Assignment report

Please note: even though best effort have been made to copy code with original formatting, it still can be affected sometimes. For proper code review, please, refer to the provided .java files

CoinSorter class source code

```java
import java.util.ArrayList;
import java.util.List;

/**
 * @author anonymous
 * This class provides attributes, methods and overall logic for Coin Sorter program.
 * All methods from this class are used for terminal version of program and majority of them are used for
 * program with GUI
 */
public class CoinSorter {

    private String currency;
    private int minCoinIn;
    private int maxCoinIn;
    private List<Integer> coinList;

    /* First constructor, assigns data to all attributes. Even though this constructor allows flexibility with
     * assigning different data to attributes, in the body of program parameters will be passed with accordance to the brief.
     * This flexibility was provided for future development and to comply with UML diagram in brief
     */
    public CoinSorter(String currencyIn, int minCoinIn, int maxCoinIn, ArrayList<Integer> coinListIn)
    {
        this.currency = currencyIn;
        this.minCoinIn = minCoinIn;
        this.maxCoinIn = maxCoinIn;
        this.coinList = coinListIn;
    }

    // Second constructor, assigns default data from brief to currency, minCoinIn, maxCoinIn and coinList
    public CoinSorter()
    {
        this.currency = "Pound sterling";
        this.minCoinIn = 0;
        this.maxCoinIn = 10000;
        setStandardCoinValues();
    }

    // Method to set currency
    public void setCurrency(String currencyIn)
    {
        this.currency = currencyIn;
    }
```

```java
// Method to set minCoinIn
public void setMinCoinIn(int minCoinIn)
{
    this.minCoinIn = minCoinIn;
}

// Method to set maxCoinIn
public void setMaxCoinIn(int maxCoinIn)
{
    this.maxCoinIn = maxCoinIn;
}

// Method to get currency
public String getCurrency()
{
    return this.currency;
}

// Method to get minCoinIn
public int getMinCoinIn()
{
    return this.minCoinIn;
}

// Method to get maxCoinIn
public int getMaxCoinIn()
{
    return this.maxCoinIn;
}

// Method to get coinList(used for GUI)
public List<Integer> getCoinList()
{
    return this.coinList;
}

/* This method both prints and returns coinList elements.
 * It does both because description of method and return data type in UML diagram contradict each other in brief
 * Method description requires to print String. UML diagram requires to return String.
 * In order to comply with brief requirements, method does both. In real life project, I would return to discuss with client
 * and suggest using only return String for this method, and use println function in main, if it needs to be printed
 */
public String printCoinList()
{
    String temp = "The current coin denominations are in circulation: ";
    if (this.coinList != null && this.coinList.size() > 0) // evaluating if there are elements to display from coinList
    {
        for (int i = 0; i < this.coinList.size(); i ++)// iterating through all elements of coinList, adding them to temp String
        {
            temp += this.coinList.get(i);
            temp += ", ";
        }
        temp = temp.substring(0, temp.length() - 2); // removing last ", "
        System.out.println(temp);
        return temp;
    }
    temp += "none"; // this case activated in case coinList is empty
    System.out.println(temp);
    return temp;

}

/* Coin calculator takes total value to calculate and in which coins should calculation happen.
 * After that method will calculate how many coins of given denomination in total value
 * and what is the remainder. For instance, with parameters (120, 50), result will be
 * 2x50p and 20p remainder
 */
public String coinCalculator(int totalValueToExchangeIn, int coinTypeIn)
{
    int coinTypeCounter = 0;
    String coinCurrency;
    while(totalValueToExchangeIn >= coinTypeIn)
    {
        totalValueToExchangeIn -= coinTypeIn;
        coinTypeCounter ++;
    }

    // Short conditional statement to use cents or pennies in printing statement, depending in current currency setting
    if(this.currency.equalsIgnoreCase("Pound sterling"))
    {
        coinCurrency = "p";
    }
    else
    {
        coinCurrency = "c";
    }
    return ("A total of " + coinTypeCounter + " x " + coinTypeIn + coinCurrency + " coints can be exchanged, with a remainder of
     " + totalValueToExchangeIn + coinCurrency + ".");
}

/* Multi coin calculator takes total value to calculate and which coin type should not be used in calculations.
 * After that, calculator will calculate to which denominators total value can be divided, starting from the biggest
 * and finishing with the smallest denominator, EXCLUDING coinTypeIn denominator. After that, result is returned with remainder.
 * For instance, with parameters (562, 50) result will be 2 x 200p, 1 x 100p, 0 x 50p, 3 x 20p, 0 x 10p, with a remainder of 2p
 */
public String multiCoinCalculator(int totalValueToExchangeIn, int coinTypeIn)
{
    int coinTypeCounter200 = 0;
    int coinTypeCounter100 = 0;
    int coinTypeCounter50 = 0;
    int coinTypeCounter20 = 0;
    int coinTypeCounter10 = 0;
    String coinCurrency;

    // Calculations for 200p
    while(totalValueToExchangeIn >= 200 && coinTypeIn != 200)
    {
        totalValueToExchangeIn -= 200;
        coinTypeCounter200 ++;
    }

    // Calculations for 100p
    while(totalValueToExchangeIn >= 100 && coinTypeIn != 100)
    {
        totalValueToExchangeIn -= 100;
        coinTypeCounter100 ++;
    }

    // Calculations for 50p
    while(totalValueToExchangeIn >= 50 && coinTypeIn != 50)
    {
        totalValueToExchangeIn -= 50;
        coinTypeCounter50 ++;
    }

    // Calculations for 20p
    while(totalValueToExchangeIn >= 20 && coinTypeIn != 20)
    {
        totalValueToExchangeIn -= 20;
        coinTypeCounter20 ++;
    }

    // Calculations for 10p
    while(totalValueToExchangeIn >= 10 && coinTypeIn != 10)
    {
        totalValueToExchangeIn -= 10;
        coinTypeCounter10 ++;
    }

    // Short conditional statement to use cents or pennies in printing statement, depending on current currency setting
    if(this.currency.equalsIgnoreCase("Pound sterling"))
    {
        coinCurrency = "p";
    }
    else
    {
        coinCurrency = "c";
    }

    return ("The coins exchanged are: " + coinTypeCounter200 + " x " + "200" + coinCurrency + ", " + coinTypeCounter100 + " x 10
    0" + coinCurrency + ", " + coinTypeCounter50 + " x 50" + coinCurrency + ", " + coinTypeCounter20 + " x 20" + coinCurrency +
    ", " + coinTypeCounter10 + " x 10" + coinCurrency + ", with a remainder of " + totalValueToExchangeIn + coinCurrency);
}
```

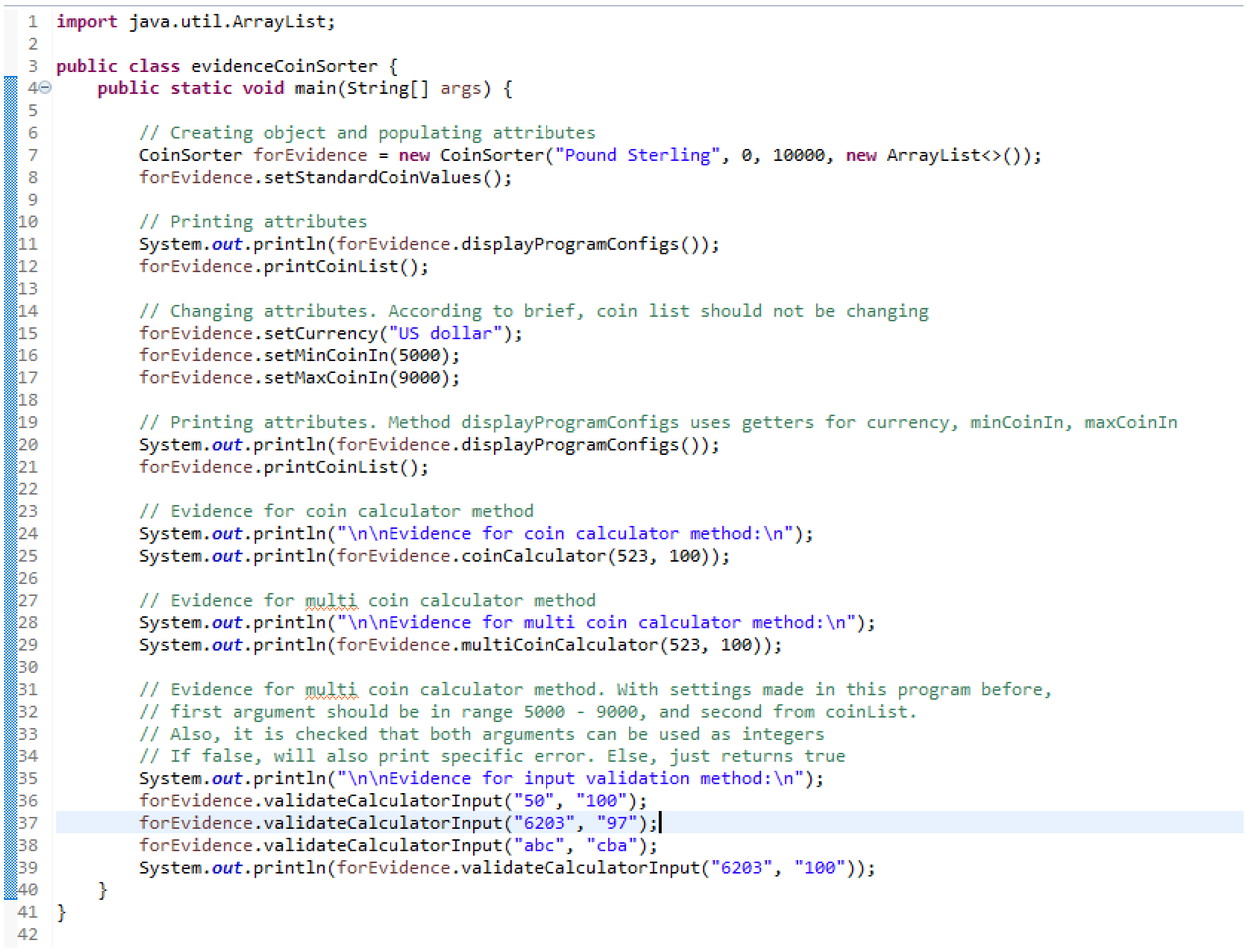
```
// Returns String with current currency and minimum and maximum value accepted as an input
public String displayProgramConfigs()
{
    String coin;
    if(this.currency.equalsIgnoreCase("Pound sterling"))
    {
        coin = "p";
    }
    else
    {
        coin = "c";
    }
    return ("The current currency is " + this.currency + " and the current minimum and maximum values accepted as an input are "
      + this.minCoinIn + coin + " and " + this.maxCoinIn + coin);
}

/* This validating method returns true if given coin type is in the coinList AND if given valueToExchange is
 * in provided range from the brief (from minCoinIn to maxCoinIn) AND if provided values can be used as integers
 */
public boolean validateCalculatorInput(String ValueToExchangeIn, String coinTypeIn)
{
    // Firstly, validating if values are numbers
    if (!validateInt(ValueToExchangeIn) || !validateInt(coinTypeIn))
    {
        System.out.println("You have to type in only integers in order to perform operation");
        return false;
    }

    // Secondly, validating value to exchange
    if(Integer.parseInt(ValueToExchangeIn) < this.minCoinIn || Integer.parseInt(ValueToExchangeIn) > this.maxCoinIn)
    {
        System.out.println("Your value to exchange should be in the range " + this.minCoinIn + " - " + this.maxCoinIn + " inclus
       ive");
        return false;
    }

    // Thridly, validating coin type
    for (int i = 0; i < coinList.size(); i ++)
    {
        if (coinList.get(i) != Integer.parseInt(coinTypeIn))
        {
            continue;
        }
        else return true;
    }
    System.out.println("The coin type you have requested is not supported. Supported coin types (denominations in circulation) a
    re 200, 100, 50, 20, 10. In order to do calculation use one of them");
    return false;
}
```

```java
    /* This validating method returns true if given String is integer
     * Otherwise, false
     */
    public boolean validateInt(String stringToValidate)
    {
        try
        {
            int temp1 = Integer.parseInt(stringToValidate.trim());
        }
        catch (NumberFormatException nfe)
        {
            return false;
        }
        if(stringToValidate.trim().contains(",") || stringToValidate.trim().contains("."))
        {
            return false;
        }
        return true;
    }

    /* Method to assign values to the second constructor
     * However, in case first constructor will be initialised, can be called from the main in order to populate coinList with
     * correct denominators from the brief (200, 100, 500, 30, 10)
     */
    public void setStandardCoinValues()
    {
        this.coinList = new ArrayList<Integer>();
        this.coinList.add(200);
        this.coinList.add(100);
        this.coinList.add(50);
        this.coinList.add(20);
        this.coinList.add(10);
    }
}
```

Evidence for CoinSorter class

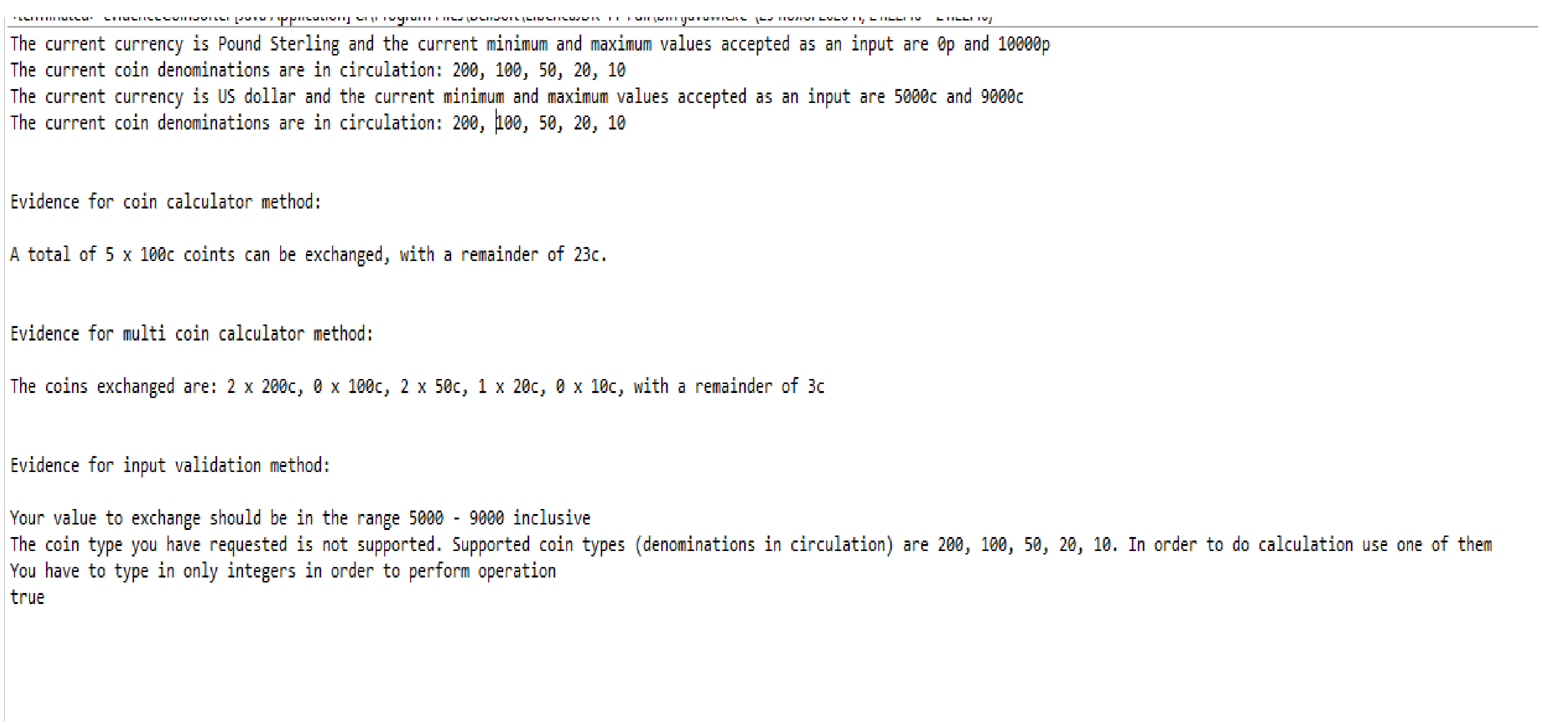
It will be demonstrated that class works by invoking its primary methods. If you want to see them invoked in terminal interface or GUI, kindly see it further in the document in evidence of testing classes with interface. Please note, that some methods do not have user input validation in their body right away, instead separate validation methods are used. In actual interface validation methods are called first to make sure that operation can be done with provided input. For instance, setter method for `getMinCoinIn` defined to setup any integer, but in body of program it would be called only after calling input validation method and passing validation successfully.

Code for evidence



```java
import java.util.ArrayList;

public class evidenceCoinSorter {
    public static void main(String[] args) {

        // Creating object and populating attributes
        CoinSorter forEvidence = new CoinSorter("Pound Sterling", 0, 10000, new ArrayList<>());
        forEvidence.setStandardCoinValues();

        // Printing attributes
        System.out.println(forEvidence.displayProgramConfigs());
        forEvidence.printCoinList();

        // Changing attributes. According to brief, coin list should not be changing
        forEvidence.setCurrency("US dollar");
        forEvidence.setMinCoinIn(5000);
        forEvidence.setMaxCoinIn(9000);

        // Printing attributes. Method displayProgramConfigs uses getters for currency, minCoinIn, maxCoinIn
        System.out.println(forEvidence.displayProgramConfigs());
        forEvidence.printCoinList();

        // Evidence for coin calculator method
        System.out.println("\n\nEvidence for coin calculator method:\n");
        System.out.println(forEvidence.coinCalculator(523, 100));

        // Evidence for multi coin calculator method
        System.out.println("\n\nEvidence for multi coin calculator method:\n");
        System.out.println(forEvidence.multiCoinCalculator(523, 100));

        // Evidence for multi coin calculator method. With settings made in this program before,
        // first argument should be in range 5000 - 9000, and second from coinList.
        // Also, it is checked that both arguments can be used as integers
        // If false, will also print specific error. Else, just returns true
        System.out.println("\n\nEvidence for input validation method:\n");
        forEvidence.validateCalculatorInput("50", "100");
        forEvidence.validateCalculatorInput("6203", "97");
        forEvidence.validateCalculatorInput("abc", "cba");
        System.out.println(forEvidence.validateCalculatorInput("6203", "100"));
    }
}
```

Output for evidence



```
[illegible]
The current currency is Pound Sterling and the current minimum and maximum values accepted as an input are 0p and 10000p
The current coin denominations are in circulation: 200, 100, 50, 20, 10
The current currency is US dollar and the current minimum and maximum values accepted as an input are 5000c and 9000c
The current coin denominations are in circulation: 200, 100, 50, 20, 10


Evidence for coin calculator method:

A total of 5 x 100c coints can be exchanged, with a remainder of 23c.


Evidence for multi coin calculator method:

The coins exchanged are: 2 x 200c, 0 x 100c, 2 x 50c, 1 x 20c, 0 x 10c, with a remainder of 3c


Evidence for input validation method:

Your value to exchange should be in the range 5000 - 9000 inclusive
The coin type you have requested is not supported. Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10. In order to do calculation use one of them
You have to type in only integers in order to perform operation
true
```

TestCoinSorter class source code

```java
import java.util.ArrayList;
import java.util.Scanner;

/**
 * @author anonymous
 * This class defines all elements of terminal menu, initialises all necessary objects and starts program
 * Can be complied and launched both in IDE and terminal
 */
public class TestCoinSorter {

    // Main method to initialise objects and launch menu
    public static void main(String[] args) {
        // Initialising user1. Interface list initialised as an array list
        CoinSorter user1 = new CoinSorter("Pound Sterling", 0, 10000, new ArrayList<>());
        user1.setStandardCoinValues();

        // Starting main menu and passing object instance as a parameter
        startMenu(user1);

    }

    // This method defines main menu and starts it
    public static void startMenu(CoinSorter objectIn)
    {

        // Printing menu
        System.out.println("\n*** Coin Sorter - Main Menu ***\n");
        System.out.println("1 - Coin Calculator\n");
        System.out.println("2 - Multiple Coin Calculator\n");
        System.out.println("3 - Print coin list\n");
        System.out.println("4 - Set details\n");
        System.out.println("5 - Display program configurations\n");
        System.out.println("6 - Quit the program\n");
        System.out.print("\nPlease type corresponding number in terminal: ");

        // Initialising scanner to take input
        Scanner sc = new Scanner(System.in);
        String input = sc.next().trim();

        /* Below menu for each choice in menu defined. As section 4 (set details) opens up
         * another menu, it was created as another class.
         * Important to not that menu is using recursion on order to be able to repeatedly prompt user for
         * input after
         * getting invalid input or after some operations were finished
         */

        // Case 1, coin calculator
        if(input.equalsIgnoreCase("1"))
```

```
{
    // Variables to take user input
    String totalValueToExchangeIn;
    String coinTypeIn;

    // This loop validates input data
    do
    {
        System.out.print("\nType in total value to exchange: ");
        totalValueToExchangeIn = sc.next();
        System.out.print("\nType in the coin type for exchange: ");
        coinTypeIn = sc.next();
    }
    while (!objectIn.validateCalculatorInput(totalValueToExchangeIn, coinTypeIn));

    // When input was validated, coinCalculator is called in order to perform calculation and prin
   t result
    System.out.print("\n");
    System.out.println(objectIn.coinCalculator(Integer.parseInt(totalValueToExchangeIn),
     Integer.parseInt(coinTypeIn)));
    System.out.print("\n");
    startMenu(objectIn);
}

// Case 2, Multiple Coin Calculator
else if(input.equalsIgnoreCase("2"))
{
    // Variables to take user input
    String totalValueToExchangeIn;
    String coinTypeIn;

    // This loop validates input data
    do
    {
        System.out.print("\nType in total value to exchange: ");
        totalValueToExchangeIn = sc.next();
        System.out.print("\nType in which coin should not be used in exchange: ");
        coinTypeIn = sc.next();
    }
    while (!objectIn.validateCalculatorInput(totalValueToExchangeIn, coinTypeIn));

    // When input was validated, coinCalculator is called in order to perform calculation and prin
   t result
    System.out.print("\n");
   System.out.println(objectIn.multiCoinCalculator(Integer.parseInt(totalValueToExchangeIn),
   Integer.parseInt(coinTypeIn)));
    System.out.print("\n");
    startMenu(objectIn);
}

// Case 3, print coin list
```

```java
        else if(input.equalsIgnoreCase("3"))
        {
            System.out.print("\n");
            objectIn.printCoinList();
            System.out.print("\n");
            startMenu(objectIn);
        }

        // Case 4, set details. As this option is the most complicated, it starts in separate method with
separate menu
        else if (input.equalsIgnoreCase("4"))
        {
            choice4(objectIn);
        }

        // Case 5, display program configurations
        else if (input.equalsIgnoreCase("5"))
        {
            System.out.print("\n");
            System.out.println(objectIn.displayProgramConfigs());
            System.out.print("\n");
            startMenu(objectIn);
        }

        // Case 6, exit
        else if (input.equalsIgnoreCase("6"))
        {
            System.out.println("\nThank you for using Coin Sorter program");
            sc.close(); // Closing scanner to avoid memory leak
        }

        // This part handles invalid input in main menu and uses recursion
        else
        {
            System.out.println("\nInvalid input. Please type number 1 - 6");
            System.out.println("Restarting program\n\n");
             startMenu(objectIn); // Recursive call to get input 1 - 6, and if not provided keep prompting
              user
        }
    }

    // Separate method for case 4, set details
    public static void choice4(CoinSorter objectIn)
    {
        // Printing menu
        System.out.println("\n*** Set Details Sub-Menu ***\n");
        System.out.println("1 - Set currency\n");
        System.out.println("2 - Set minimum coin input value\n");
        System.out.println("3 - Set maximum coin input value\n");
        System.out.println("4 - Return to main menu\n");
        System.out.print("\nPlease type corresponding number in terminal: ");
```

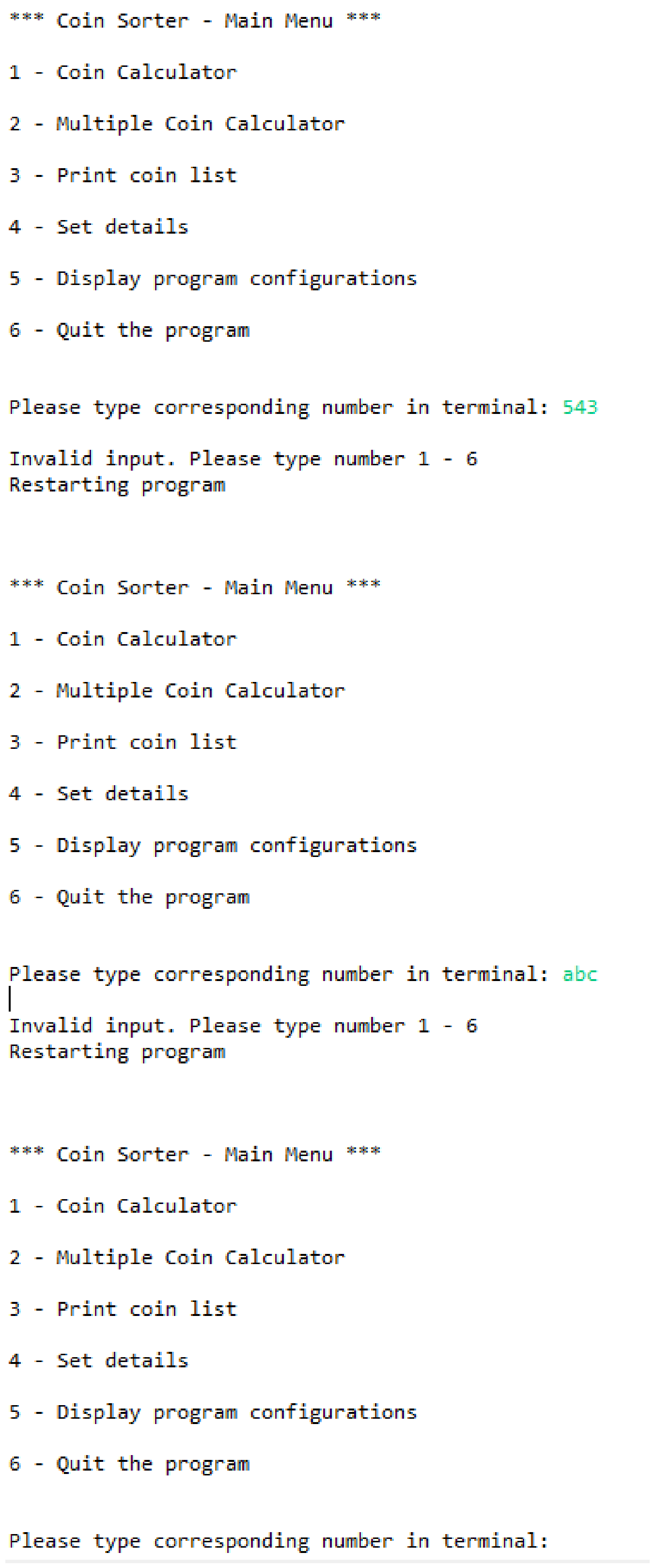
```
// Scanner for user input
Scanner sc = new Scanner(System.in);
String input = sc.next().trim();

// Case 4.1
if(input.equalsIgnoreCase("1"))
{
    System.out.print("Type in the currency you would like to set. GBP or USD: ");
    input = sc.next().trim();
    // As provided per brief and per extra instructions from from course coordinator, only two cur
   rencies will be accepted in this section: GBP and USD
    // Checking if user input requests to change currency to GBP
     if (input.equalsIgnoreCase("sterling") || input.equalsIgnoreCase("pound")
     || input.equalsIgnoreCase("pounds") || input.equalsIgnoreCase("GBP"))
    {
        objectIn.setCurrency("Pound sterling");
        System.out.print("\n");
        System.out.println("Currency has been setup to Pound sterling");
        System.out.print("\n");
    }
    // Checking if user input requests to change currency to USD
      else if(input.equalsIgnoreCase("US dollar") ||
      input.equalsIgnoreCase("US")  || input.equalsIgnoreCase("USD") ||
      input.equalsIgnoreCase("Dollar") || input.equalsIgnoreCase("Dollars"))
    {
        objectIn.setCurrency("US dollar");
        System.out.print("\n");
        System.out.println("Currency has been setup to US dollar");
        System.out.print("\n");
    }
    // In all other cases user will be explained which currencies are possible and how to set them
    up
    else
    {
        System.out.print("\n");
        System.out.println("Due to the brief instructions provided this program should only work w
        ith Pound sterling or US dollars. Try again, and type in \"GBP\" or \"USD\"");
        System.out.print("\n");
    }
    // Returning to set details menu
    choice4(objectIn);
}

// Case 4.2, set minimum coin input

else if(input.equalsIgnoreCase("2"))
{
    // Explaining user which input is allowed and taking input
    System.out.print("Type in the minimum coin input you would like to set. Should be in range 0 -
    10,000 inclusive, and no more than the maximum coin input: ");
```

```
        String inputInt = sc.next();

        // Validating that input is an integer (not data type, passed data type is String). Printing e
       rror if not
        if(!objectIn.validateInt(inputInt))
        {
            System.out.print("\n");
            System.out.println("You have to type in only integers in order to perform operation. Try a
            gain");
            System.out.print("\n");
            choice4(objectIn);
        }

        // Validating that input is within given parameters (0 - 10,000, no more than MaxCoinInput.
        if (Integer.parseInt(inputInt) >= 0 && Integer.parseInt(inputInt) <= 10000 &&
       Integer.parseInt(inputInt) <= objectIn.getMaxCoinIn())
        {
            // If validated, setting up value
            objectIn.setMinCoinIn(Integer.parseInt(inputInt));
            System.out.print("\n");
            System.out.println("The minimum coin input has been setup to " + inputInt + " coins");
            System.out.print("\n");
        }

        // In case input invalid, explaining error and prompting again
        else
        {
            System.out.print("\n");
            System.out.println("Due to brief provided, the minimum and maximum coin input should be in
           range 0 - 10,000 inclusive, and the minimum input "
                   + "should not be more than the maximum coin input. Try again, and set value from t
            his range as in input");
            System.out.print("\n");
        }
        choice4(objectIn);
    }

    // Case 4.2, set maximum coin input
    else if(input.equalsIgnoreCase("3"))
    {
        // Explaining user which input is allowed and taking input
        System.out.print("Type in the maximum coin input you would like to set. Should be in range 0 -
        10,000 inclusive, "
                + "and no less than the minimum coin input: ");
        String inputInt = sc.next();

        // Validating that input is an integer (not data type, passed data type is String). Printing e
       rror if not
        if(!objectIn.validateInt(inputInt))
        {
            System.out.print("\n");
```

```
                System.out.println("You have to type in only integers in order to perform operation. Try a
                gain");
                System.out.print("\n");
                choice4(objectIn);
            }

            // Validating that input is within given parameters (0 - 10,000, no less than MinCoinInput.
            if (Integer.parseInt(inputInt) >= 0 && Integer.parseInt(inputInt) <= 10000 &&
           Integer.parseInt(inputInt) >= objectIn.getMinCoinIn())
            {
                // If validated, setting up value
                objectIn.setMaxCoinIn(Integer.parseInt(inputInt));
                System.out.print("\n");
                System.out.println("The maximum coin input has been setup to " + inputInt + " coins");
                System.out.print("\n");
            }

            // In case input invalid, explaining error and prompting again
            else
            {
                System.out.print("\n");
                System.out.println("Due to brief provided, the minimum and maximum coin input should be in
                range 0 - 10,000 inclusive, and the maximum input "
                        + "should not be less than the minimum coin input. Try again, and set value from t
                his range as in input");
                System.out.print("\n");
            }
            choice4(objectIn);
        }

        // Case 4.4, exit to main menu
        else if(input.equalsIgnoreCase("4"))
        {
            startMenu(objectIn);
            sc.close(); // Closing scanner to avoid memory leak
        }

        // Handling invalid input in case 4 menu
        else
        {
            System.out.println("\nInvalid input. Please type number 1 - 4");
            System.out.println("Restarting program\n\n");
           choice4(objectIn); // Recursive call to get input 1 - 4, and if not provided keep prompting us
           er
        }
    }

}
```

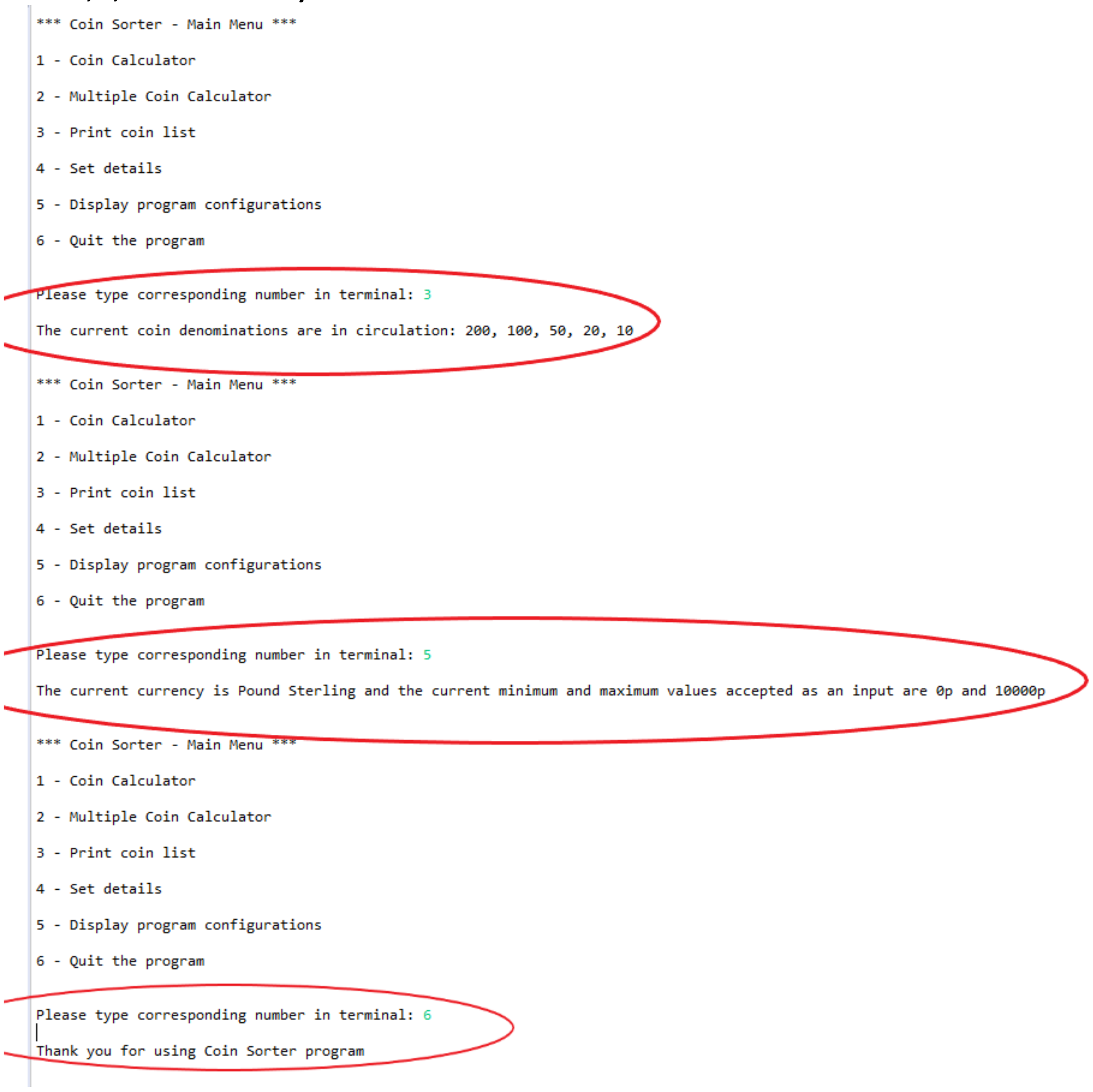
Evidence for TestCoinSorter class

This class can be launched from terminal
Main menu printed, with invalid user input gives error and prompts again



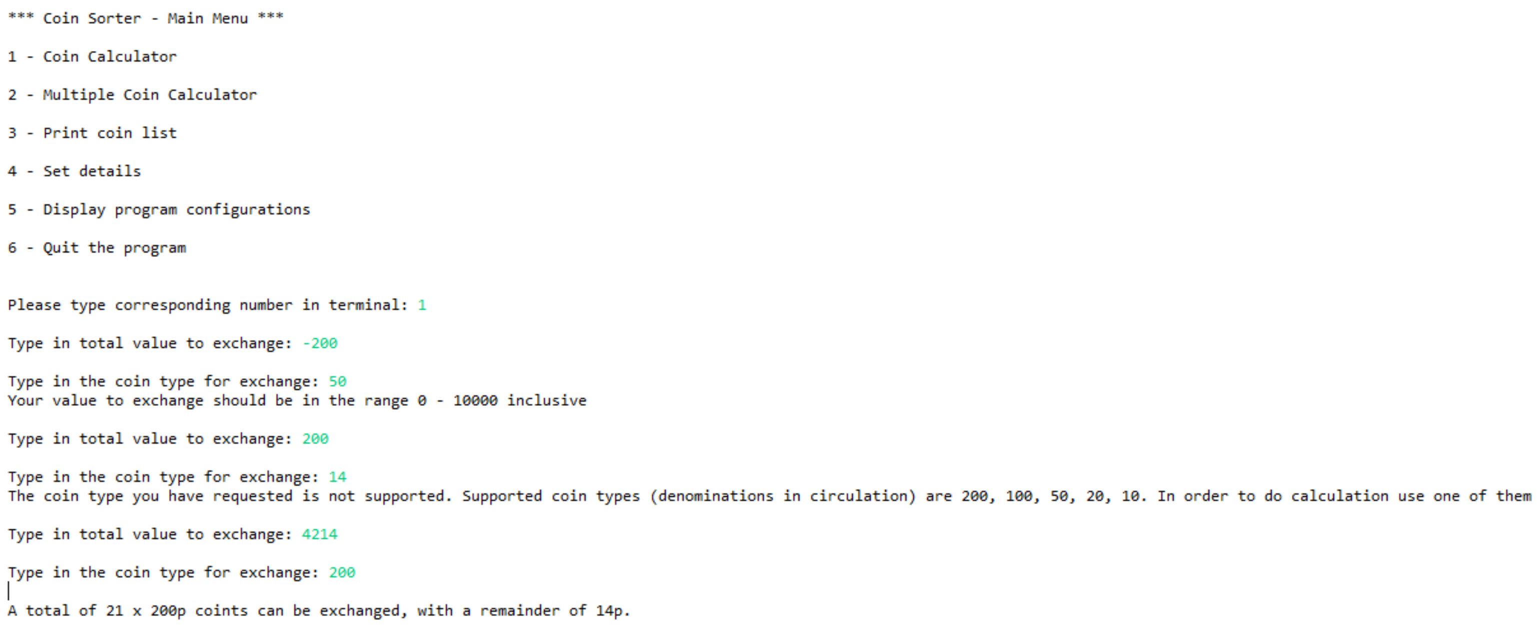
```
*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program


Please type corresponding number in terminal: 543

Invalid input. Please type number 1 - 6
Restarting program



*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program


Please type corresponding number in terminal: abc

Invalid input. Please type number 1 - 6
Restarting program



*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program


Please type corresponding number in terminal:
```

Cases 3, 5, 6 work correctly



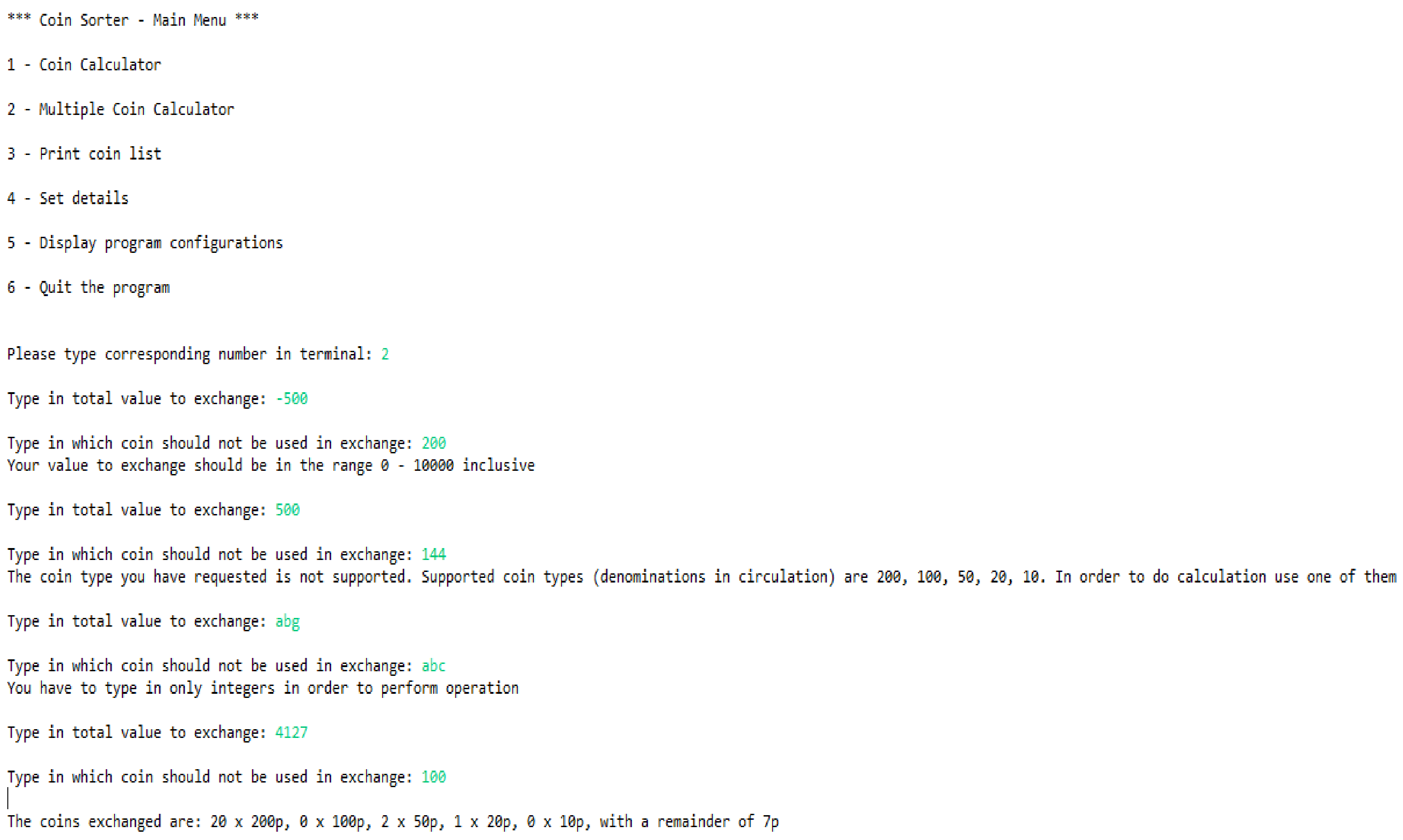
```
*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 3

The current coin denominations are in circulation: 200, 100, 50, 20, 10

*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 5

The current currency is Pound Sterling and the current minimum and maximum values accepted as an input are 0p and 10000p

*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 6

Thank you for using Coin Sorter program
```

Case 1 works correctly and it validates user input



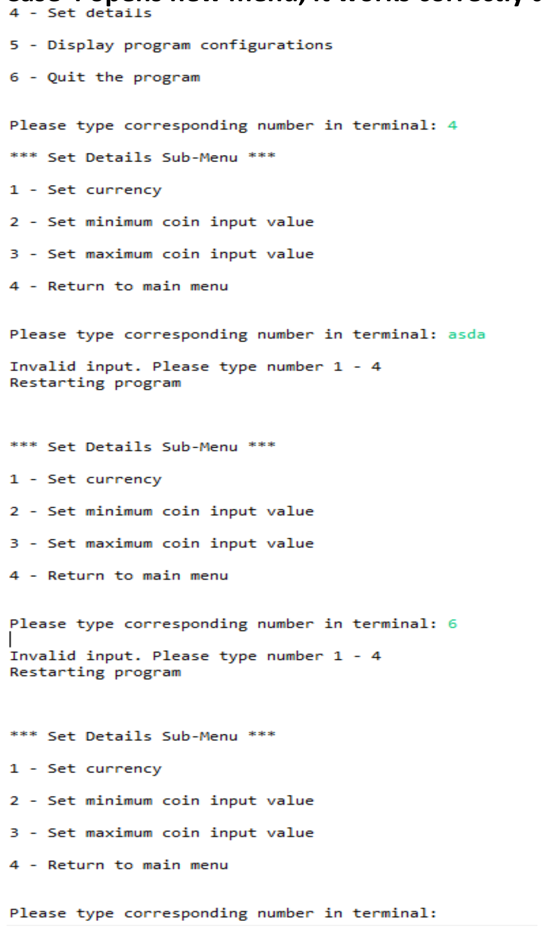
```
*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 1

Type in total value to exchange: -200

Type in the coin type for exchange: 50
Your value to exchange should be in the range 0 - 10000 inclusive

Type in total value to exchange: 200

Type in the coin type for exchange: 14
The coin type you have requested is not supported. Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10. In order to do calculation use one of them

Type in total value to exchange: 4214

Type in the coin type for exchange: 200

A total of 21 x 200p coints can be exchanged, with a remainder of 14p.
```

Case 2 works correctly and it validates user input



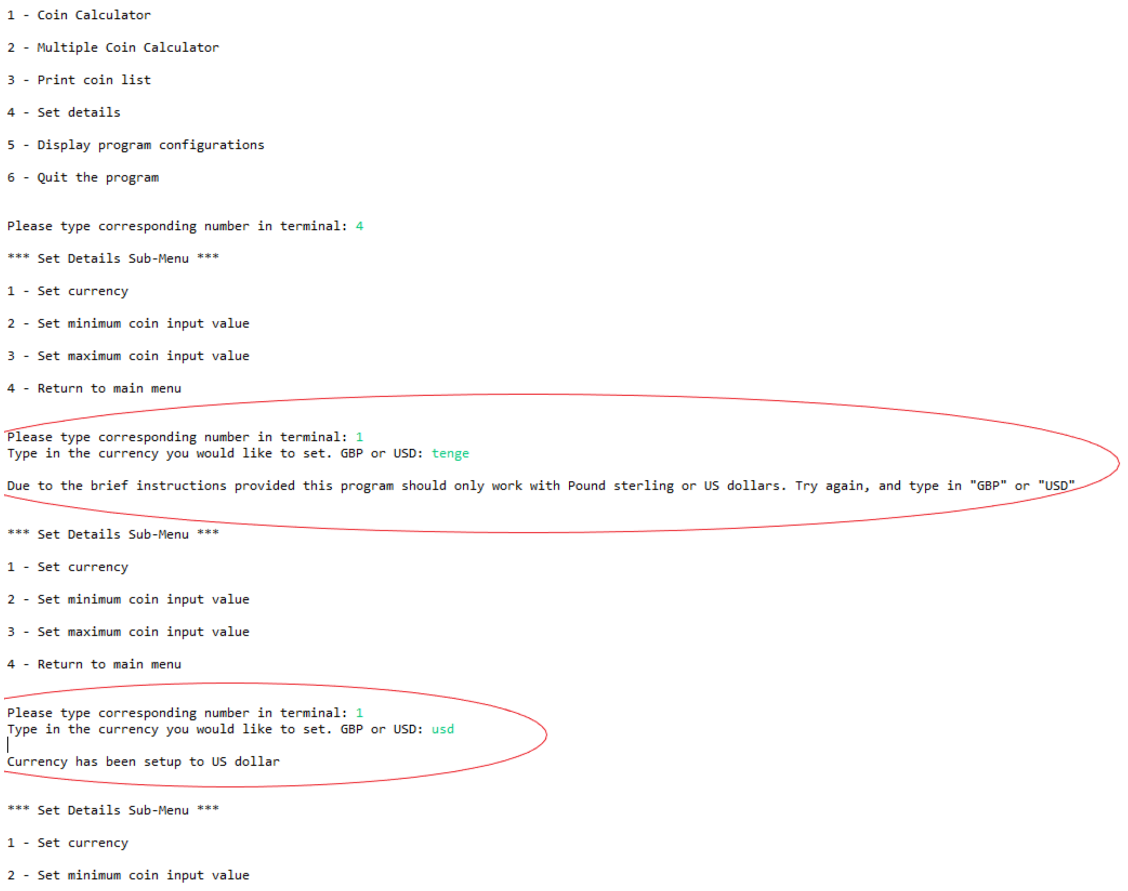
```
*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program


Please type corresponding number in terminal: 2

Type in total value to exchange: -500

Type in which coin should not be used in exchange: 200
Your value to exchange should be in the range 0 - 10000 inclusive

Type in total value to exchange: 500

Type in which coin should not be used in exchange: 144
The coin type you have requested is not supported. Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10. In order to do calculation use one of them

Type in total value to exchange: abg

Type in which coin should not be used in exchange: abc
You have to type in only integers in order to perform operation

Type in total value to exchange: 4127

Type in which coin should not be used in exchange: 100

The coins exchanged are: 20 x 200p, 0 x 100p, 2 x 50p, 1 x 20p, 0 x 10p, with a remainder of 7p
```

Case 4 opens new menu, it works correctly and it validates user input, prompts user again if input is invalid



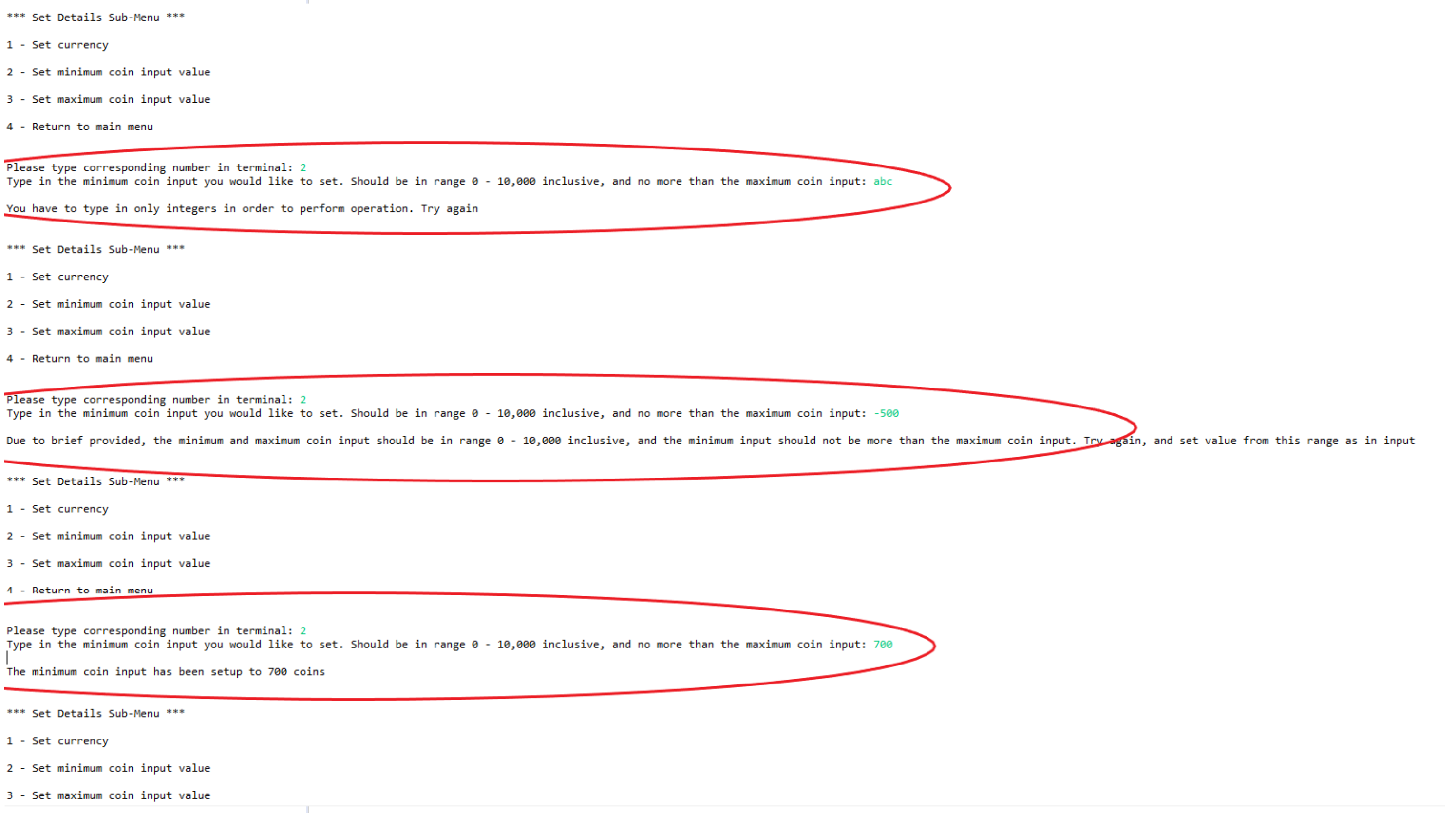
```
4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 4

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: asda

Invalid input. Please type number 1 - 4
Restarting program


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: 6

Invalid input. Please type number 1 - 4
Restarting program


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal:
```

Case 4.1 opens set currency menu, it works correctly and it validates user input. If invalid, returns error and instructions with available currencies (available currencies were provided to me by course coordinator: US dollar and Pound Sterling)



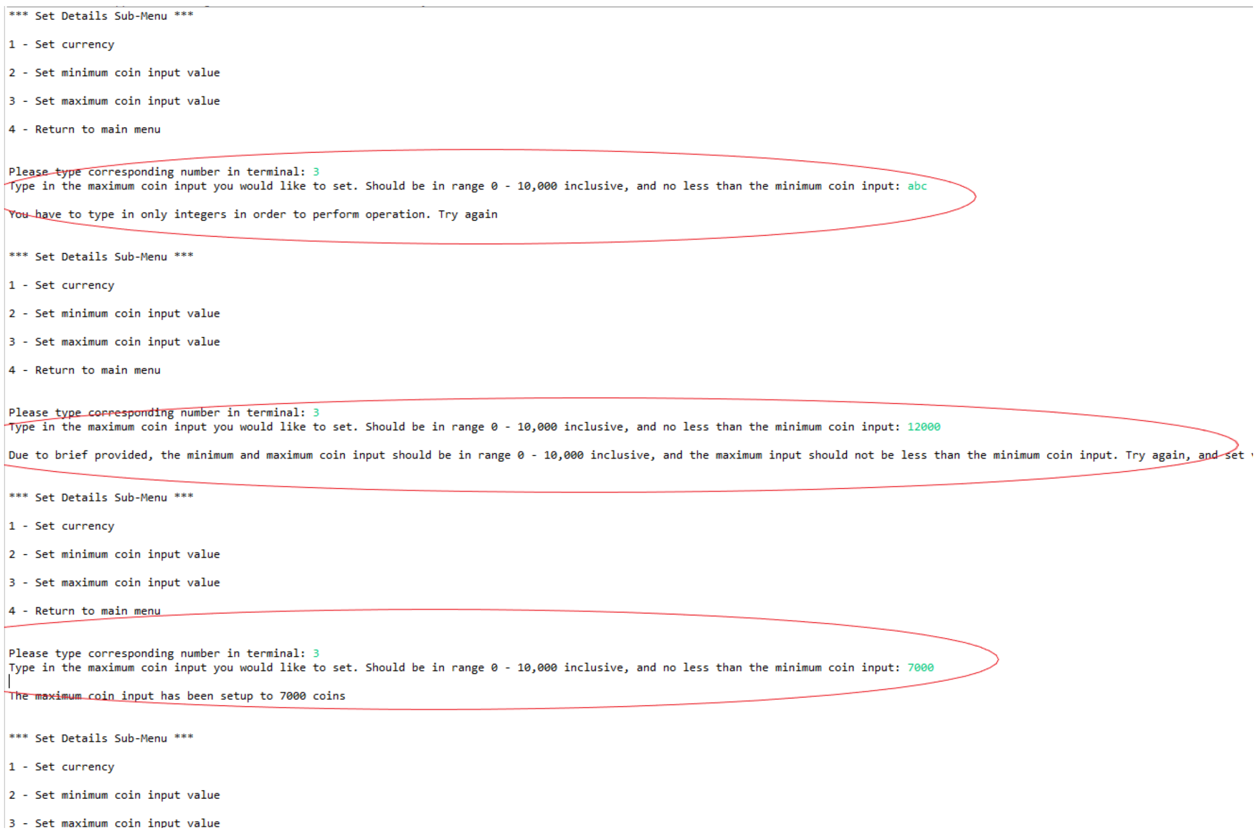
```
1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program


Please type corresponding number in terminal: 4

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu


Please type corresponding number in terminal: 1
Type in the currency you would like to set. GBP or USD: tenge

Due to the brief instructions provided this program should only work with Pound sterling or US dollars. Try again, and type in "GBP" or "USD"


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu


Please type corresponding number in terminal: 1
Type in the currency you would like to set. GBP or USD: usd

Currency has been setup to US dollar


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value
```

Case 4.2 opens set min coin input value menu, it works correctly and it validates user input. If invalid, returns error and instructions with available range of input



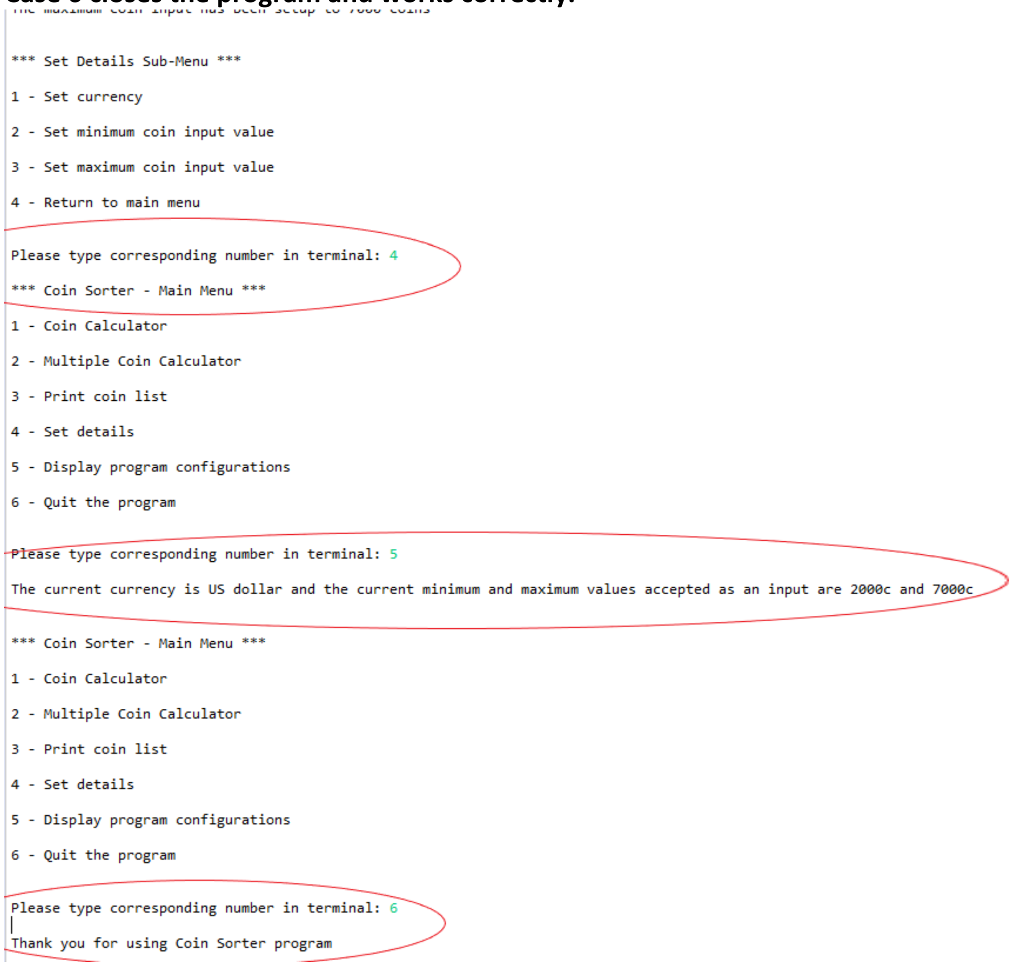
```
*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu


Please type corresponding number in terminal: 2
Type in the minimum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no more than the maximum coin input: abc

You have to type in only integers in order to perform operation. Try again


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu


Please type corresponding number in terminal: 2
Type in the minimum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no more than the maximum coin input: -500

Due to brief provided, the minimum and maximum coin input should be in range 0 - 10,000 inclusive, and the minimum input should not be more than the maximum coin input. Try again, and set value from this range as in input


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu


Please type corresponding number in terminal: 2
Type in the minimum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no more than the maximum coin input: 700

The minimum coin input has been setup to 700 coins


*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value
```

Case 4.3 opens set max coin input value menu, it works correctly and it validates user input. If invalid, returns error and instructions with available range of input



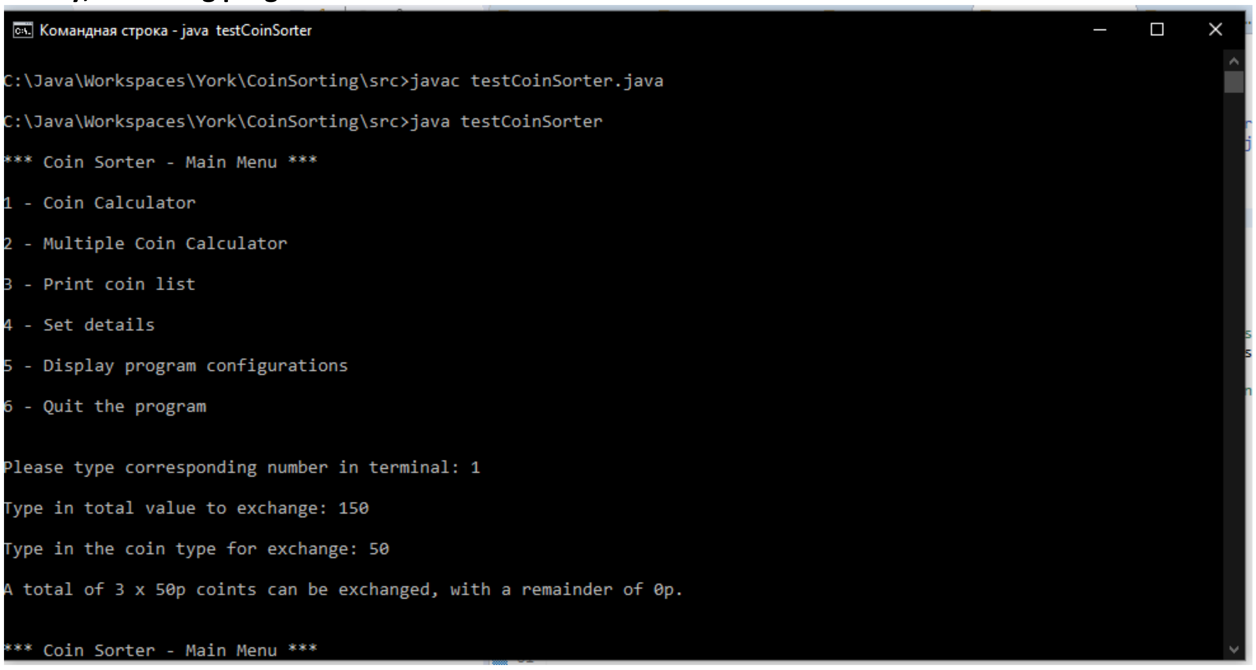
```
*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: 3
Type in the maximum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no less than the minimum coin input: abc

You have to type in only integers in order to perform operation. Try again

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: 3
Type in the maximum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no less than the minimum coin input: 12000

Due to brief provided, the minimum and maximum coin input should be in range 0 - 10,000 inclusive, and the maximum input should not be less than the minimum coin input. Try again, and set

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: 3
Type in the maximum coin input you would like to set. Should be in range 0 - 10,000 inclusive, and no less than the minimum coin input: 7000
|
The maximum coin input has been setup to 7000 coins

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value
```

Case 4.4 returns to main menu. Also, we try case 5 to see if changes after manilupating setters have been applied. Case 6 closes the program and works correctly.



```
The maximum coin input has been setup to 7000 coins

*** Set Details Sub-Menu ***

1 - Set currency

2 - Set minimum coin input value

3 - Set maximum coin input value

4 - Return to main menu

Please type corresponding number in terminal: 4

*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 5

The current currency is US dollar and the current minimum and maximum values accepted as an input are 2000c and 7000c

*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 6
|
Thank you for using Coin Sorter program
```

Finally, launching program from terminal



```
C:\Java\Workspaces\York\CoinSorting\src>javac testCoinSorter.java

C:\Java\Workspaces\York\CoinSorting\src>java testCoinSorter

*** Coin Sorter - Main Menu ***

1 - Coin Calculator

2 - Multiple Coin Calculator

3 - Print coin list

4 - Set details

5 - Display program configurations

6 - Quit the program

Please type corresponding number in terminal: 1

Type in total value to exchange: 150

Type in the coin type for exchange: 50

A total of 3 x 50p coints can be exchanged, with a remainder of 0p.

*** Coin Sorter - Main Menu ***
```

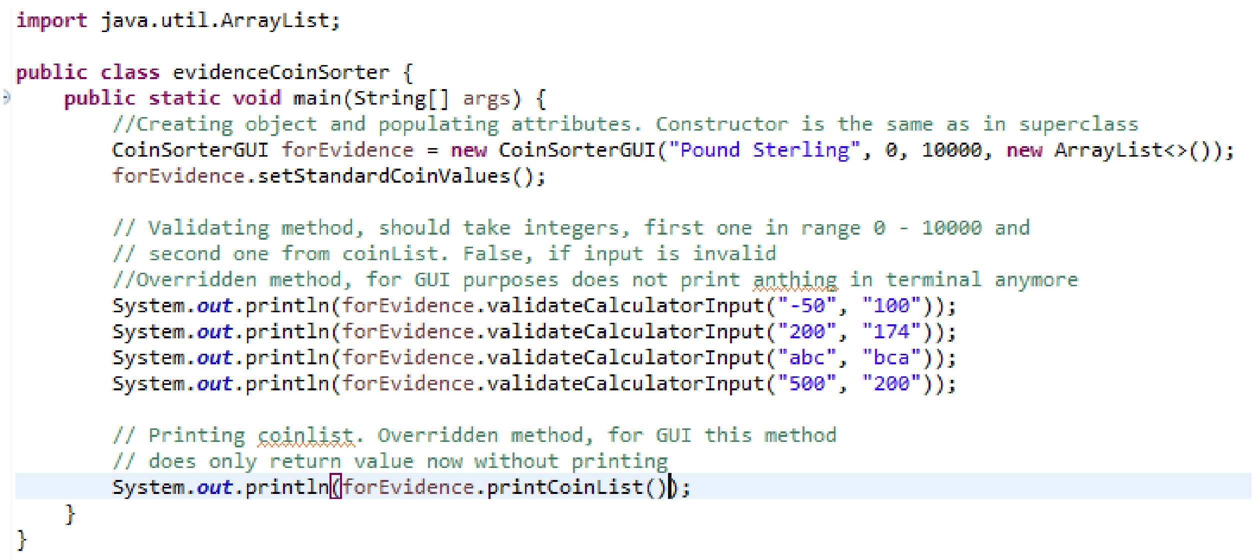
CoinSorterGUI class source code

```java
import java.util.ArrayList;

/**
 * @author anonymous
 * This class extends original CoinSorter class, and overrides some of its methods in order for them to
 * work properly with GUI version of program instead of terminal version of program
 */
public class CoinSorterGUI extends CoinSorter
{
    // Constructors duplicate constructors from superclass
    public CoinSorterGUI(String currencyIn, int minCoinIn, int maxCoinIn, ArrayList<Integer> coinListIn)
    {
        super(currencyIn, minCoinIn, maxCoinIn, coinListIn);
    }

    // Constructors duplicate constructors from superclass
    public CoinSorterGUI()
    {
        super();
    }

    /*
     * Method was overridden in order to adapt it to GUI instead of terminal output
     */
    @Override
    public boolean validateCalculatorInput(String ValueToExchangeIn, String coinTypeIn) {
        // Firstly, validating if values are numbers
        if (!validateInt(ValueToExchangeIn) || !validateInt(coinTypeIn))
        {
            return false;
        }

        // Secondly, validating value to exchange
        if(Integer.parseInt(ValueToExchangeIn) < getMinCoinIn() ||
        Integer.parseInt(ValueToExchangeIn) > getMaxCoinIn())
        {
            return false;
        }

        // Thridly, validating coin type
        for (int i = 0; i < getCoinList().size(); i ++)
        {
            if (getCoinList().get(i) != Integer.parseInt(coinTypeIn))
            {
                continue;
            }
            else return true;
        }
        return false;
    }


    /*
     * Method was overridden in order to adapt it to GUI instead of terminal output
     */
    @Override
    public String printCoinList() {
        String temp = "The current coin denominations are in circulation: ";
        if (getCoinList() != null && getCoinList().size() > 0) // evaluating if there are elements to disp
        lay from coinList
        {
             for (int i = 0; i < getCoinList().size(); i ++)// iterating through all elements of coinList
             and adding them to temporary String
            {
                temp += getCoinList().get(i);
                temp += ", ";
            }
            temp = temp.substring(0, temp.length() - 2); // removing last ", "
            return temp;
        }
        temp += "none"; // this case activated in case coinList is empty
        return temp;
    }

}
```

Evidence for CoinSorterGUI class

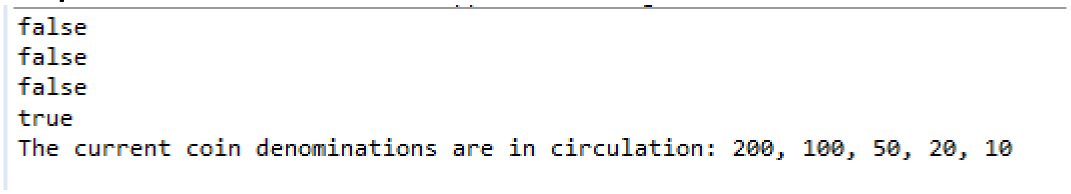
This class extends CoinSorter, and takes majority of logic from it and the same constructor. Additionally, it overrides several methods and adopts them to GUI program instead of terminal program.

Code for evidence



```java
import java.util.ArrayList;

public class evidenceCoinSorter {
    public static void main(String[] args) {
        //Creating object and populating attributes. Constructor is the same as in superclass
        CoinSorterGUI forEvidence = new CoinSorterGUI("Pound Sterling", 0, 10000, new ArrayList<>());
        forEvidence.setStandardCoinValues();

        // Validating method, should take integers, first one in range 0 - 10000 and
        // second one from coinList. False, if input is invalid
        //Overridden method, for GUI purposes does not print anthing in terminal anymore
        System.out.println(forEvidence.validateCalculatorInput("-50", "100"));
        System.out.println(forEvidence.validateCalculatorInput("200", "174"));
        System.out.println(forEvidence.validateCalculatorInput("abc", "bca"));
        System.out.println(forEvidence.validateCalculatorInput("500", "200"));

        // Printing coinlist. Overridden method, for GUI this method
        // does only return value now without printing
        System.out.println(forEvidence.printCoinList());
    }
}
```

Output for evidence



```
false
false
false
true
The current coin denominations are in circulation: 200, 100, 50, 20, 10
```

TestCoinSorterGUI class source code

```java
import java.util.ArrayList;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.Background;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import javafx.geometry.Pos;

/**
 * @author anonymous
 * This class defines GUI version of menu, initialises all necessary objects and
starts the program
 * Can be complied and launched both in IDE and terminal
 */
```

```
public class TestCoinSorterGUI extends Application {

    // Initialising instance, passing standard parameters according to brief. Int
erface list initialised as an array list
    CoinSorterGUI user1 = new CoinSorterGUI("Pound Sterling", 0, 10000,
    new ArrayList<>());

    // This method defines main menu and starts it
    public void start(Stage stage)
    {
        // Populating array list parameter according to brief
        user1.setStandardCoinValues();

        // Node with output text. Invisible unless text will be set
        Label outputTextLabel= new Label();
        outputTextLabel.setTextFill(Color.BLACK);
        outputTextLabel.setFont(Font.font("Arial", 15));
        outputTextLabel.setText("Coin Sorter - Main Menu");

        // Option 1, Coin Calculator. Defining button and action. Called in separ
ate window and using separate method below
        Button buttonCoinCalculator = new Button();
        buttonCoinCalculator.setText("Coin Calculator");
        buttonCoinCalculator.setOnAction(e -> coinCalculatorMenu());

        // Option 2, Multiple Coin Calculator. Defining button and action. Called
        in separate window and using separate method below
        Button buttonMultiCoinCalculator = new Button();
        buttonMultiCoinCalculator.setText("Multiple Coin Calculator");
        buttonMultiCoinCalculator.setOnAction(e -> multiCoinCalculatorMenu());

        // Option 3, Print coin list. Defining button and action
        Button buttonPrintCoinList = new Button();
        buttonPrintCoinList.setText("Print coin list");
        buttonPrintCoinList.setOnAction(e -
> outputTextLabel.setText(user1.printCoinList()));

        // Option 4, Set details. Opens separate window which defined in separate
        method below
        Button buttonSetDetails = new Button();
        buttonSetDetails.setText("Set details");
        buttonSetDetails.setOnAction(e -> setDetailsMenu());

        // Option 5, Display program configurations. Defining button and action
        Button buttonDisplayConfig = new Button();
        buttonDisplayConfig.setText("Display program configurations");
        buttonDisplayConfig.setOnAction(e
        -> outputTextLabel.setText(user1.displayProgramConfigs()));

        // Option 6, Quit the program, alternative way to close program window
        Button buttonQuitProgram = new Button();
        buttonQuitProgram.setText("Quit the program");
        buttonQuitProgram.setOnAction(e -> stage.close());

        // Grouping buttons together in horisontal box
        HBox mainMenuOptions = new HBox(10);
        mainMenuOptions.setAlignment(Pos.CENTER);
        mainMenuOptions.getChildren().addAll(buttonCoinCalculator,
       buttonMultiCoinCalculator, buttonPrintCoinList, buttonSetDetails,
       buttonDisplayConfig, buttonQuitProgram);

        // Creating root node and adding all elements from main menu to it
        VBox root = new VBox(10);
        root.setBackground(Background.EMPTY);
        root.setAlignment(Pos.CENTER);

        root.getChildren().addAll(mainMenuOptions, outputTextLabel);
        Scene scene = new Scene(root, 1000, 300, Color.WHITE);

        // Passing scene to the stage and starting menu application
        stage.setScene(scene);
        stage.setTitle("Coin Sorter");
        stage.show();

    }

    // Used to launch application from IDE
    public static void main(String[] args)
     {
         launch(args);
     }

    /*Coin calculator menu defined here as a separate stage
      *
      */
    public void coinCalculatorMenu()
     {
         // Initialising new stage
         Stage coinCalculatorMenu = new Stage();

         // Creating label for text and putting it VBox in order to be able to p
        osition in properly in the centre
         Label textLabel = new Label();
         textLabel.setTextFill(Color.BLACK);
         textLabel.setFont(Font.font("Arial", 20));
         textLabel.setText("Type in total value to exchange in the first field
         and the coin type for exchange in the second field");
         VBox textBox = new VBox();
         textBox.getChildren().addAll(textLabel);
         textBox.setAlignment(Pos.CENTER);

         // Text field to take first input
         TextField textFieldTotalValue = new TextField();
         textFieldTotalValue.setMaxWidth(250);
         // Text field to take second input
         TextField textFieldCoinType = new TextField();
         textFieldCoinType.setMaxWidth(250);

         // Creating and defining action for calculation button
         Button buttonCalculateCoins = new Button();
         buttonCalculateCoins.setText("Calculate");
         buttonCalculateCoins.setOnAction(e -> {
             // Validatin data, if correct, performing calculations and putting
             them in textLabel
             if (user1.validateCalculatorInput(textFieldTotalValue.getText(),
             textFieldCoinType.getText()))
             {
                 textLabel.setText(user1.coinCalculator(Integer.parseInt(textFie
                 ldTotalValue.getText()), Integer.parseInt(textFieldCoinType.get
                 Text())));
                 textLabel.setTextFill(Color.BLACK);
             }
             // Otherwise, requesting the correct input
             else
             {
                 textLabel.setText("ERROR:\nYou have to type in only integers.\nY
                 our value to exchange should be in the range " + user1.getMinCo
                 inIn() + " - " + user1.getMaxCoinIn() + " inclusive. \n"
                        + "Supported coin types (denominations in circulation) ar
                 e 200, 100, 50, 20, 10. \nType in different values and try agai
                 n");
                 textLabel.setTextFill(Color.RED);
             }
         });

         // Creating root node and addint elements to it
         VBox root = new VBox(10);
         root.setSpacing(10);
         root.setAlignment(Pos.CENTER);
         root.getChildren().addAll(textLabel, textFieldTotalValue,
         textFieldCoinType, buttonCalculateCoins);

         // Creating and launching scene
         Scene scene = new Scene(root, 1000, 300);
         coinCalculatorMenu.setScene(scene);
         coinCalculatorMenu.setTitle("Coin calculator");
         coinCalculatorMenu.show();

     }

    /*Multiple Coin calculator menu defined here as a separate stage
      *
      */
```

```java
public void multiCoinCalculatorMenu()
 {
     // Initialising object of Stage
     Stage multiCoinCalculatorMenu = new Stage();

     // Creating new text label in order to print output and errors
     Label textLabel = new Label();
     textLabel.setTextFill(Color.BLACK);
     textLabel.setFont(Font.font("Arial", 20));
     textLabel.setText("Type in total value to exchange in the first field
     and which coin should not be used in exchange in the second field");
     // Using VBox in order to be able to centrally allign output test in
     the GUI
     VBox textBox = new VBox();
     textBox.getChildren().addAll(textLabel);
     textBox.setAlignment(Pos.CENTER);

     // Creating text field in order to take input 1
     TextField textFieldTotalValue = new TextField();
     textFieldTotalValue.setMaxWidth(250);

     // Creating text field in order to take input 2
     TextField textFieldCoinTypeToExclude = new TextField();
     textFieldCoinTypeToExclude.setMaxWidth(250);

     // Creating and assigning action to button
     Button buttonCalculateCoins = new Button();
     buttonCalculateCoins.setText("Calculate");
     buttonCalculateCoins.setOnAction(e -> {

         // Validating user input and performing calculation
         if (user1.validateCalculatorInput(textFieldTotalValue.getText(),
         textFieldCoinTypeToExclude.getText()))
         {
              textLabel.setText(user1.multiCoinCalculator(Integer.parseInt(t
              extFieldTotalValue.getText()), Integer.parseInt(textFieldCoinT
              ypeToExclude.getText())));
             textLabel.setTextFill(Color.BLACK);
         }

        // In case of invalid input providing error message
        with possible problems
         else
         {
             textLabel.setText("ERROR:\nYou have to type in only integers.
             \nYour value to exchange should be in the range " + user1.getMi
             nCoinIn() + " - " + user1.getMaxCoinIn() + " inclusive. \n"
             + "Supported coin types (denominations in circulation)
             are 200, 100, 50, 20, 10. \nType in different values and try
             again");
             textLabel.setTextFill(Color.RED);
```

```
            }
        });

        // Root node to group all nodes together
        VBox root = new VBox(10);
        root.setSpacing(10);
        root.setAlignment(Pos.CENTER);
        root.getChildren().addAll(textLabel, textFieldTotalValue,
        textFieldCoinTypeToExclude, buttonCalculateCoins);

        // And initialise and launch a scene
        Scene scene = new Scene(root, 1200, 300);
        multiCoinCalculatorMenu.setScene(scene);
        multiCoinCalculatorMenu.setTitle("Multiple coin calculator");
        multiCoinCalculatorMenu.show();

    }

    /* Set details menu defined here as a separate stage
      *
      */
    public void setDetailsMenu()
    {
        // Initialising stage
        Stage setDetailsMenu = new Stage();

        // Creating text label and putting in the box for allignment functions
        Label textLabel = new Label();
        textLabel.setTextFill(Color.BLACK);
        textLabel.setFont(Font.font("Arial", 20));
        textLabel.setText("Type in value and click button to set values");
        VBox textBox = new VBox();
        textBox.getChildren().addAll(textLabel);
        textBox.setAlignment(Pos.CENTER);

        // Creating new text field for input
        TextField textFieldSetlValue = new TextField();
        textFieldSetlValue.setMaxWidth(250);

        // As terminal version of program, case 4 offers 4 extra options in sepa
        rate window

        // Case 4.1, set currency. Creating button and validating input
        Button buttonSetCurrency = new Button();
        buttonSetCurrency.setText("Set currency");
        buttonSetCurrency.setOnAction(e -> {
            // As described in brief and by course coordinator, the only
           possible currencies are GBP and USD
            // Validating if text is requesting to change currency to GBP
            if (textFieldSetlValue.getText().trim().equalsIgnoreCase("sterling p
ound") || textFieldSetlValue.getText().trim().equalsIgnoreCase("pound") || textFi
```

```
eldSetlValue.getText().trim().equalsIgnoreCase("pound sterling") || textFieldSetl
Value.getText().trim().equalsIgnoreCase("GBP"))
                {
                    user1.setCurrency("Pound sterling");
                    textLabel.setText("Currency has been setup to Pound sterling");
                    textLabel.setTextFill(Color.BLACK);
                }

               // Validating if text is requesting to change currency to USD
                else if (textFieldSetlValue.getText().trim().equalsIgnoreCase("US do
llar") || textFieldSetlValue.getText().trim().equalsIgnoreCase("USD") || textFiel
dSetlValue.getText().trim().equalsIgnoreCase("Dollar") || textFieldSetlValue.getT
ext().trim().equalsIgnoreCase("dollar US"))
                {
                    user1.setCurrency("US dollar");
                    textLabel.setText("Currency has been setup to US dollar");
                    textLabel.setTextFill(Color.BLACK);
                }
                // Else providing error and giving user possible values to pass
                else
                {
                    textLabel.setText("ERROR:\nDue to the brief instructions provide
                    d this program \nshould only work with Pound sterling or US dol
                    lars. \nTry again, and type in \"Pound sterling\" or \"US dolla
                    r\"");
                    textLabel.setTextFill(Color.RED);
                }
            });

            // Case 4.2, set minimum input value
            Button buttonMinCoinIn = new Button();
            buttonMinCoinIn.setText("Set minimum coin input value");
            buttonMinCoinIn.setOnAction(e -> {

                // Validating input, if validated, printing sentence with result
                if (user1.validateInt(textFieldSetlValue.getText().trim()) &&
                 Integer.parseInt(textFieldSetlValue.getText().trim()) >= 0 &&
                 Integer.parseInt(textFieldSetlValue.getText().trim()) <= 10000 &&
                 Integer.parseInt(textFieldSetlValue.getText().trim()) <=
                 user1.getMaxCoinIn())
                {
                    user1.setMinCoinIn(Integer.parseInt
                   (textFieldSetlValue.getText().trim()));
                    textLabel.setText("The minimum coin input has been setup to " +
                    user1.getMinCoinIn() +  " coins");
                    textLabel.setTextFill(Color.BLACK);
                }

                // In case invalid input, giving error and explaining possible probl
ems, and asking to try again
                else
```

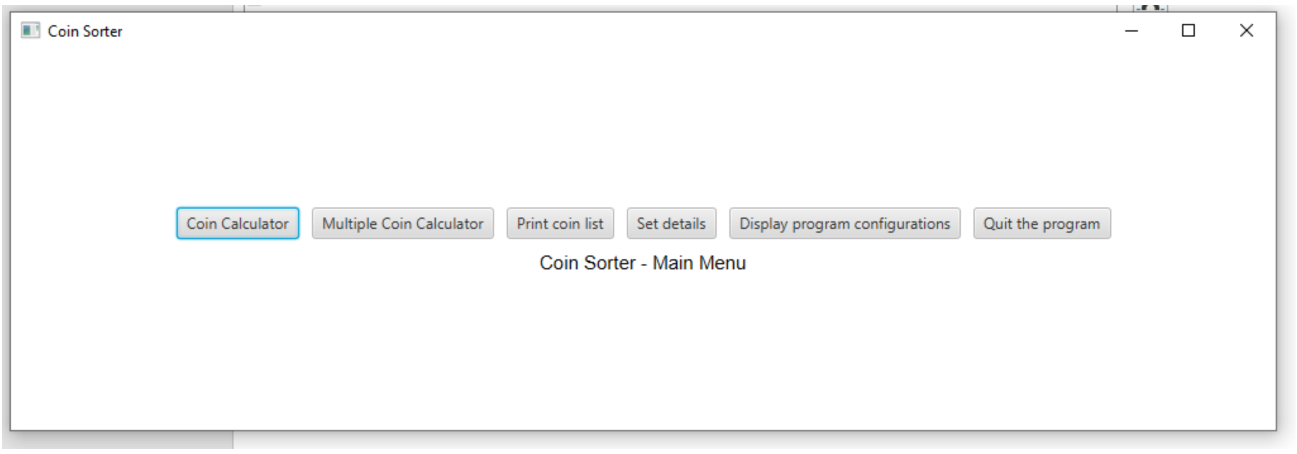
```
    {
        textLabel.setText("ERROR:\nDue to brief provided, input value sh
        ould be integers only. \nThe minimum and maximum coin input sho
        uld be in range 0 - 10,000 inclusive, \nand the minimum input s
        hould not be more than the maximum coin input. \nType in differ
        ent values and try again");
        textLabel.setTextFill(Color.RED);
    }
});

// Case 4.3 set maximum input value
Button buttonMaxCoinIn = new Button();
buttonMaxCoinIn.setText("Set maximum coin input value");
buttonMaxCoinIn.setOnAction(e -> {

    // Validating input, if validated, printing sentence with result
    if (user1.validateInt(textFieldSetlValue.getText().trim()) &&
    Integer.parseInt(textFieldSetlValue.getText().trim()) >= 0 &&
    Integer.parseInt(textFieldSetlValue.getText().trim()) <= 10000 &&
     Integer.parseInt(textFieldSetlValue.getText().trim()) >=
     user1.getMinCoinIn())
    {
        user1.setMaxCoinIn(Integer.parseInt
        (textFieldSetlValue.getText().trim()));
        textLabel.setText("The maximum coin input has been setup to " +
         user1.getMaxCoinIn() +  " coins");
        textLabel.setTextFill(Color.BLACK);
    }

    // In case invalid input, giving error and explaining possible
   problems, and asking to try again
    else
    {
        textLabel.setText("ERROR:\nDue to brief provided, input value sh
        ould be integers only. \nThe minimum and maximum coin input sho
        uld be in range 0 - 10,000 inclusive, \nand the maximum input s
        hould not be less than the minimum coin input. \nType in differ
        ent values and try again");
        textLabel.setTextFill(Color.RED);
    }
});

// Case 4.4, additional button to close window
Button buttonExitMenu = new Button();
buttonExitMenu.setText("Close window");
buttonExitMenu.setOnAction(e -> setDetailsMenu.hide());

// Grouping buttons together in horisontal node
HBox buttons = new HBox(10);
buttons.setSpacing(10);
buttons.setAlignment(Pos.CENTER);
```

```
                buttons.getChildren().addAll(buttonSetCurrency, buttonMinCoinIn,
                 buttonMaxCoinIn, buttonExitMenu);

                // Creating root node and adding all nodes to it
                VBox root = new VBox(10);
                root.setSpacing(10);
                root.setAlignment(Pos.CENTER);
                root.getChildren().addAll(textLabel, textFieldSet1Value, buttons);

                // Creating and launching scene
                Scene scene = new Scene(root, 1200, 300);
                setDetailsMenu.setScene(scene);
                setDetailsMenu.setTitle("Set details");
                setDetailsMenu.show();

            }

        }
```

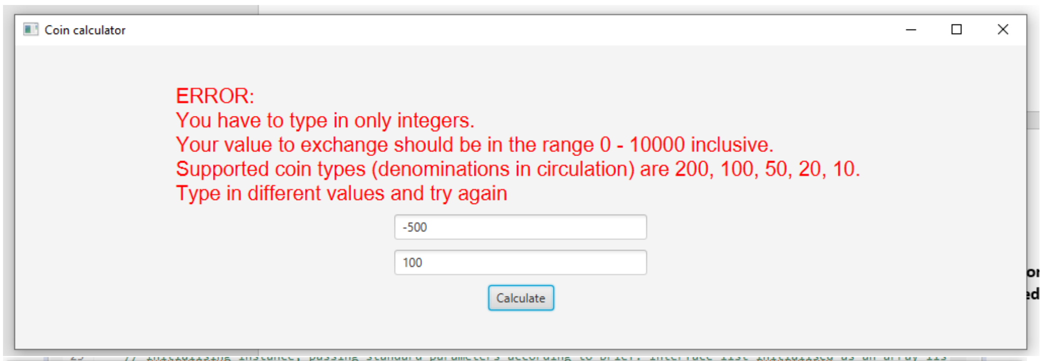
Evidence for TestCoinSorterGUI class

This class initialises all necessary objects, defines GUI version of program using JavaFX application and launches it. It takes logic for calculations from CoinSorter and CoinSorter GUI classes. Can be launched from terminal

Main menu launched



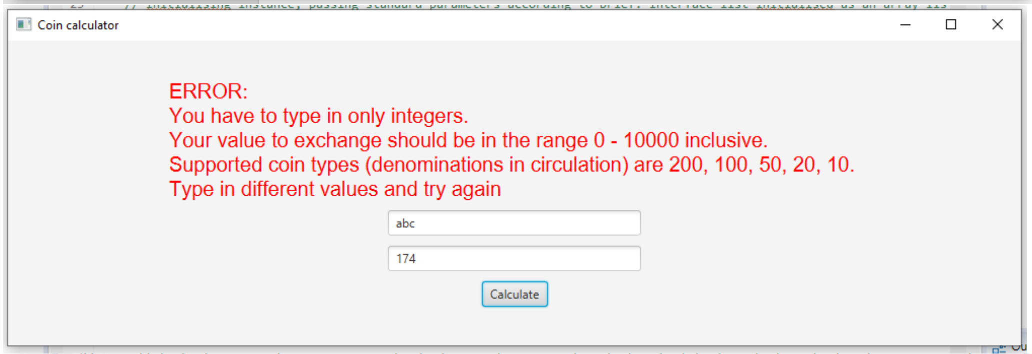
Coin calculator button launches new window, validates input and does calculations. If input is invalid, error message will appear with instructions for correct input



ERROR:
You have to type in only integers.
Your value to exchange should be in the range 0 - 10000 inclusive.
Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10.
Type in different values and try again

-500

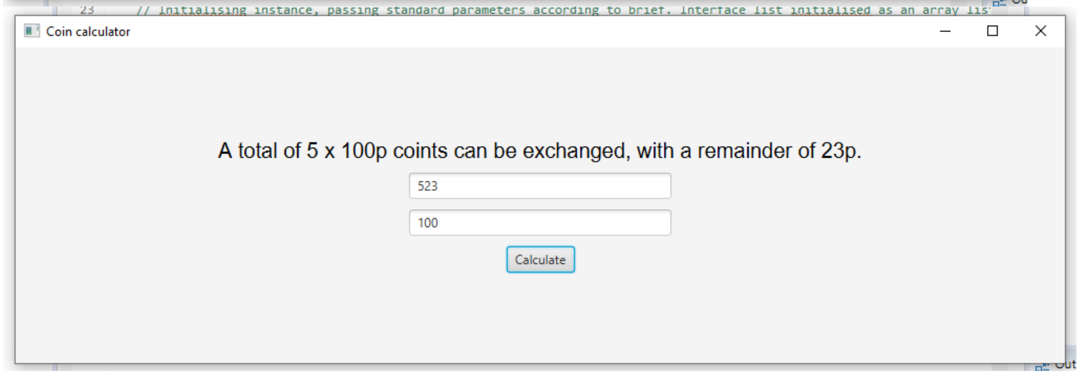
100



ERROR:
You have to type in only integers.
Your value to exchange should be in the range 0 - 10000 inclusive.
Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10.
Type in different values and try again

abc

174

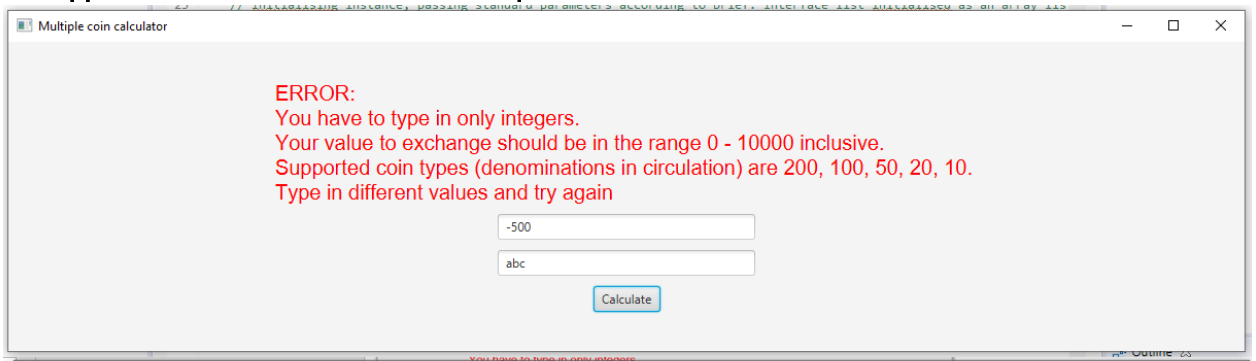


A total of 5 x 100p coints can be exchanged, with a remainder of 23p.

523

100

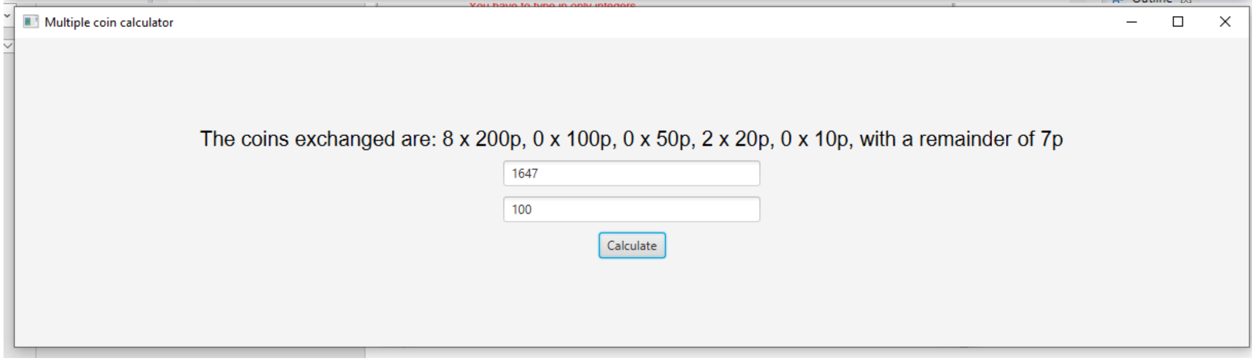
Multiple coin calculator button launches new window, validates input and does calculations. If input is invalid, error message will appear with instructions for correct input



ERROR:
You have to type in only integers.
Your value to exchange should be in the range 0 - 10000 inclusive.
Supported coin types (denominations in circulation) are 200, 100, 50, 20, 10.
Type in different values and try again

-500

abc

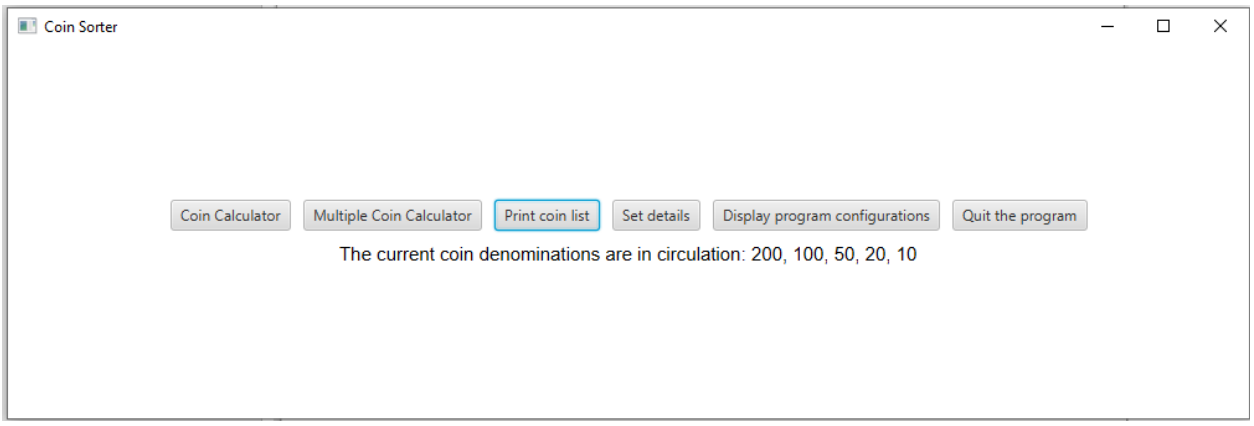


The coins exchanged are: 8 x 200p, 0 x 100p, 0 x 50p, 2 x 20p, 0 x 10p, with a remainder of 7p

1647

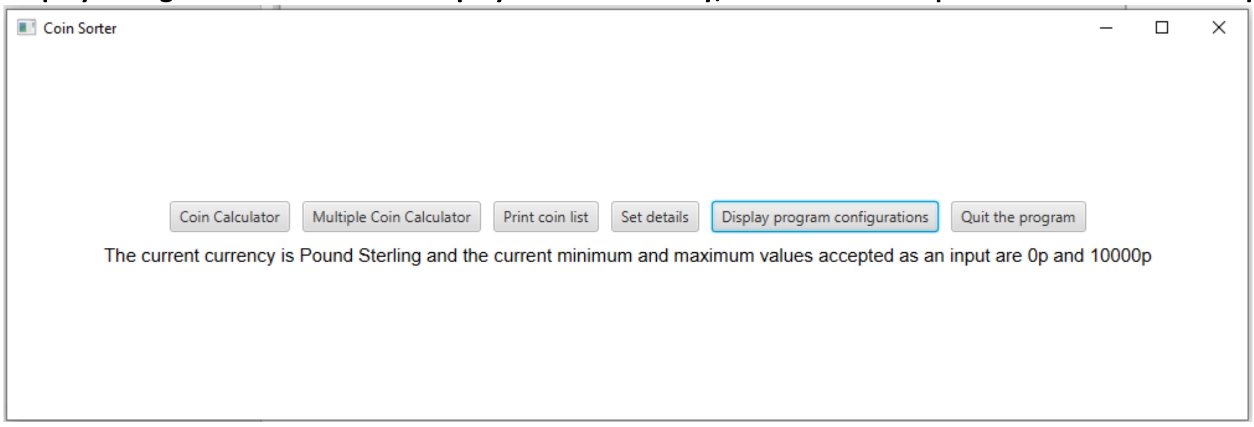
100

Print coin list button will display coin list that can be used



The current coin denominations are in circulation: 200, 100, 50, 20, 10

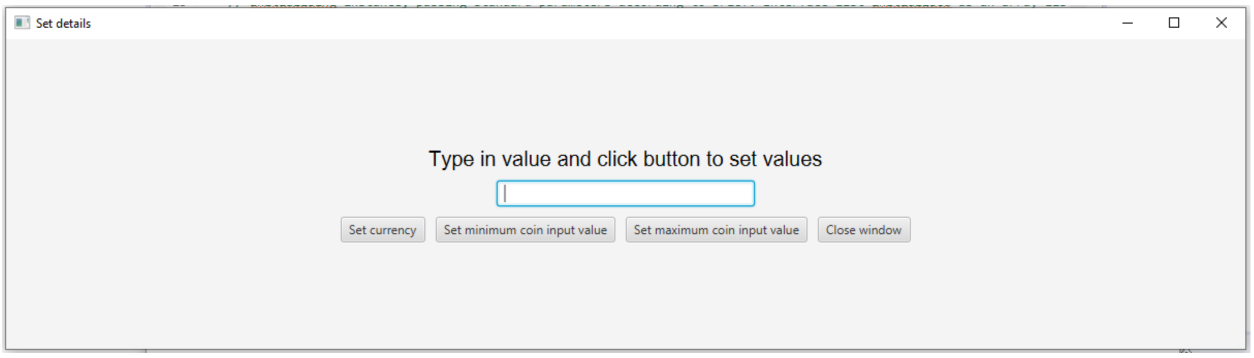
Display configurations button will display current currency, minimum coin input and maximum coin input



The current currency is Pound Sterling and the current minimum and maximum values accepted as an input are 0p and 10000p

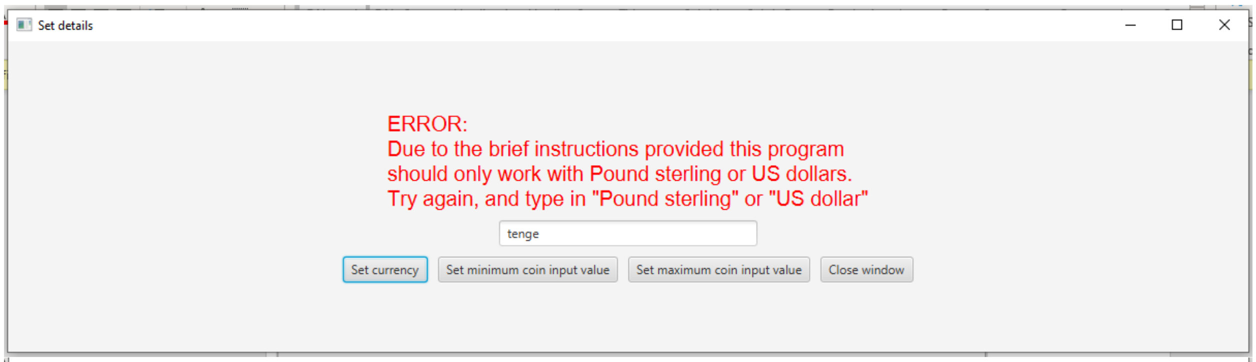
Quit the program button is extra way to close the program in the same way as cross on the top right of the window

Set details button will open another windows with setter options. It validates provided input for each button. Last button is closing the window in the same way as cross on the top right of the window



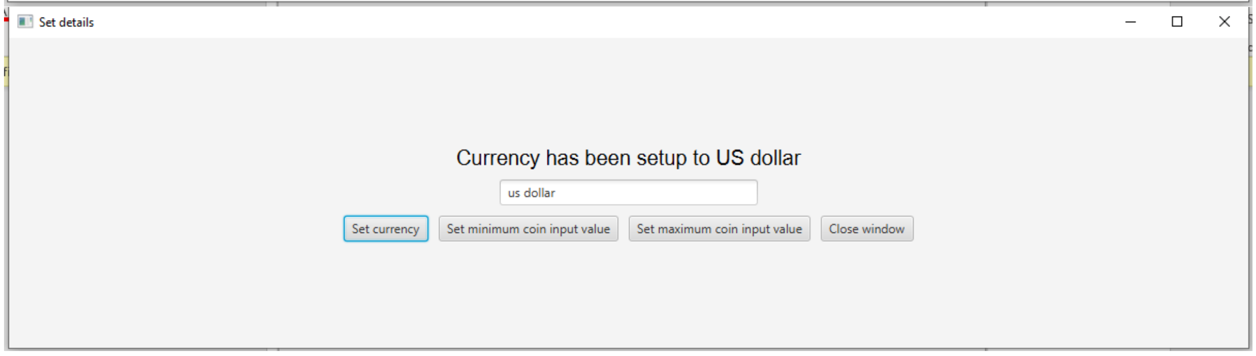
Type in value and click button to set values

Set currency button validates input, if valid sets new currency. If invalid, gives error with information which input can be valid


Set details

ERROR:
Due to the brief instructions provided this program
should only work with Pound sterling or US dollars.
Try again, and type in "Pound sterling" or "US dollar"

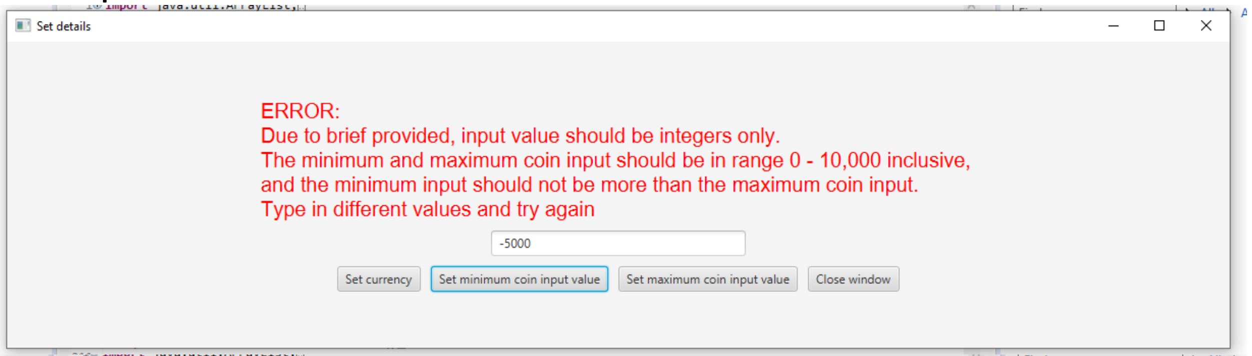
tenge


Set details

Currency has been setup to US dollar

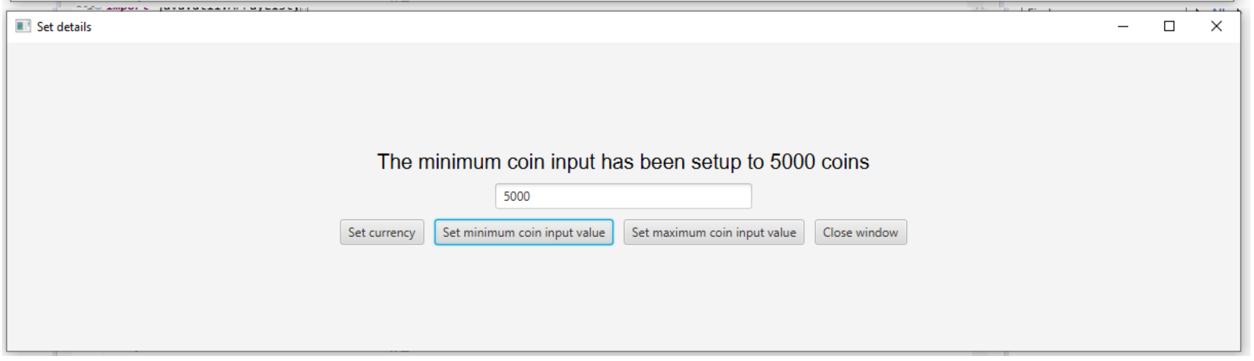
us dollar

Set minimum coin input value button validates input to check if it can be used as an integer, if it is within limits 0 – 10000 and if it is no more than maximum coin input value. If valid, value is set, otherwise, error message will be displayed with information which input can be valid


Set details

ERROR:
Due to brief provided, input value should be integers only.
The minimum and maximum coin input should be in range 0 - 10,000 inclusive,
and the minimum input should not be more than the maximum coin input.
Type in different values and try again

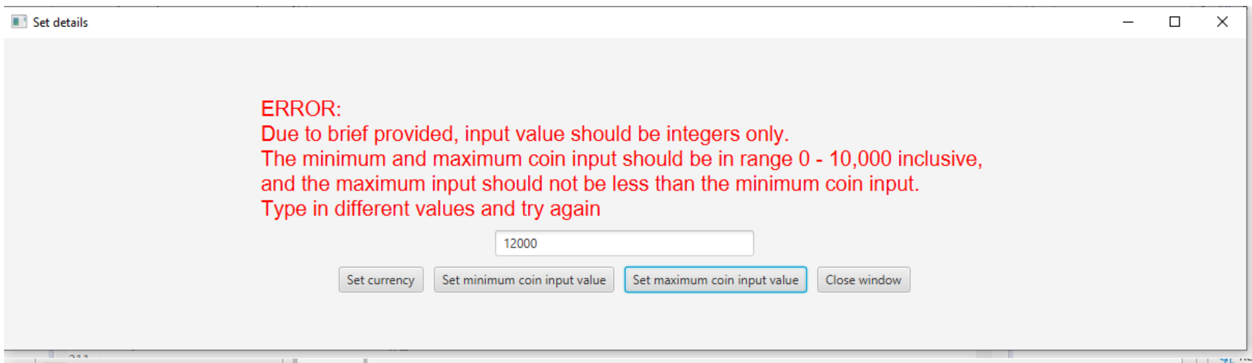
-5000


Set details

The minimum coin input has been setup to 5000 coins

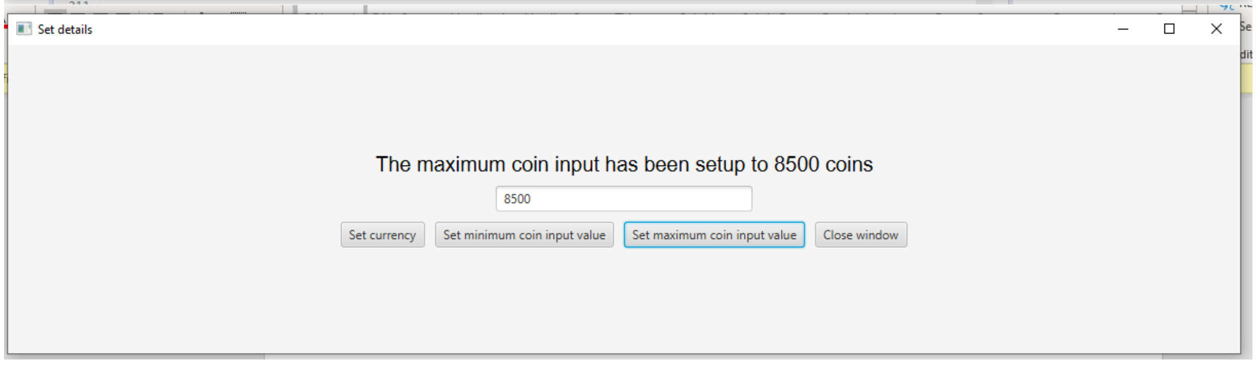
5000

Set maximum coin input value button validates input to check if it can be used as an integer, if it is within limits 0 – 10000 and if it is no more less than minimum coin input value. If valid, value is set, otherwise, error message will be displayed with information which input can be valid



ERROR:
Due to brief provided, input value should be integers only.
The minimum and maximum coin input should be in range 0 - 10,000 inclusive,
and the maximum input should not be less than the minimum coin input.
Type in different values and try again

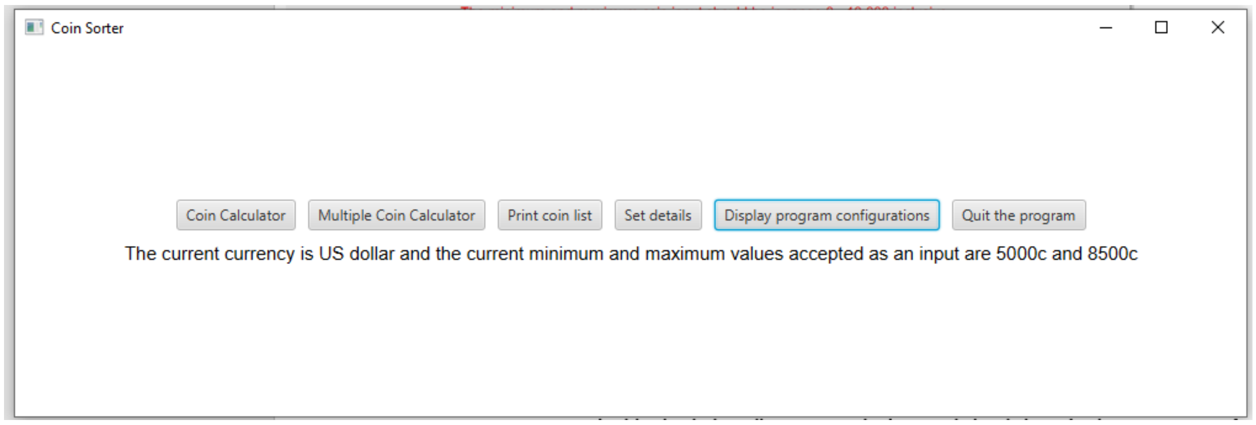
12000



The maximum coin input has been setup to 8500 coins

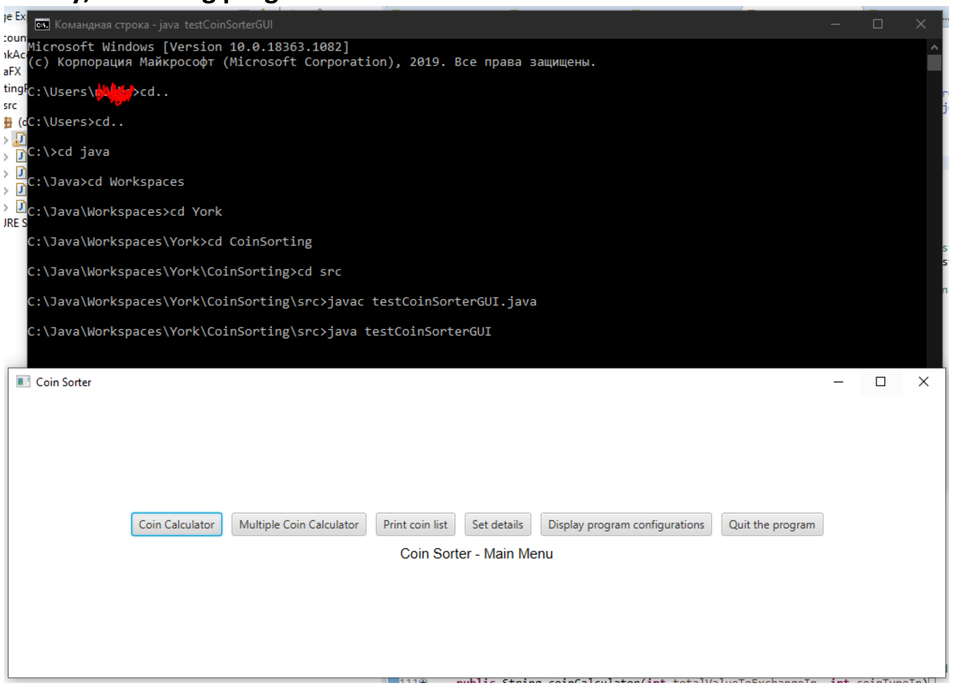
8500

Now we can return to main menu to double check that all setters worked properly by clicking display program configurations button



The current currency is US dollar and the current minimum and maximum values accepted as an input are 5000c and 8500c

Finally, launching program from terminal



```
Microsoft Windows [Version 10.0.18363.1082]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\[redacted]>cd..

C:\Users>cd..

C:\>cd java

C:\Java>cd Workspaces

C:\Java\Workspaces>cd York

C:\Java\Workspaces\York>cd CoinSorting

C:\Java\Workspaces\York\CoinSorting>cd src

C:\Java\Workspaces\York\CoinSorting\src>javac testCoinSorterGUI.java

C:\Java\Workspaces\York\CoinSorting\src>java testCoinSorterGUI
```

Coin Sorter - Main Menu