Project Title

Coin Collection Tracker

Team: Bug Bashers

Team Members

1. Kondba Jatale
2. Durgesh Ingale
3. Prem Sawant
4. Abhijeet Desai
5. Priyanka Gaikwad

INDEX

Case Study

Managing coin collection:
Mr. X has a large collection of Coins that he has collected over the years and currently adding to
the collection. He wants to build an application to manage the collection. It should have the following features.
1. Addition.
a. The user should be able to add a new coin to the collection.
b. The coin has following details available
i. Country
ii. Denomination
iii. Year of minting
iv. Current value
v. Acquired date
c. User should also be able to bulk upload the data by reading from a file

2. Search
a. User should be able to create a list on:
i. Country
ii. Year of Minting
iii. Current Value (sorted)

b. User should be able to search a specific coin based on:
i. Country + Denomination
ii. Country + Year of Minting
iii. Country + Denomination + Year of Minting
iv. Acquired Date + Country
2. Persist

a. On startup the application should load the existing coin collection from database into a collection in the application.

b. The application should provide a choice to the user, that will enable the current state of the collection to be stored in the database.

CHAPTER 1: INTRODUCTION

1.1) Problem Statement

The "Coin Collection Tracker " aims to address the need for an efficient and user-friendly system for managing a large collection of coins. Mr. X, an avid collector, currently lacks an organized method to catalog and retrieve information about his coins.

Impact:

This can lead to a number of problems, including:

- Lost or misplaced coins
- Difficulty finding specific coins
- Difficulty valuing the collection
- Difficulty making informed decisions about the collection

1.2) Abstract

The "Coin Collection Tracker" is a software application designed to assist coin collectors in organizing and managing their collections. This system provides a set of features enabling users to add new coins, perform searches based on various criteria (such as country, year of minting, current value), and persistently store the collection's state. Additionally, the application supports bulk data upload from external files, enhancing efficiency in cataloging.

1.3) Scope of Work

1. **Coin Class**
   Responsible Member: Priyanka Gaikwad

Tasks:

- Implement the Coin class with attributes (Country, Denomination, Year of Minting, Current Value, Acquired Date).
- Include methods for getting and setting attributes.
- Implement a toString method for displaying coin information.

2. **CoinCollection Class**
   Responsible Member: Kondba Jatale and Priyanka Gaikwad

Tasks:

- Implement the CoinCollection class for managing the collection of coins.
- Include methods for adding new coins, searching, and any other collection-related operations.
- Handle storage of coins (e.g., in a Set or other appropriate data structure).
- Use the classes developed by other team members to perform operations based on user input.
- Handle input validation and user prompts.

3. **Searching Class**
   Responsible Member: Abhijeet Desai

Tasks:

- Implement various search functionalities in the Searching class, allowing users to search for coins based on different criteria, including Country and Denomination, Country and Year of Minting, Country, Denomination, and Year of Minting, as well as Acquired Date and Country.
- responsible for one of these specific search functions, ensuring comprehensive search capabilities.

4. **CreateList Class**
   Responsible Member: Prem Sawant and Abhijeet Desai
   Tasks:

- Implement the CreateList class for creating different lists based on user input (Country, Year of Minting, Current Value).
- Include methods for generating and displaying these lists.

5. **File Handling Class**

- Responsible Member: Durgesh Ingale

Tasks:

- Implement the File handling class for handling file operations.
- Include methods for reading from and writing to a file (e.g., CSV or text file) for bulk upload of coin data.

6. **Exception Handling**

- Responsible Member: Durgesh Ingale

Tasks:

- Implement the CustomException class for handling specific exceptions that may occur during the execution of the application. This class should extend the standard Java exception classes and provide custom messages for different scenarios, ensuring clear and informative error handling.

7. **DatabaseOperations Class**
   Responsible Member: Prem Sawant

   Tasks:

- Implement the DatabaseOperations class for performing CRUD operations on the coin collection.
- Include methods for inserting, updating, retrieving, and deleting coin records from the database.

8. **DatabaseUtils Class**

Responsible Member: Kondaba Jatale

Tasks:

- Implement the DatabaseUtils class to manage database connections and queries.
- Include methods for establishing a connection.

9. Testing

Responsible Member: All Team Members

Tasks:

- Write unit tests for individual classes and methods
- Collaborate to ensure comprehensive test coverage.

1.3 Operating Environment:

1.3.1 Software Requirements:

Name of Component	Specification
Operating System	Windows 7 or above
Tools	Eclipse

1.3.2 Hardware Requirements:

Name of Component	Specification
Processor	2, 3, 4 or any higher version
RAM	1 GB or Higher
Hard Disk	8 GB or above

CHAPTER 2- PROPOSED SYSTEM

2.1 Proposed System:

The proposed Coin Collection Tracker System is designed to provide an efficient and user-friendly platform for coin collectors to organize and manage their collections effectively. The system aims to address the limitations of manual collection management and enhance the overall user experience.

2.3 Objective of System:

The primary objectives of the Coin Collection Tracker System are as follows:

Efficient Coin Management: To Provide a platform for users to efficiently manage their coin collections, including adding new coins, searching for specific coins, and generating lists based on various criteria.

Flexibility in Searching: To Enable users to perform detailed searches based on various parameters, including country, denomination, year of minting, acquired date, and current value, providing a comprehensive search experience.

Bulk Data Upload: To Facilitate the bulk upload of coin data through file processing, allowing users to quickly add multiple coins to their collection from an external source.

Database Integration: To Integrate a database to persistently store coin data, ensuring that users can access their collection across sessions and allowing for efficient retrieval and management of coin information.

CHAPTER 3-ANALYSIS & DESIGN

1. **E-R Diagram:**

2. **Class Diagram:**

3. **Table Design:**

CHAPTER 4- TESTING

Test Class: CoinTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC1	Test the Coin constructor	1. Create a new Coin object with valid parameters. 2. Assert that the Coin object has the correct values for all of its properties.	The Coin object should have the correct values for all of its properties.	The Coin object has the correct values for all of its properties.	Pass
TC2	Test the Coin equals() method	1. Create two Coin objects with the same values for all of their properties. 2. Assert that the two Coin objects are equal.	The two Coin objects should be equal.	The two Coin objects are equal.	Pass
TC3	Test the Coin equals() method with different values	1. Create two Coin objects with different values for one or more of their properties. 2. Assert that the two Coin objects are not equal.	The two Coin objects should not be equal.	The two Coin objects are not equal.	Pass

Test Class: CoinCollectionAppTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/ Fail

TC1	Add a new coin with valid parameters	1. Call the addNewCoinToTheCollection() method with valid parameters. 2. Assert that the new coin is added to the set.	The new coin should be added to the set.	The new coin is added to the set.	Pass
TC2	Add a new coin with invalid parameters	1. Call the addNewCoinToTheCollection() method with invalid parameters. 2. Assert that the new coin is not added to the set.	The new coin should not be added to the set.	The new coin is not added to the set.	Pass
TC3	Add a new coin that already exists in the set	1. Call the addNewCoinToTheCollection() method with a new coin that already exists in the set. 2. Assert that the new coin is not added to the set.	The new coin should not be added to the set.	The new coin is not added to the set.	Pass
TC4	Add a new coin with a null value	1. Call the addNewCoinToTheCollection() method with a new coin with a null value for one of its properties. 2. Assert that the new coin is not added to the set.	The new coin should not be added to the set.	The new coin is not added to the set.	Pass
TC5	Add a new coin with an empty value	1. Call the addNewCoinToTheCollection() method with a new coin with an empty value for one of its properties. 2. Assert that the new coin is not added to the set.	The new coin should not be added to the set.	The new coin is not added to the set.	Pass

Test Class: SearchingTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC1	Search for coins by country and denomination	1. Call the Searching.countryAndDenomination() method with valid parameters. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC2	Search for coins by country and year of minting	1. Call the Searching.countryAndYearOfMinting() method with valid parameters. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass

TC3	Search for coins by country, denomination, and year of minting	1. Call the Searching.countryAndDenominationAndYearOfMinting() method with valid parameters. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC4	Search for coins by country and acquired date	1. Call the Searching.countryAndAcquiredDate() method with valid parameters. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC5	Search for coins with invalid parameters	1. Call the Searching method with invalid parameters. 2. Assert that no coins are returned.	No coins should be returned.	No coins are returned.	Pass

Test Class: CreateListTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC1	Create a list of coins by country	1. Call the CreateList.countryList() method with a valid country. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC2	Create a list of coins by year of minting	1. Call the CreateList.yearOfMiningList() method with a valid year of minting. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC3	Create a list of coins by current value	1. Call the CreateList.currentValueList() method with a valid current value. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC4	Create a list of coins with invalid parameters	1. Call the CreateList method with invalid parameters. 2. Assert that no coins are returned.	No coins should be returned.	No coins are returned.	Pass

Test Class: DatabaseOperationsTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC1	Get all coins from database	1. Call the DatabaseOperations.getCoins() method. 2. Assert that the expected coins are returned.	The expected coins should be returned.	The expected coins are returned.	Pass
TC2	Save coins to database	1. Call the DatabaseOperations.saveDataToDatabase() method with a valid set of coins. 2. Assert that the coins are saved successfully.	The coins should be saved successfully.	The coins are not saved successfully.	fail
TC3	Save coins to database with invalid parameters	1. Call the DatabaseOperations.saveDataToDatabase() method with an invalid set of coins. 2. Assert that the coins are not saved to the database.	The coins should not be saved to the database.	The coins are not saved to the database.	Pass
TC4	Save coins to database with empty set	1. Call the DatabaseOperations.saveDataToDatabase() method with an empty set of coins. 2. Assert that the coins are not saved to the database.	The coins should not be saved to the database.	The coins are not saved to the database.	Pass

Test Class: DBUtilsTest

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC1	Initialize database	1. Call the DbUtils.initializeDatabase() method. 2. Assert that the connection to the database is established.	The connection to the database should be established.	The connection to the database is established.	Pass
TC2	Get connection	1. Call the DbUtils.getConnection() method. 2. Assert that the connection is valid.	The connection should be valid.	The connection is valid.	Pass
TC3	Close connection	1. Call the DbUtils.closeConnection() method. 2. Assert that the connection is closed.	The connection should be closed.	The connection is closed.	Pass

TC4	Close resource	1. Call the DbUtils.closeResource() method with a valid resource. 2. Assert that the resource is closed.	The resource should be closed.	The resource is not closed.	Fail
TC5	Close resource with null parameter	1. Call the DbUtils.closeResource() method with a null parameter. 2. Assert that the method does not throw an exception.	The method should not throw an exception.	The method does not throw an exception.	Pass

CONCLUSION

- In conclusion, the development of the Coin Collection Management System has successfully addressed the challenges faced by coin collectors in organizing and managing their collections effectively. The system provides a comprehensive set of features, including streamlined coin addition, powerful search capabilities, list generation, bulk data upload, and secure database integration.