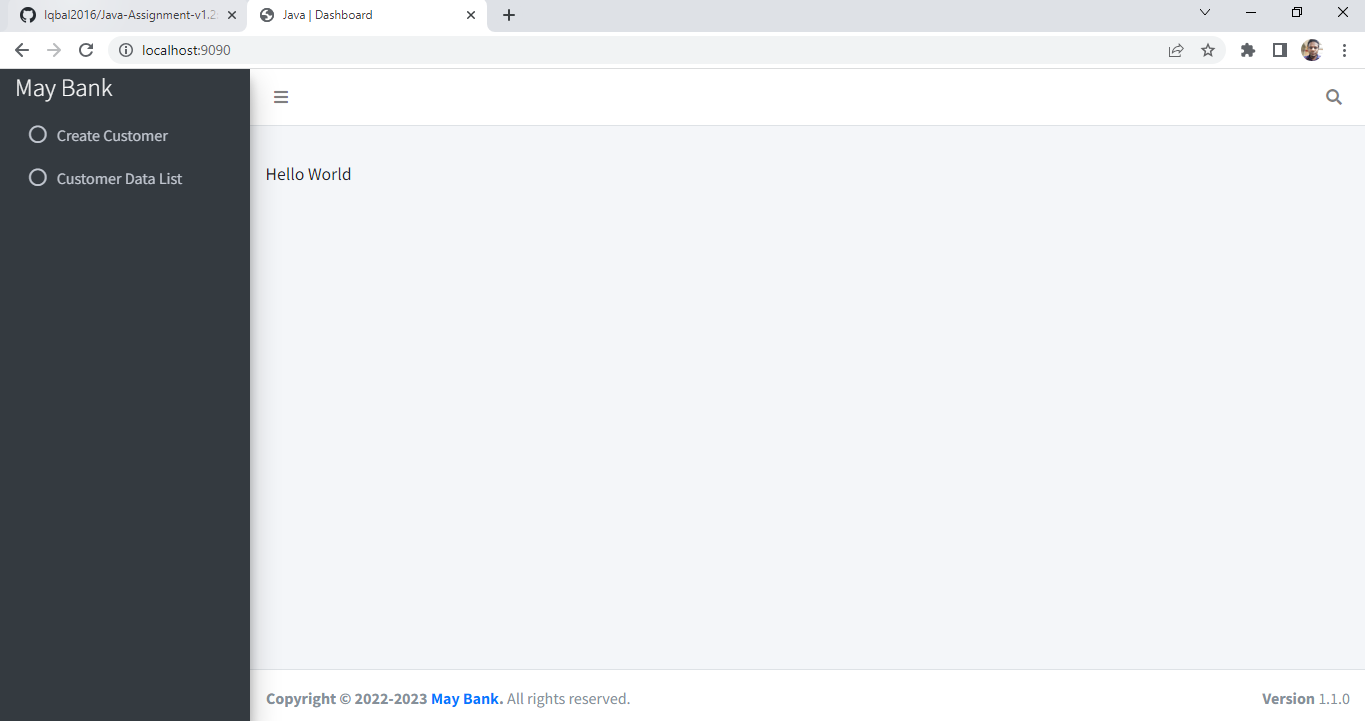
**a) Use any view page like jsp, html etc.**

**b) Use web framework as controller**

**c) Use MYSQL db to storing values**

**d) Can we build web project without web.xml?**

**Java Spring Boot project for display hello world**



1. For view page used Spring Boot and Thymeleaf.
2. Used Spring web framework as controller HomeController
3. MYSQL db name JavaAssignmentDB
4. Yes, starting from Servlet 3.0 specification, it is possible to build a Java web project without using the traditional web.xml deployment descriptor. Servlet 3.0 introduced a new feature called "Servlet annotations" that allows you to configure servlets, filters, and listeners using annotations instead of web.xml.

6.1) Create a DB with Customer ID, Short Name, Full Name, Address 1, Address 2 and Address

3(each Address line is 80 Chars), City, and Postal Code.

6.2) Prepare a screen to list Customers, and select Customer to display the Address.

6.3) Address screen will have ADD, Modify and Delete.

6.4) All the Customers max can have 3 address.

6.5) Validations on Postal Code required.

6.1) Table create command:

CREATE TABLE customer\_tbl (

id bigint,

customerid varchar(10),

short\_name varchar(20),

full\_name varchar(60),

address1 varchar(80),

address2 varchar(80),

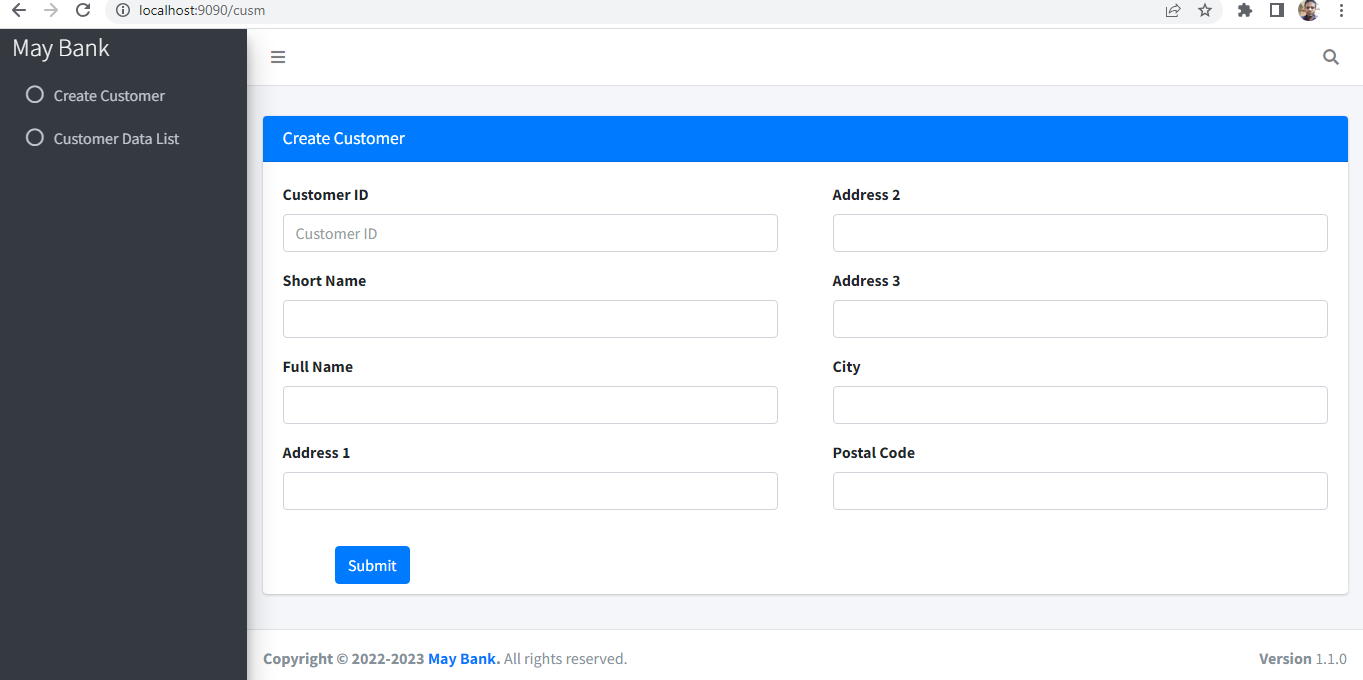
address3 varchar(80),

city varchar(60),

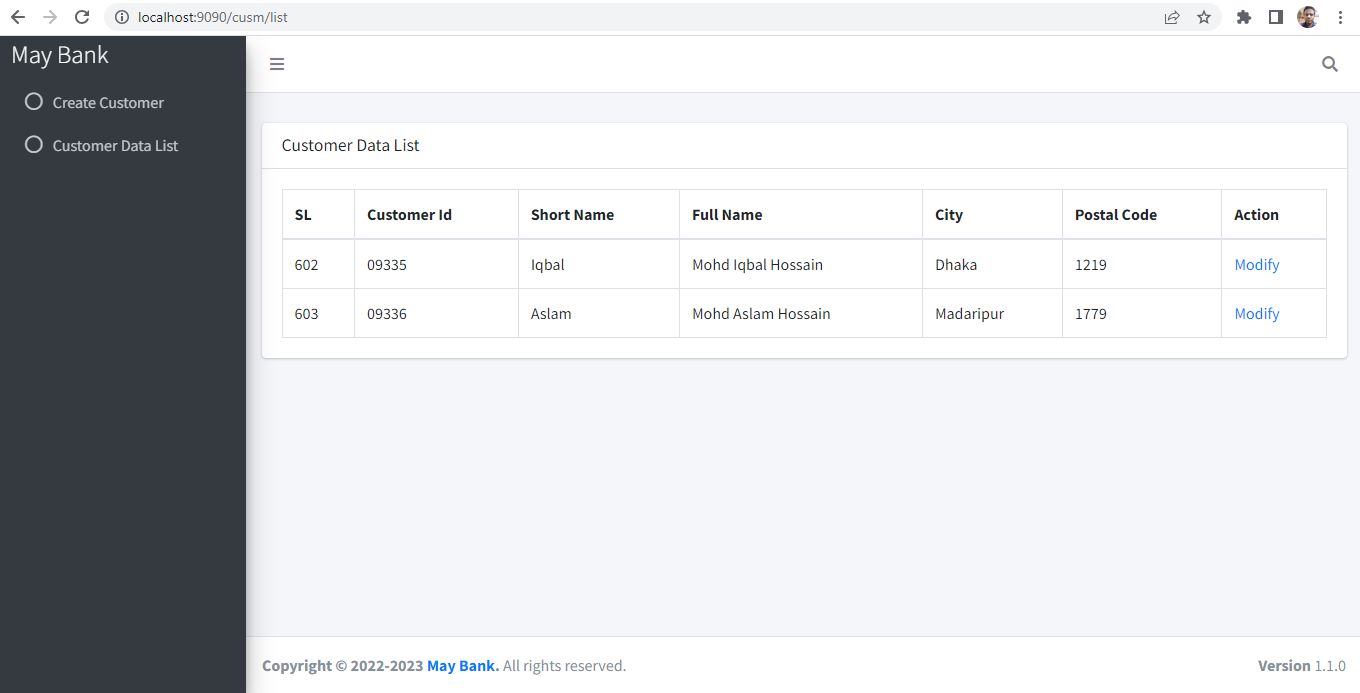
postal\_code varchar(10) NOT NULL,

PRIMARY KEY (id)

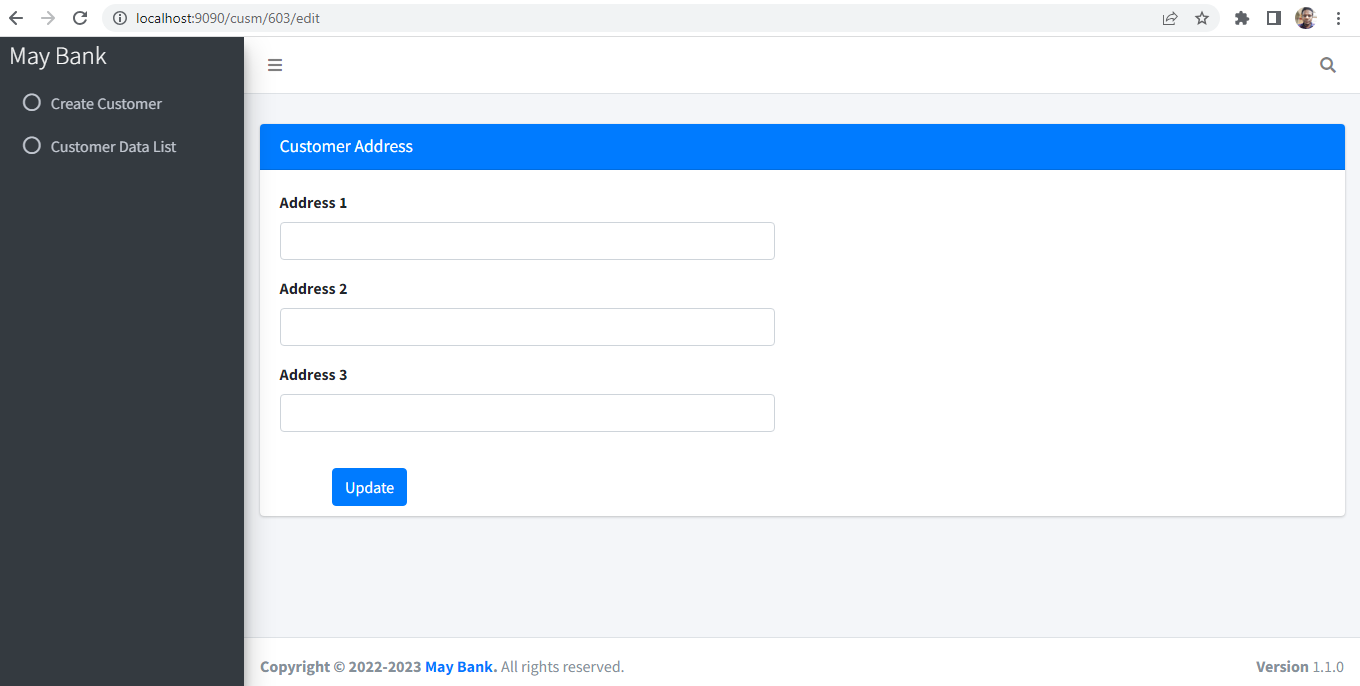
);

6.2) Add Customer page

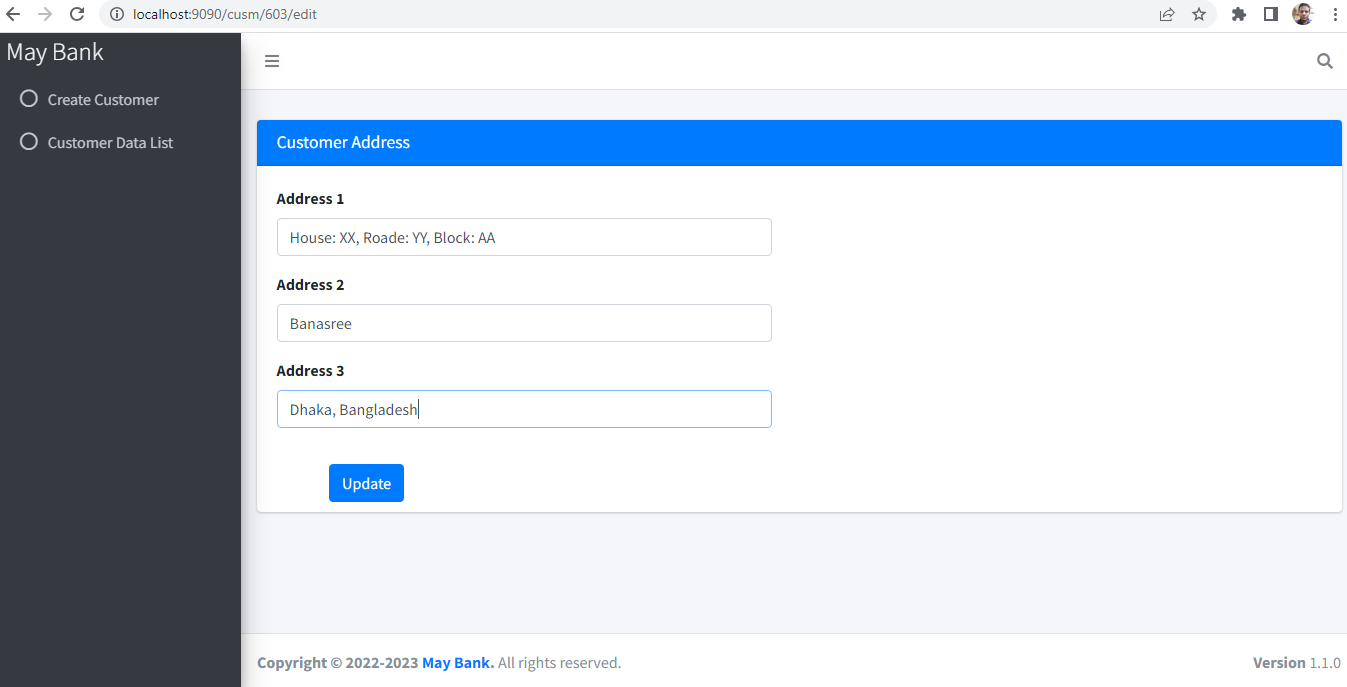
Customer List



Click Modify > Go to > Current Customer Update screen



Add Customer 3 Address



For this process I can update customer addresses from the Customer Address page. No need to delete button. If required to delete customer address, just remove customer address and update customer address