**Table of content**

|  |  |  |
| --- | --- | --- |
| **No** | **Content** | **Page** |
| 1 | **Part B: System Requirements Specification**  B1 Introduction  B2 Overall Description  B3 Functional Requirement & Non-Functional Requirement & Assumption  B4 Workload allocation | **3 - 8**  3  4  5 – 7  8 |
| 2 | **Part C: Class Diagram** | **8 – 9** |
| 3 | **Part D: Source code, sample of input data, sample output**  Source code  Sample of input data  Sample output | **10 – 82**  10 – 55  56  57 – 82 |

**Part B: System Requirements Specification**

**B1 Introduction**

**Purpose**

This application aims to manage and track UX Gallery’s operations which are customer data, artists and their works, and the gallery’s inventory. The gallery can maintain records of sales and client preferences by tracking customer information and art purchases. Tracking artists and their creations can benefit future exhibitions by keeping track of the art that has been displayed in the gallery. The gallery can manage its art stock and make sure it is well-maintained and organized by keeping track of its inventory. This application enhances business process consistency and efficiency by reducing the possibility of human error.

**Project Scope**

The main goal of the art gallery management system is to provide an easy-to-use application for the end users, which are the gallery’s staff to keep track of customer and their art purchases, artists and their works that have appeared in the gallery, and the current inventory of the gallery that is simple and easy to use.

This application offers features such as adding, deleting, modifying, searching, and listing customer, artist, and artwork data. The inventory displays the artworks’ purchase price and selling price when sold.

**Overview**

This report is as organized as follows:

* Part B1 is the introduction to the system where it states the purpose and the project scope of the system.
* Part B2 is the overall description where it states the user and product perspectives, and characteristics.
* Part B3 shows the workload allocation, functional requirements, non-functional requirements, and assumptions of the system.
* Part C displays the class diagram draft, and the completed class diagram.

**B2 Overall Description**

**User perspective**

User of the system is the staffs of UX Gallery

The primary user will be the manager of UX Gallery.

The secondary users are the staff in UX Gallery who have been given access to use the system.

**Product perspective**

This system is to allow the user to manage the following information:

* Customers and their art purchases
* Artist and their works
* Current inventory

All the information will be saved in a text file automatically every time user ends the session.

**Characteristics**

The user should be able to do the following functions:

* Create a new customer/artist/artwork.
* Search and list the customer/artist/artwork.
* Modify the customer/artist/artwork.
* Delete the customer/artist/artwork.

**B3 Functional Requirement & Non-Functional Requirement & Assumption**

**Functional Requirement**

|  |  |
| --- | --- |
| **ID** | **Functional Requirements** |
| **FR01** | The application shall allow the user to add a new customer, including their name, contact information, and artist preferences. |
| **FR02** | The application shall allow the user to modify an existing customer's information, including their name, contact information, and artwork purchases. |
| **FR03** | The application shall allow the user to delete a customer's information from the system. |
| **FR04** | The application shall allow the user to search for a customer by entering customer ID. |
| **FR05** | The application shall allow the user to list all customers in the system. |
| **FR06** | The application shall allow the user to add a new artist, including their name, specialty, whether they are alive or deceased, and price ranges of artwork. |
| **FR07** | The application shall allow the user to modify an existing artist's information, including their name, specialty, whether they are alive or deceased, and price ranges of artwork. |
| **FR08** | The application shall allow the user to delete an artist's information from the system. |
| **FR09** | The application shall allow the user to search for an artist by entering artist ID. |
| **FR10** | The application shall allow the user to list all artists in the system. |
| **FR11** | The application shall allow the user to add a new artwork, including the title, date purchased, date sold, artist, purchase price, and selling price. |
| **FR12** | The application shall allow the user to modify an existing artwork's information, including the title, date purchased, date sold, artist, purchase price, and selling price. |
| **FR13** | The application shall allow the user to delete an artwork's information from the system. |
| **FR14** | The application shall allow the user to search for an artwork entering by artwork ID. |
| **FR15** | The application shall allow the user to list all artwork in the system. |
| **FR16** | The application shall allow the user to generate a report of the current inventory, including the purchase price and selling price of each artwork. |

**Non-Functional Requirement**

|  |  |  |
| --- | --- | --- |
| **ID** | **Non-functional requirement** | **Category** |
| NFR01 | The system shall be built in Java | Development |
| NFR02 | The system interface shall be simple and responsive | Usability |
| NFR03 | The system shall be available all the time | Availability |
| NFR04 | The system shall conform to the Data Protection Act 2002 | Legislative |
| NFR05 | The system shall be compatible with operating system window 10 and above | Compatibility |
| NFR06 | The system shall be able to handle 100 artworks, 100 artists, 100 customers without slowing system performance | Scalability |

**Assumption**

1. Gallery staff will update inventory manually.
2. Data will be stored in text file.
3. System end user is gallery staff.
4. User inputs should be valid, and a prompt will be prompted if user inputted invalid input.
5. The interface will return to main menu if the user inputs invalid artwork or artist ID.
6. All artwork title, artist name and customer name are unique.
7. Each artwork purchase is associated with only one customer, and that each customer may have multiple artwork purchases.
8. Each artwork is associated with one and only one artist, and that each artist may have multiple artworks.
9. Purchase price and selling price of each artwork are specified in the same currency.
10. The inventory is updated in real-time as artwork is sold, and the selling price is recorded at the time of sale.
11. User has access to a text file for saving and loading information, and that the file is in a specified location.
12. User has permission to read and write to the text file, and that the file is not being used by another program while the application is running.

**B4 Workload allocation**

|  |  |
| --- | --- |
| Name | Tasks |
| Choo Jia Zheng | Non-functional requirements, assumptions, class diagrams, main menu class, source code, sample input/output data. |
| Lee Jia Wei | Functional requirements, assumptions, Artwork Menu class, Artwork Method class. |
| Loke Weng Yan | B1 and B2 of report, Customer Menu class, Customer Method class. |
| Tin Hui Hui | B1 and B2 of report, Artist Menu class, Artist Method class. |

**Part C(i): Draft Class Diagram**

Diagram

Description automatically generated

**Part C(ii): Class Diagram**

Diagram, schematic

Description automatically generated

**Part D: Source code, sample of input data, sample output**

**Source code**

**Main.java**

|  |
| --- |
| package group\_2;  import java.io.IOException;  import java.io.FileNotFoundException;  import java.io.File;  import java.util.\*;  public class Main {  public static void main(String[] args) {  System.out.println("--------------------------------------------------------------");  System.out.println(" WELCOME TO UX GALLERY PROGRAM ");  System.out.println("--------------------------------------------------------------");  System.out.println("1. CUSTOMER");  System.out.println("2. ARTIST");  System.out.println("3. ARTWORK");  System.out.println("4. EXIT");  System.out.print("\nPlease enter your choice : ");  Scanner input = new Scanner(System.in);  int choice = input.nextInt();    switch(choice) {  case 1:  customerInformationMenu();  break;  case 2:  artistInformationTest();  break;  case 3:  artworkInformationMenu();  break;  case 4:  System.out.println("Thank you for using our application!!!");  return;  }  }    public static void customerInformationMenu()  {  CustomerInformation customerInfomation = new CustomerInformation();  ArrayList <CustomerInformation> customerList = customerInfomation.getList();  customerList.clear();  customerInfomation.readCustomerFile();    try {  Scanner input = new Scanner(System.in);  int choice;  boolean loop = true;    while(loop)  {    System.out.println("------------------------------------------------------------------------------------------");  System.out.printf("%1$55s%n", "CUSTOMER INFORMATION");  System.out.println("------------------------------------------------------------------------------------------");    System.out.printf("%1$60s%n%n","List of the customer information");    System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n", "ID", "NAME", "CONTACT", "ARTWORK PURCHASED", "ARTIST PREFERENCES");  System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n", "----", "-----------------", "-----------", "-----------------", "------------------");    for (CustomerInformation customerInformation : customerList)  {  System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n",customerInformation.getCustomerId(), customerInformation.getCustomerName(),  customerInformation.getCustomerContact(), customerInformation.getCustomerArtworkPurchased(), customerInformation.getCustomerArtistPreferences());  }    System.out.println("\n------------------------------------------------------------------------------------------");    System.out.println("Do you want to : ");  System.out.println("1. Add\n2. Delete\n3. Modify\n4. Search\n5. Exit\n");  System.out.print("Enter option : ");    choice = input.nextInt();  input.nextLine();    boolean loop2 = true;  while(loop2)  {  if (choice == 1)  {  CustomerInformation.addCustomer();  loop2 = false;  }  else if (choice == 2)  {  CustomerInformation.deleteCustomer();  loop2 = false;  }  else if (choice == 3)  {  CustomerInformation.modifyCustomer();  loop2 = false;  }  else if (choice == 4)  {  CustomerInformation.searchCustomer();  loop2 = false;  }  else if (choice == 5)  {  loop2 = false;  loop = false;  Main.main(null);  }  else  {  System.out.print("Please enter your choice : ");  choice = input.nextInt();  input.nextLine();  }  }  }  input.close();  }catch (Exception ex){  ex.getMessage();  }  }    public static void artistInformationTest()  {  // Create an object of the ArtistInformation class and initialize an ArrayList of ArtistInformation objects  ArtistInformation artistInformation = new ArtistInformation();  ArrayList<ArtistInformation> artistList = artistInformation.getList();  // Read artist information from a file  artistInformation.readArtistFile();    // Create a Scanner object to read user input  Scanner input = new Scanner(System.in);    try {  int opt;  boolean loop = true;    // Enter a loop that displays the menu until the user chooses to exit  while (loop) {  // Display the header and artist information table  System.out.println("------------------------------------------------------------------------------------------");  System.out.printf("%1$49s%n", "ARTIST INFORMATION");  System.out.println("------------------------------------------------------------------------------------------");  System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n", "ID", "NAME", "SPECIALTY", "STATUS", "PRICE RANGE");  System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n", "----", "-----------------", "-----------", "-----------------", "------------------");  for (ArtistInformation artistInfo : artistList) {  System.out.printf("%1$-5s %2$-20s %3$-15s %4$-20s %5$-20s%n", artistInfo.getArtistID(), artistInfo.getArtistName(),  artistInfo.getSpecialty(), artistInfo.getStatus(), artistInfo.getPriceRange());  }  System.out.println("\n------------------------------------------------------------------------------------------");    // Display the menu and prompt the user for input  System.out.println("Do you want to: ");  System.out.print("\n1. Add\n2. Delete\n3. Modify\n4. Search\n5. Exit\nEnter option: ");    opt = input.nextInt();    // Enter a loop that executes the chosen operation and repeats until a valid option is chosen  boolean loop2 = true;  while(loop2) {  if(opt == 1) {  ArtistInformation.addArtist(); // Call the addArtist() method of the ArtistInformation class  loop2 = false;  }  else if(opt == 2) {  ArtistInformation.deleteArtist(); // Call the deleteArtist() method of the ArtistInformation class  loop2 = false;  }  else if(opt == 3) {  ArtistInformation.modifyArtist(); // Call the modifyArtist() method of the ArtistInformation class  loop2 = false;  }  else if(opt == 4) {  ArtistInformation.searchArtist(); // Call the searchArtist() method of the ArtistInformation class  loop2 = false;  }  else if(opt == 5) {  System.out.println("Exiting to the Main Menu...");  loop2 = false;  loop = false; // Exit the loop that displays the menu  Main.main(null); // Call the main() method of the Main class to return to the main menu  }  else {  // Display an error message and prompt the user for a valid option  System.out.println("Invalid option. Only 1 - 5 is allowed.");  System.out.println("Please enter an option: ");  opt = input.nextInt();  input.nextLine();  }  }  }  input.close(); // close the Scanner object  } catch (Exception ex) {  ex.getMessage();  }  }    public static void artworkInformationMenu()  {  // Creating scanner object to get user input  Scanner scanner = new Scanner(System.in);  // Creating artwork manager object to access the inventory.txt file  ArtworkManager artworkManager = new ArtworkManager("inventory.txt");    // Display the header information for the artwork  System.out.println("--------------------------------------------------------------------------------------------------------------------------------------------------------------------");  System.out.printf("%80s\n", "ARTWORK INFORMATION");  System.out.println("--------------------------------------------------------------------------------------------------------------------------------------------------------------------");  System.out.println(String.format("%-10s%-30s%-30s%-20s%-20s%-20s%-20s%s", "ID", "Name", "Artist", "DatePurchase", "DateSold", "PurchasePrice", "SellingPrice", "Status"));    // Display the artwork information in a tabular format  for (Artwork artwork : artworkManager.listArtwork()) {  System.out.println(String.format("%-10s%-30s%-30s%-20s%-20s%-20s%-20s%s",  artwork.getId(),  artwork.getTitle(),  artwork.getArtist(),  artwork.getDatePurchased(),  artwork.getDateSold(),  artwork.getPurchasePrice(),  artwork.getSellingPrice(),  artwork.getStatus()));  }  // Loop until the user chooses to exit the program  boolean exit = false;  while (!exit) {  // Display the options menu to the user  System.out.println("\nDo you want to:");  System.out.println("1. Find");  System.out.println("2. Add");  System.out.println("3. Modify");  System.out.println("4. Delete");  System.out.println("5. Exit");  System.out.print("\nPlease enter your choice : ");  // Read the user's choice from the console input  int choice = scanner.nextInt();  scanner.nextLine(); // consume the new line character left over from scanner.nextInt() call  // Switch case for menu options  switch (choice) {  // Finding an artwork  case 1:  System.out.print("\nEnter the ID of the artwork:");  String id = scanner.next();  scanner.nextLine();  Artwork artwork = artworkManager.findArtwork(id);  if (artwork != null) {  System.out.println(artwork.toText());  } else {  System.out.println("Artwork not found.");  }  break;  // Adding an artwork  case 2:  // Prompt the user to enter a new artwork ID and validate it  String newId;  while (true) {  System.out.print("\nEnter the ID of the artwork (4 digits):");  newId = scanner.next();  scanner.nextLine();  if (newId.length() != 4 || !newId.matches("\\d+")) {  System.out.println("Please enter a valid 4-digit number.");  } else if (artworkManager.findArtworkById(newId) != null) {  System.out.println("An artwork with that ID already exists.");  } else {  break;  }  }    // Prompt the user to enter the title of the artwork  System.out.print("Enter the title of the artwork:");  String title = scanner.nextLine();    // Prompt the user to enter the artist of the artwork and validate it  System.out.print("Enter the artist of the artwork:");  String artist = scanner.nextLine();  boolean artistFound = false;  try (Scanner artistFileScanner = new Scanner(new File("artist.txt"))) {  while (artistFileScanner.hasNextLine()) {  String line = artistFileScanner.nextLine();  String[] parts = line.split("\\|");  if (parts.length >= 2 && parts[1].equalsIgnoreCase(artist)) {  artistFound = true;  break;  }  }  } catch (FileNotFoundException e) {  System.out.println("Error: artist.txt file not found");  }    // If the artist is not found in the artist.txt file, return to the main menu  if (!artistFound) {  System.out.println("The artist is not found in the artist.txt file. Please add the artist first.");  Main.main(null);  return;  }    // Prompt the user to enter the date purchased of the artwork and validate it  System.out.print("Enter the date purchased (yyyy-mm-dd) of the artwork:");  String datePurchased = scanner.nextLine();  if (!datePurchased.matches("\\d{4}-\\d{2}-\\d{2}")) {  System.out.println("Invalid date format. Please enter the date in the format yyyy-mm-dd.");  scanner.nextLine();  }    // Prompt the user to enter the date sold of the artwork and validate it  System.out.print("Enter the date sold (yyyy-mm-dd) of the artwork (press enter if not sold yet):");  String dateSold = scanner.nextLine();  if (!dateSold.isEmpty() && !dateSold.matches("\\d{4}-\\d{2}-\\d{2}")) {  System.out.println("Invalid date format. Please enter the date in the format yyyy-mm-dd or press enter if not sold yet.");  scanner.nextLine(); // clear the buffer  }    // Prompt the user to enter the purchase price of the artwork and validate it  System.out.print("Enter the purchase price of the artwork:");  double purchasePrice;  while (true) {  String purchasePriceStr = scanner.nextLine();  if (purchasePriceStr.matches("\\d+")) {  purchasePrice = Double.parseDouble(purchasePriceStr);  break;  }  System.out.println("Invalid input. Please enter only digits.");  }    // Prompt the user to enter the selling price of the artwork and validate it  System.out.print("Enter the selling price of the artwork (press enter if not sold yet):");  double sellingPrice = 0.0;  while (true) {  String sellingPriceStr = scanner.nextLine();  if (sellingPriceStr.isEmpty()) {  break;  }  if (sellingPriceStr.matches("\\d+")) {  sellingPrice = Double.parseDouble(sellingPriceStr);  break;  }  System.out.println("Invalid input. Please enter only digits or press enter if not sold yet.");  }  System.out.print("Enter the status of the artwork (sold/available):");  String status;  while (true) {  status = scanner.nextLine();  if (status.equals("Sold") || status.equals("Available")) {  break;  }  System.out.println("Invalid input. Please enter either 'sold' or 'available'.");  }  Artwork newArtwork = new Artwork(newId, title, artist, datePurchased, dateSold, purchasePrice, sellingPrice, status);  artworkManager.addArtwork(newArtwork);  System.out.println("Artwork added successfully.");  break;  // Modifying an artwork  case 3:  // Find the artwork to be modified by its ID  System.out.print("\nEnter the ID of the artwork:");  String modifyId = scanner.next();  scanner.nextLine();  Artwork artwork1 = artworkManager.findArtwork(modifyId);  if (artwork1 != null) {  // Print the current information of the artwork  System.out.println("\nArtwork information:");  System.out.println("ID: " + artwork1.getId());  System.out.println("Name: " + artwork1.getTitle());  System.out.println("Artist: " + artwork1.getArtist());  System.out.println("Purchased Date: " + artwork1.getDatePurchased());  System.out.println("Sold Date: " + artwork1.getDateSold());  System.out.println("Purchased Price: " + artwork1.getPurchasePrice());  System.out.println("Sold Price: " + artwork1.getSellingPrice());  System.out.println("Status: " + artwork1.getStatus());  String newTitle = "";  while (newTitle.equals("")) {  // Prompt the user to enter the new title of the artwork  System.out.print("\nEnter the new title of the artwork:");  newTitle = scanner.nextLine();  }  artwork1.setTitle(newTitle);  // Prompt the user to enter the new artist of the artwork  System.out.print("\nEnter the new artist of the artwork:");  String newArtist = scanner.nextLine();  boolean artistFound1 = false;  try (Scanner artistFileScanner = new Scanner(new File("artist.txt"))) {  // Check if the new artist is in the artist.txt file  while (artistFileScanner.hasNextLine()) {  String line = artistFileScanner.nextLine();  String[] parts = line.split("\\|");  if (parts.length >= 2 && parts[1].equalsIgnoreCase(newArtist)) {  artistFound1 = true;  break;  }  }  } catch (FileNotFoundException e) {  System.out.println("Error: artist.txt file not found");  }  //If the new artist is not in the artist.txt file, prompt the user to add the artist and return to the main menu  if (!artistFound1) {  System.out.println("The artist is not found in the artist.txt file. Please add the artist first.");  // Return to the main menu  Main.main(null);  return;  }  artwork1.setArtist(newArtist);  String newPurchasedDate = "";  while (true) {  // Prompt the user to enter the new purchased date of the artwork  System.out.print("\nEnter the new purchased date of the artwork (yyyy-MM-dd): ");  newPurchasedDate = scanner.nextLine();  // Check if the input date is valid  if (newPurchasedDate.equals("N/A") || newPurchasedDate.matches("\\d{4}-\\d{2}-\\d{2}")) {  break;  } else {  System.out.println("Error: Invalid input format. Please enter a valid date in the format yyyy-MM-dd or N/A.");  }  }  artwork1.setDatePurchased(newPurchasedDate);  String newSoldDate = "";  while (true) {  // Prompt the user to enter the new sold date of the artwork  System.out.print("\nEnter the new sold date of the artwork (yyyy-MM-dd or N/A, press Enter to skip): ");  newSoldDate = scanner.nextLine();  if (newSoldDate.equals("") || newSoldDate.equals("N/A") || newSoldDate.matches("\\d{4}-\\d{2}-\\d{2}")) {  break;  } else {  System.out.println("Error: Invalid input format. Please enter a valid date in the format yyyy-MM-dd or N/A, or press Enter to skip.");  }  }  artwork1.setDateSold(newSoldDate);  double newPurchasePrice = 0.0;  while (newPurchasePrice == 0.0) {  System.out.print("\nEnter the new purchase price of the artwork:");  try {  newPurchasePrice = Double.parseDouble(scanner.nextLine());  } catch (NumberFormatException e) {  System.out.println("Invalid input. Please enter a valid number.");  }  }  artwork1.setPurchasePrice(newPurchasePrice);  double newSellingPrice = -1.0;  while (newSellingPrice < 0) {  System.out.print("\nEnter the new selling price of the artwork (Press enter to skip):");  String input = scanner.nextLine();  if (input.equals("")) {  newSellingPrice = 0.0;  } else {  try {  newSellingPrice = Double.parseDouble(input);  if (newSellingPrice < 0) {  System.out.println("Error: Selling price cannot be negative.");  }  } catch (NumberFormatException e) {  System.out.println("Invalid input. Please enter a valid number.");  }  }  }  artwork1.setSellingPrice(newSellingPrice);  String newStatus = "";  while (!newStatus.equalsIgnoreCase("Sold") && !newStatus.equalsIgnoreCase("Available")) {  System.out.print("\nEnter the new status of the artwork (Sold or Available):");  newStatus = scanner.nextLine();  if (!newStatus.equalsIgnoreCase("Sold") && !newStatus.equalsIgnoreCase("Available")) {  System.out.println("Error: Invalid input. Please enter either Sold or Available.");  }  }  artwork1.setStatus(newStatus);  artworkManager.modifyArtwork(artwork1);  System.out.println("Artwork modified successfully.");  } else {  System.out.println("Artwork not found.");  }  break;  // Deleting an artwork  case 4:  System.out.print("\nEnter the ID of the artwork to delete:");  String deleteId = scanner.next();  scanner.nextLine();  if (artworkManager.deleteArtwork(deleteId)) {  System.out.println("Artwork deleted successfully.");  } else {  System.out.println("Artwork not found.");  }  break;  // Exiting the program  case 5:  System.out.println("\nExiting the artwork program...");  Main.main(null);  break;  default:  System.out.println("wrong option. Only 1-6 is allowed");  }  }  }  } |

**ArtworkManager.java**

|  |
| --- |
| package group\_2;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.ArrayList;  import java.io.FileNotFoundException;  //This class manages a collection of artworks  public class ArtworkManager {  private ArrayList<Artwork> artworks; // the collection of artworks  private String filename; // the name of the file to read/write artworks from/to  // Constructor for ArtworkManager  public ArtworkManager(String filename) {  artworks = new ArrayList<>(); // Initialize the artworks array list  this.filename = filename;  readFromFile();// Read artworks from file  }  // Adds an artwork to the collection  public void addArtwork(Artwork artwork) {  artworks.add(artwork);  writeToFile();  }  // Deletes an artwork from the collection by ID  public boolean deleteArtwork(String id) {  for (Artwork artwork : artworks) {  if (artwork.getId().equals(id)) {  artworks.remove(artwork);  writeToFile();  return true;  }  }  return false;  }  // Modifies an artwork in the collection  public void modifyArtwork(Artwork artwork) {  for (int i = 0; i < artworks.size(); i++) {  if (artworks.get(i).getId() == artwork.getId()) {  artworks.set(i, artwork);  writeToFile();  return;  }  }  }  public Artwork findArtwork(String id) {  for (Artwork artwork : artworks) {  if (artwork.getId().equals(id)) {  return artwork;  }  }  return null;  }  // Finds an artwork in the collection by ID  public Artwork findArtworkById(String id) {  for (Artwork artwork : listArtwork()) {  if (artwork.getId().equalsIgnoreCase(id)) {  return artwork;  }  }  return null; // Return null if the artwork is not found  }  // Returns the entire collection of artworks  public ArrayList<Artwork> listArtwork() {  return artworks;  }  // Reads the artworks from the file  private void readFromFile() {  try {  BufferedReader reader = new BufferedReader(new FileReader(filename));  String line = reader.readLine();  while (line != null) {  if (!line.isEmpty()) {  Artwork artwork = Artwork.fromText(line);  artworks.add(artwork);  } else {  System.out.println("Skipping empty line in input file");  }  line = reader.readLine();  }  reader.close();  } catch (FileNotFoundException e) {  System.out.println("File not found: " + filename);  } catch (IOException e) {  System.out.println("Error reading file: " + filename);  e.printStackTrace();  } catch (IllegalArgumentException e) {  System.out.println("Error parsing input data in file: " + filename);  e.printStackTrace();  }  }  private void writeToFile() {  try {  BufferedWriter writer = new BufferedWriter(new FileWriter(filename));  for (Artwork artwork : artworks) {  writer.write(artwork.toText() + "\n");  }  writer.close();  } catch (IOException e) {  e.printStackTrace();  }  }  }  class Artwork {  private String id;  private String title;  private String artist;  private String datePurchased;  private String dateSold;  private double purchasePrice;  private double sellingPrice;  private String status;  public Artwork(String id, String title, String artist, String datePurchased, String dateSold, double purchasePrice,  double sellingPrice, String status) {  this.id = id;  this.title = title;  this.artist = artist;  this.datePurchased = datePurchased;  this.dateSold = dateSold;  this.purchasePrice = purchasePrice;  this.sellingPrice = sellingPrice;  this.status = status;  }  //mutator & getter  public String getId() {  return id;  }  public String getTitle() {  return title;  }  public String getArtist() {  return artist;  }  public String getDatePurchased() {  return datePurchased;  }  public String getDateSold() {  return dateSold;  }  public double getPurchasePrice() {  return purchasePrice;  }  public double getSellingPrice() {  return sellingPrice;  }  public String getStatus() {  return status;  }  public void setID(String id) {  this.id = id;  }    public void setStatus(String status) {  this.status = status;  }    public void setTitle(String title) {  this.title = title;  }    public void setArtist(String artist) {  this.artist = artist;  }    public void setDatePurchased(String datePurchased) {  this.datePurchased = datePurchased;  }    public void setDateSold(String dateSold) {  this.dateSold = dateSold;  }    public void setPurchasePrice(double purchasePrice) {  this.purchasePrice = purchasePrice;  }    public void setSellingPrice(double sellingPrice) {  this.sellingPrice = sellingPrice;  }  public static Artwork fromText(String line) {  if (line.isEmpty()) {  throw new IllegalArgumentException("Input string is empty");  }  String[] tokens = line.split("\\|");  if (tokens.length < 8) {  throw new IllegalArgumentException("Input string does not contain the expected number of tokens: " + line);  }  String id = tokens[0];  String title = tokens[1];  String artist = tokens[2];  String datePurchased = tokens[3];  String dateSold = tokens[4];  double purchasePrice = 0.0;  double sellingPrice = 0.0;  if (!tokens[5].isEmpty()) {  try {  purchasePrice = Double.parseDouble(tokens[5]);  } catch (NumberFormatException e) {  }  }  if (!tokens[6].isEmpty()) {  try {  sellingPrice = Double.parseDouble(tokens[6]);  } catch (NumberFormatException e) {  }  }  String status = tokens[7];  return new Artwork(id, title, artist, datePurchased, dateSold, purchasePrice, sellingPrice, status);  }  public String toText() {  return String.format("%s|%s|%s|%s|%s|%.2f|%.2f|%s", id, title, artist, datePurchased, dateSold, purchasePrice,  sellingPrice, status);  }  } |

**ArtistInformation.java**

|  |
| --- |
| package group\_2;  import java.io.BufferedWriter;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.ArrayList;  import java.util.Scanner;  public class ArtistInformation {  // Instance variables  private String artistID;  private String artistName;  private String specialty;  private String status;  private String priceRange;  private static ArrayList<ArtistInformation> artistList = new ArrayList<>();    // Accessor methods  public String getArtistID() {  return artistID;  }    public String getArtistName() {  return artistName;  }  public String getSpecialty() {  return specialty;  }    public String getStatus() {  return status;  }    public String getPriceRange() {  return priceRange;  }    public ArrayList<ArtistInformation> getList() {  return artistList;  }    // Mutator methods  public void setArtistID(String artistID) {  this.artistID = artistID;  }    public void setArtistName(String artistName) {  this.artistName = artistName;  }    public void setSpecialty(String specialty) {  this.specialty = specialty;  }    public void setStatus(String status) {  this.status = status;  }    public void setPriceRange(String priceRange) {  this.priceRange = priceRange;  }    // Constructors  public ArtistInformation(String artistID, String artistName, String specialty, String status, String priceRange) {  this.artistID = artistID;  this.artistName = artistName;  this.specialty = specialty;  this.status = status;  this.priceRange = priceRange;  }    public ArtistInformation() {    }    // Read artist data text file method  public void readArtistFile() {  // Read the file and  try {  // Create a new file object  FileReader fileReader = new FileReader ("artist.txt");  BufferedReader bufferedReader = new BufferedReader(fileReader);    // Read each line of the file and add it to the ArrayList  String line;  while ((line = bufferedReader.readLine()) != null) {  String[] fields = line.split("\\|");  String artistID = fields[0];  String artistName = fields[1];  String specialty = fields[2];  String status = fields[3];  String priceRange = fields[4];    ArtistInformation artistInfo = new ArtistInformation(artistID, artistName, specialty, status, priceRange);  artistList.add(artistInfo);  }  // Close the BufferedWriter and FileWriter  bufferedReader.close();  fileReader.close();  }  catch(IOException e) {  System.out.println("An error has occurred.");  e.printStackTrace();  }  }    // Write artist data text file method  public void writeFile() {  try {  // Create a new FileWriter object with the file path as parameter  FileWriter fileWriter = new FileWriter("artist.txt");    // Create a new BufferedWriter object to write data to the file  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);    // Write data to the file  for (ArtistInformation artistInformation : artistList) {  String line = String.format("%s|%s|%s|%s|%s",  artistInformation.getArtistID(),  artistInformation.getArtistName(),  artistInformation.getSpecialty(),  artistInformation.getStatus(),  artistInformation.getPriceRange());  bufferedWriter.write(line);  bufferedWriter.newLine(); // add new line character  }    // Close the BufferedWriter and FileWriter  bufferedWriter.close();  fileWriter.close();  } catch (IOException e) {  e.printStackTrace();  }  }  // Create add artist method  public static void addArtist() {  Scanner input = new Scanner(System.in);    System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$44s%n", "ADD ARTIST");  System.out.println("--------------------------------------------------------------------------------------");    // Create array for storing only artist ID  String[] idList = new String[artistList.size()];  for (int i = 0; i < artistList.size(); i++) {  idList[i] = artistList.get(i).getArtistID();  }    // Prompt user to input customer ID  System.out.print("Enter artist ID : ");  String id = input.next();  // Validate the customer ID preventing users from entering an existing ID  boolean idExists = true;  while (idExists) {  for (int j = 0; j < idList.length; j++) {  if (idList[j].equals(id)) {  System.out.println("The ID already exists. Please use another ID.");  System.out.print("Enter artist ID : ");  id = input.next();  break; // Break out of the for loop so we don't keep checking the same ID  }  if (j == idList.length - 1) {  idExists = false; // Exit the while loop because the ID doesn't exist in the array  }  }  }  while (true)  {  if (id.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter artist ID : ");  id = input.next();  }  }  // Prompt user to input artist name, specialty, artist status, price range  System.out.print("Enter artist name : ");  input.nextLine();  String name = input.nextLine();    System.out.print("Enter artist specialty : ");  String specialty = input.next();    System.out.print("Enter artist status : ");  String status = input.next();    // Validate artist status to be either 'Alive' or 'Deceased' only  boolean loop = true;  while(loop) {  if (status.equals("Alive") || status.equals("Deceased")) {  break;  }  else {  System.out.println("Invalid input. Please enter either 'Alive' or 'Deceased'.");  System.out.print("Enter artist status : ");  status = input.next();  }  }  System.out.print("Enter artist price range : ");  String priceRange = input.next();    // Double confirm on user input  while (true) {  System.out.println("\nPlease check your newly added artist data.");  System.out.println("The artist ID : " + id);  System.out.println("The artist name : " + name);  System.out.println("The artist specialty : " + specialty);  System.out.println("The artist status : " + status);  System.out.println("The artist price range : " + priceRange);    System.out.println("\nAre you sure you want to add?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  // Validate option so that user enters Y, y, N, or n only. Error message otherwise.  if (option == 'Y' || option == 'y') {  ArtistInformation artistInformation = new ArtistInformation(id, name, specialty, status, priceRange);  artistList.add(artistInformation);    artistInformation.writeFile();  System.out.println("\nThe artist " + id + " has been added successfully.");  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe add artist action is cancelled.");  break;  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");    }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }  // Create delete artist method  public static void deleteArtist() {  Scanner input = new Scanner(System.in);    System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "DELETE ARTIST");  System.out.println("--------------------------------------------------------------------------------------");    // Create array for storing only artist ID  String[] idList = new String[artistList.size()];  for (int i = 0; i < artistList.size(); i++) {  idList[i] = artistList.get(i).getArtistID();  }  // Prompt user to input artist ID  System.out.print("Enter artist ID : ");  String id = input.next();  // Validate the artist ID and double confirm on user input  boolean found = true;  boolean idExists = true;  while (idExists) {  for (int j = 0; j < idList.length; j++) {  if (idList[j].equals(id)) {  System.out.println("\nThe ID " + id + " is in the artist list.");  System.out.println("The artist ID : " + id );  System.out.println("The artist name : " + artistList.get(j).getArtistName());  System.out.println("The artist specialty : " + artistList.get(j).getSpecialty());  System.out.println("The artist status : " + artistList.get(j).getStatus());  System.out.println("The artist price range : " + artistList.get(j).getPriceRange());    System.out.println("\nAre you sure you want to delete?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  // Validate option so that user enters Y, y, N, or n only. Error message otherwise.  if(option == 'Y' || option == 'y' || option == 'N' || option == 'n') {  while(found){  if (option == 'Y' || option == 'y') {  artistList.remove(j); // Remove artist with the ID the user entered  ArtistInformation artistInformation = new ArtistInformation();  artistInformation.writeFile();    System.out.println("\nThe artist information of " + id + " was deleted successfully.");  idExists = false;  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe delete artist action is cancelled");  idExists = false;  break;  }  }  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");  System.out.print("Yes(Y) No(N) : ");  option = input.next().charAt(0);  }  break; // Break out of the for loop so we don't keep checking the same ID  }  // Validate that user enters an existing ID only.  if (j == idList.length - 1) {  System.out.println("The ID does not exist. Please use another ID.");  System.out.print("Enter artist ID : ");  id = input.next();  }  }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }  // Create modify artist method  public static void modifyArtist() {  Scanner input = new Scanner (System.in);  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "MODIFY ARTIST");  System.out.println("--------------------------------------------------------------------------------------");    // Prompt user to input artist ID  System.out.print("Enter artist ID : ");  String id = input.next();    // Validate the artist ID to make sure it exists  boolean idExists = true;  while (idExists) {  for (int i = 0; i < artistList.size(); i++) {  if (artistList.get(i).getArtistID().equals(id)) {  idExists = false;  System.out.println("\nThe ID " + id + " is in the artist list.");  System.out.println("The artist ID : " + id );  System.out.println("The artist name : " + artistList.get(i).getArtistName());  System.out.println("The artist specialty : " + artistList.get(i).getSpecialty());  System.out.println("The artist status : " + artistList.get(i).getStatus());  System.out.println("The artist price range : " + artistList.get(i).getPriceRange());    // Prompt user to input new artist info and validate the new artist info  System.out.println("\nPlease enter new artist info.");  System.out.print("Enter new ID : ");  String newID = input.next();    while (true) {  // Check if ID is 4 digits  if (newID.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter customer ID : ");  newID = input.next();  }  }    int index = 0;  String[] idList = new String[artistList.size()];  for (int j = 0; j < artistList.size(); j++) {  idList[j] = artistList.get(j).getArtistID();  }    boolean idNewExists = true;  while (idNewExists) {  if(id.equals(newID)) {  for (int l = 0; l < idList.length; l++) {  if (idList[l].equals(newID)) {  index = l;    break;  }  }  break;  }  for (int k = 0; k < idList.length; k++) {  if (idList[k].equals(newID) ) {  System.out.println("The ID already exists. Please use another ID.");  System.out.print("Enter new artist ID : ");  newID = input.next();    break; // Break out of the for loop so we don't keep checking the same ID  }  if (k == idList.length - 1) {  idNewExists = false; // Exit the while loop because the ID doesn't exist in the array  }  }  }  // Prompt user to input new name, specialty, and status.  System.out.print("Enter new name : ");  input.nextLine();  String newName = input.nextLine();  System.out.print("Enter new specialty : ");  String newSpecialty = input.next();  System.out.print("Enter new status : ");  String newStatus = input.next();    // Validate new artist status to be either 'Alive' or 'Deceased' only  boolean loop = true;  while(loop) {  if (newStatus.equals("Alive") || newStatus.equals("Deceased")) {  break;  }  else {  System.out.println("Invalid input. Please enter either 'Alive' or 'Deceased'.");  System.out.print("Enter new status : ");  newStatus = input.next();  }  }    // Prompt user to enter new price range  System.out.print("Enter new price range : ");  String newPriceRange = input.next();    while (true) {  // Displays details entered by user to double confirm  System.out.println("\nPlease check your newly modified artist data.");  System.out.println("The new ID : " + newID);  System.out.println("The new name : " + newName);  System.out.println("The new specialty : " + newSpecialty);  System.out.println("The new status : " + newStatus);  System.out.println("The new price range : " + newPriceRange);    // Prompt user to confirm the changes  System.out.println("\nAre you sure you want to modify?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  // Validate option so that user enters Y, y, N, or n only. Error message otherwise.  if (option == 'Y' || option == 'y') {  artistList.get(i).setArtistID(newID);  artistList.get(i).setArtistName(newName);  artistList.get(i).setSpecialty(newSpecialty);  artistList.get(i).setStatus(newStatus);  artistList.get(i).setPriceRange(newPriceRange);  ArtistInformation artistInfomation = new ArtistInformation();  artistInfomation.writeFile();  System.out.println("\nThe artist data has been modified successfully.");  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe modify artist action is cancelled.");  break;  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");  }  }  break;  }  }  if (idExists == true) {  System.out.println("The ID does not exist. Please enter a valid ID.");  System.out.print("Enter artist ID : ");  id = input.next();  }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }  // Create search artist method  public static void searchArtist() {  Scanner input = new Scanner (System.in);  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "SEARCH ARTIST");  System.out.println("--------------------------------------------------------------------------------------");  // Prompt user to input artist ID  System.out.print("Enter artist ID : ");  String id = input.next();  // Validate the customer ID and print the artist info user search  boolean idExists = true;  boolean found = false;  while (idExists) {  for (int i = 0; i < artistList.size(); i++) {  if (artistList.get(i).getArtistID().equals(id)) {    System.out.println("\nThe ID " + id + " is in the artist list.");  System.out.println("The artist ID : " + id );  System.out.println("The artist name : " + artistList.get(i).getArtistName());  System.out.println("The artist specialty : " + artistList.get(i).getSpecialty());  System.out.println("The artist status : " + artistList.get(i).getStatus());  System.out.println("The artist price range : " + artistList.get(i).getPriceRange());  found = true;  idExists = false;  break; // Exit the loop while customer info is found  }  }  if (!found) {  // Prompt the user for a valid artist ID, error message if ID doesn't exist  System.out.println("\nThe ID does not exist. Please enter a valid ID.");  System.out.print("Enter artist ID : ");  id = input.next();  }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }  } |

**CustomerInformation.java**

|  |
| --- |
| package group\_2;  import java.io.BufferedWriter;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.ArrayList;  import java.util.Scanner;  public class CustomerInformation {  // Instance variables  private String customerId;  private String customerName;  private String customerContact;  private String customerArtworkPurchased;  private String customerArtistPreferences;  private static ArrayList<CustomerInformation> customerList = new ArrayList<>();    // Accessor methods  public String getCustomerId() {  return customerId;  }    public String getCustomerName() {  return customerName;  }    public String getCustomerContact() {  return customerContact;  }    public String getCustomerArtworkPurchased() {  return customerArtworkPurchased;  }    public String getCustomerArtistPreferences() {  return customerArtistPreferences;  }    public ArrayList<CustomerInformation> getList() {  return customerList;  }    // Mutator methods  public void setCustomerId(String customerId) {  this.customerId = customerId;  }    public void setCustomerName(String name) {  this.customerName = name;  }    public void setCustomerContact(String contact) {  this.customerContact = contact;  }    public void setCustomerArtworkPurchased(String artworkPurchased) {  this.customerArtworkPurchased = artworkPurchased;  }    public void setCustomerArtistPreferences(String artistPreferences) {  this.customerArtistPreferences = artistPreferences;  }    // Constructors  public CustomerInformation(String id, String name, String contact, String artworkPurchased, String artistPreferences) {  this.customerId = id;  this.customerName = name;  this.customerContact = contact;  this.customerArtworkPurchased = artworkPurchased;  this.customerArtistPreferences = artistPreferences;  }  public CustomerInformation() {    }    // Read customer data text file method  public void readCustomerFile() {  try {  // Create a new file object  FileReader fileReader = new FileReader ("customer.txt");  BufferedReader bufferedReader = new BufferedReader(fileReader);    // Read each line of the file and add it to the ArrayList  String line;  while ((line = bufferedReader.readLine()) != null) {  String[] fields = line.split("\\|");  String customerId = fields[0];  String customerName = fields[1];  String customerContact = fields[2];  String customerArtworkPurchased = fields[3];  String customerArtistPreferences = fields[4];    CustomerInformation customerInfo = new CustomerInformation(customerId, customerName, customerContact, customerArtworkPurchased, customerArtistPreferences);  customerList.add(customerInfo);  }    // Close the BufferedWriter and FileWriter  bufferedReader.close();  fileReader.close();  } catch(IOException e) {  e.printStackTrace();  }  }    // Write customer data text file method  public void writeFile() {  try {  // Create a new FileWriter object with the file path as parameter  FileWriter fileWriter = new FileWriter("customer.txt");    // Create a new BufferedWriter object to write data to the file  BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);    // Write data to the file  for (CustomerInformation customerInformation : customerList) {  String line = String.format("%s|%s|%s|%s|%s",  customerInformation.getCustomerId(),  customerInformation.getCustomerName(),  customerInformation.getCustomerContact(),  customerInformation.getCustomerArtworkPurchased(),  customerInformation.getCustomerArtistPreferences());  bufferedWriter.write(line);  bufferedWriter.newLine(); // add new line character  }    // Close the BufferedWriter and FileWriter  bufferedWriter.close();  fileWriter.close();  } catch (IOException e){  e.printStackTrace();  }  }  // Create add customer method  public static void addCustomer() {  Scanner input = new Scanner(System.in);  // Print the user interface  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "ADD CUSTOMER");  System.out.println("--------------------------------------------------------------------------------------");    // Create array for storing only customer ID  String[] idList = new String[customerList.size()];  for (int i = 0; i < customerList.size(); i++)  {  idList[i] = customerList.get(i).getCustomerId();  }  // Prompt user to input customer ID  System.out.print("Enter customer ID : ");  String id = input.next();  // Check the id is in 4 digit number  while (true)  {  if (id.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }    // Validate the customer ID  boolean idExists = true;  while (idExists) {  for (int j = 0; j < idList.length; j++) {  if (idList[j].equals(id)) {  System.out.println("The ID already exists. Please use another ID.");  System.out.print("Enter customer ID : ");  id = input.next();  while (true)  {  if (id.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }  break; // Break out of the for loop so we don't keep checking the same ID  }  if (j == idList.length - 1) {  idExists = false; // Exit the while loop because the ID doesn't exist in the array  }  }  }  // Prompt user to input customer name, contact  System.out.print("Enter customer name : ");  input.nextLine();  String name = input.nextLine();  System.out.print("Enter customer contact : ");  String contact = input.next();    // Prompt user to input customer artwork purchased ID  System.out.print("Enter artwork purchased ID : ");  String artworkPurchased = input.next();  // Validate the customer artwork purchased ID  ArtworkManager artworkManager = new ArtworkManager("inventory.txt");  ArrayList<Artwork> artworks = artworkManager.listArtwork();  String[] artworkIdList = new String[artworks.size()];  String[] artworkStatusList = new String[artworks.size()];  for (int i = 0; i < artworks.size(); i++) {  artworkIdList[i] = artworks.get(i).getId();  artworkStatusList[i] = artworks.get(i).getStatus();  }  boolean artworkIdExists = true;  boolean artworkAvailable = true;  while (artworkIdExists && artworkAvailable) {  boolean found = false;  for (int j = 0; j < artworkIdList.length; j++) {  if (artworkIdList[j].equals(artworkPurchased)) {  found = true;  if (artworkStatusList[j].equals("Sold")) {  System.out.println("The artwork already sold. Please enter another ID.");  System.out.print("Enter artwork ID : ");  artworkPurchased = input.next();  break;  }  else if (artworkStatusList[j].equals("Available")) {  artworkIdExists = false;  artworkAvailable = false;  break;  }  }  }  if (!found) {  System.out.println("Artwork ID not found. Exiting to Main Menu.");  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  artworkIdExists = false;  artworkAvailable = false;    Main.main(null);    }  }    // Prompt user to input customer artist preferences ID  System.out.print("Enter artist preferences ID : ");  String artistPreferences = input.next();  // Validate the customer artist preferences ID  ArtistInformation artistInformation = new ArtistInformation();  artistInformation.readArtistFile();  ArrayList<ArtistInformation> artists = artistInformation.getList();  String[] artistIdList = new String[artists.size()];  for (int i = 0; i < artists.size(); i++) {  artistIdList[i] = artists.get(i).getArtistID();  }    boolean artistExists = true;  while (artistExists) {  boolean found = false;  for (int j = 0; j < artistIdList.length; j++) {    if (artistIdList[j].equals(artistPreferences)) {  found = true;  System.out.println("The artist ID is exists.");  artistExists = false;  break; // Break out of the for loop so we don't keep checking the same ID  }  }  if (!found) {  System.out.println("Artist ID not found. Exiting method to Main Menu.");  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  artistExists = false;  Main.main(null);  }  }    // Double confirm on user input  while (true) {  System.out.println("\nPlease check your newly added customer data.");  System.out.println("The customer ID : " + id);  System.out.println("The customer name : " + name);  System.out.println("The customer contact : " + contact);  System.out.println("The customer artwork purchased ID : " + artworkPurchased);  System.out.println("The customer artist preferences ID : " + artistPreferences);    System.out.println("\nAre you sure you want to add?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  if (option == 'Y' || option == 'y') {  CustomerInformation customerInformation = new CustomerInformation(id, name, contact, artworkPurchased, artistPreferences);  customerList.add(customerInformation);    customerInformation.writeFile();  System.out.println("\nThe customer " + id + " has been added successfully.");  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe add customer action is cancelled.");  break;  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");    }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }    // Create delete customer method  public static void deleteCustomer() {  Scanner input = new Scanner(System.in);  // Print the user interface  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "DELETE CUSTOMER");  System.out.println("--------------------------------------------------------------------------------------");    // Create array for storing only customer ID  String[] idList = new String[customerList.size()];  for (int i = 0; i < customerList.size(); i++) {  idList[i] = customerList.get(i).getCustomerId();  }  // Prompt user to input customer ID  System.out.print("Enter customer ID : ");  String id = input.next();  // Validate the customer ID and double confirm on user input  boolean found = true;  boolean idExists = true;  while (idExists) {  for (int j = 0; j < idList.length; j++) {  if (idList[j].equals(id)) {  System.out.println("\nThe ID " + id + " is in the customer list.");  System.out.println("The customer ID : " + id );  System.out.println("The customer name : " + customerList.get(j).getCustomerName());  System.out.println("The customer contact : " + customerList.get(j).getCustomerContact());  System.out.println("The artwork puchased ID : " + customerList.get(j).getCustomerArtworkPurchased());  System.out.println("The artist preferences ID : " + customerList.get(j).getCustomerArtistPreferences());    System.out.println("\nAre you sure you want to delete?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  if(option == 'Y' || option == 'y' || option == 'N' || option == 'n') {  while(found) {  if (option == 'Y' || option == 'y') {  customerList.remove(j);  CustomerInformation customerInfomation = new CustomerInformation();  customerInfomation.writeFile();    System.out.println("\nThe customer information of " + id + " was deleted successfully.");  idExists = false;  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe delete customer action is cancelled");  idExists = false;  break;  }  }  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");  System.out.print("Yes(Y) No(N) : ");  option = input.next().charAt(0);  }  break; // Break out of the for loop so we don't keep checking the same ID  }  if (j == idList.length - 1) {  System.out.println("The ID does not exist. Please use another ID.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }    // Create modify customer method  public static void modifyCustomer() {  Scanner input = new Scanner (System.in);  // Print the user interface  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "MODIFY CUSTOMER");  System.out.println("--------------------------------------------------------------------------------------");    // Prompt user to input customer ID  System.out.print("Enter customer ID : ");  String id = input.next();  // Validate the customer ID into 4 digit  while (true)  {  if (id.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }    // Validate the customer ID  boolean idExists = true;  while (idExists) {  for (int i = 0; i < customerList.size(); i++) {  if (customerList.get(i).getCustomerId().equals(id)) {  idExists = false;  System.out.println("\nThe ID " + id + " is in the customer list.");  System.out.println("The customer ID : " + id );  System.out.println("The customer name : " + customerList.get(i).getCustomerName());  System.out.println("The customer contact : " + customerList.get(i).getCustomerContact());  System.out.println("The artwork puchased ID : " + customerList.get(i).getCustomerArtworkPurchased());  System.out.println("The artist preferences ID : " + customerList.get(i).getCustomerArtistPreferences());    // Prompt user to input new customer info and validate the new customer info  System.out.println("\nPlease enter new customer info.");  System.out.print("Enter new ID : ");  String newID = input.next();  // Check the new ID is 4 digit number  while (true)  {  if (newID.matches("\\d{4}")) {  break;  } else {  System.out.println("Invalid input. Please enter a 4 digit number.");  System.out.print("Enter customer ID : ");  newID = input.next();  }  }    int index = 0;  String[] idList = new String[customerList.size()];  String[] artworkPurchasedList = new String[customerList.size()];  for (int j = 0; j < customerList.size(); j++) {  idList[j] = customerList.get(j).getCustomerId();  artworkPurchasedList[j] = customerList.get(j).getCustomerArtworkPurchased();  }    // Validate the customer ID  boolean idNewExists = true;  while (idNewExists) {  if(id.equals(newID)) {  for (int l = 0; l < idList.length; l++) {  if (idList[l].equals(newID)) {  index = l;  break;  }  }  break;  }  for (int k = 0; k < idList.length; k++) {  if (idList[k].equals(newID) ) {  System.out.println("The ID already exists. Please use another ID.");  System.out.print("Enter new customer ID : ");  newID = input.next();    break; // Break out of the for loop so we don't keep checking the same ID  }  if (k == idList.length - 1) {  idNewExists = false; // Exit the while loop because the ID doesn't exist in the array  }  }  }  // Prompt use to input the name, contact and artwork purchased ID  System.out.print("Enter new name : ");  input.nextLine();  String newName = input.nextLine();  System.out.print("Enter new contact : ");  String newContact = input.next();    System.out.print("Enter new artwork purchased : ");  String newArtworkPurchased = input.next();    // Validate the artwork purchased ID  ArtworkManager artworkManager = new ArtworkManager("inventory.txt");  ArrayList<Artwork> artworks = artworkManager.listArtwork();  String[] artworkIdList = new String[artworks.size()];  String[] artworkStatusList = new String[artworks.size()];  for (int m = 0; m < artworks.size(); m++) {  artworkIdList[m] = artworks.get(m).getId();  artworkStatusList[m] = artworks.get(m).getStatus();  }    boolean artworkIdExists = true;  boolean artworkAvailable = true;  while (artworkIdExists == true && artworkAvailable == true) {  if (artworkPurchasedList[index].equals(newArtworkPurchased)) {  break;  }    boolean found = false;  for (int j = 0; j < artworkIdList.length; j++) {  if (artworkIdList[j].equals(newArtworkPurchased)) {  found = true;  if (artworkStatusList[j].equals("Sold")) {  System.out.println("The artwork is already sold. Please enter another ID.");  System.out.print("Enter artwork ID : ");  newArtworkPurchased = input.next();  break;  }  else if (artworkStatusList[j].equals("Available")) {  artworkIdExists = false;  artworkAvailable = false;  break;  }  }  }  if (!found) {  break;  }  }    // Prompt the user to input the artist preferences ID  System.out.print("Enter new artist preferences : ");  String newArtistPreferences = input.next();    // Validate the artist preferences  ArtistInformation artistInformation = new ArtistInformation();  artistInformation.readArtistFile();  ArrayList<ArtistInformation> artists = artistInformation.getList();  String[] artistIdList = new String[artists.size()];  for (int m = 0; m < artists.size(); m++) {  artistIdList[m] = artists.get(m).getArtistID();  }    boolean artistExists = true;  while (artistExists) {  boolean found = false;  for (int j = 0; j < artistIdList.length; j++) {  if (artistIdList[j].equals(newArtistPreferences)) {  found = true;  System.out.println("The artist ID is exists.");  artistExists = false;  break; // Break out of the for loop so we don't keep checking the same ID  }  }  if (!found) {  System.out.println("Artist ID not found. Exiting to Main Menu.");  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  artistExists = false;  Main.main(null);  }  }    // Double confirm on user input  while (true) {  System.out.println("\nPlease check your newly modified customer data.");  System.out.println("The new ID : " + newID);  System.out.println("The new name : " + newName);  System.out.println("The new contact : " + newContact);  System.out.println("The new artwork purchased : " + newArtworkPurchased);  System.out.println("The new artist preferences : " + newArtistPreferences);    System.out.println("\nAre you sure you want to modify?");  System.out.print("Yes(Y) No(N) : ");  char option = input.next().charAt(0);  if (option == 'Y' || option == 'y') {  customerList.get(i).setCustomerId(newID);  customerList.get(i).setCustomerName(newName);  customerList.get(i).setCustomerContact(newContact);  customerList.get(i).setCustomerArtworkPurchased(newArtworkPurchased);  customerList.get(i).setCustomerArtistPreferences(newArtistPreferences);  CustomerInformation customerInfomation = new CustomerInformation();  customerInfomation.writeFile();  System.out.println("\nThe customer data has been modified successfully.");  break;  }  else if (option == 'N' || option == 'n') {  System.out.println("\nThe modify customer action is cancelled.");  break;  }  else {  System.out.println("Wrong option. Only Y and N is allowed.");  }  }  break;  }  }  if (idExists == true) {  // Prompt the user for a valid customer id  System.out.println("The ID does not exist. Please enter a valid ID.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }  System.out.println("Press any key to continue.....");  input.nextLine();  }    // Create search customer method  public static void searchCustomer() {  Scanner input = new Scanner (System.in);  System.out.println("--------------------------------------------------------------------------------------");  System.out.printf("%1$50s%n", "SEARCH CUSTOMER");  System.out.println("--------------------------------------------------------------------------------------");    // Prompt user to input customer ID  System.out.print("Enter customer ID : ");  String id = input.next();  // Print the customer info user search  boolean idExists = true;  boolean found = false;  while (idExists) {  for (int i = 0; i < customerList.size(); i++) {  if (customerList.get(i).getCustomerId().equals(id)) {  System.out.println("\nThe ID " + id + " is in the customer list.");  System.out.println("The customer ID : " + id );  System.out.println("The customer name : " + customerList.get(i).getCustomerName());  System.out.println("The customer contact : " + customerList.get(i).getCustomerContact());  System.out.println("The artwork puchased ID : " + customerList.get(i).getCustomerArtworkPurchased());  System.out.println("The artist preferences ID : " + customerList.get(i).getCustomerArtistPreferences());    found = true;  idExists = false;  break; // Exit the loop while customer info is found  }  }  if (!found) {  // Prompt the user for a valid customer ID if the ID doesn't exist in the database  System.out.println("\nThe ID does not exist. Please enter a valid ID.");  System.out.print("Enter customer ID : ");  id = input.next();  }  }  System.out.println("Press any key to continue...");  input.nextLine();  input.nextLine();  }  } |

**Sample Input Data**

|  |
| --- |
| customer.txt - customer information |
| 1898|Wong Xiao Hua|012564815|9786|6248  1567|Wong Xiao Lee|0152854645|7591|5367  1344|Wong Xiao Zheng|0123456451|7182|3067  4561|Wong Xiao Mei|0123456764|1945|2905  1564|Wong Xiao Ting|0126786164|6038|1093 |

|  |
| --- |
| artist.txt - artist information |
| 8742|Frida Kahlo|Painting|Deceased|150000-200000  2190|Ai Wei Wei|Sculpture|Alive|75000-100000  3067|David Hockney|Drawing|Alive|40000-60000  4512|Yayoi Kusama|Printmaking|Alive|80000-120000  6178|Basquiat|Collage|Deceased|100000-150000  2905|Tracey Emin|Digital Art|Alive|30000-50000  1294|Annie Leibovitz|Photography|Alive|50000-75000  9250|Matthew Barney|VideoArt|Alive|60000-80000  7485|Marina Abramovic|Performance Art|Alive|90000-120000  5367|Jean Michel Basquiat|Painting|Deceased|200000-250000  3572|Damien Hirst|Sculpture|Alive|150000-180000  1093|Gerhard Richter|Drawing|Alive|70000-90000  6248|Takashi Murakami|Printmaking|Alive|100000-130000  9637|JR|Collage|Alive|40000-60000  2389|Olafur Eliasson|Digital Art|Alive|80000-100000 |

|  |
| --- |
| inventory.txt - artwork information |
| 3956|The Broken Silence|Marina Abramovic|2023-04-05| |4500.00| |Available  6038|Fragmented Truth|Gerhard Richter|2023-04-06|2023-04-08|3200.00|4800.00|Sold  7654|The Isolated Vision|Takashi Murakami|2023-04-07| |8000.00| |Available  1945|Twisted Reality|Tracey Emin|2023-04-08|2023-04-12|1800.00|2700.00|Sold  3871|Digital Echoes|Olafur Eliasson|2023-04-10| |5500.00| |Available  5739|The Fragile Touch|Yayoi Kusama|2023-04-12| |3800.00| |Available  7182|The Mysterious Gaze|David Hockney|2023-04-13|2023-04-15|2800.00|4200.00|Sold  8314|Undefined Boundaries|Basquiat|2023-04-15| |7000.00| |Available  4653|Illusionary Memories|Annie Leibovitz|2023-04-16| |4200.00| |Available  2022|The Shattered Dream|Damien Hirst|2023-04-18| |9500.00| |Available  5118|Untold Story|JR|2023-04-20| |3000.00| |Available  7591|Enigmatic Connection|Jean Michel Basquiat|2023-04-22|2023-04-25|7500.00|11250.00|Sold  6724|The Interwoven Emotions|Marina Abramovic|2023-04-23| |6200.00| |Available  3012|Complex Realities|Gerhard Richter|2023-04-25| |3800.00| |Available  9786|The Layered Thoughts|Takashi Murakami|2023-04-27|2023-04 29|12000.00|18000.00|Sold |

**Sample Output Data**

**Text

Description automatically generated**

*Figure 1: Menu of the UX Gallery Program*

Figure 1 shows the main menu of the UX Gallery Program. Inputting ‘1’ leads to customer class, inputting ‘2’ leads to the artist class, inputting ‘3’ leads to the artwork class and inputting ‘4’ exits the application.

**1) CUSTOMER**

*Graphical user interface, text

Description automatically generated*

*Figure 1.1a: Entering the customer menu.*

Figure 1.1a shows when the user enters option ‘1’ as their input from main menu and the customer information is listed as shown. There are ID, Name, Contact, Artwork Purchased, Artist Preference displayed for each of the customers. The output will then prompt the user to input 5 different options, which is to Add, Delete, Modify, Search, or Exit the program.

*Graphical user interface, text

Description automatically generated*

*Figure 1.1b: User enters invalid options from the Customer menu.*

**1.1 ADD CUSTOMER**

*Text

Description automatically generated*

*Figure 1.1.1: User enters all valid inputs.*

*Text

Description automatically generated*

*Figure 1.1.2: User enters all valid inputs but cancelled adding customer action.*

*Text

Description automatically generated*

*Figure 1.1.3: User enters invalid customer ID.*

*Text

Description automatically generated*

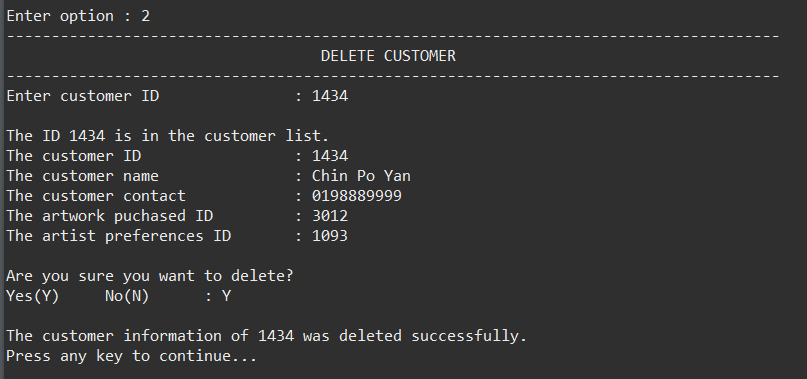
*Figure 1.1.4: User enters invalid Artwork ID.*

*Text

Description automatically generated*

*Figure 1.1.5: User enters invalid Artist ID.*

**1.2 DELETE CUSTOMER**

**

*Figure 1.2.1: User enters valid inputs for deletion of customer.*

*Text

Description automatically generated*

*Figure 1.2.3: User decides to not delete the valid customer ID.*

*Text

Description automatically generated*

*Figure 1.2.2: User enters invalid customer ID to be deleted.*

**1.3 MODIFY CUSTOMER**

*Text

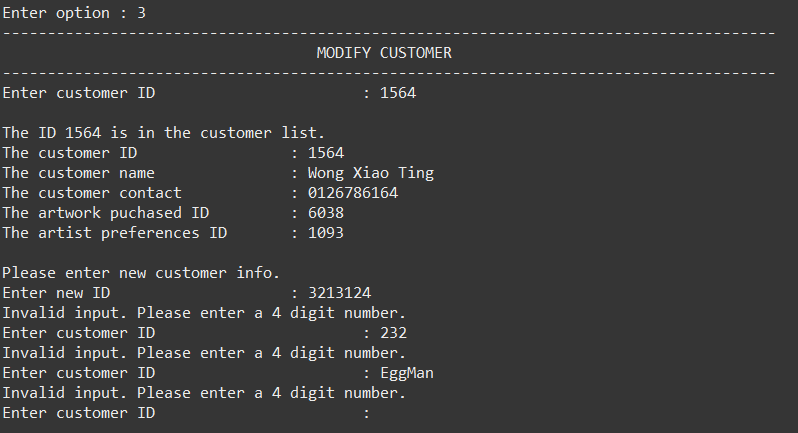
Description automatically generated*

*Figure 1.3.1: User enters the correct details for modification.*

*Text

Description automatically generated*

*Figure 1.3.2: User enters the invalid inputs for the customer ID.*

**

*Figure 1.3.3: User enters the invalid input for the new customer information.*

*Text

Description automatically generated*

*Figure 1.3.4: User enters the invalid Artwork ID for the new customer information.*

*Text

Description automatically generated*

*Figure 1.3.5: User enters the invalid Artist ID for the new customer information.*

**1.4 SEARCH CUSTOMER**

*Text

Description automatically generated*

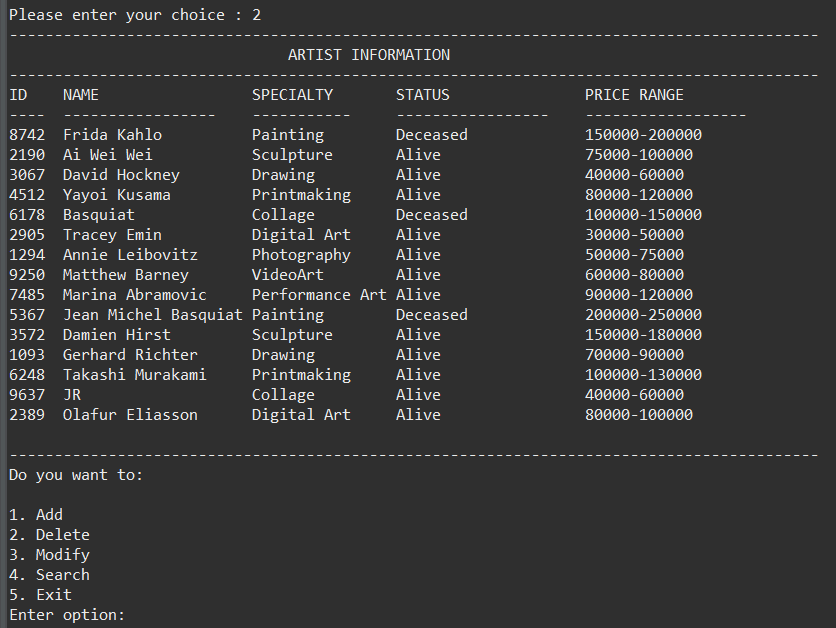
*Figure 1.4.1: User enters a valid customer ID to search for the corresponding customer.*

*Text

Description automatically generated*

*Figure 1.4.2: User enters the invalid customer ID.*

**2) ARTIST**

**

*Figure 2a: User enters Artist menu.*

Figure 2a shows when the user enters option ‘2’ as their input from main menu and the artist information is listed as shown. There is ID, Name, Specialty, Status, and Price Range displayed for each of the artists. The output will then prompt the user to input 5 different options, which is to Add, Delete, Modify, Search, or Exit the program.

*Text

Description automatically generated*

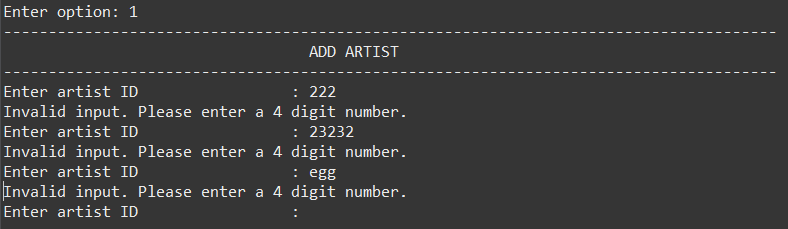
*Figure 2b: User enters invalid options from the Artist menu.*

**2.1 ADDING ARTIST**

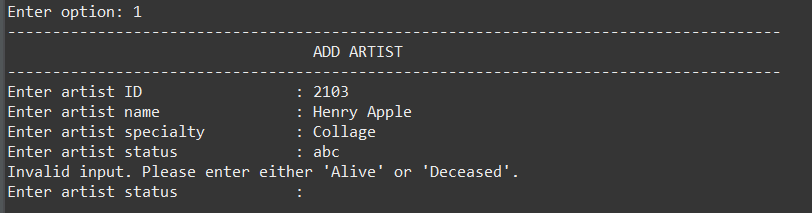
*Text

Description automatically generated*

*Figure 2.1.1: User enters valid inputs and added a new Artist.*

**

*Figure 2.1.2: User enters invalid inputs for adding new Artist.*

**

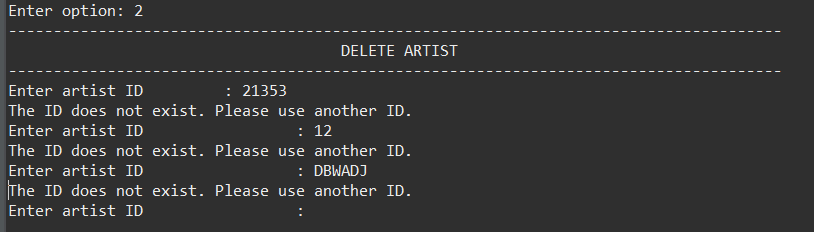
*Figure 2.1.3: User enters invalid Artist status.*

**2.2 DELETE ARTIST**

*Text

Description automatically generated*

*Figure 2.2.1: User enters valid input of Artist ID and deleted the inputted Artist.*

**

*Figure 2.2.2: User enters invalid Artist ID.*

**2.3 MODIFY ARTIST**

*Text

Description automatically generated*

*Figure 2.3.1: User enters valid modification details.*

*Text

Description automatically generated*

*Figure 2.3.2: User enters invalid Artist ID for modification.*

*Text

Description automatically generated*

*Figure 2.3.3: User enters invalid new artist ID.*

**2.4 SEARCH ARTIST**

*Text

Description automatically generated*

*Figure 2.4.1: User enters valid Artist ID to be searched.*

*Text

Description automatically generated*

*Figure 2.4.2: User enters invalid Artist ID.*

**3) ARTWORK**

*Graphical user interface, text

Description automatically generated*

*Figure 3a: User enters Artwork menu.*

Figure 3a shows when the user enters option ‘3’ as their input from main menu and the artwork information is listed as shown. There is ID, Name of artwork, Artist name, Date purchased, Date sold, Purchase price, Selling price, and Status displayed for each of the artwork. The output will then prompt the user to input 5 different options, which is to Add, Delete, Modify, Search, or Exit the program.

*Text

Description automatically generated*

*Figure 3b: User enters invalid options from the Artwork menu.*

**3.1 SEARCH ARTWORK**

*Text

Description automatically generated*

*Figure 3.1.1: User enters valid Artwork ID to be searched.*

*Text

Description automatically generated*

*Figure 3.1.2: User enters invalid Artwork ID.*

**3.2 ADD ARTWORK**

*Text

Description automatically generated*

*Figure 3.2.1: User enters all valid inputs to add a new Artwork.*

*Text

Description automatically generated*

*Figure 3.2.2: User enters invalid new Artwork ID to be added.*

*Text

Description automatically generated*

*Figure 3.2.3: User enters new valid Artwork ID but enters Artist that is unavailable.*

*Text

Description automatically generated*

*Figure 3.2.4: User enters new valid Artwork ID and Artist name but enters incorrect date purchased format.*

*Text

Description automatically generated*

*Figure 3.2.5: User enters valid new Artwork ID, Artist name, and date purchased format but enters invalid purchase price input.*

**3.3 MODIFY ARTWORK**

*Text

Description automatically generated*

*Figure 3.3.1: User enters valid inputs to modify an existing Artwork from Available to Sold status.*

*Text

Description automatically generated*

*Figure 3.3.2: User enters invalid Artwork ID to be modified.*

*Text

Description automatically generated*

*Figure 3.3.3: User enters valid Artwork ID and new Artwork title but enters invalid Artist name.*

*Text

Description automatically generated*

*Figure 3.3.4: User enters invalid Artist name.*

*Text

Description automatically generated*

*Figure 3.3.5: User enters invalid format of the purchase date.*

*Text

Description automatically generated*

*Figure 3.3.6: User enters invalid purchase price format.*

**3.4 DELETE ARTWORK**

*Text

Description automatically generated*

*Figure 3.4.1: User enters valid Artwork ID to be deleted.*

*Text

Description automatically generated*

*Figure 3.4.2: User enters invalid Artwork ID to be deleted.*

**4) EXIT**

Text

Description automatically generated

*Figure 4: User exits the program.*

Figure 4 shows when the user enters option ‘4’ as their input from main menu then a thank you prompt is outputted and the end.