Coin Management App

Problem Statement:

Develop a coin collection management application that allows users to perform various operations, including loading data from a database, inserting data from a file, displaying coins, adding new coins, creating lists based on criteria, and searching for coins based on specific attributes.

Subtask 1: Initialize the Project

- Problem Statement: Create a project structure and initialize the necessary files and classes.
- Acceptance Criteria:
  1. The project directory and basic files are created.
  2. Necessary Java classes (Coin, DatabaseOperations, FileHandling, CreateList, Searching, CoinCollectionApp) are defined.

Subtask 2: Load Data from Database

- Problem Statement:
  1. Implement the functionality to load data from a database into a collection.

- Algorithm:
  1. Open a connection to the database.
  2. Create a SQL query to retrieve all coin data from the "coins" table.
  3. Execute the query and obtain a result set.
  4. Iterate through the result set and for each row:
  5. a. Extract coin attributes (country, denomination, yearOfMinting, currentValue, currency, acquiredDate).
  6. b. Create a new Coin object with the extracted attributes.
  7. c. Add the Coin object to the coins collection.
  8. Close the result set and the database connection
  9.
- Acceptance Criteria:
  1. The application can connect to the database and retrieve coin data.
  2. The loaded data is stored in the `coins` set.
  3. Users can select option 1 in the main menu to load data from the database.

Coin Management App

Subtask 3: Insert Data from File to Database

- Problem Statement:
  Implement the functionality to insert data from a file into the database.

- Algorithm:
  1. Open and read a CSV file containing coin data.
     For each line in the CSV file:
     1. Parse the line to extract coin attributes (country, denomination, yearOfMinting, currentValue, currency, acquiredDate).
     2. Create a SQL INSERT statement with placeholders for the coin attributes.
     3. Prepare a SQL statement and set values for the placeholders using the extracted attributes.
     4. Execute the SQL statement to insert the coin data into the "coins" table.
     5. Close the file.
- Acceptance Criteria:
     1. Users can select option 2 in the main menu to insert data from a file.
     2. Data from the CSV file is successfully inserted into the database.

Subtask 4: Display All Available Coins

- Problem Statement:
  Implement the functionality to display all available coins.
- Algorithm:
  1. Prompt the user to enter details of the new coin (country, denomination, yearOfMinting, currentValue, currency, acquiredDate).
  2. Validate user input to ensure it matches expected formats and criteria.
  3. Create a new Coin object with the entered attributes.
  4. Add the new Coin object to both the coins and dummyCoins collections.
- Acceptance Criteria
  1. Users can select option 4 in the main menu to add a new coin to the collection.
  2. The new coin is successfully added to both the coins set and the dummyCoins set.

Coin Management App

Subtask 5: Add New Coin to Collection

- Problem Statement:
  Implement the functionality to manually add a new coin to the collection.
- Algorithm:
  1. Prompt the user to enter details of the new coin (country, denomination, yearOfMinting, currentValue, currency, acquiredDate).
  2. Validate user input to ensure it matches expected formats and criteria.
  3. Create a new Coin object with the entered attributes.
  4. Add the new Coin object to both the coins and dummyCoins collections.
- Acceptance Criteria:
  1. Users can select option 4 in the main menu to add a new coin to the collection.
  2. The new coin is successfully added to both the `coins` set and the `dummyCoins` set.

Subtask 6: Save Newly Added Data to Database

- Problem Statement:
  Implement the functionality to save only newly added data from the `dummyCoins` set to the database.
- Algorithm:
  1. Open a connection to the database.
  2. Create a SQL INSERT statement with placeholders for the coin attributes.
  3. For each Coin object in the dummyCoins collection:
     a. Prepare a SQL statement and set values for the placeholders using the coin's attributes.
     b. Execute the SQL statement to insert the coin data into the "coins" table.
  4. Close the database connection.
  5. Clear the dummyCoins collection.
- Acceptance Criteria:
  1. Users can select option 5 in the main menu to save newly added data to the database.
  2. Only data that is in the `dummyCoins` set is inserted into the database.

Subtask 7: Show Newly Added Data

- Problem Statement:
  Implement the functionality to display newly added data from the `dummyCoins` set.
- Algorithm:
  1. Create a method in `CoinCollectionApp` to iterate through the `dummyCoins` set and print coin details.
- Acceptance Criteria:
  1. Users can select option 6 in the main menu to show newly added data.

Coin Management App

Subtask 8: Create Lists Based on Criteria

- Problem Statement:
  Implement the functionality to create lists of coins based on criteria such as country, year of minting, and current value.
- Algorithm:
  Display a menu to let the user choose the criteria for creating a list (e.g., country, year of minting, current value).
  Based on the user's choice:
  1. For country:
     a. Iterate through the coins collection.
     b. Create a list of coins with matching countries.
     c. Display the list of coins.
  2. For year of minting:
     a. Iterate through the coins collection.
     b. Create a list of coins with matching year of minting.
     c. Display the list of coins.
  3. For current value:
     a. Iterate through the coins collection.
     b. Create a list of coins with matching current values.
     c. Display the list of coins.
- Acceptance Criteria:
  1. Users can create lists based on country, year of minting, and current value.
  2. Lists are displayed according to the selected criteria.

Coin Management App

Subtask 9: Searching for Coins Based on Criteria

- Problem Statement: Implement the functionality to search for coins based on various criteria, such as country, denomination, year of minting, and acquired date.
- Algorithm:
  1. Display a menu to let the user choose the search criteria (e.g., country, denomination, year of minting, acquired date).
  2. Based on the user's choice:
     a. For country and denomination:
        A. Prompt the user to enter a country and denomination.
        B. Iterate through the coins collection.
        C. Display coins that match both the country and denomination.
     b. For country and year of minting:
        A. Prompt the user to enter a country and year of minting.
        B. Iterate through the coins collection.
        C. Display coins that match both the country and year of minting.
     c. For country and denomination and year of minting:
        A. Prompt the user to enter a country, denomination, and year of minting.
        B. Iterate through the coins collection.
        C. Display coins that match all three criteria.
     d. For country and acquired date:
        A. Prompt the user to enter a country and acquired date.
        B. Iterate through the coins collection.
        C. Display coins that match both the country and acquired date.
- Acceptance Criteria:
  1. Users can perform searches based on country, denomination, year of minting, and acquired date.
  2. Search results are displayed according to the selected criteria.

Coin Management App

Subtask 10: Main Menu and User Interaction

- Problem Statement:
  Implement the main menu and user interactions to navigate through the application.
- Algorithm:
  1. Create a main menu in the `CoinCollectionApp` class with options to perform different actions.
  2. Implement user interactions to select options and execute corresponding functionalities.
- Acceptance Criteria:
  1. Users can navigate through the application using the main menu.
  2. All implemented functionalities can be accessed from the main menu.

Subtask 11: Exception Handling and Input Validation

- Problem Statement:
  Add error handling and input validation to ensure a smooth user experience.
- Algorithm:
  1. Implement error handling for potential exceptions in database operations, file reading, and user input.
  2. Validate user input to ensure it matches expected formats and criteria.
- Acceptance Criteria:
  1. The application handles exceptions gracefully and provides informative error messages.
  2. User input is validated to prevent invalid entries.

Coin Management App

Coin Collection Management Application Testing Documentation

Test Case 1: Load Data from Database

- Test Description:
  This test case checks if the application successfully loads coin data from the database.

- Test Steps:
  1. Open a connection to a test database.
  2. Insert sample data into the test database.
  3. Call the `loadAllDataFromDatabase` method to load data from the test database.
  4. Assert that the data loaded matches the sample data.

- Expected Result:
  a. The data loaded from the database should match the sample data.
  b. The test should pass without any exceptions or errors.

Test Case 2: Insert Data from File to Database

- Test Description:

- This test case checks if the application correctly inserts data from a CSV file into the database.

- Test Steps:
  1. Create a temporary test CSV file with sample coin data.
  2. Call the `insertDataFromFileToDatabase` method to insert data from the test file into a test database.
  3. Query the test database to check if the inserted data matches the sample data.

- Expected Result:
  1. The data inserted into the database should match the sample data from the test CSV file.
  2. The test should pass without any exceptions or errors.

Coin Management App

Test Case 3: Add New Coin to Collection

- Test Description:
  This test case checks if the application properly adds a new coin to the collection.

- Test Steps:
  1. Call the `addNewCoinToCollection` method with sample coin data.
  2. Check if the new coin is in the collection.

- Expected Result:
  1. The new coin should be successfully added to the collection.
  2. The test should pass without any exceptions or errors.

Test Case 4: Save Newly Added Data to Database

- Test Description:
  This test case checks if the application saves newly added data to the database.

- Test Steps:
  1. Call the `saveNewlyAddedDataToDatabase` method with a new coin added to the collection.
  2. Query the database to check if the new data matches the added coin.

- Expected Result:
  1. The newly added data should be successfully saved to the database.
  2. The test should pass without any exceptions or errors.

Test Case 5: Create Lists Based on Criteria

- Test Description:
  This test case checks if the application correctly creates lists of coins based on specified criteria.

- Test Steps:
  1. Call the `createListsBasedOnCriteria` method with specific criteria.
  2. Check if the generated list matches the expected result.

- Expected Result:
  1. The application should generate lists of coins that match the specified criteria.
  2. The test should pass without any exceptions or errors.

Coin Management App

Test Case 6: Searching for Coins Based on Criteria

- Test Description:
  This test case checks if the application successfully searches for coins based on various criteria.

- Test Steps:
  1. Call the `searchForCoinsBasedOnCriteria` method with specific search criteria.
  2. Check if the search results match the expected coins.

- Expected Result:
  1. The application should find and display coins that match the specified search criteria.
  2. The test should pass without any exceptions or errors.