Practical Programming Assignment 1 (COM00141M)

[Week 3 Coin Sorter Assignment]

Part A :

CoinSort Source Code

```java
import java.util.List;

/**
* @author Student
*
* This class is to sort coins
*/
public class CoinSorter {

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

  public CoinSorter() {
    this.currency = "sterlin";
    this.minCoinIn = 0;
    this.maxCoinIn = 10000;
    this.coinList = List.of(200, 100, 50, 20, 10);
  }

  public CoinSorter(String currency, int minCoinIn, int maxCoinIn, List<Integer> coinList) {
    this.currency = currency;
    this.minCoinIn = minCoinIn;
    this.maxCoinIn = maxCoinIn;
    this.coinList = coinList;
  }

  /**
   * Gets currency
   *
   * @return the currency
   */
  public String getCurrency() {
    return currency;
  }

  /**
   * Sets Currency
   *
   * @param currency the currency to set
   */
```

```java
public void setCurrency(String currency) {
  this.currency = currency;
}

/**
 * Gets Minimum Coin Value
 *
 * @return the minCoinIn
 */
public int getMinCoinIn() {
  return minCoinIn;
}

/**
 * Sets Minimum Coin Value
 *
 * @param minCoinIn the minCoinIn to set
 */
public void setMinCoinIn(int minCoinIn) {
  this.minCoinIn = minCoinIn;
}

/**
 * Gets Maximum Coin Value
 *
 * @return the maxCoinIn
 */
public int getMaxCoinIn() {
  return maxCoinIn;
}

/**
 * Sets Minimum Coin Value
 *
 * @param maxCoinIn the maxCoinIn to set
 */
public void setMaxCoinIn(int maxCoinIn) {
  this.maxCoinIn = maxCoinIn;
}

/**
 * Prints all coin list
 *
 * @return
 */
public String printCoinList() {
  String coins = "";
  for (int type : coinList) {
    coins += String.valueOf(type);
    if(type != coinList.get(coinList.size()-1)) {
      coins += ", ";
    }
  }
  return String.format("The current coin denominations are in circulation: %s", coins);
}

/**
 * Calculate coins with coin type
```

```
     *
     * @param totalValue
     * @param coinType
     * @return result of the calculation
     */
    public String coinCalculator(int totalValue, int coinType) {
        String errorMessage = validate(totalValue, coinType);
        if (!errorMessage.isEmpty()) {
            return errorMessage;
        }
        int division = totalValue / coinType;
        int remainder = totalValue % coinType;
        return String.format("A total of %s x %sp coins can be exchanged, with a remainder of %sp", division,
coinType,
                remainder);
    }

    /**
     * Calculate coin without excluded coin type
     *
     * @param totalValue
     * @param excludedCoinType
     * @return result of the calculation
     */
    public String multiCoinCalculator(int totalValue, int excludedCoinType) {
        String errorMessage = validate(totalValue, excludedCoinType);
        if (!errorMessage.isEmpty()) {
            return errorMessage;
        }
        String result = "The coins exchanged are: ";
        int remainder = totalValue;
        for (int type : coinList) {
            if (type == excludedCoinType) {
                result += String.format("0 x %sp, ", excludedCoinType);
            }else {
                int division = remainder / type;
                remainder = remainder % type;
                result += String.format("%s x %sp, ", division, type);
            }

            if (type == coinList.get(coinList.size() - 1)) {
                result += String.format("with a remainder of %sp", remainder);
                break;
            }
        }
        return result;
    }

    /**
     * @return Program config items
     */
    public String displayProgramConfigs() {
        return String.format("The current currency %s and the current minimum %s and maximum value %s
accepted as input.",
                this.getCurrency(), this.getMinCoinIn(), this.getMaxCoinIn());
    }
```

```
  /**
   * Validates the inputs
   * @param totalValue
   * @param coinType
   * @return error message
   */
  private String validate(int totalValue, int coinType) {
    String errorMessage = "";
    if (totalValue < getMinCoinIn()) {
      errorMessage += String.format("Total amount can not be less than %s", getMinCoinIn());
    } else if (totalValue > getMaxCoinIn()) {
      errorMessage += String.format("Total amount can not be bigger than %s", getMaxCoinIn());
    } else if (!coinList.contains(coinType)) {
      errorMessage += String.format("Coin Type is not valid : %s", coinType);
    }
    return errorMessage;
  }
}
```

testCounSorter Source Code

```
import java.util.InputMismatchException;
import java.util.Scanner;

/**
* @author Student
*
*        This class tests the functions and methods of CoinSorter class. This
*        class is running on command line
*/
public class testCoinSorter {

  private static CoinSorter coinSorter;

  /**
   * Main method
   *
   * @param args
   */
  public static void main(String[] args) {
    testCoinSorter sorter = new testCoinSorter();
    coinSorter = new CoinSorter();
    Scanner sc = new Scanner(System.in);
    int command = -1;
    // until user press quit action, main menu will prompt to user
    do {
      try {
        // prints main menu option
        System.out.println("***Coin Sorter - Main Menu***\r\n" + "1 - Coin calculator\r\n"
            + "2 - Multiple coin calculator\r\n" + "3 - Print coin list\r\n" + "4 - Set details\r\n"
            + "5 - Display program configurations\r\n" + "6 - Quit the program\r\n");
        command = sc.nextInt();
        // runs main menu commands
        sorter.runMainMenuCommands(command, sc);
      } catch (InputMismatchException ex) {
        System.out.println("Main menu command has to be an integer, invalid command:" + sc.nextLine());
```

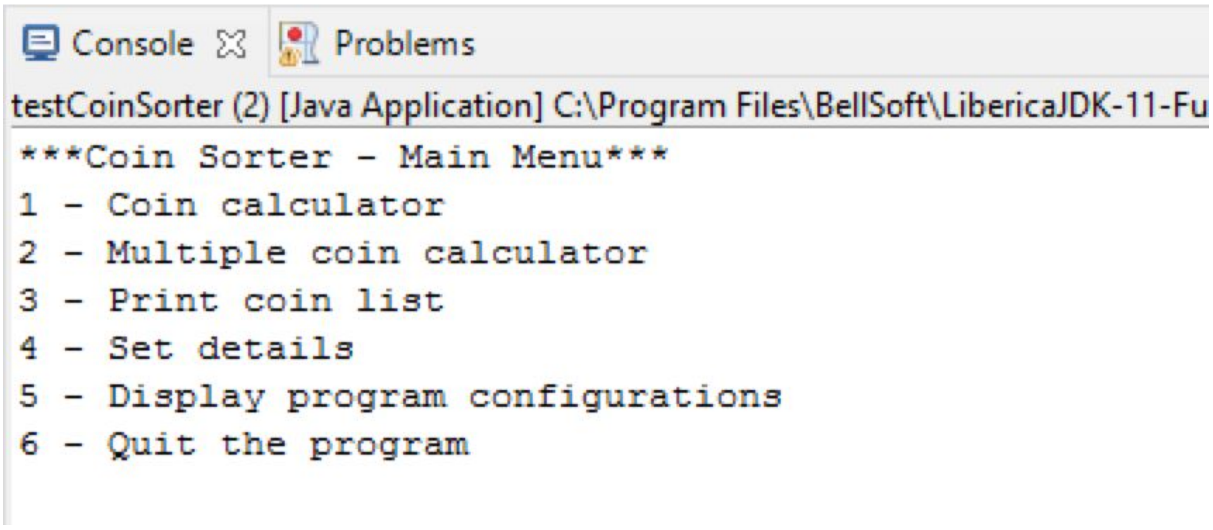
```java
        }

    } while (command != 6);
  }

  /**
   * runs main method commands
   *
   * @param command
   * @param scanner
   */
  private void runMainMenuCommands(int command, Scanner sc) {
    // command run decision point
    switch (command) {
    case 1:
      calculateCoin(sc);
      break;
    case 2:
      calculateMultiCoin(sc);
      break;
    case 3:
      printCoinList();
      break;
    case 4:
      goToSubMenu(sc);
      break;
    case 5:
      displayProgramConfigs();
      break;
    case 6:
      System.out.println("Quited");
      break;
    default:
      System.out.println("Command is not valid :" + command);
      break;
    }

  }

  /**
   * calculates the coins
   *
   * @param scanner
   */
  private void calculateCoin(Scanner sc) {
    // title
    System.out.println("------------------------------");
    System.out.println("1: Coin Calculator ");
    System.out.println("------------------------------");
    System.out.println(
        "You can exchange total amount of coins with the maximum number of coins of the input coin
type that can be exchanged");
    int totalAmount = -1;
    int coinType = -1;
    // will prompt to user until total amount and currency type is inserted
    do {
      try {
        System.out.println("Total Amount : ");
        totalAmount = sc.nextInt();
        if (totalAmount > coinSorter.getMaxCoinIn()) {
          System.out.println("Total amount can not be bigger than " + coinSorter.getMaxCoinIn());
        } else if (totalAmount < coinSorter.getMinCoinIn()) {
          System.out.println("Total amount can not be less than " + coinSorter.getMinCoinIn());
        } else if (totalAmount > coinSorter.getMaxCoinIn()) {
          System.out.println("Total amount can not be bigger than " + coinSorter.getMaxCoinIn());
        } else {
          // if total amount is valid go and ask currency type
          do {
            try {
              System.out.println("Coin Type : ");
              coinType = sc.nextInt();
            } catch (InputMismatchException ex) {
              // if currency type is not valid format as integer them prompt error
              System.out.println("Coin type has to be an integer, invalid value " + sc.nextLine());
            }
          } while (coinType < 0);
        }
      } catch (InputMismatchException ex) {
        // if total amount is not valid format as integer them prompt error
        System.out.println("Total amount has to be an integer, invalid value " + sc.nextLine());
      }
    } while (totalAmount < 0);

    // calculates the result and return
    String result = coinSorter.coinCalculator(totalAmount, coinType);
    System.out.println("Result : " + result);
    System.out.println();
  }

  /**
   * calculates multiple coins
   *
   * @param scanner
   */
  private void calculateMultiCoin(Scanner sc) {
    // title
    System.out.println("------------------------------");
    System.out.println("2: Multi Coin Calculator ");
    System.out.println("------------------------------");
    System.out.println("You can exchange total amount of coins by excluding with the input of coin type");
    int totalAmount = -1;
    int coinType = -1;
    // will prompt to user until total amount and currency type is inserted
    do {
      try {
        System.out.println("Total Amount : ");
        totalAmount = sc.nextInt();
        if (totalAmount > coinSorter.getMaxCoinIn()) {
          System.out.println("Total amount can not be bigger than " + coinSorter.getMaxCoinIn());
        } else if (totalAmount < coinSorter.getMinCoinIn()) {
          System.out.println("Total amount can not be less than " + coinSorter.getMinCoinIn());
        } else if (totalAmount > coinSorter.getMaxCoinIn()) {
          System.out.println("Total amount can not be bigger than " + coinSorter.getMaxCoinIn());
        } else {
          // if total amount is valid go and ask currency type
          do {
                try {
                    System.out.println("Excluded Coin Type : ");
                    coinType = sc.nextInt();
                } catch (InputMismatchException ex) {
                    // if currency type is not valid format as integer them prompt error
                    System.out
                            .println("Excluded Coin type has to be an integer, invalid value " + sc.nextLine());
                }
            } while (coinType < 0);
        }
    } catch (InputMismatchException ex) {
        // if total amount is not valid format as integer them prompt error
        System.out.println("Total amount has to be an integer, invalid value " + sc.nextLine());
    }
} while (totalAmount < 0);

// calculates multiple coin sort then returns result
String result = coinSorter.multiCoinCalculator(totalAmount, coinType);
System.out.println("Result : " + result);
System.out.println();
}

/**
 * prints coin sorter list
 */
private void printCoinList() {
    // title
    System.out.println("------------------------------");
    System.out.println("3: Print Coin List ");
    System.out.println("------------------------------");
    // returns coin list detail
    String result = coinSorter.printCoinList();
    System.out.println("Result : " + result);
    System.out.println();
}

/**
 * displays coin sorter configurations
 */
private void displayProgramConfigs() {
    // title
    System.out.println("------------------------------");
    System.out.println("5: Display Program Configs ");
    System.out.println("------------------------------");
    // returns coin sorter config details
    String result = coinSorter.displayProgramConfigs();
    System.out.println("Result : " + result);
    System.out.println();
}

/**
 * goes to sub menu
 *
 * @param sc
 */
private void goToSubMenu(Scanner sc) {
    // title
    System.out.println("------------------------------");
```

```java
  System.out.println("4: Set Details ");
  System.out.println("-----------------------------");
  int command = -1;
  // prompt until user set quit command from sum menu
  do {
    try {
      System.out.println("***Set Details Sub-Menu***\r\n" + "1 - Set currency\r\n"
          + "2 - Set minimum coin input value\r\n" + "3 - Set maximum coin input value\r\n"
          + "4 - Return to main menu\r\n");
      command = sc.nextInt();
      // runs sub menu commands
      runSubMenuCommands(command, sc);
    } catch (InputMismatchException ex) {
      // if command is not a valid command as integer, retrieve error
      System.out.println("Sub menu command has to be an integer, invalid command:" + sc.nextLine());
    }
  } while (command != 4);

}

/**
 * runs sub menu commands
 *
 * @param command
 * @param scanner
 */
private void runSubMenuCommands(int command, Scanner sc) {
  // command run decision point
  switch (command) {
  case 1:
    setCurrency(sc);
    break;
  case 2:
    setMinCoin(sc);
    break;
  case 3:
    setMaxCoin(sc);
    break;
  case 4:
    break;
  default:
    System.out.println("Sub menu command is not valid :" + command);
    break;
  }
}

/**
 * sets currency info
 *
 * @param scanner
 */
private void setCurrency(Scanner sc) {
  // title
  System.out.println("-----------------------------");
  System.out.println("1: Set Currency ");
  System.out.println("-----------------------------");
  System.out.println("You can set coin sorter currency");
  String currency = "";
```

```
        // prompt until user set valid input
        do {
            System.out.println("Currency : ");
            currency = sc.next();
            // checks input is empty or not
            if (currency.isEmpty()) {
                // if empty retrieve an error
                System.out.println("Currency can not be empty  ");
            }

        } while (currency.isEmpty());

        coinSorter.setCurrency(currency);
        System.out.println("Currency updated with " + currency);
        System.out.println();
    }

    /**
     * sets minimum coin value
     *
     * @param scanner
     */
    private void setMinCoin(Scanner sc) {
        // title
        System.out.println("------------------------------");
        System.out.println("2: Set Minimum Coin ");
        System.out.println("------------------------------");
        System.out.println("You can set coin sorter minimum coin value");
        int minValue = -1;
        // prompt until user set valid input
        do {
            try {
                System.out.println("Minimum Value Amount : ");
                minValue = sc.nextInt();
                // checks inserted min value can not be less than 0
                if (minValue < 0) {
                    System.out.println("Minimum value can not be less then 0");
                }
            } catch (InputMismatchException ex) {
                // if input value is not valid as integer, retrieves an error
                System.out.println("Minimum value has to be an integer, invalid value: " + sc.nextLine());
            }
        } while (minValue < 0);

        // sets minimum coin value
        coinSorter.setMinCoinIn(minValue);
        System.out.println("Minimum coin value updated with " + minValue);
        System.out.println();
    }

    /**
     * sets maximum coin value
     *
     * @param sc
     */
    private void setMaxCoin(Scanner sc) {
        // title
        System.out.println("------------------------------");
    System.out.println("3: Set Maximum Coin ");
    System.out.println("-----------------------------");
    System.out.println("You can set coin sorter maximum coin value");
    int maxValue = -1;
    // prompt until user set valid input
    do {
      try {
        System.out.println("Maximum Value Amount : ");
        maxValue = sc.nextInt();
        // checks inserted max value can not be less then 0
        if (maxValue < 0) {
          System.out.println("Maximum value can not be less then 0");
        }
      } catch (InputMismatchException ex) {
        // if input value is not valid as integer, retrieves an error
        System.out.println("Maximum value has to be an integer, invalid value : " + sc.nextLine());
      }

    } while (maxValue < 0);

    // sets maximum coin information
    coinSorter.setMaxCoinIn(maxValue);
    System.out.println("Maximum coin value updated with " + maxValue);
    System.out.println();
  }
}
```

Evidence for Test Coin Sorter class

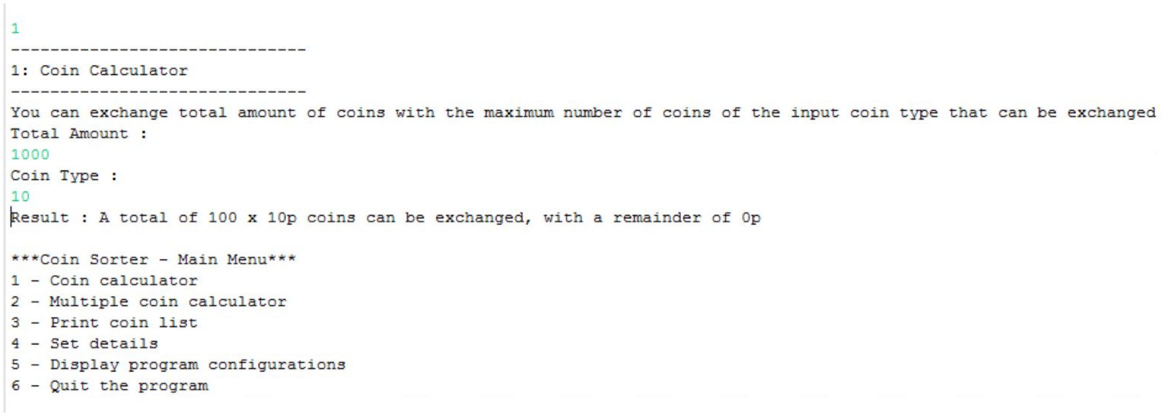
Main Menu



```
testCoinSorter (2) [Java Application] C:\Program Files\BellSoft\LibericaJDK-11-Fu
***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
```

Command 1 Coin Calculator

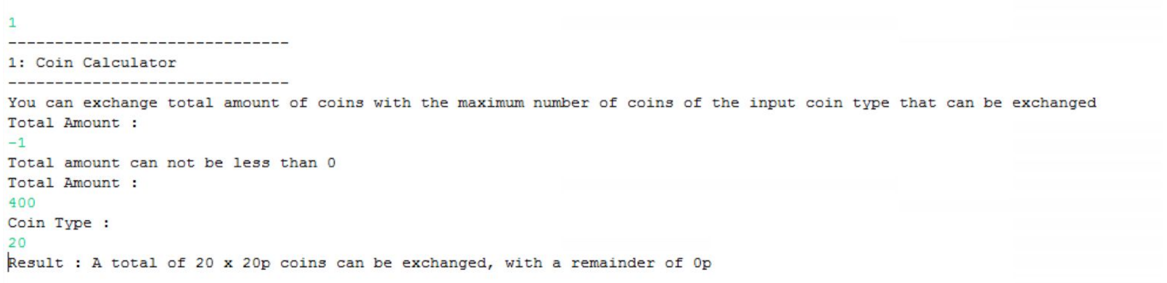
Success:



```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
1000
Coin Type :
10
Result : A total of 100 x 10p coins can be exchanged, with a remainder of 0p

***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
```

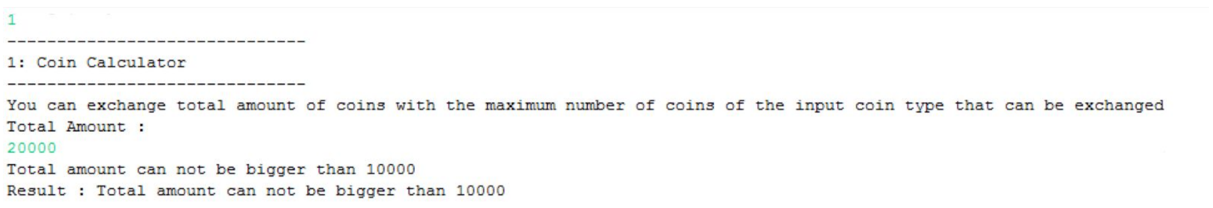
Fail Scenarios:

1- Total amount less than minimum value:



```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
-1
Total amount can not be less than 0
Total Amount :
400
Coin Type :
20
Result : A total of 20 x 20p coins can be exchanged, with a remainder of 0p
```

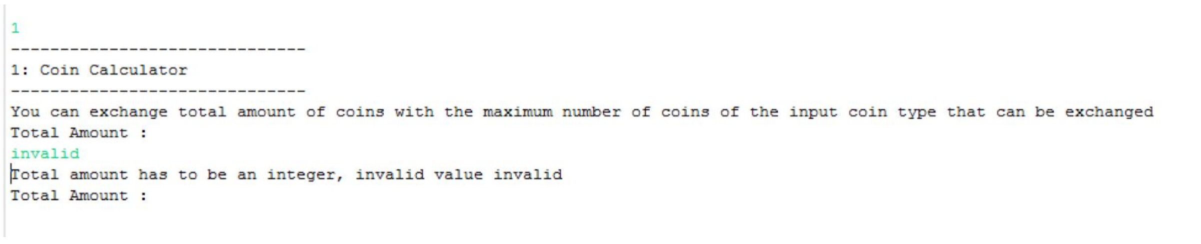
2- Total amount bigger than maximum value:



```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
20000
Total amount can not be bigger than 10000
Result : Total amount can not be bigger than 10000
```

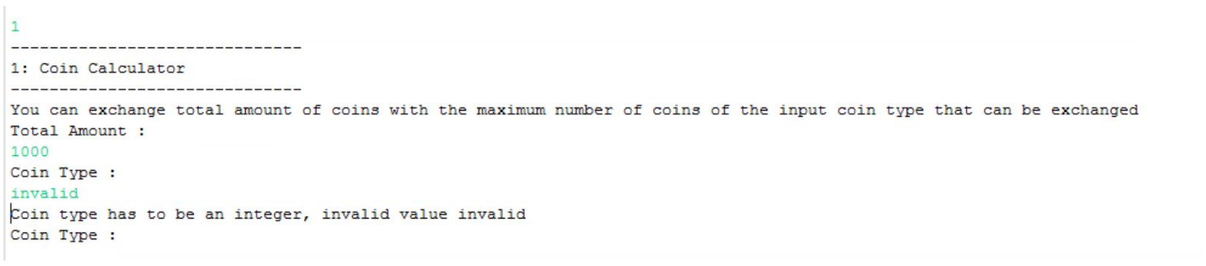
3- Total amount and currency type has to be integer value :

Total amount:



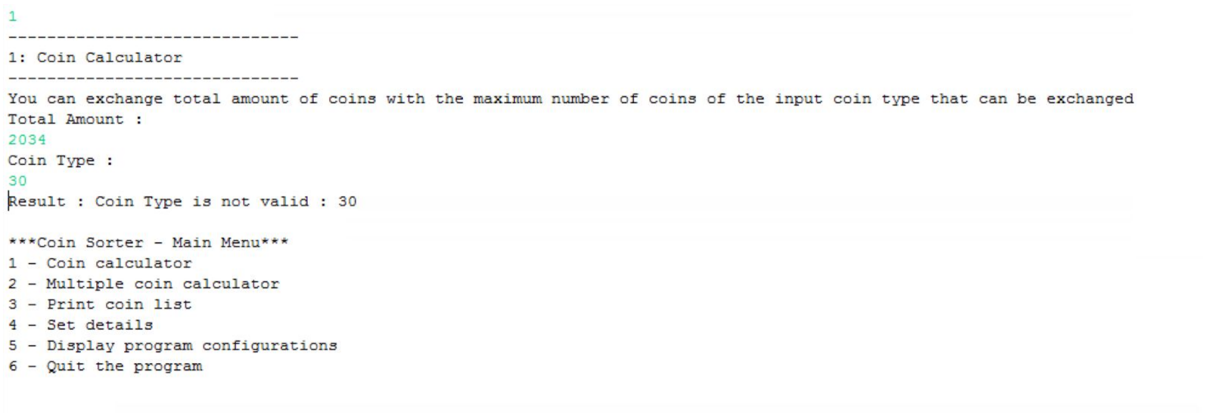
```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
invalid
Total amount has to be an integer, invalid value invalid
Total Amount :
```

Coin type:



```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
1000
Coin Type :
invalid
Coin type has to be an integer, invalid value invalid
Coin Type :
```

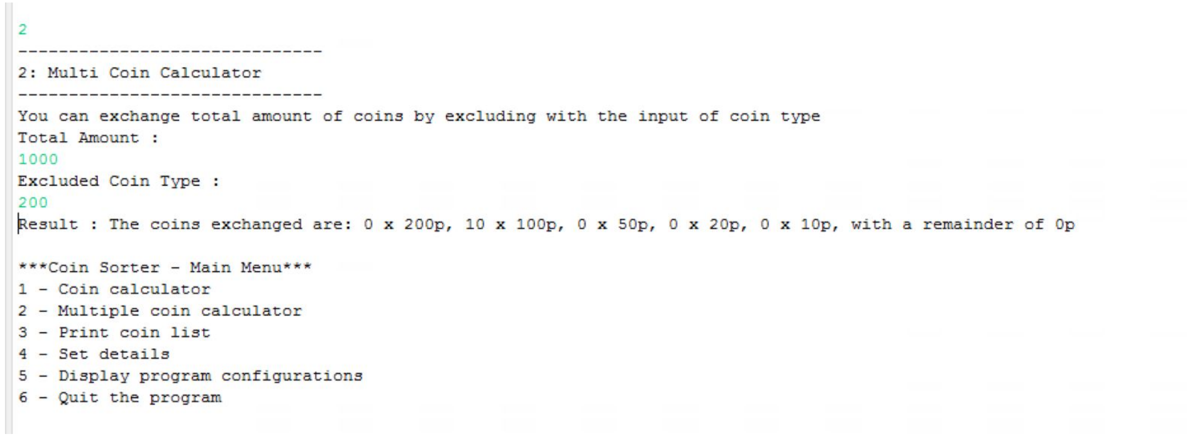
4- Coin type is not in coin list:



```
1
------------------------------
1: Coin Calculator
------------------------------
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged
Total Amount :
2034
Coin Type :
30
Result : Coin Type is not valid : 30

***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
```

Command 2 Multiple Coin Calculator

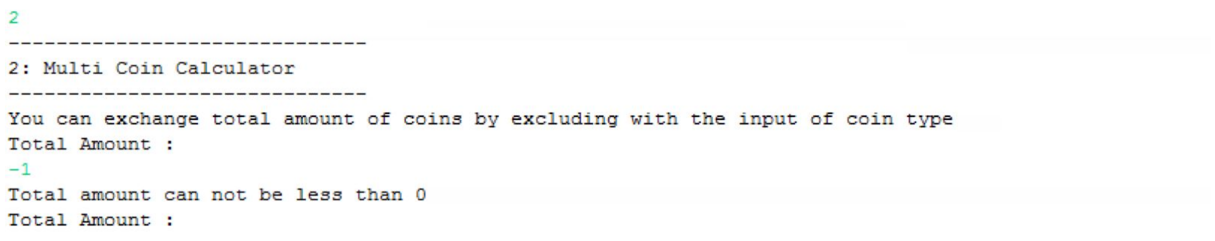
Success:



```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
1000
Excluded Coin Type :
200
Result : The coins exchanged are: 0 x 200p, 10 x 100p, 0 x 50p, 0 x 20p, 0 x 10p, with a remainder of 0p

***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
```

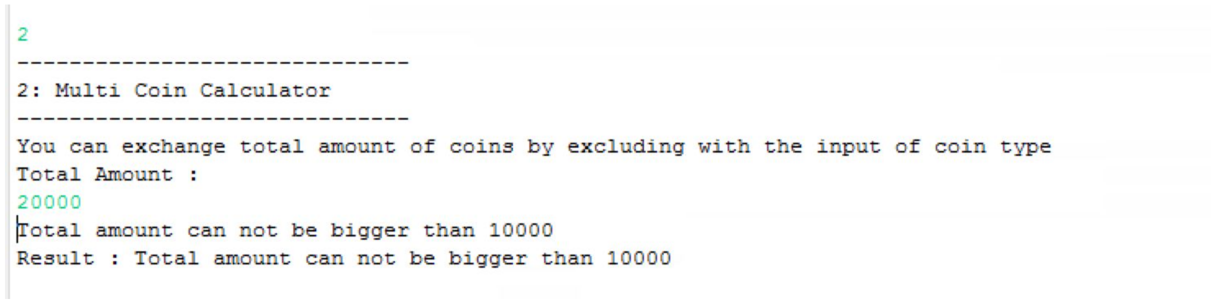
Fail Scenarios:

1- Total amount less than minimum value:



```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
-1
Total amount can not be less than 0
Total Amount :
```

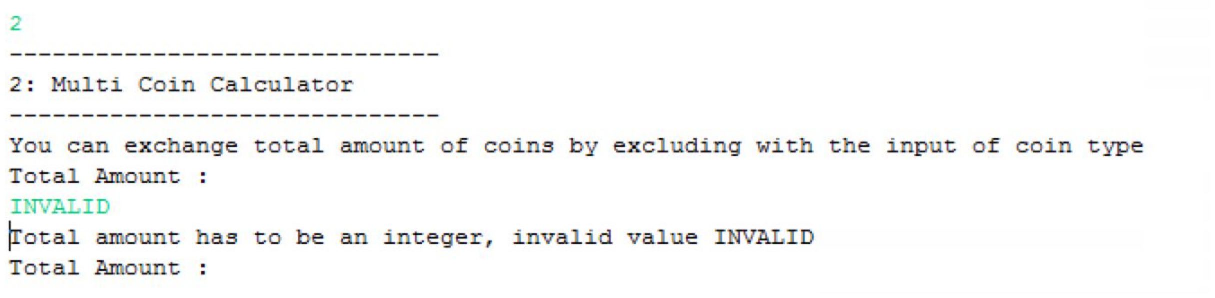
2- Total amount bigger than maximum value:



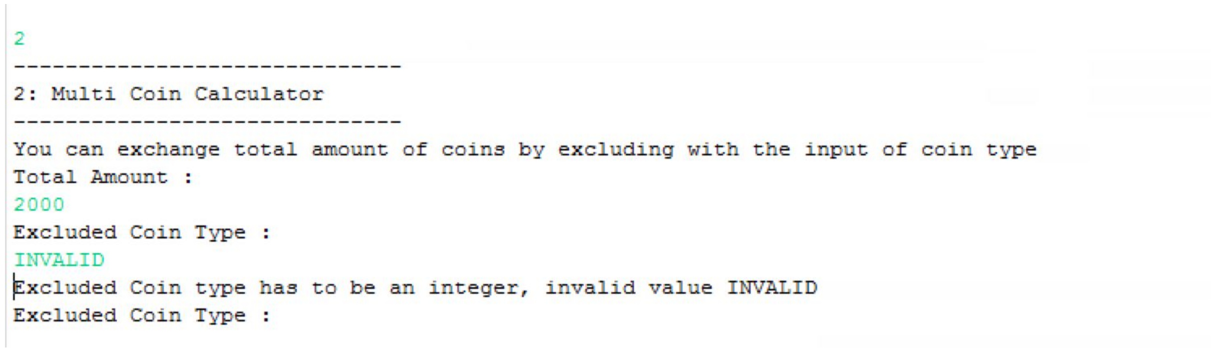
```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
20000
Total amount can not be bigger than 10000
Result : Total amount can not be bigger than 10000
```

3- Total amount and currency type has to be integer value :

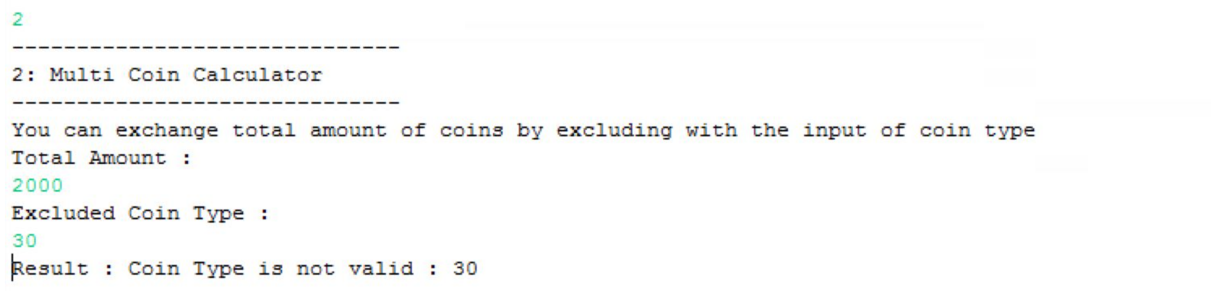
Total amount:


```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
INVALID
Total amount has to be an integer, invalid value INVALID
Total Amount :
```

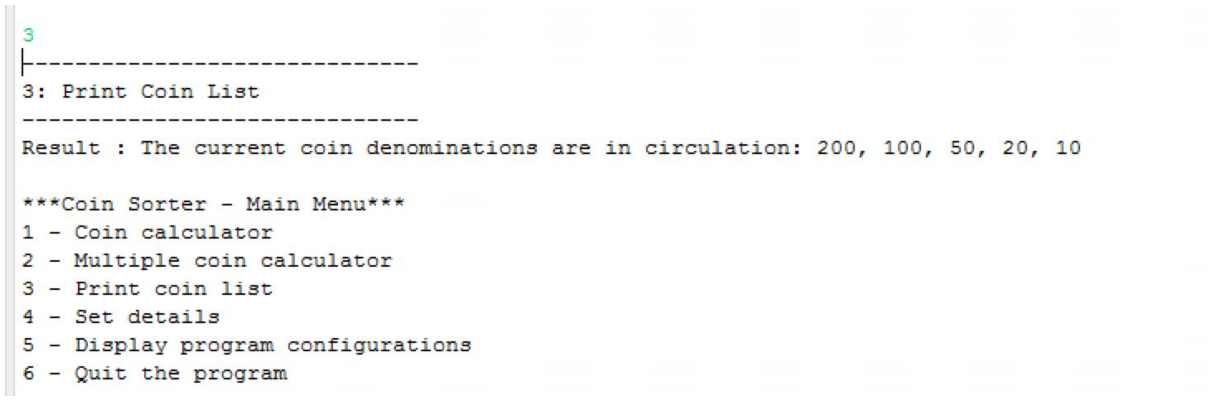
Coin type:


```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
2000
Excluded Coin Type :
INVALID
Excluded Coin type has to be an integer, invalid value INVALID
Excluded Coin Type :
```

4- Coin type is not in coin list:


```
2
------------------------------
2: Multi Coin Calculator
------------------------------
You can exchange total amount of coins by excluding with the input of coin type
Total Amount :
2000
Excluded Coin Type :
30
Result : Coin Type is not valid : 30
```

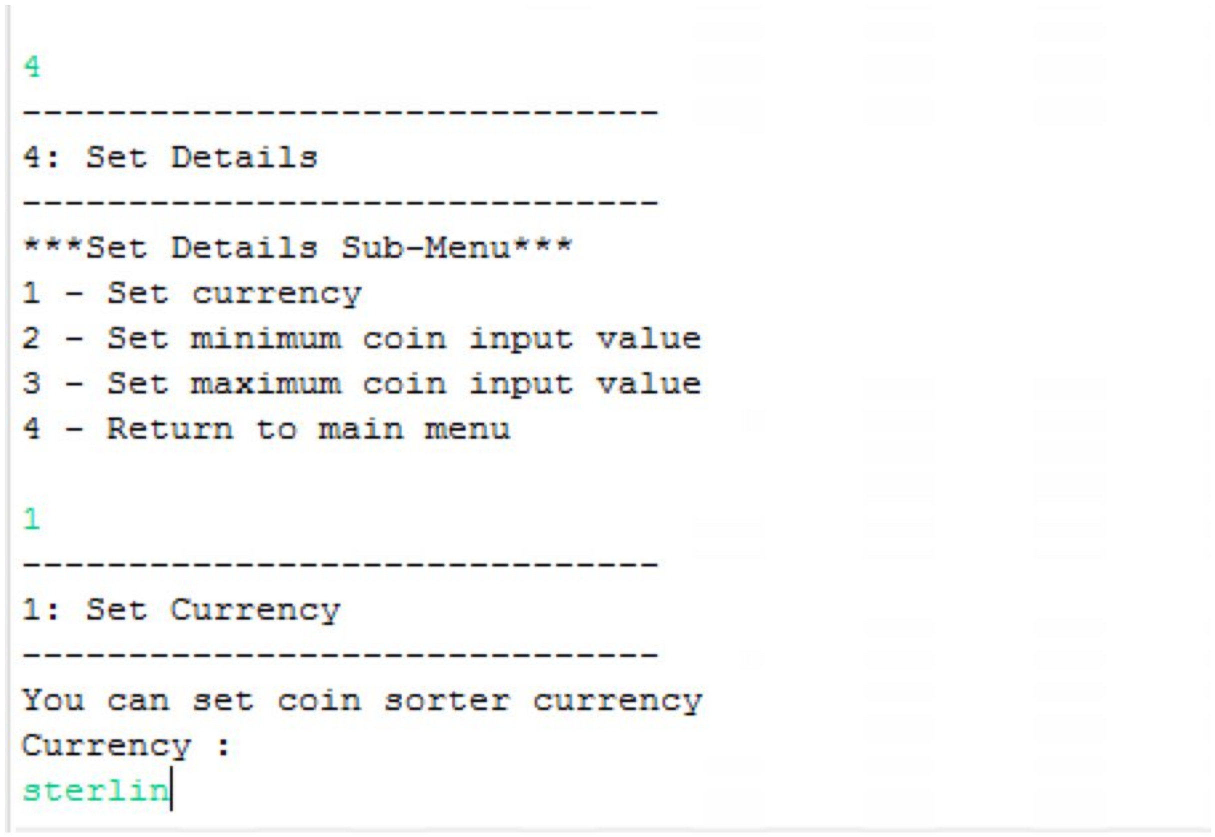
Command 3 Print Coin List


```
3
------------------------------
3: Print Coin List
------------------------------
Result : 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
```

Command 4 Set Details Sub Menu

Command 4.1 Set Currency

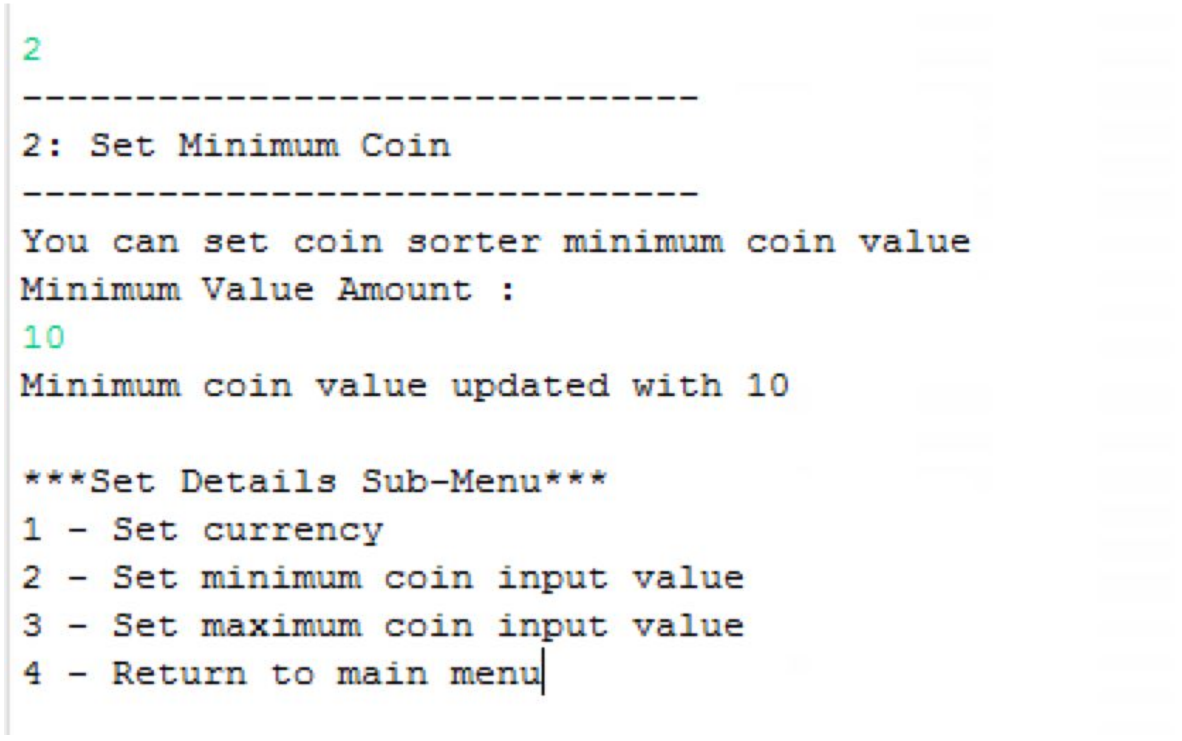
Success:



```
4
------------------------------
4: Set Details
------------------------------
***Set Details Sub-Menu***
1 - Set currency
2 - Set minimum coin input value
3 - Set maximum coin input value
4 - Return to main menu

1
------------------------------
1: Set Currency
------------------------------
You can set coin sorter currency
Currency :
sterlin
```

Command 4.2 Set Minimum Coin

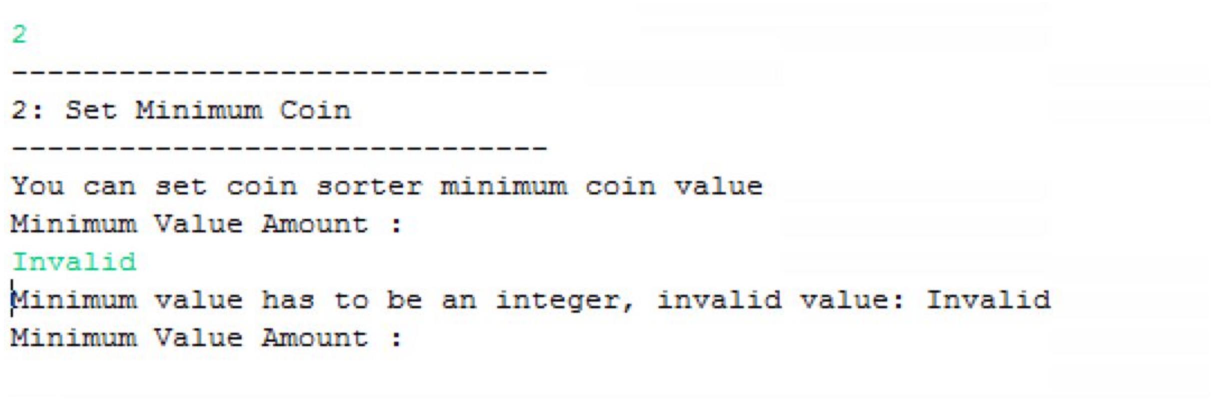
Succes:



```
2
------------------------------
2: Set Minimum Coin
------------------------------
You can set coin sorter minimum coin value
Minimum Value Amount :
10
Minimum coin value updated with 10

***Set Details Sub-Menu***
1 - Set currency
2 - Set minimum coin input value
3 - Set maximum coin input value
4 - Return to main menu
```

Fail Scenarios:

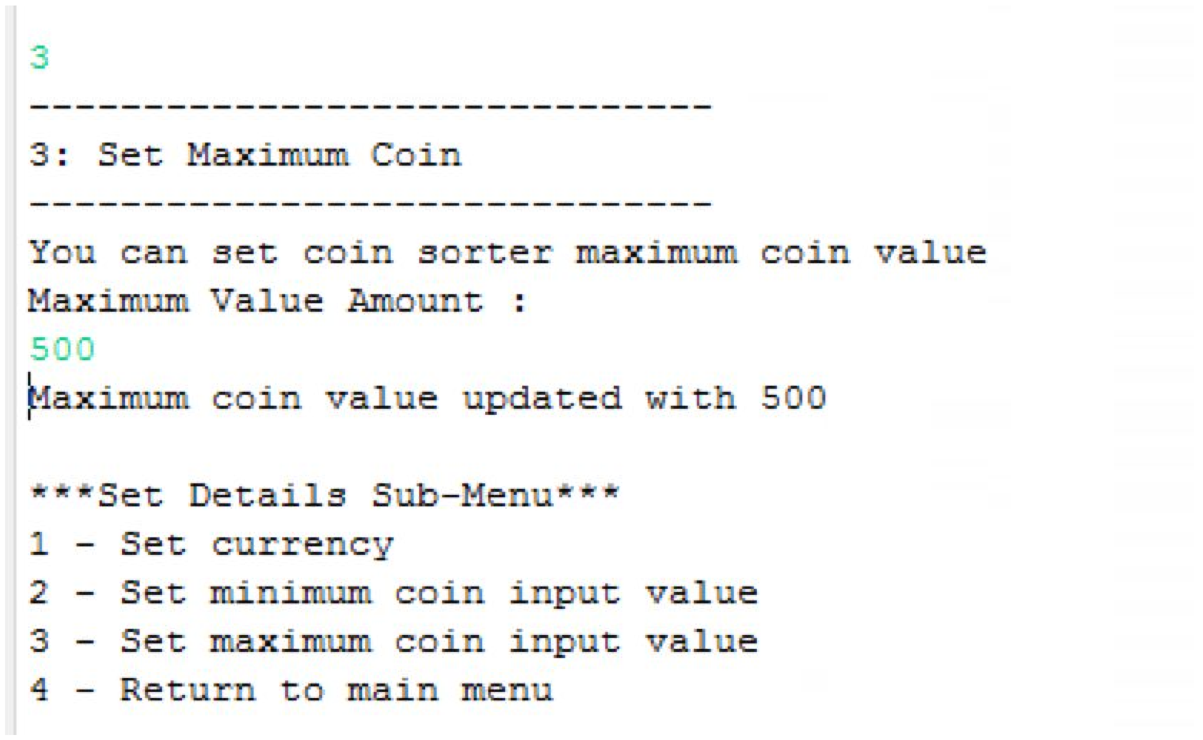
1- Min Value type is invalid:



```
2
------------------------------
2: Set Minimum Coin
------------------------------
You can set coin sorter minimum coin value
Minimum Value Amount :
Invalid
Minimum value has to be an integer, invalid value: Invalid
Minimum Value Amount :
```

Command 4.3 Set Maximum Coin

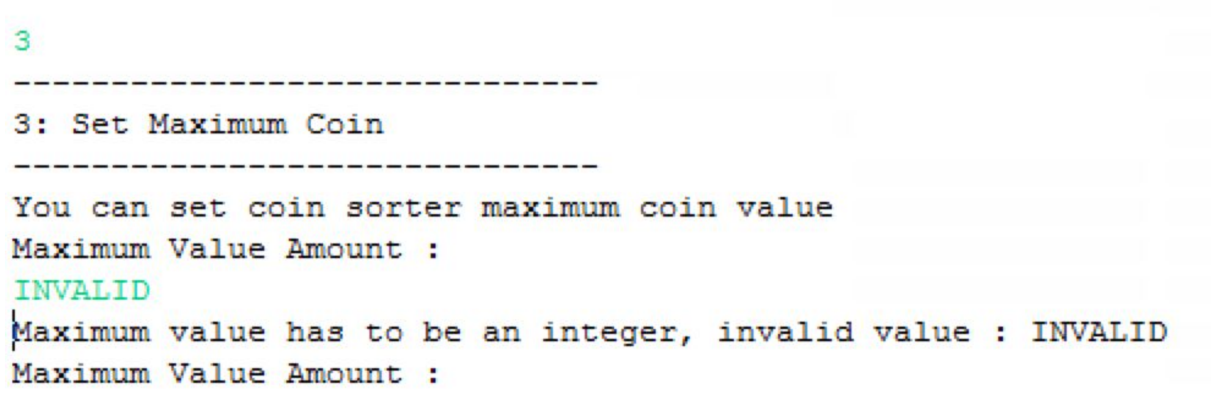
Success:



```
3
------------------------------
3: Set Maximum Coin
------------------------------
You can set coin sorter maximum coin value
Maximum Value Amount :
500
Maximum coin value updated with 500

***Set Details Sub-Menu***
1 - Set currency
2 - Set minimum coin input value
3 - Set maximum coin input value
4 - Return to main menu
```

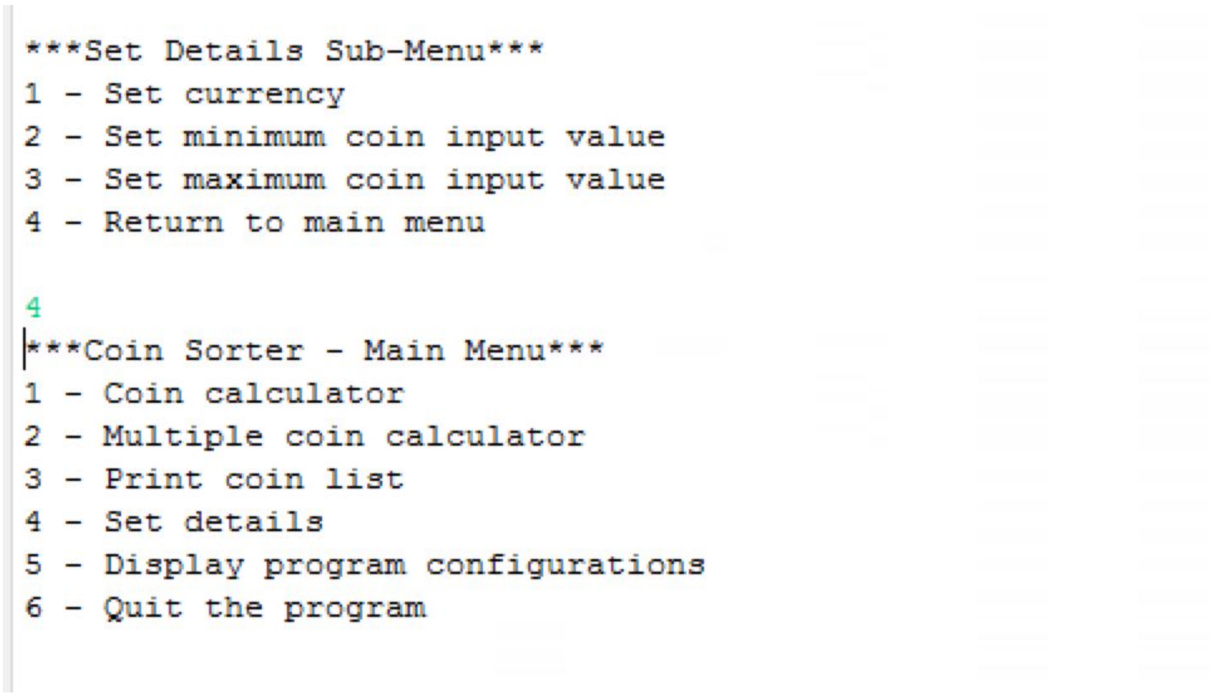
Fail Scenarios:

1- Max Value type is invalid:



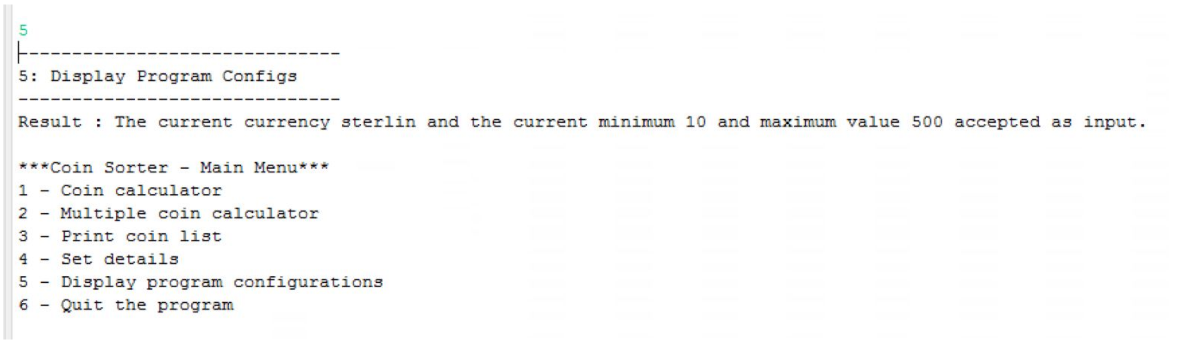
```
3
------------------------------
3: Set Maximum Coin
------------------------------
You can set coin sorter maximum coin value
Maximum Value Amount :
INVALID
Maximum value has to be an integer, invalid value : INVALID
Maximum Value Amount :
```

Command 4.4 Return Main Menu



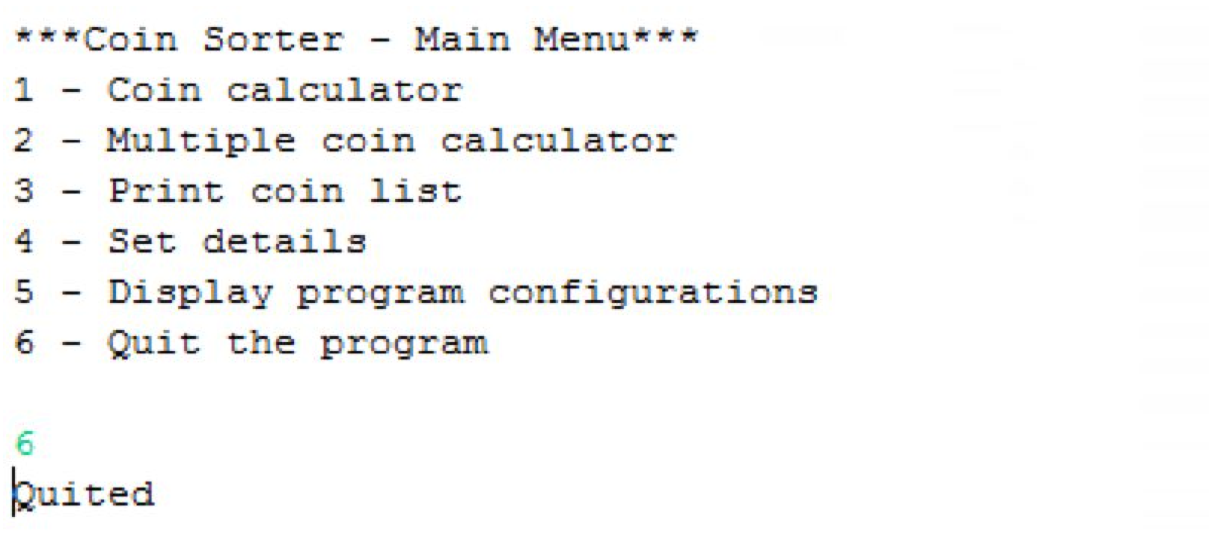
```
***Set Details Sub-Menu***
1 - Set currency
2 - Set minimum coin input value
3 - Set maximum coin input value
4 - Return to main menu

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
```

Command 5 Display Program Configs



```
5
------------------------------
5: Display Program Configs
------------------------------
Result : The current currency sterlin and the current minimum 10 and maximum value 500 accepted as 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
```

Command 6 Quit


```
***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

6
Quited
```

Part B – Graphical Menu

CoinSorterGUI Source Code:

```
/**
*
* @author Student
*
* This class is to sort coins
*
*/
public class CoinSorterGUI extends CoinSorter{
}
```

testCoinSortGUI Source Code:

```
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
```

```java
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;

/**
* @author Student
*
*        This class tests the functions and methods of the CoinSorterGUI
*        class. This class is a GUI application which is sorting coins.
*/
public class testCoinSorterGUI extends Application {

  private Stage window;
  private Scene mainScene;
  private Scene subMenuScene;
  private CoinSorterGUI coinSorterGUI;

  @Override
  public void start(Stage primaryStage) {
    try {
      window = primaryStage;
      window.setTitle("Coin Sorter - Main Menu");
      // initialise coinSorterGUI object
      coinSorterGUI = new CoinSorterGUI();
      // creates the root pane
      GridPane rootGrid = createMainMenuPane(coinSorterGUI);
      // initialise main scene
      mainScene = new Scene(rootGrid, 350, 300);
      window.setScene(mainScene);
      // shows main scene
      window.show();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  public static void main(String[] args) {
    launch(args);
  }

  /**
   * creates main menu pane
   *
   * @param coinSorter
   * @return main menu grid pane
   */
  private GridPane createMainMenuPane(CoinSorter coinSorter) {
    GridPane rootGrid = createGridPane();
    VBox titleBox = createTitleBox("Welcome", "This application was designed to help you to sort your
coins");
    rootGrid.add(titleBox, 0, 0, 2, 1);
    // creates the grid pane components
    Button coinCalculatorButton = addButtonWithLabel(rootGrid, 1, "1:", "Coin Calculator");
    coinCalculatorButton.setOnAction(e -> createCoinCalculatorPane(coinSorter));

    Button multiCoinCalculatorButton = addButtonWithLabel(rootGrid, 2, "2:", "Multi Coin Calculator");
    multiCoinCalculatorButton.setOnAction(e -> createMultiCoinCalculatorPane(coinSorter));
```

```java
        Button printCoinButton = addButtonWithLabel(rootGrid, 3, "3:", "Print Coin List");
        printCoinButton.setOnAction(e -> createPrintCoinListPane(coinSorter));

        Button setDetailsButton = addButtonWithLabel(rootGrid, 4, "4:", "Set Details");
        setDetailsButton.setOnAction(e -> createSubMenuPane(coinSorter));

        Button displayButton = addButtonWithLabel(rootGrid, 5, "5:", "Display Program Configurations");
        displayButton.setOnAction(e -> createDisplayConfigurationDetailsPane(coinSorter));

        Button quitButton = addButtonWithLabel(rootGrid, 6, "6:", "Quit The Program");
        quitButton.setOnAction(e -> quit());
        return rootGrid;
    }

    /**
     * creates sub menu pane
     *
     * @param coinSorter
     */
    private void createSubMenuPane(CoinSorter coinSorter) {
        GridPane grid = createGridPane();
        Text scenetitle = new Text("You can update Coin Sorter application config values from this menu");
        scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 10));
        scenetitle.setWrappingWidth(350);
        grid.add(scenetitle, 0, 0, 2, 1);
        // creates the grid pane components
        Button setCurrencyButton = addButtonWithLabel(grid, 1, "1:", "Set Currency");
        setCurrencyButton.setOnAction(e -> createSetCurrencyPane(coinSorter));

        Button setMinButton = addButtonWithLabel(grid, 2, "2:", "Set Minimum Coin Input Value");
        setMinButton.setOnAction(e -> createSetMinimumValuePane(coinSorter));

        Button setMaxButton = addButtonWithLabel(grid, 3, "3:", "Set Maximum Coin Input Value");
        setMaxButton.setOnAction(e -> createSetMaximumValuePane(coinSorter));

        Button quitButton = addButtonWithLabel(grid, 4, "4:", "Return to Main Menu");
        quitButton.setOnAction(e -> goMainMenu());

        subMenuScene = new Scene(grid, 400, 275);
        window.setTitle("Set Details - Sub Menu");
        showScene(window, subMenuScene);
    }

    /**
     * create set currency pane
     *
     * @param coinSorter
     */
    private void createSetCurrencyPane(CoinSorter coinSorter) {
        GridPane grid = createGridPane();
        VBox titleBox = createTitleBox("Set Currency", "You can set the application currency type");
        grid.add(titleBox, 0, 0, 2, 1);
        // creates the grid pane components
        TextField currency = addTextFieldWithLabel(grid, 1, "Currency :");
        currency.setText(coinSorter.getCurrency());
        final Text result = new Text();
        result.setWrappingWidth(250);
        grid.add(result, 0, 2, 2, 1);

        HBox hbBtn = createConfirmButtonHBox(Pos.BOTTOM_RIGHT);
        Button btnOk = (Button) hbBtn.getChildren().get(0);
        Button btnCancel = (Button) hbBtn.getChildren().get(1);
        grid.add(hbBtn, 1, 3);
        btnOk.setOnAction(e -> {
            // checks for currency text is empty or not
            if (currency.getText().isEmpty()) {
                result.setFill(Color.RED);
                result.setText("Currency can not be empty");
            } else {
                coinSorter.setCurrency(currency.getText());
                goSubMenu();
            }
        });
        btnCancel.setOnAction(e -> goSubMenu());
        moveNext(grid, 400, 200, "Set Currency");
    }

    /**
     * creates minimum value pane
     *
     * @param coinSorter
     */
    private void createSetMinimumValuePane(CoinSorter coinSorter) {
        GridPane grid = createGridPane();
        VBox titleBox = createTitleBox("Set Minimum Coin Input Value",
                "You can set the application minimum coin value");
        grid.add(titleBox, 0, 0, 2, 1);
        // creates the grid pane components
        TextField minValue = addTextFieldWithLabel(grid, 1, "Minimum Coin Value :");
        minValue.setText(String.valueOf(coinSorter.getMinCoinIn()));
        final Text result = new Text();
        result.setWrappingWidth(250);
        grid.add(result, 0, 2, 2, 1);
        HBox hbBtn = createConfirmButtonHBox(Pos.BOTTOM_RIGHT);
        Button btnOk = (Button) hbBtn.getChildren().get(0);
        Button btnCancel = (Button) hbBtn.getChildren().get(1);
        grid.add(hbBtn, 1, 3);
        btnOk.setOnAction(e -> {
            // checks for minimum value text is empty or not
            if (minValue.getText().isEmpty()) {
                result.setFill(Color.RED);
                result.setText("Minimum coin value can not be empty");
            } else {
                try {
                    coinSorter.setMinCoinIn(Integer.valueOf(minValue.getText()));
                    goSubMenu();
                } catch (NumberFormatException ex) {
                    // if minimum value is not an integer value, retrieve an error to the user.
                    result.setFill(Color.RED);
                    result.setText("Minimum coin value has to be an integer");
                }
            }
        });
        btnCancel.setOnAction(e -> goSubMenu());
        moveNext(grid, 400, 200, "Set Minimum Coin Value");
```

```
}

/**
 * creates set maximum value pane
 *
 * @param coinSorter
 */
private void createSetMaximumValuePane(CoinSorter coinSorter) {
  GridPane grid = createGridPane();
  VBox titleBox = createTitleBox("Set Maximum Coin Input Value",
      "You can set the application maximum coin value");
  grid.add(titleBox, 0, 0, 2, 1);
  // creates the grid pane components
  TextField maxValue = addTextFieldWithLabel(grid, 1, "Maximum Coin Value :");
  maxValue.setText(String.valueOf(coinSorter.getMaxCoinIn()));
  final Text result = new Text();
  result.setWrappingWidth(250);
  grid.add(result, 0, 2, 2, 1);

  HBox hbBtn = createConfirmButtonHBox(Pos.BOTTOM_RIGHT);
  Button btnOk = (Button) hbBtn.getChildren().get(0);
  Button btnCancel = (Button) hbBtn.getChildren().get(1);
  grid.add(hbBtn, 1, 3);
  btnOk.setOnAction(e -> {
    // checks for maximum value text is empty or not
    if (maxValue.getText().isEmpty()) {
      result.setFill(Color.RED);
      result.setText("Maximum coin value can not be empty");
    } else {
      try {
        coinSorter.setMaxCoinIn(Integer.valueOf(maxValue.getText()));
        goSubMenu();
      } catch (NumberFormatException ex) {
        // if maximum value is not an integer value, retrieve an error to the user.
        result.setFill(Color.RED);
        result.setText("Maximum coin value has to be an integer");
      }
    }
  });
  btnCancel.setOnAction(e -> goSubMenu());
  moveNext(grid, 400, 200, "Set Minimum Coin Value");
}

/**
 * created coin calculator pane
 *
 * @param coinSorter
 */
private void createCoinCalculatorPane(CoinSorter coinSorter) {
  GridPane grid = createGridPane();
  VBox titleBox = createTitleBox("Coin Calculator",
      "You can exchange total amount of coins with the maximum number of coins of the input coin
type that can be exchanged");
  grid.add(titleBox, 0, 0, 2, 1);
  // creates the grid pane components
  TextField totalAmount = addTextFieldWithLabel(grid, 1, "Total Amount:");
  TextField coinType = addTextFieldWithLabel(grid, 2, "Coin Type :");
```

```java
    final Text result = new Text();
    result.setWrappingWidth(300);
    grid.add(result, 0, 4, 2, 1);

    HBox hbBtn = createConfirmButtonHBox(Pos.BOTTOM_RIGHT);
    Button btnOk = (Button) hbBtn.getChildren().get(0);
    btnOk.setText("Calculate");
    Button btnCancel = (Button) hbBtn.getChildren().get(1);
    btnCancel.setText("Return To Main Menu");
    grid.add(hbBtn, 1, 3);
    btnOk.setOnAction(e -> {
      // checks for total amount value text is empty or not
      if (totalAmount.getText().isEmpty()) {
        result.setFill(Color.RED);
        result.setText("Total amount field can not be empty!");
      }
      // checks for coin type value text is empty or not
      else if (coinType.getText().isEmpty()) {
        result.setFill(Color.RED);
        result.setText("Coin type field can not be empty!");
      } else {
        try {
          int tAmount = Integer.valueOf(totalAmount.getText());
          int currenyType = Integer.valueOf(coinType.getText());
          // if total amount less than minimum amount
          if (tAmount < coinSorter.getMinCoinIn()) {
            result.setFill(Color.RED);
            result.setText(String.format("Total amount has to be bigger than minimum amount :
%s",coinSorter.getMinCoinIn()));
          }else if (tAmount > coinSorter.getMaxCoinIn()) {
            // if total amount bigger than maximum amount
            result.setFill(Color.RED);
            result.setText(String.format("Total amount has to be less than maximum amount :
%s",coinSorter.getMaxCoinIn()));
          }else {
            result.setFill(Color.BLUEVIOLET);
            result.setText(coinSorter.coinCalculator(tAmount, currenyType));
          }
        } catch (NumberFormatException ex) {
          // if total amount and currency type format is not integer, retrieve an error
          result.setFill(Color.RED);
          result.setText(String.format("Please check your input has to be integer : %s", ex.getMessage()));
        }
      }
    });
    btnCancel.setOnAction(e -> goMainMenu());
    moveNext(grid, 400, 250, "Coin Calculate");
  }

  /**
   * created multiple coin calculator pane
   *
   * @param coinSorter
   */
  private void createMultiCoinCalculatorPane(CoinSorter coinSorter) {
    GridPane grid = createGridPane();
    VBox titleBox = createTitleBox("Multi Coin Calculator",
        "You can exchange total amount of coins by excluding with the input of coin type");
    grid.add(titleBox, 0, 0, 2, 1);
    // creates the grid pane components
    TextField totalAmount = addTextFieldWithLabel(grid, 1, "Total Amount:");
    TextField coinType = addTextFieldWithLabel(grid, 2, "Coin Type :");

    final Text result = new Text();
    result.setWrappingWidth(250);
    grid.add(result, 0, 4, 2, 1);

    HBox hbBtn = createConfirmButtonHBox(Pos.BOTTOM_RIGHT);
    Button btnOk = (Button) hbBtn.getChildren().get(0);
    btnOk.setText("Calculate");
    Button btnCancel = (Button) hbBtn.getChildren().get(1);
    btnCancel.setText("Return To Main Menu");
    grid.add(hbBtn, 1, 3);
    btnOk.setOnAction(e -> {
      // checks for total amount value text is empty or not
      if (totalAmount.getText().isEmpty()) {
        result.setFill(Color.RED);
        result.setText("Total amount field can not be empty!");
      }
      // checks for coin type value text is empty or not
      else if (coinType.getText().isEmpty()) {
        result.setFill(Color.RED);
        result.setText("Coin type field can not be empty!");
      } else {
        try {
          int tAmount = Integer.valueOf(totalAmount.getText());
          int curreny Type = Integer.valueOf(coinType.getText());
          // if total amount less than minimum amount
          if (tAmount < coinSorter.getMinCoinIn()) {
            result.setFill(Color.RED);
            result.setText(String.format("Total amount has to be bigger than minimum amount :
%s",coinSorter.getMinCoinIn()));
          }else if (tAmount > coinSorter.getMaxCoinIn()) {
            // if total amount bigger than maximum amount
            result.setFill(Color.RED);
            result.setText(String.format("Total amount has to be less than maximum amount :
%s",coinSorter.getMaxCoinIn()));
          }else {
            result.setFill(Color.BLUEVIOLET);
            result.setText(coinSorter.multiCoinCalculator(tAmount, currenyType));
          }
        } catch (NumberFormatException ex) {
          // if total amount and currency type format is not integer, retrieve an error
          result.setFill(Color.RED);
          result.setText(String.format("Please check your input has to be integer : %s", ex.getMessage()));
        }
      }
    });
    btnCancel.setOnAction(e -> goMainMenu());
    moveNext(grid, 400, 250, "Multi Coin Calculate");
  }

  /**
   * created print coin list pane
   *
   * @param coinSorter
```

```
 */
private void createPrintCoinListPane(CoinSorter coinSorter) {
   // creates base pane as border pane
   BorderPane bPane = createPaneWithBackButton();
   // creates child pane of the main border pane
   GridPane grid = createGridPane();
   // creates child pane title
   Text scenetitle = createPaneTitle("Coin List");
   grid.add(scenetitle, 0, 0, 2, 1);
   // gets coin list info and set as text value
   Text coinListText = new Text(String.valueOf(coinSorter.printCoinList()));
   coinListText.setWrappingWidth(300);
   grid.add(coinListText, 0, 1, 2, 1);
   bPane.setCenter(grid);
   // moves to the scene
   moveNext(bPane, 350, 250, "Print Coin List");
}

/**
 * creates display configuration detail pane
 *
 * @param coinSorter
 */
private void createDisplayConfigurationDetailsPane(CoinSorter coinSorter) {
   // created base pane with back buttons
   BorderPane bPane = createPaneWithBackButton();
   GridPane grid = createGridPane();
   Text scenetitle = createPaneTitle("Program Configs");
   grid.add(scenetitle, 0, 0, 2, 1);
   // creates the grid pane components
   Label currencyLabel = new Label("Currency :");
   grid.add(currencyLabel, 0, 1);
   Text currencyText = new Text(String.valueOf(coinSorter.getCurrency()));
   grid.add(currencyText, 1, 1);

   Label minCoinLabel = new Label("Minimum Coin :");
   grid.add(minCoinLabel, 0, 2);
   Text minCoinInText = new Text(String.valueOf(coinSorter.getMinCoinIn()));
   grid.add(minCoinInText, 1, 2);

   Label maxCoinLabel = new Label("Maximum Coin :");
   grid.add(maxCoinLabel, 0, 3);
   Text maxCoinInText = new Text(String.valueOf(coinSorter.getMaxCoinIn()));
   grid.add(maxCoinInText, 1, 3);
   bPane.setCenter(grid);
   // moves to the scene
   moveNext(bPane, 350, 250, "Display Program Config");
}

/**
 * created grid pane
 *
 * @return grid pane
 */
private GridPane createGridPane() {
   GridPane grid = new GridPane();
   grid.setAlignment(Pos.CENTER);
   grid.setHgap(10);
```

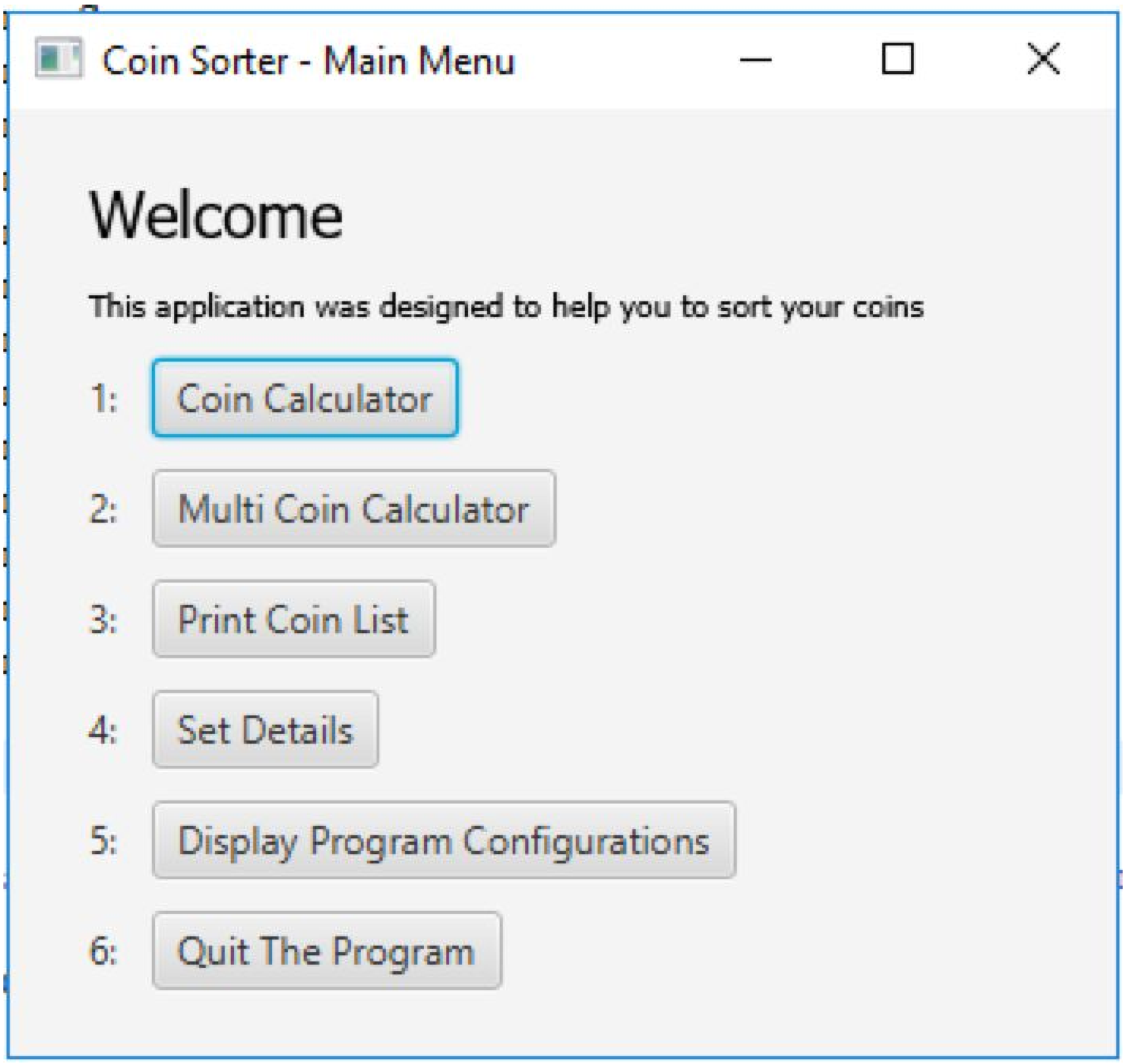
```
  grid.setVgap(10);
  return grid;
}

/**
 * moves to the next scene
 *
 * @param pane
 * @param width
 * @param height
 * @param title
 */
private void moveNext(Pane pane, int width, int height, String title) {
  Scene scene = new Scene(pane, width, height);
  window.setTitle(title);
  showScene(window, scene);
}

/**
 * shows scene on the stage
 *
 * @param stage
 * @param scene
 */
private void showScene(Stage stage, Scene scene) {
  stage.setScene(scene);
  stage.show();
}

/**
 * creates pane with back main menu button
 *
 * @return border pane
 */
private BorderPane createPaneWithBackButton() {
  BorderPane bPane = new BorderPane();
  Button btn = new Button("Return To Main Menu");
  HBox hbBtn = new HBox(10);
  hbBtn.setMinHeight(40);
  hbBtn.setAlignment(Pos.BASELINE_CENTER);
  hbBtn.getChildren().add(btn);
  btn.setOnAction(e -> goMainMenu());
  bPane.setBottom(hbBtn);
  return bPane;
}

/**
 * adds text field with label
 *
 * @param grid
 * @param order
 * @param title
 * @return text field
 */
private TextField addTextFieldWithLabel(GridPane panel, int order, String title) {
  Label label = new Label(title);
  panel.add(label, 0, order);
  TextField texField = new TextField();
```

```
    panel.add(texField, 1, order);
    return texField;
}

/**
 * creates title vertical box
 *
 * @param title
 * @param subTitle
 * @return vertical box
 */
private VBox createTitleBox(String title, String subTitle) {
    VBox titleBox = new VBox(10);
    Text scenetitle = new Text(title);
    scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
    Text sub = new Text(subTitle);
    sub.setWrappingWidth(300);
    sub.setFont(Font.font("Tahoma", FontWeight.NORMAL, 10));
    titleBox.getChildren().addAll(scenetitle, sub);
    return titleBox;
}

/**
 * creates confirm button horizontal Box
 *
 * @param position
 * @return horizontal box
 */
private HBox createConfirmButtonHBox(Pos position) {
    Button btnOk = new Button("OK");
    Button btnCancel = new Button("Cancel");
    HBox hbBtn = new HBox(10);
    hbBtn.setAlignment(position);
    hbBtn.getChildren().addAll(btnOk, btnCancel);
    return hbBtn;
}

/**
 * adds button with label
 *
 * @param grid
 * @param order
 * @param title
 * @param buttonName
 * @return button
 */
private Button addButtonWithLabel(GridPane grid, int order, String title, String buttonName) {
    Label label = new Label(title);
    grid.add(label, 0, order);
    Button button = new Button(buttonName);
    grid.add(button, 1, order);
    return button;
}

/**
 * creates pane title
 *
 * @param title
```

```
     * @return text
     */
    private Text createPaneTitle(String title) {
        Text scenetitle = new Text(title);
        scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
        return scenetitle;
    }

    /**
     * quits the application
     */
    private void quit() {
        Platform.exit();
    }

    /**
     * goes main menu
     */
    public void goMainMenu() {
        showScene(window, mainScene);
    }

    /**
     * goes sub menu
     */
    public void goSubMenu() {
        showScene(window, subMenuScene);
    }

}
```

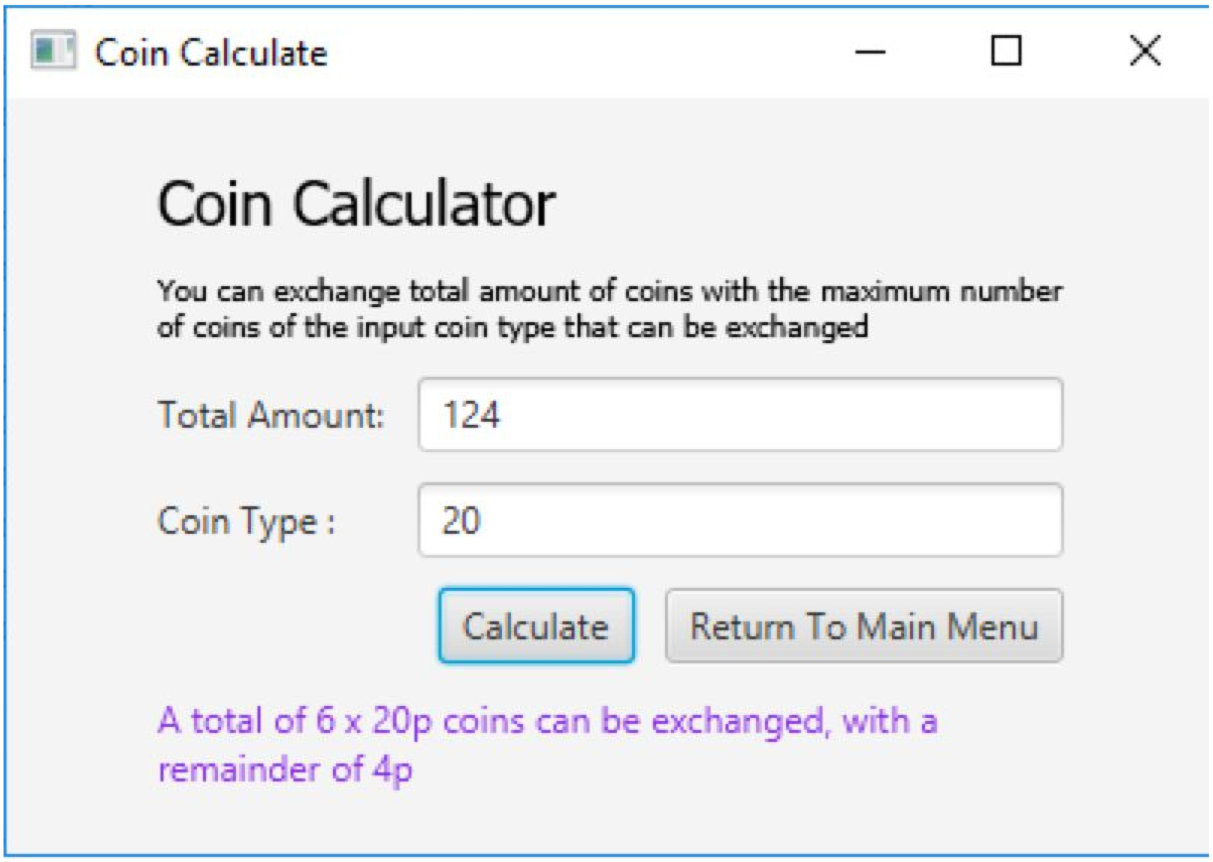
Evidence for Test Coin Sorter GUI class

Main Menu


Coin Sorter - Main Menu
Welcome
This application was designed to help you to sort your coins
1:
Coin Calculator
2:
Multi Coin Calculator
3:
Print Coin List
4:
Set Details
5:
Display Program Configurations
6:
Quit The Program

Command 1 Coin Calculator

Success:



Coin Calculator

You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

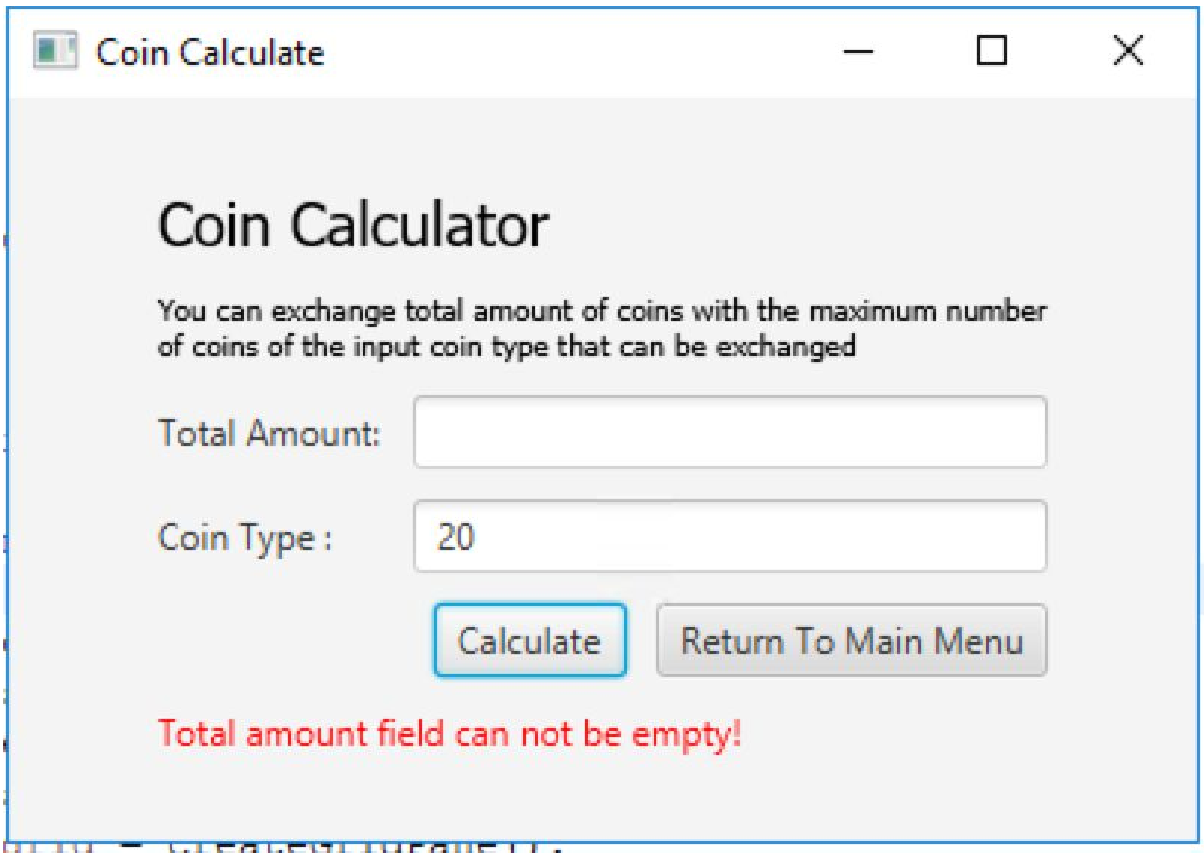
Total Amount: 124

Coin Type : 20

A total of 6 x 20p coins can be exchanged, with a remainder of 4p

Fail Scenarios:

1- Total amount can not be empty:


Coin Calculate

Coin Calculator

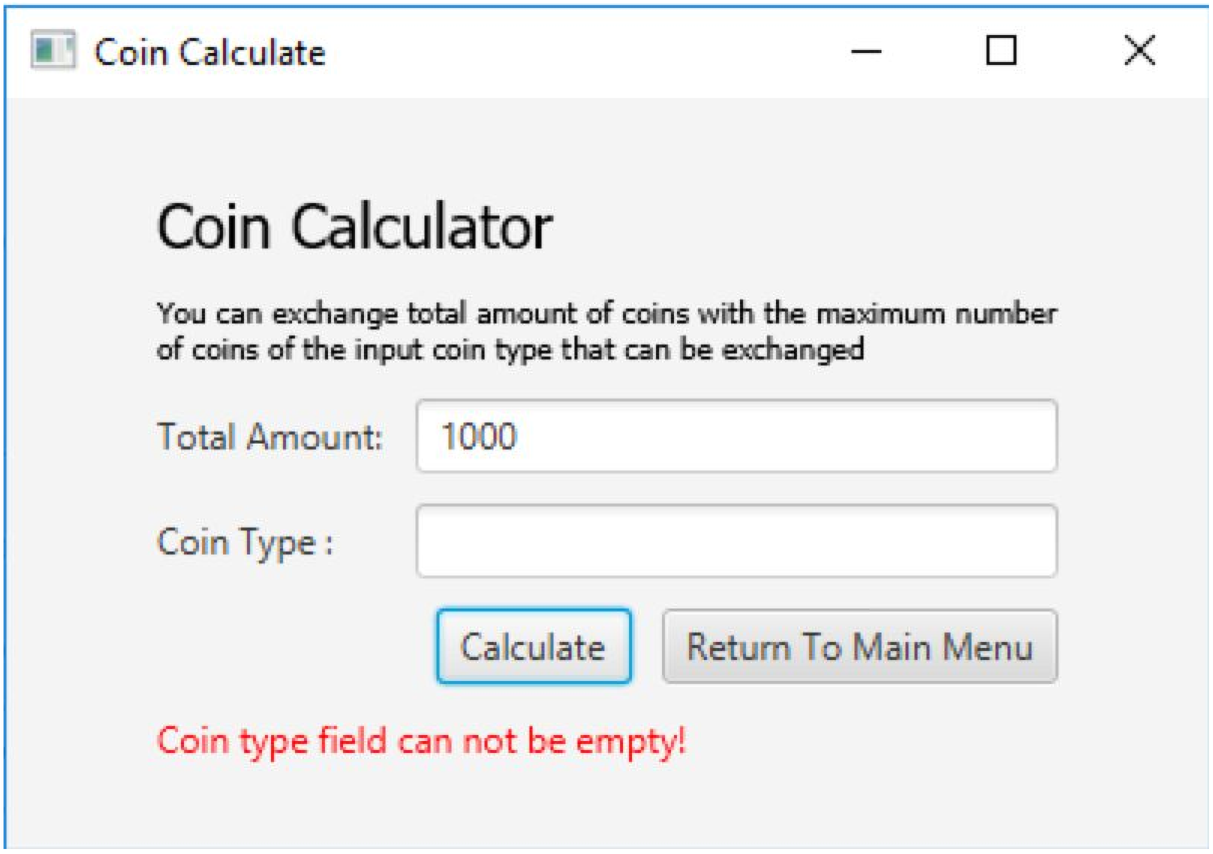
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

Total Amount:

Coin Type : 20

Total amount field can not be empty!

2- Currency type can not be empty:



Coin Calculate

Coin Calculator

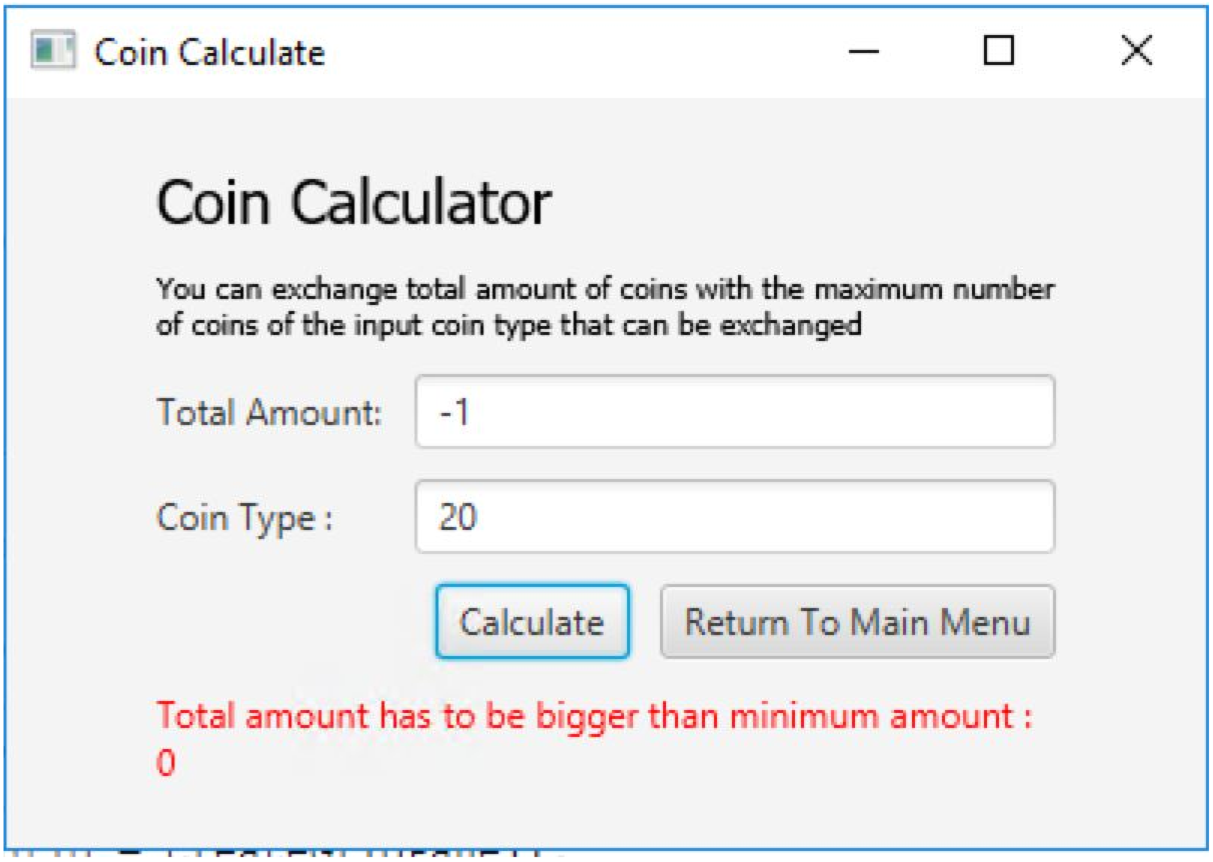
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

Total Amount: 1000

Coin Type :

Coin type field can not be empty!

3- Total amount less than minimum value:



Coin Calculate

Coin Calculator

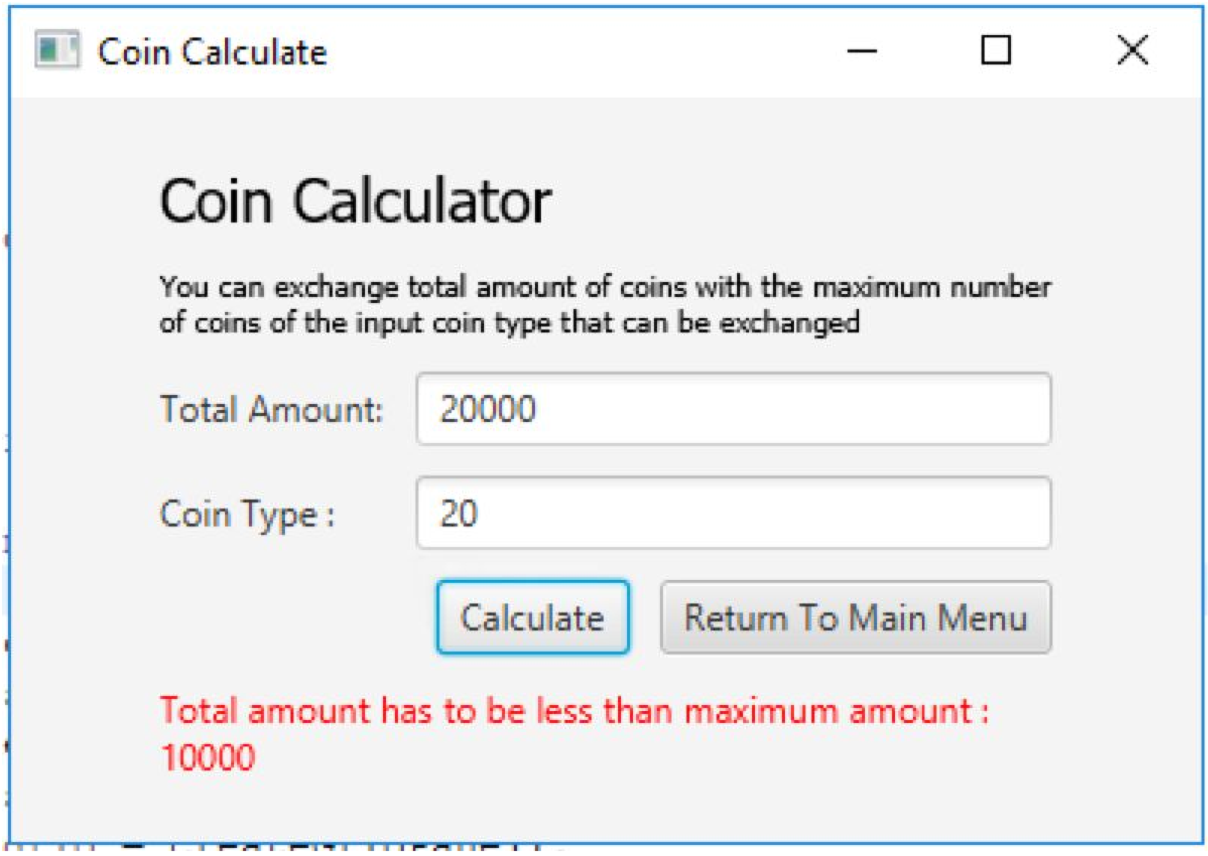
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

Total Amount: -1

Coin Type : 20

Total amount has to be bigger than minimum amount : 0

4- Total amount bigger than maximum value:



Coin Calculate

Coin Calculator

You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

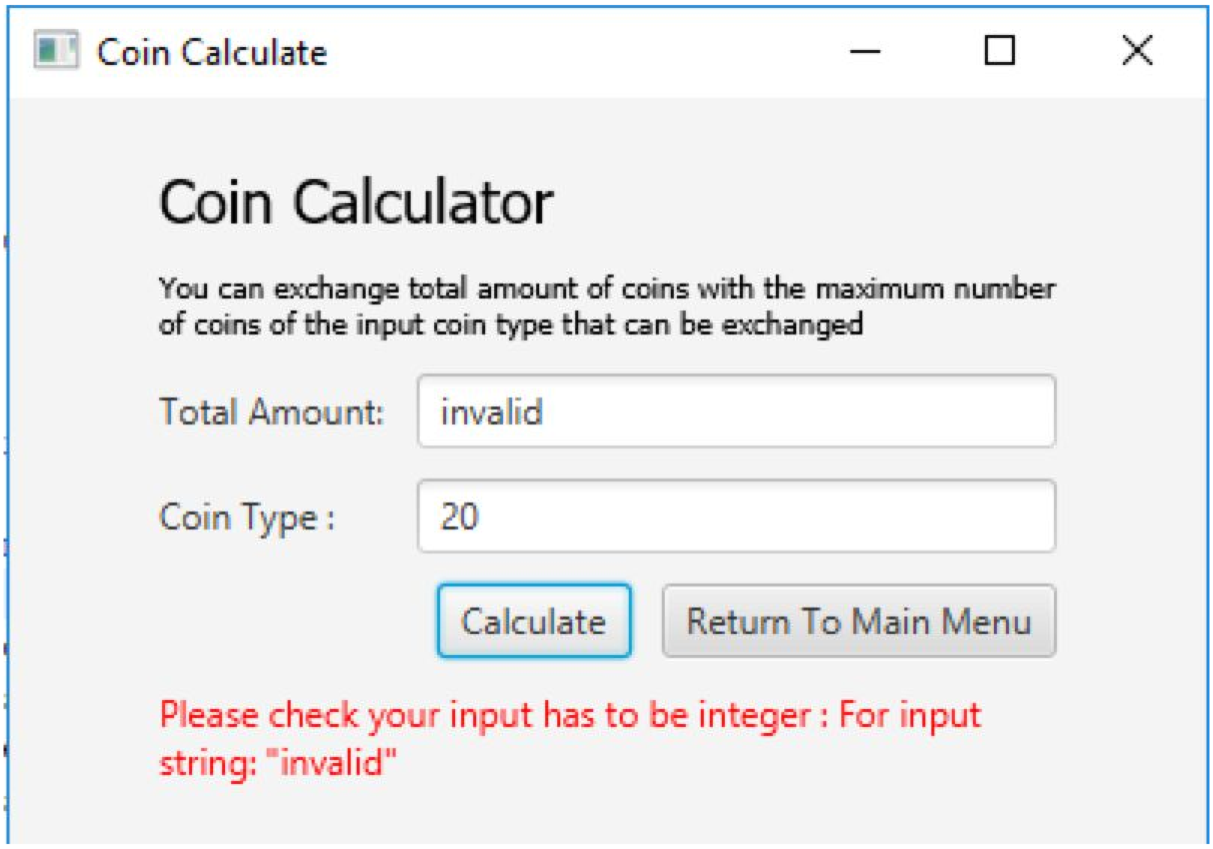
Total Amount: 20000

Coin Type : 20

Total amount has to be less than maximum amount : 10000

5- Total amount and currency type has to be integer value :

Total amount:


Coin Calculate

Coin Calculator

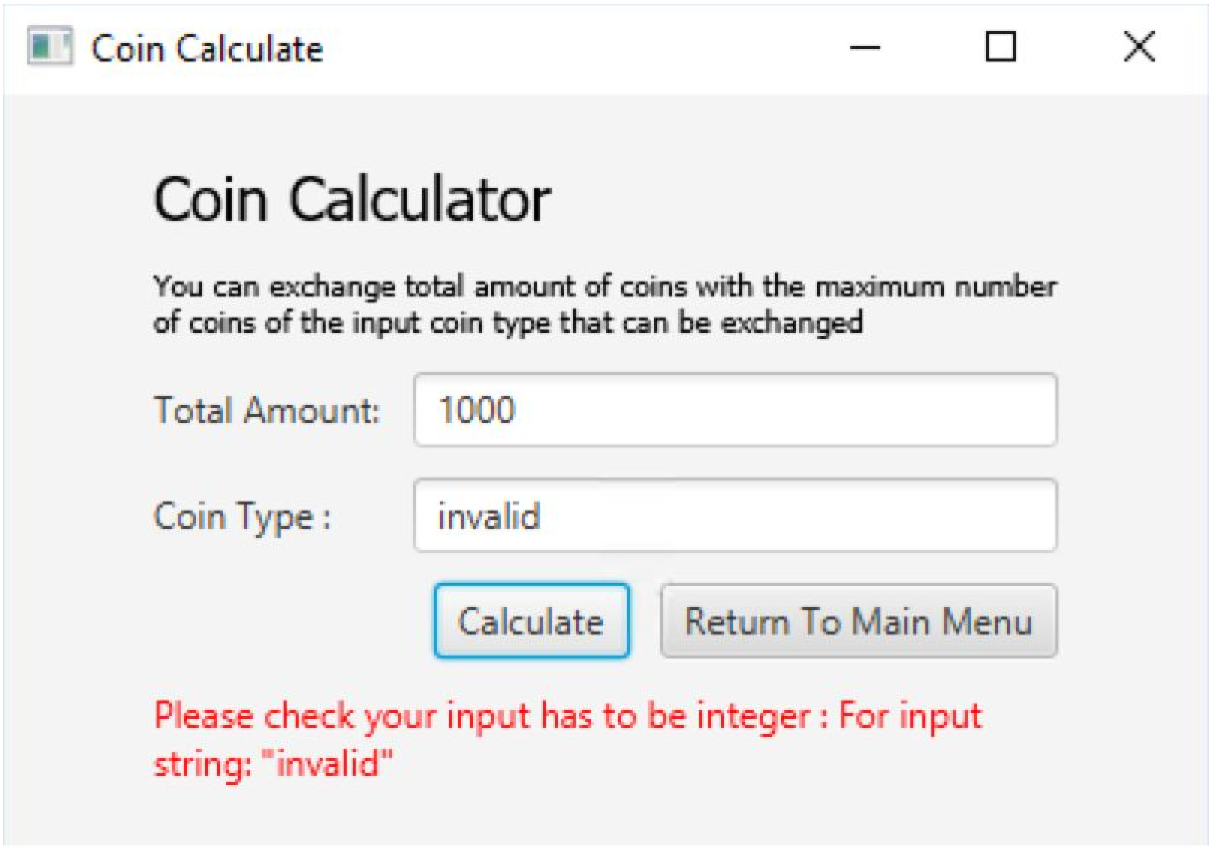
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

Total Amount: invalid

Coin Type : 20

Please check your input has to be integer : For input string: "invalid"

Coin type:


Coin Calculate

Coin Calculator

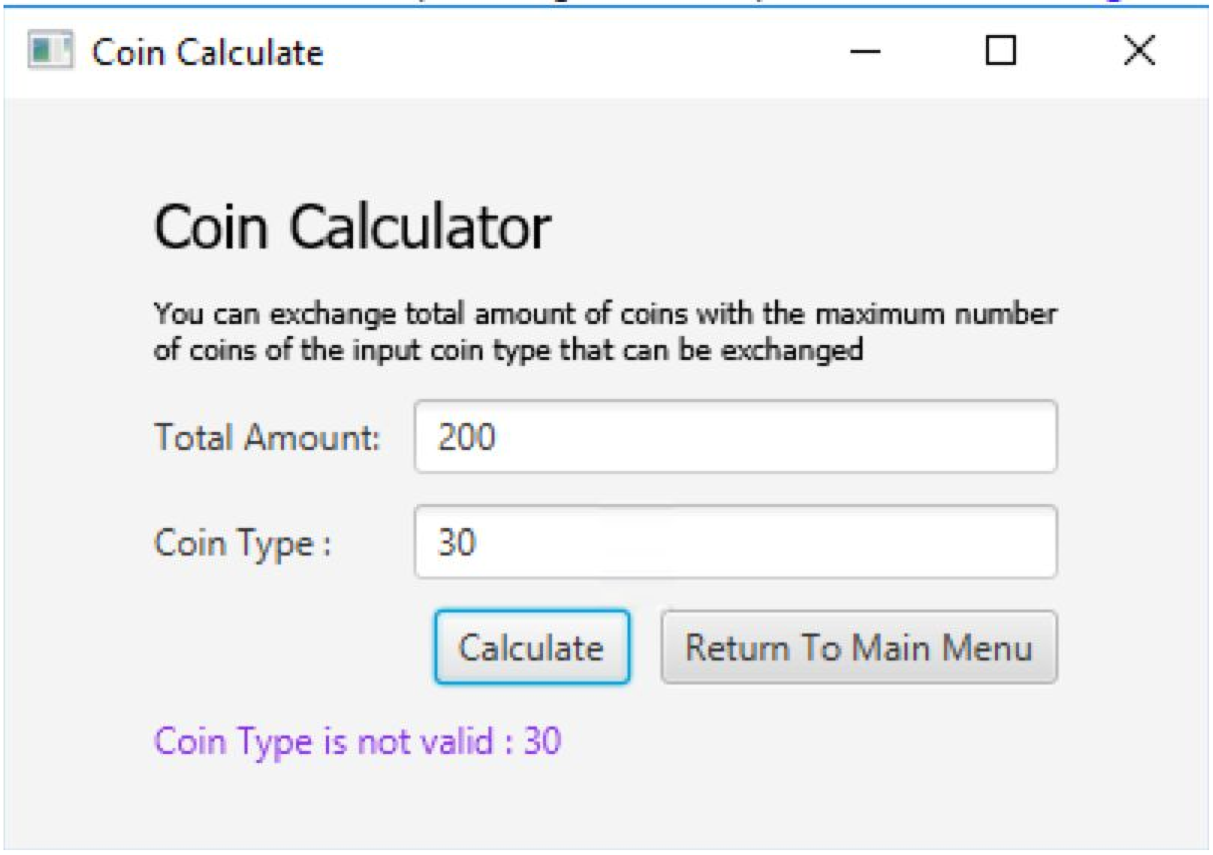
You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

Total Amount: 1000

Coin Type : invalid

Please check your input has to be integer : For input string: "invalid"

6- Coin type is not in coin list:



Coin Calculate

Coin Calculator

You can exchange total amount of coins with the maximum number of coins of the input coin type that can be exchanged

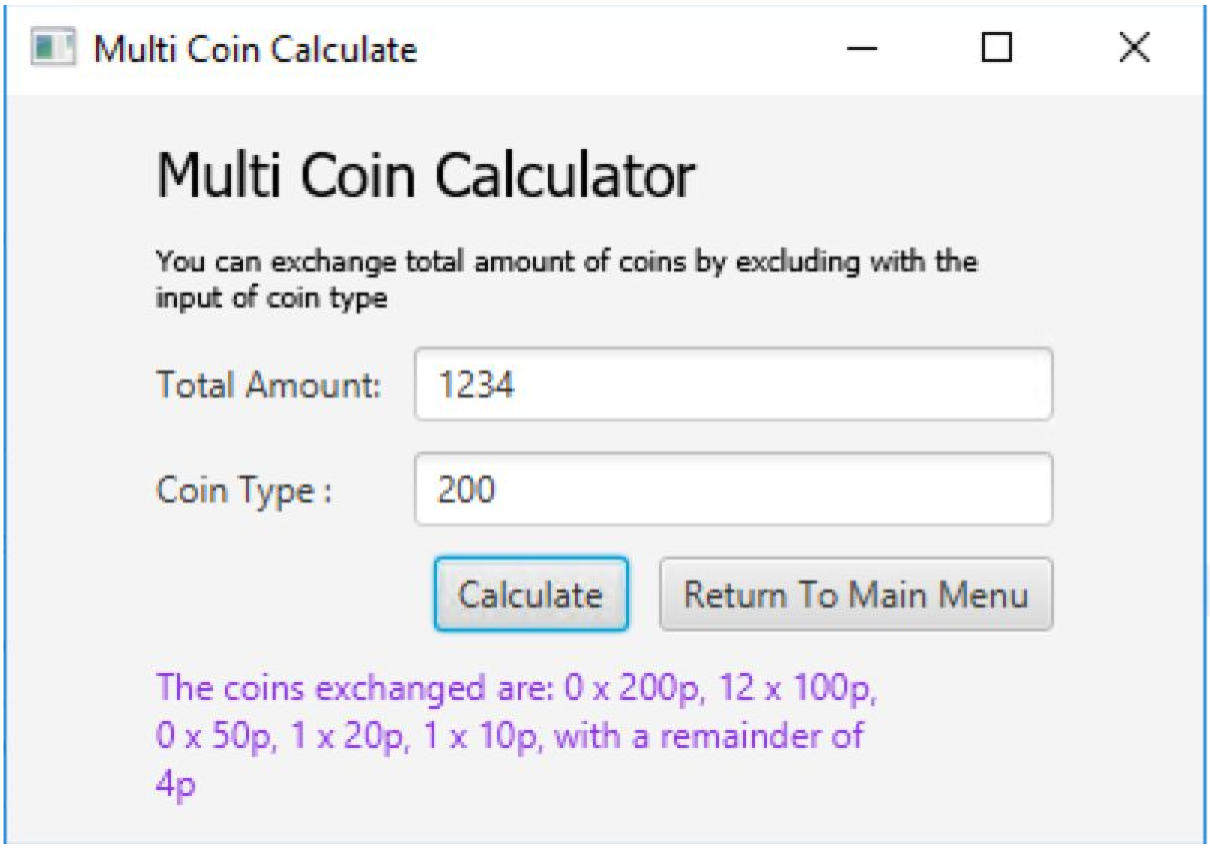
Total Amount: 200

Coin Type : 30

Coin Type is not valid : 30

Command 2 Multiple Coin Calculator

Success:



Multi Coin Calculate

Multi Coin Calculator

You can exchange total amount of coins by excluding with the input of coin type

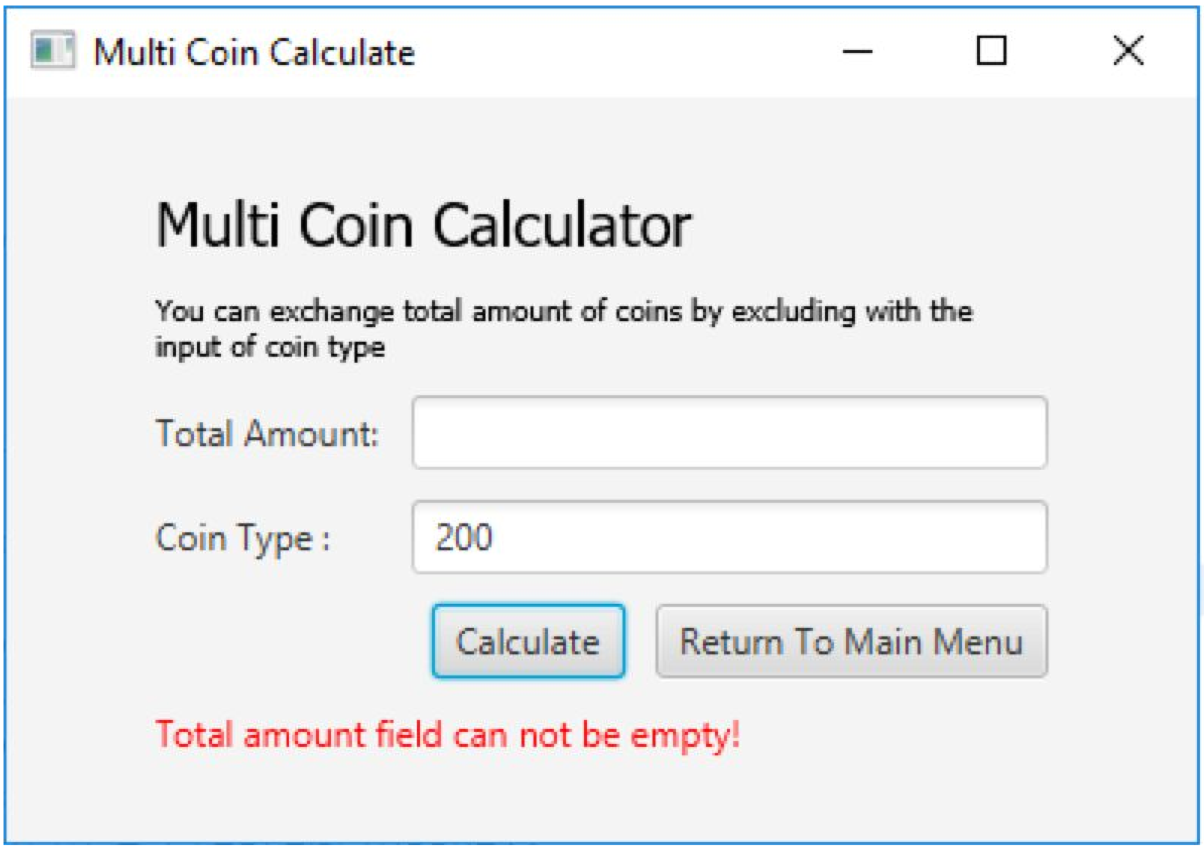
Total Amount: 1234

Coin Type : 200

The coins exchanged are: 0 x 200p, 12 x 100p, 0 x 50p, 1 x 20p, 1 x 10p, with a remainder of 4p

Fail Scenarios:

1- Total amount can not be empty:



Multi Coin Calculate

Multi Coin Calculator

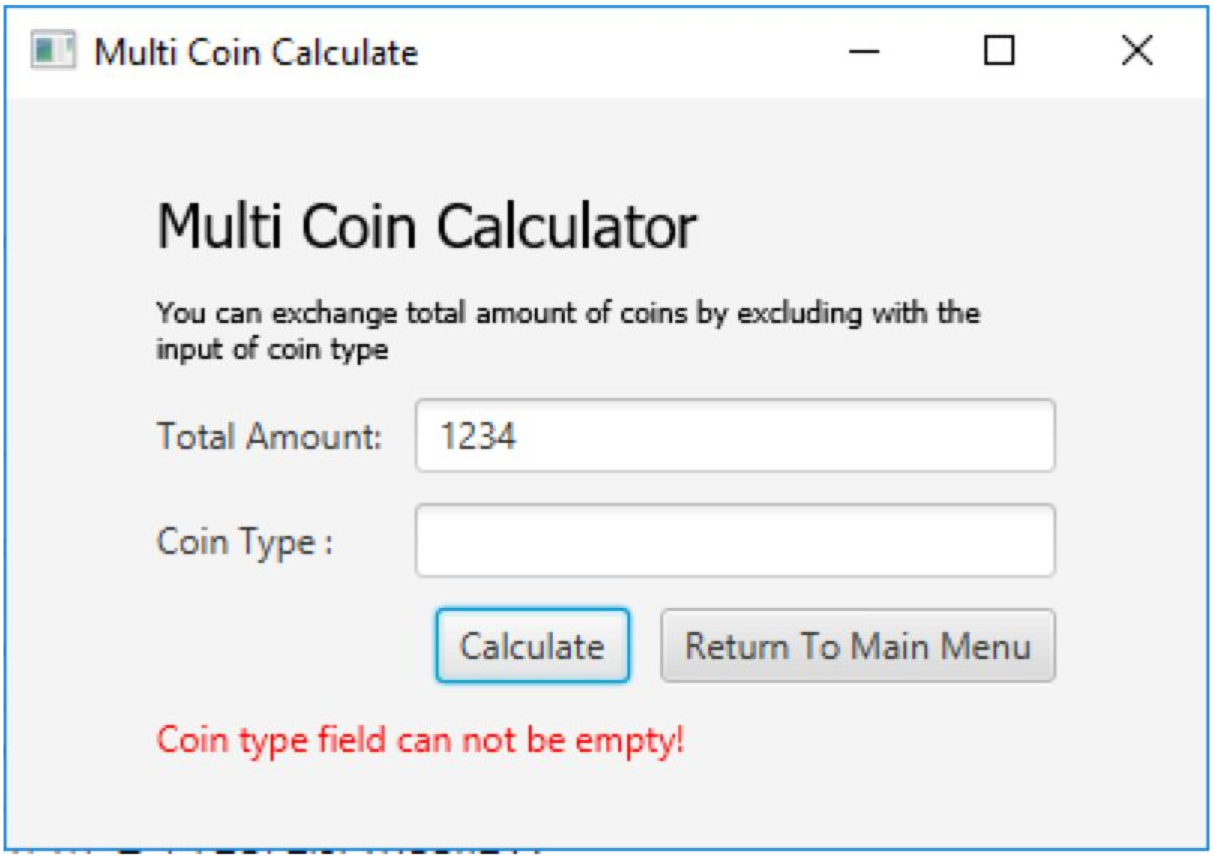
You can exchange total amount of coins by excluding with the input of coin type

Total Amount:

Coin Type : 200

Total amount field can not be empty!

2- Currency type can not be empty:



Multi Coin Calculate

Multi Coin Calculator

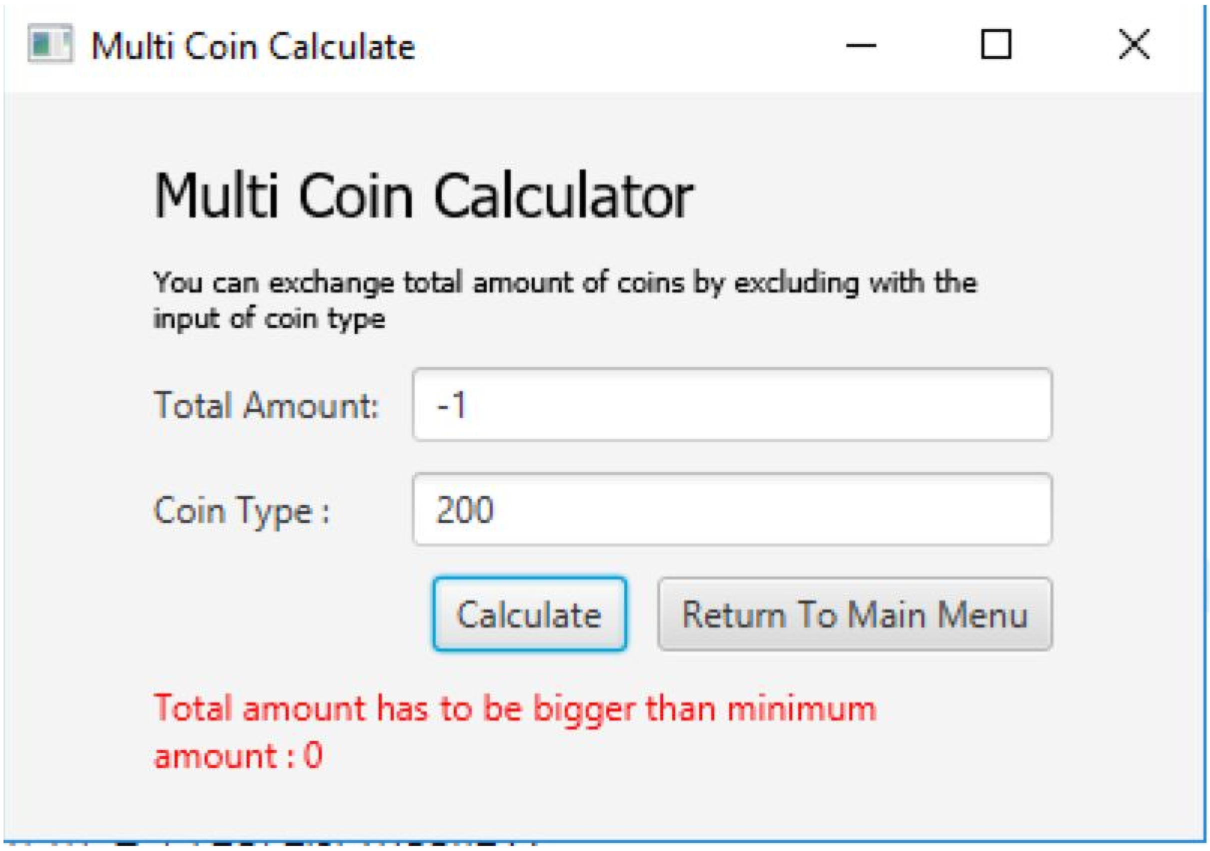
You can exchange total amount of coins by excluding with the input of coin type

Total Amount: 1234

Coin Type :

Coin type field can not be empty!

3- Total amount less than minimum value:



Multi Coin Calculate

Multi Coin Calculator

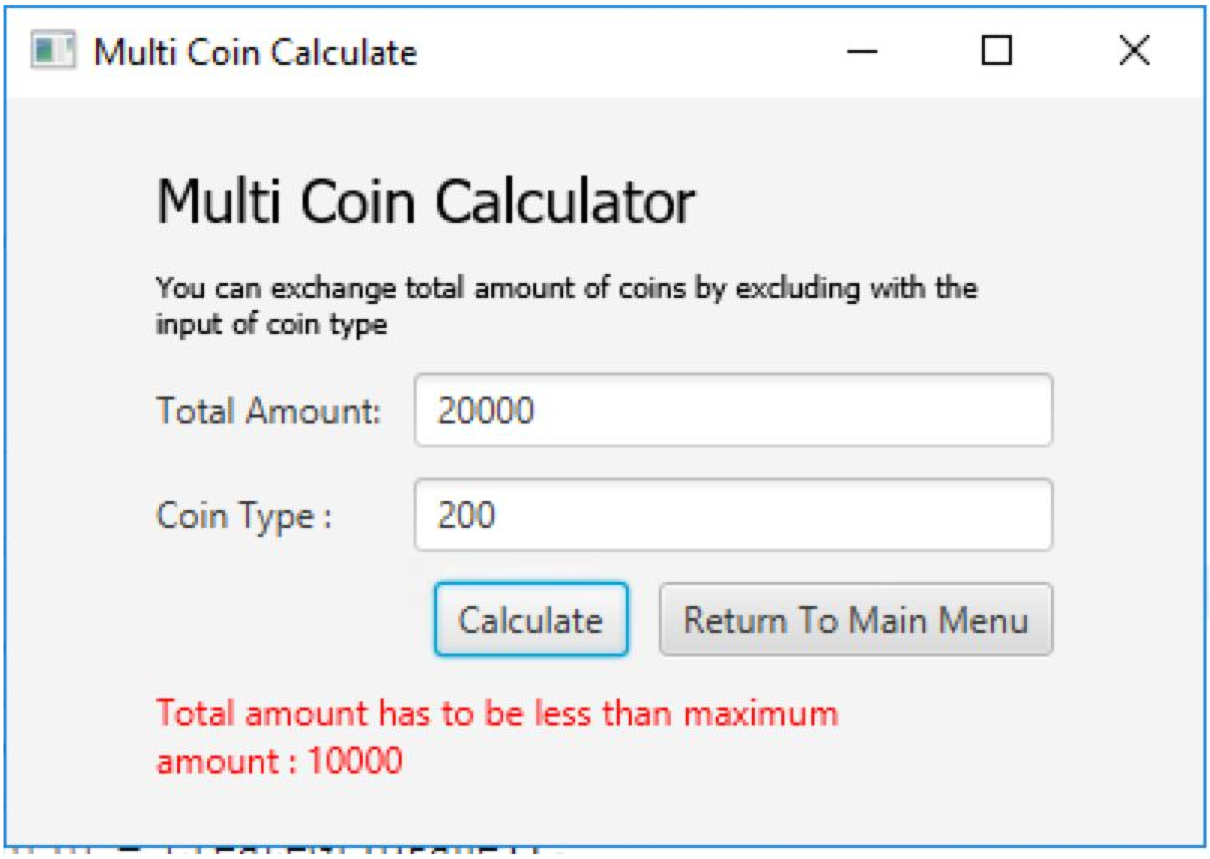
You can exchange total amount of coins by excluding with the input of coin type

Total Amount: -1

Coin Type : 200

Total amount has to be bigger than minimum amount : 0

4- Total amount bigger than maximum value:



Multi Coin Calculator

You can exchange total amount of coins by excluding with the input of coin type

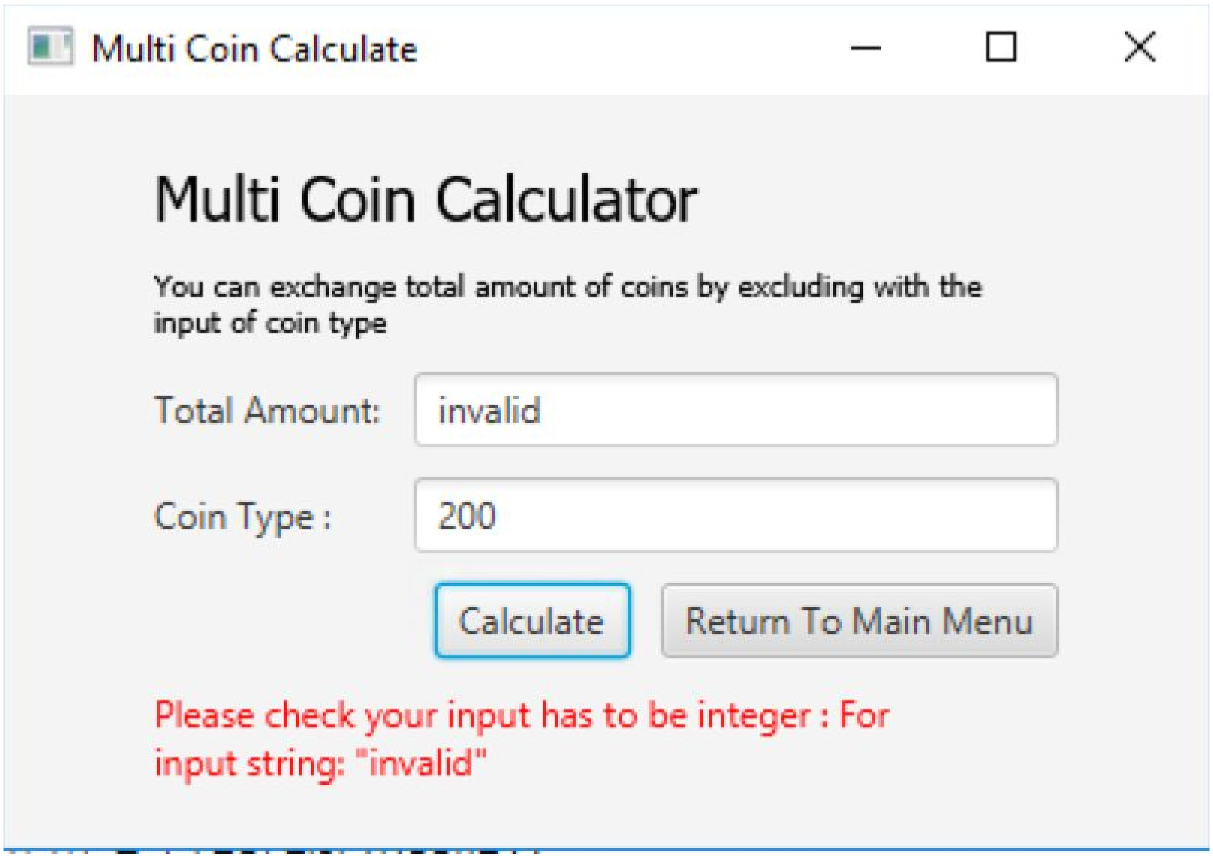
Total Amount: 20000

Coin Type : 200

Total amount has to be less than maximum amount : 10000

5- Total amount and currency type has to be integer value :

Total amount:


Multi Coin Calculate

Multi Coin Calculator

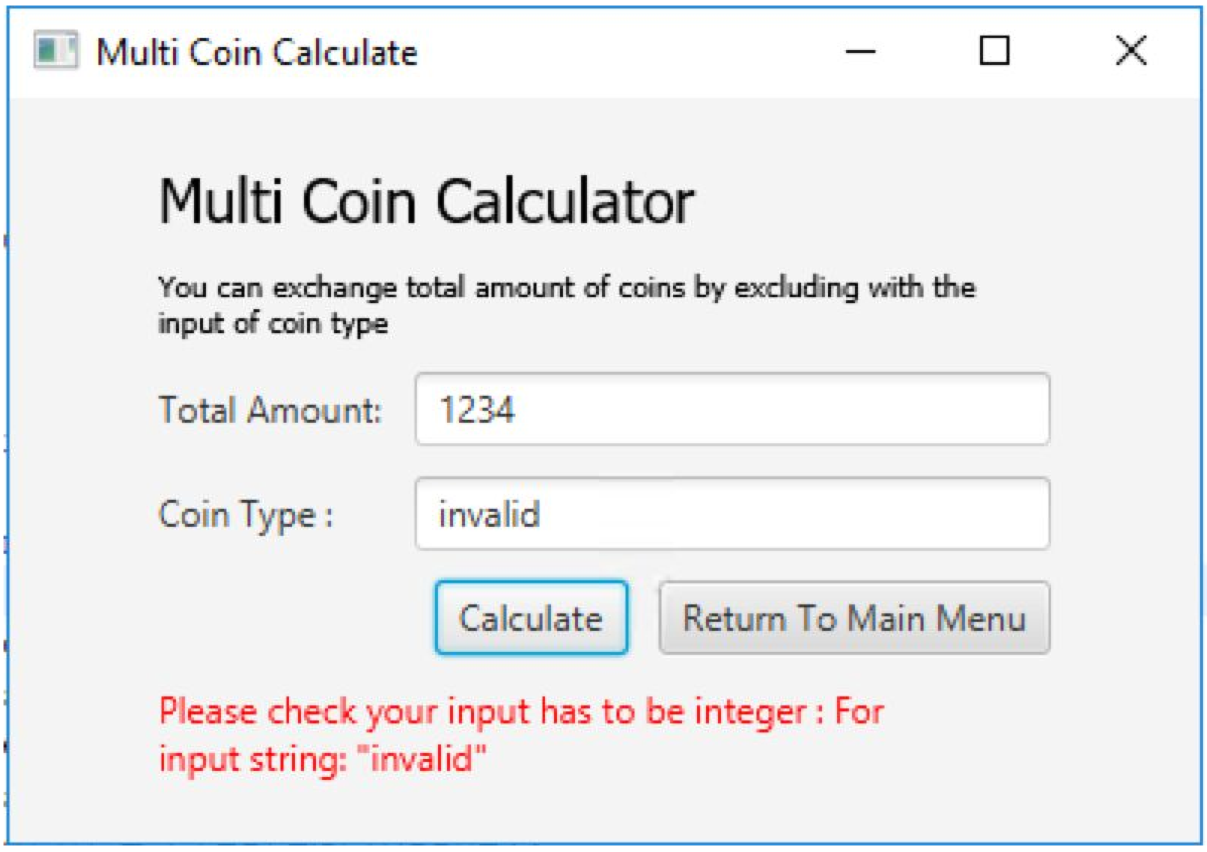
You can exchange total amount of coins by excluding with the input of coin type

Total Amount: invalid

Coin Type : 200

Please check your input has to be integer : For input string: "invalid"

Coin type:


Multi Coin Calculate

Multi Coin Calculator

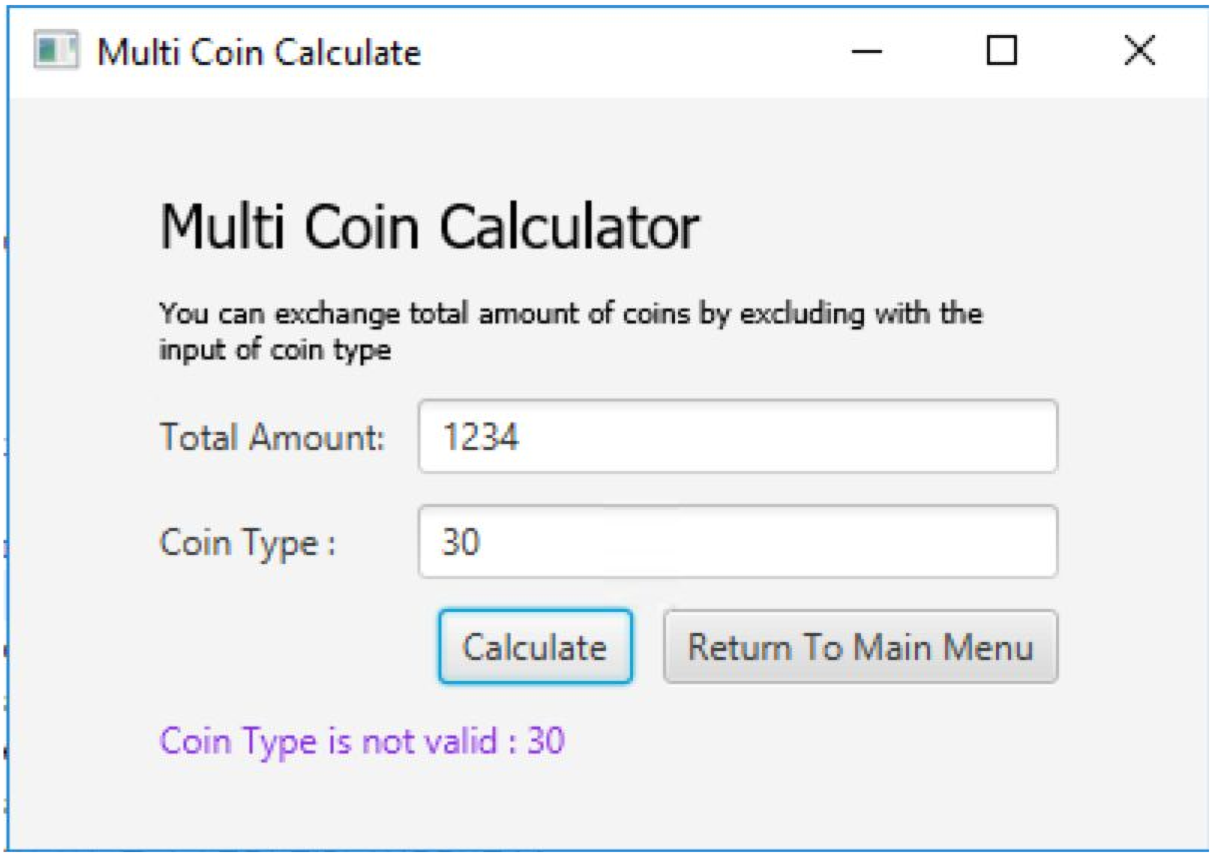
You can exchange total amount of coins by excluding with the input of coin type

Total Amount: 1234

Coin Type : invalid

Please check your input has to be integer : For input string: "invalid"

6- Coin type is not in coin list:



Multi Coin Calculator

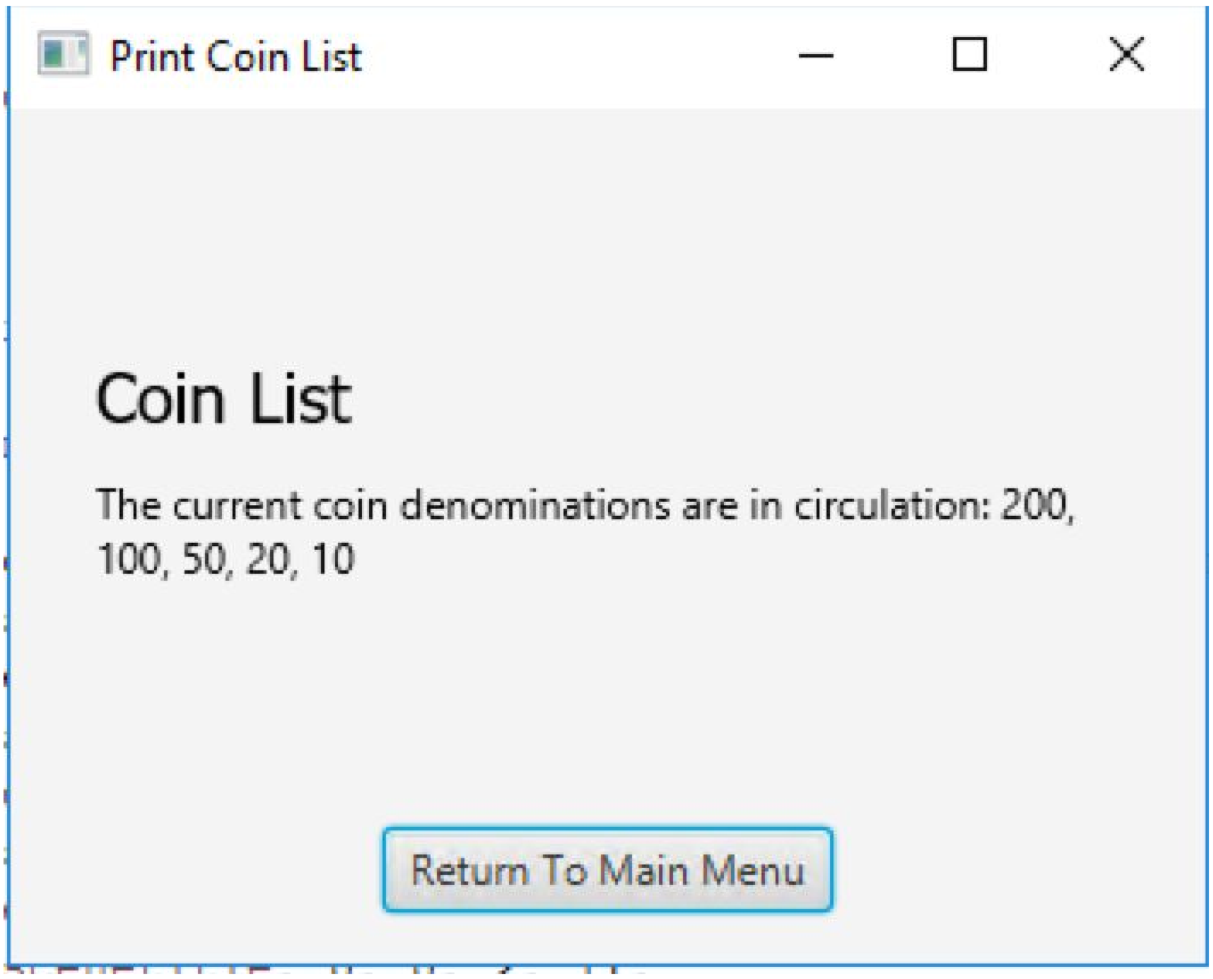
You can exchange total amount of coins by excluding with the input of coin type

Total Amount: 1234

Coin Type : 30

Coin Type is not valid : 30

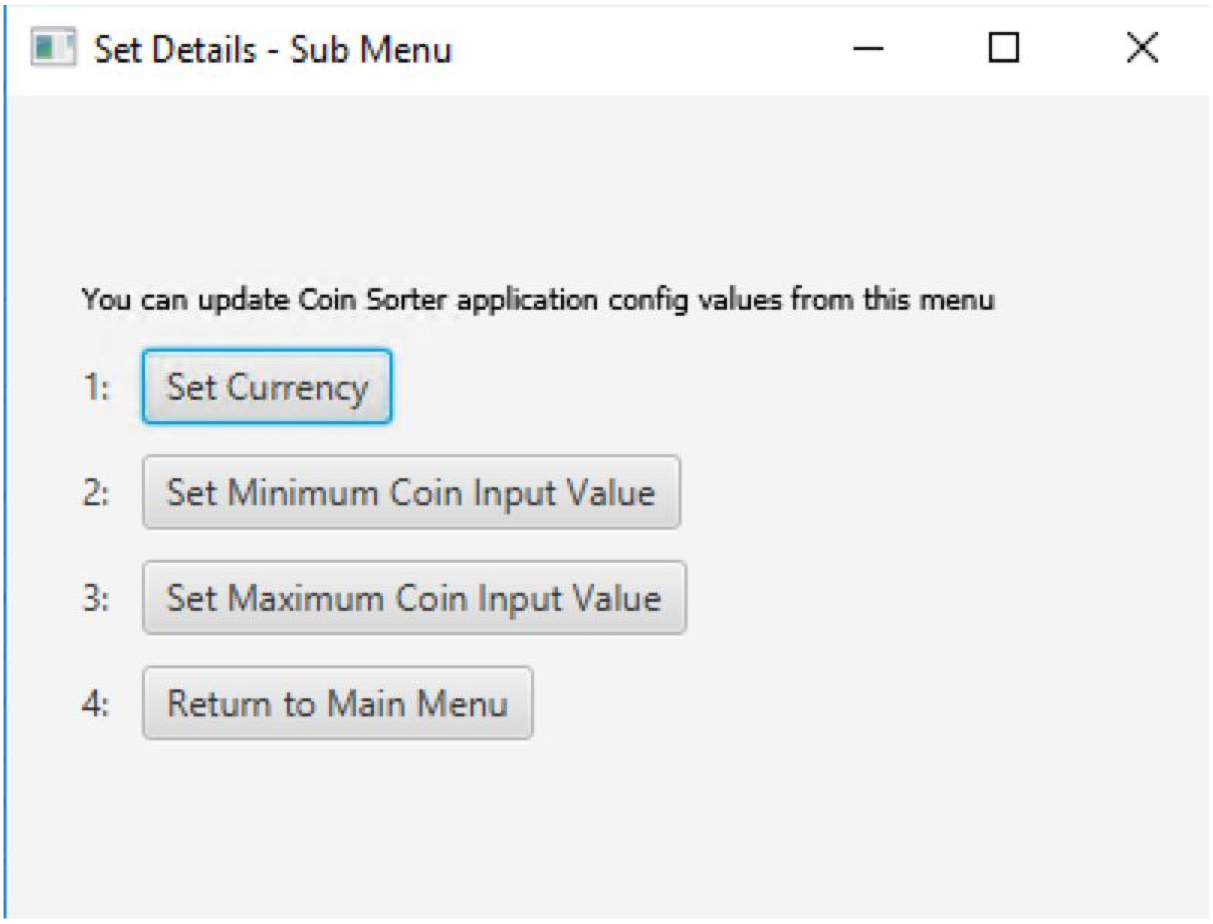
Command 3 Print Coin List



Coin List

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

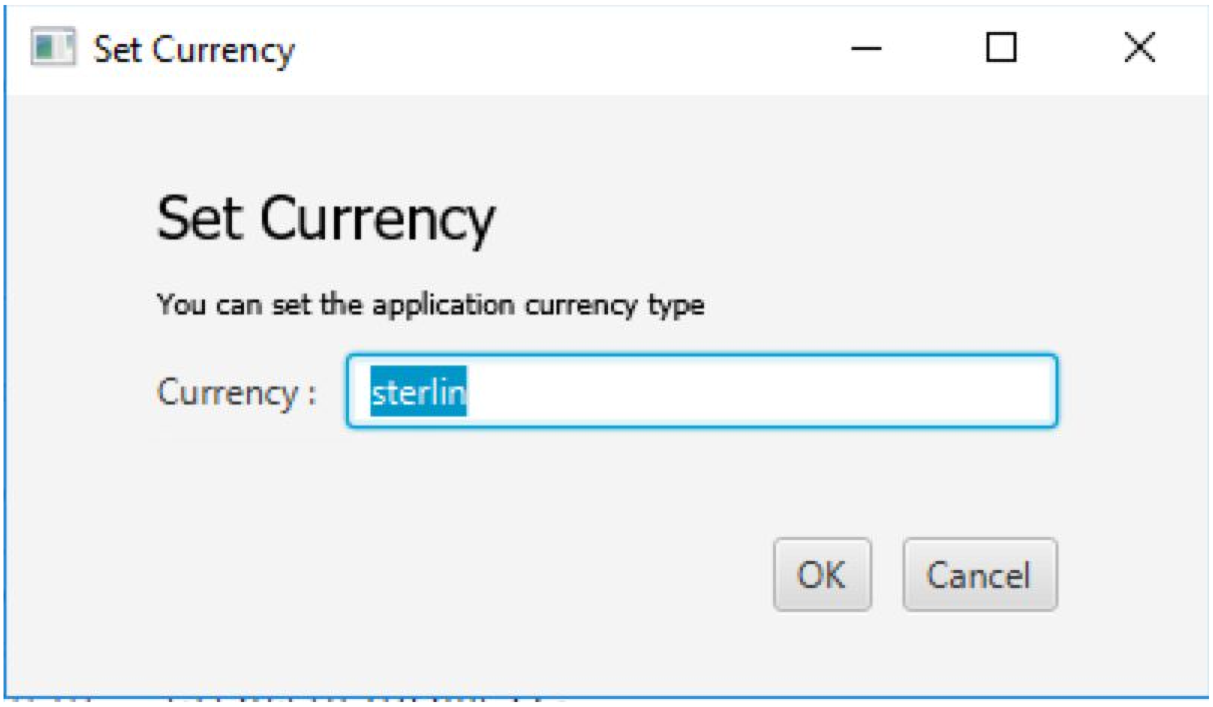
Command 4 Set Details Sub Menu



You can update Coin Sorter application config values from this menu

Command 4.1 Set Currency

Success:



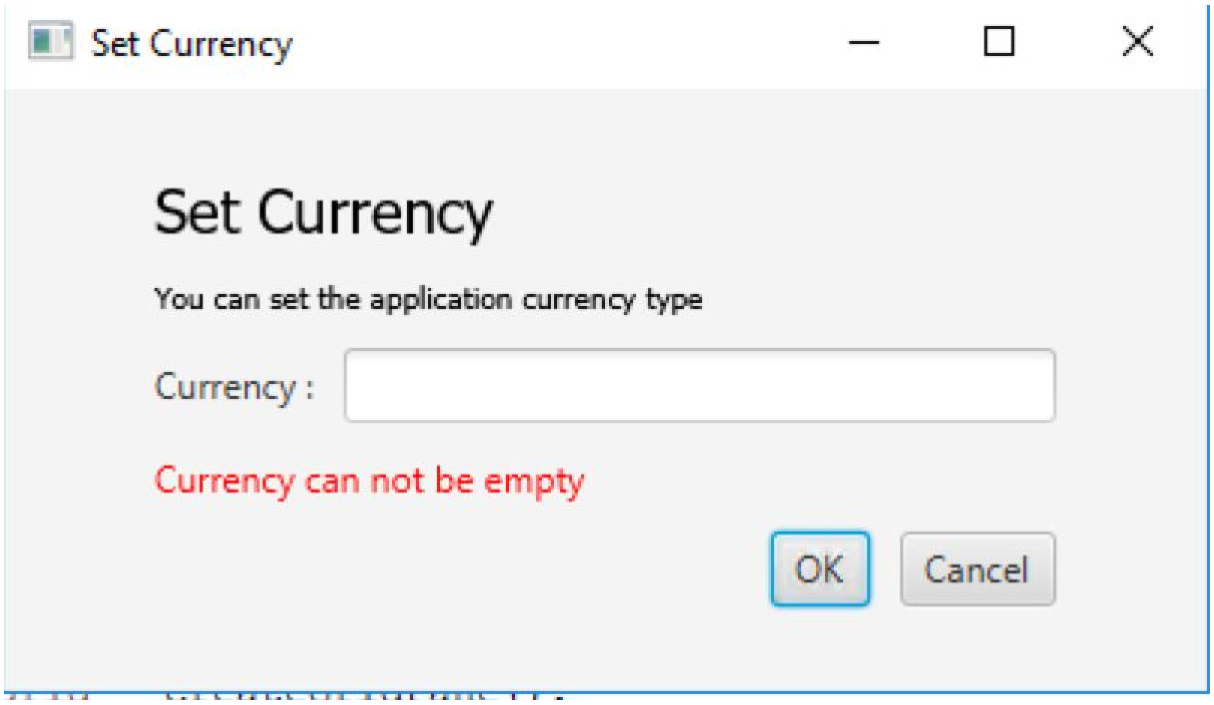
Set Currency

You can set the application currency type

Currency : sterlin

Fail Scenarios:

1- Currency can not be empty:



Set Currency

Set Currency

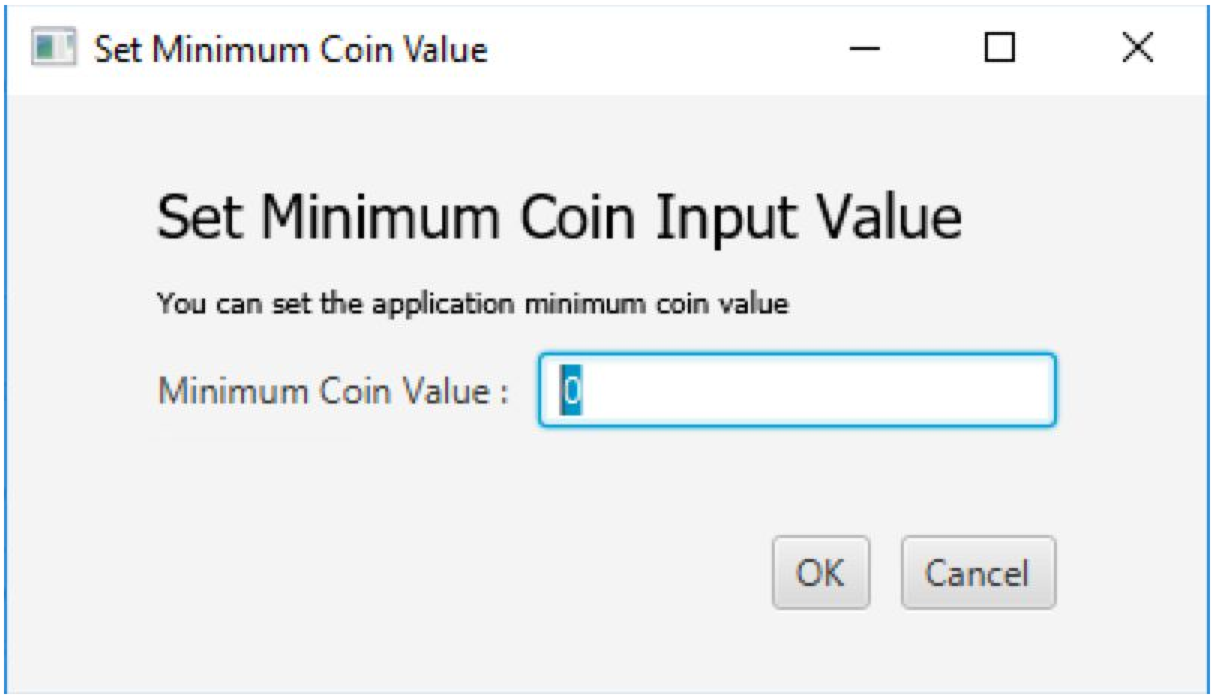
You can set the application currency type

Currency :

Currency can not be empty

Command 4.2 Set Minimum Coin

Succes:



Set Minimum Coin Value

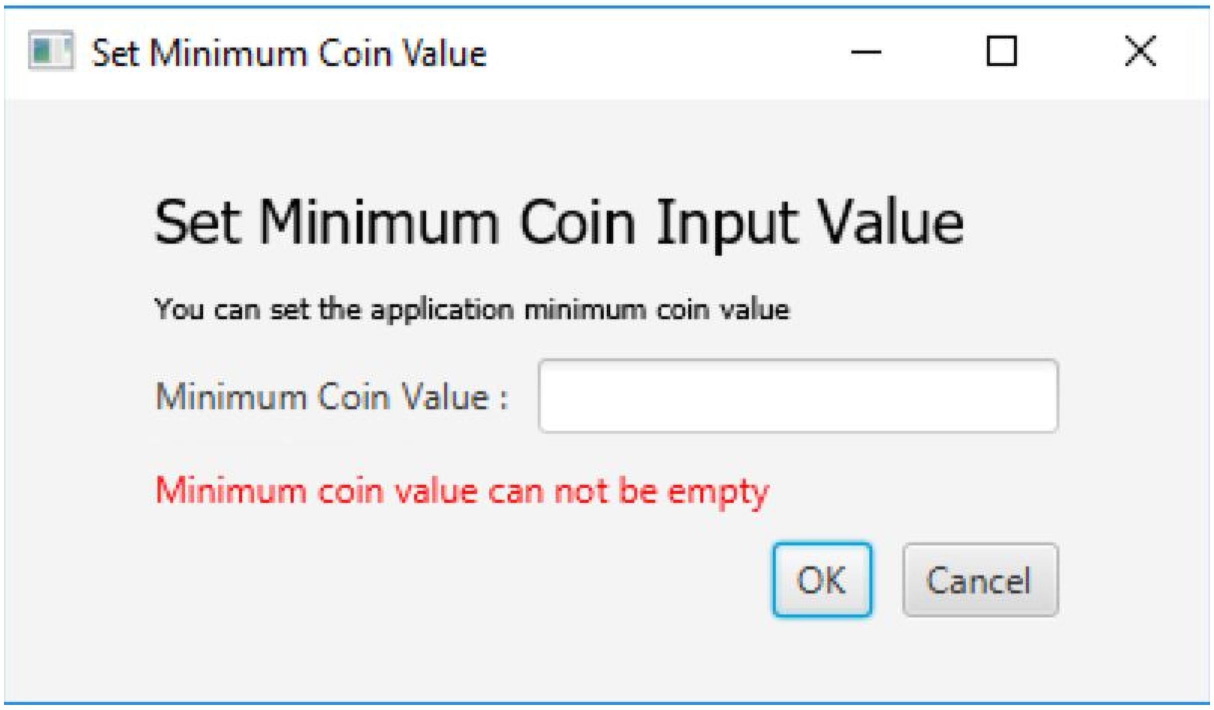
Set Minimum Coin Input Value

You can set the application minimum coin value

Minimum Coin Value : 0

Fail Scenarios:

1- Min Value can not be empty:


Set Minimum Coin Value

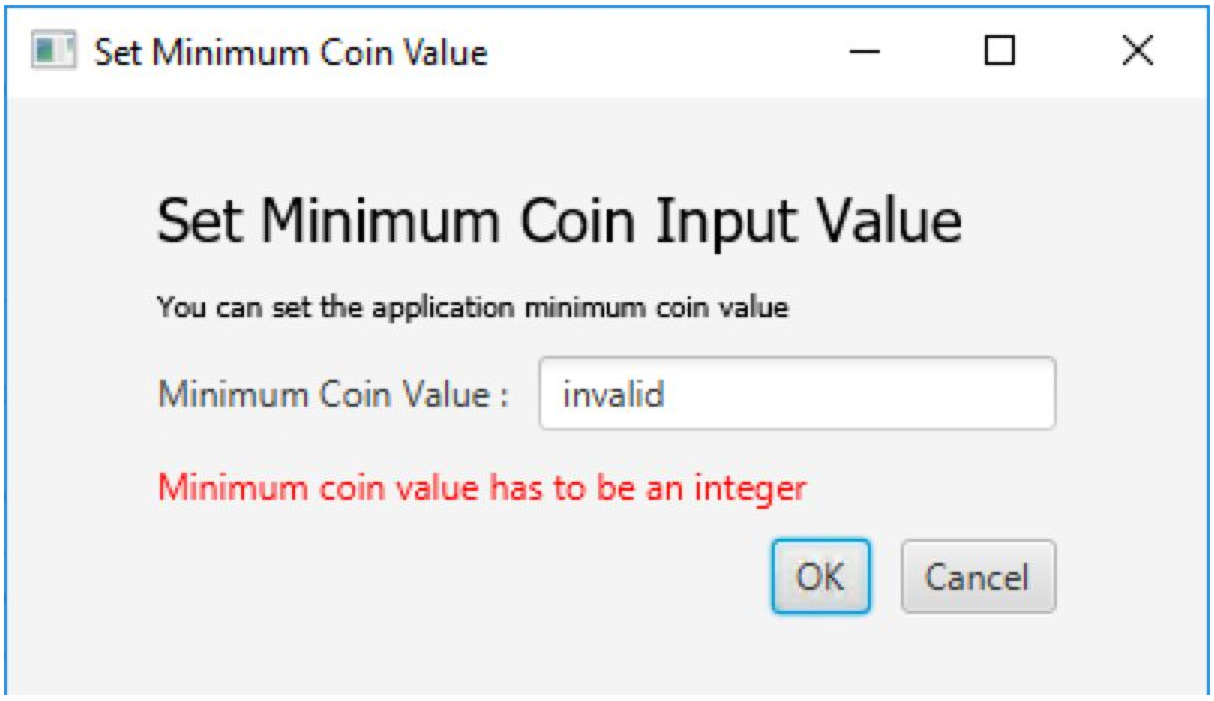
Set Minimum Coin Input Value

You can set the application minimum coin value

Minimum Coin Value :

Minimum coin value can not be empty

2- Min Value type is invalid:


Set Minimum Coin Value

Set Minimum Coin Input Value

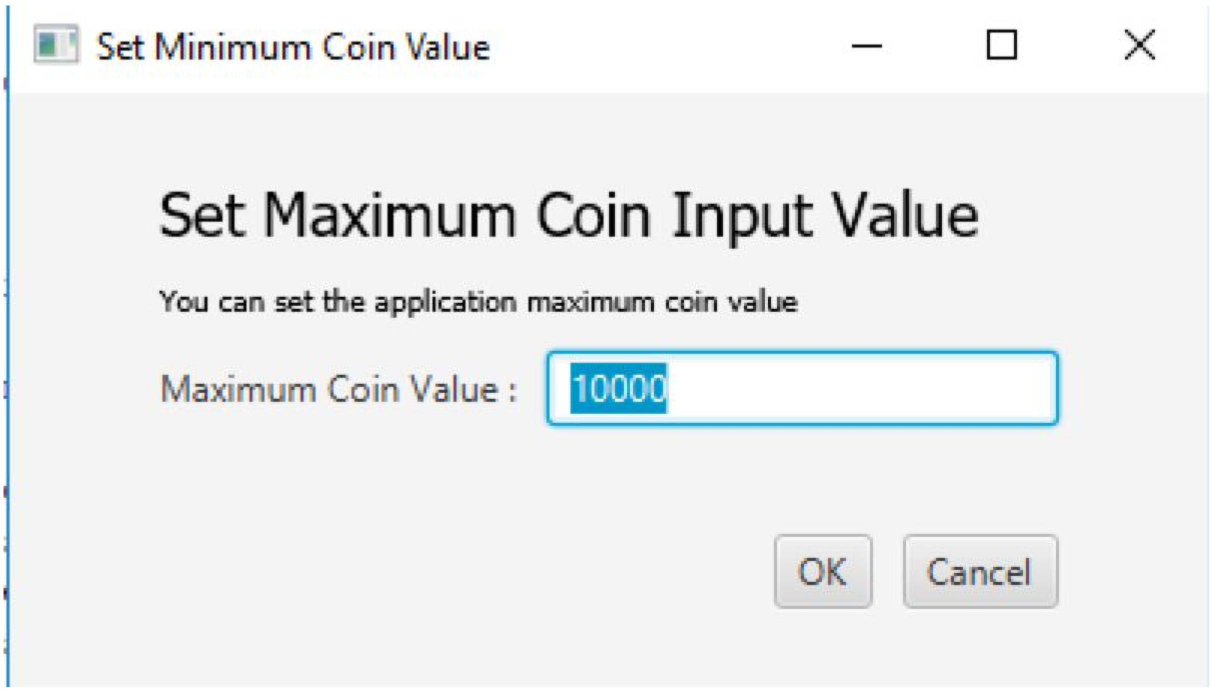
You can set the application minimum coin value

Minimum Coin Value : invalid

Minimum coin value has to be an integer

Command 4.3 Set Maximum Coin

Success:


Set Minimum Coin Value

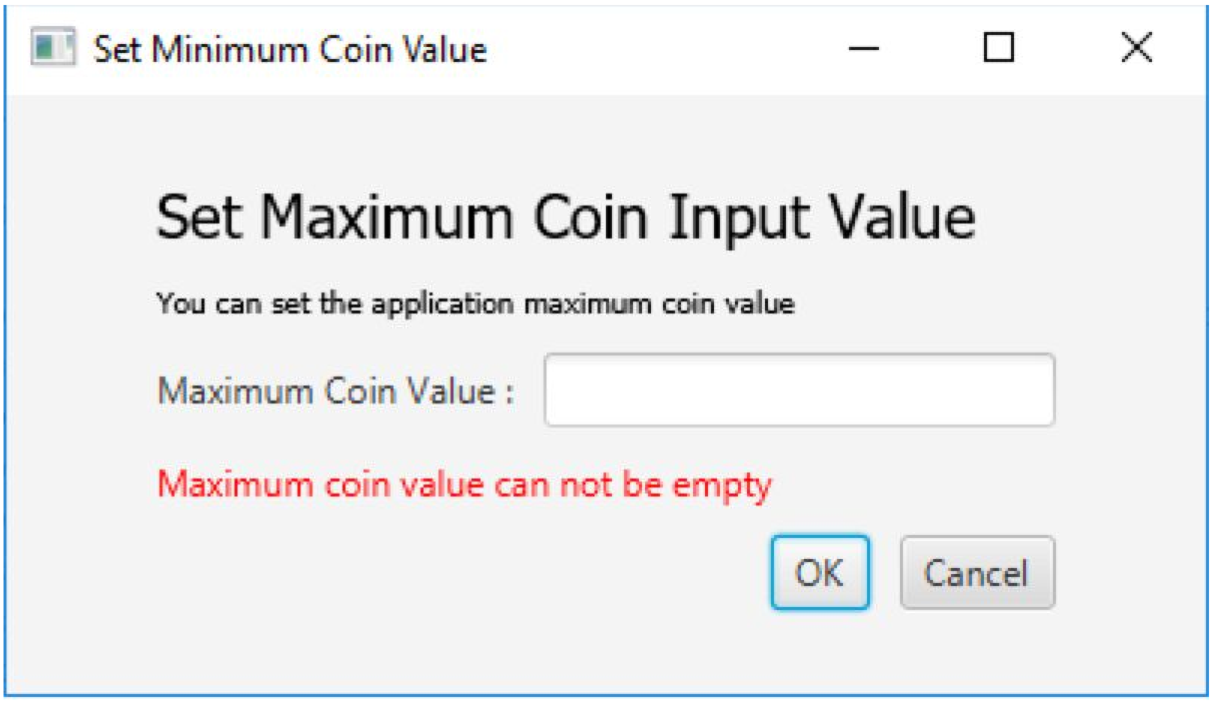
Set Maximum Coin Input Value

You can set the application maximum coin value

Maximum Coin Value : 10000

Fail Scenarios:

1- Max Value can not be empty:


Set Minimum Coin Value

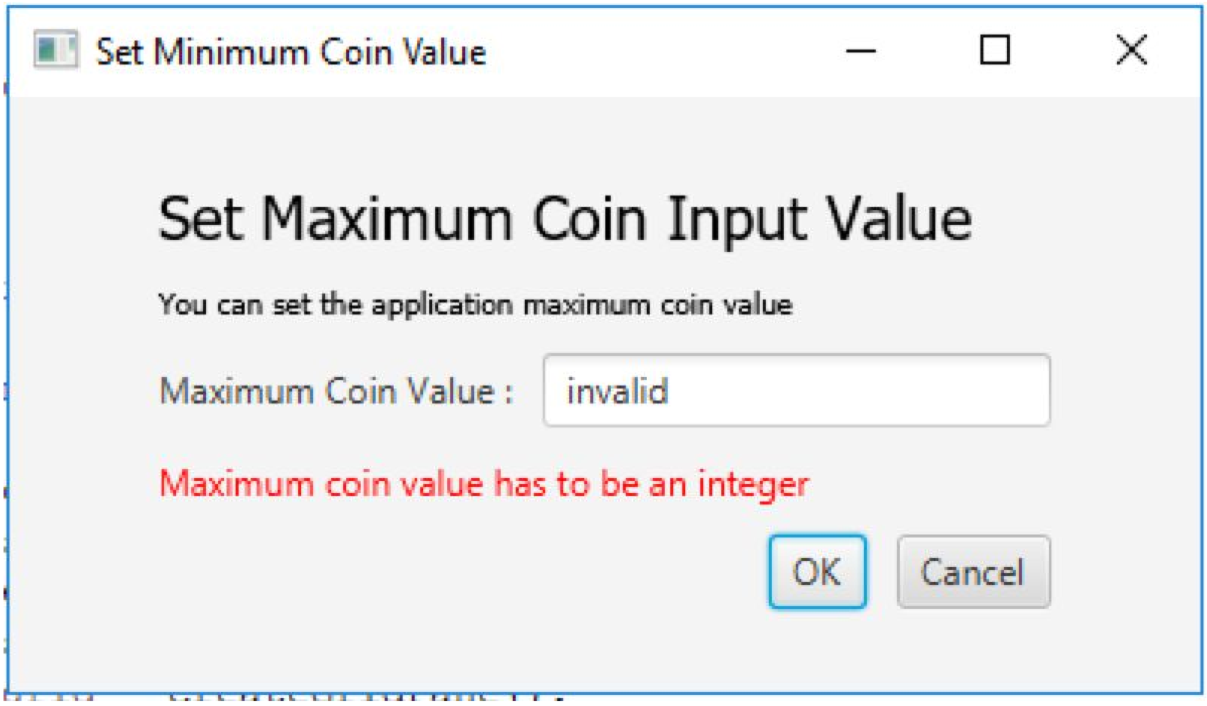
Set Maximum Coin Input Value

You can set the application maximum coin value

Maximum Coin Value :

Maximum coin value can not be empty

2- Max Value type is invalid:



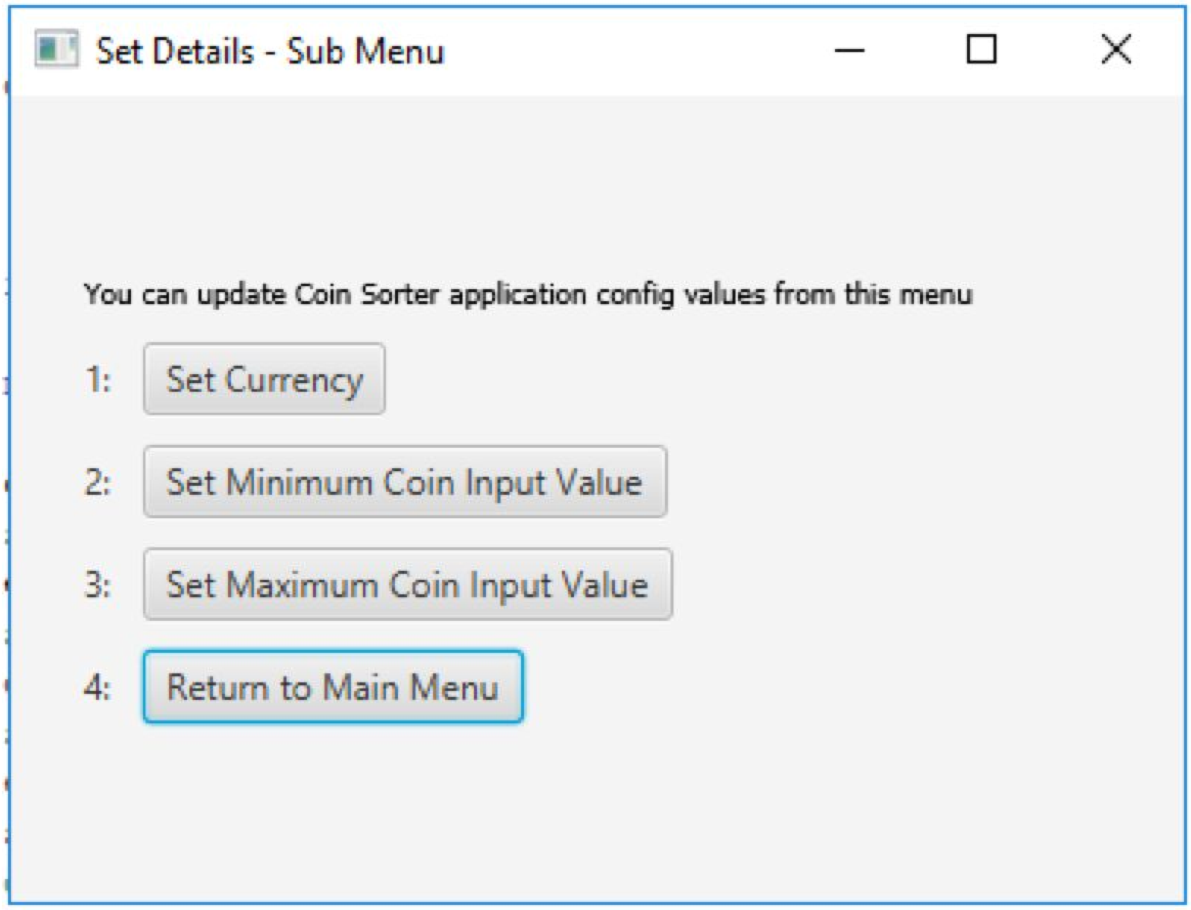
Set Maximum Coin Input Value

You can set the application maximum coin value

Maximum Coin Value : invalid

Maximum coin value has to be an integer

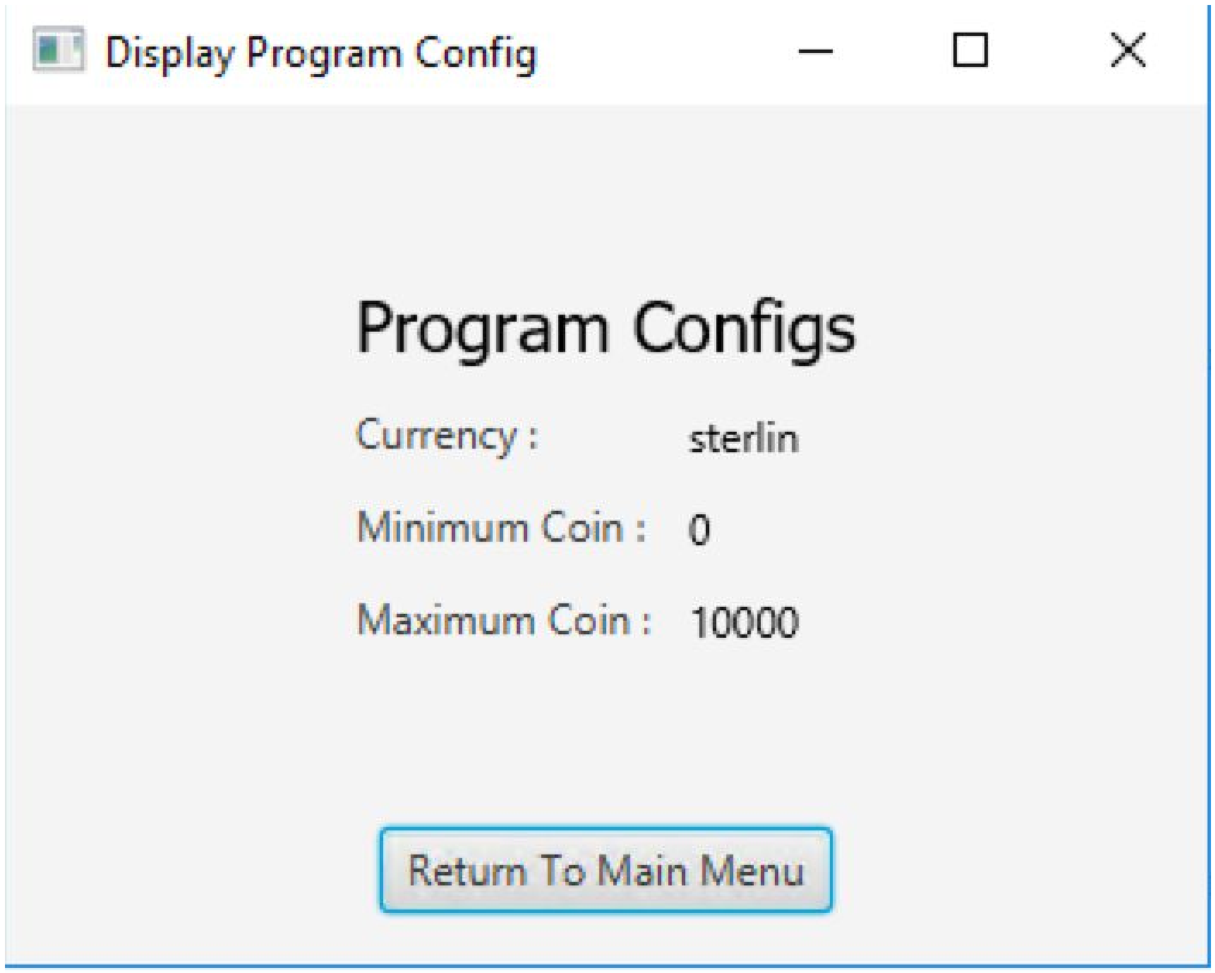
Command 4.4 Return Main Menu



You can update Coin Sorter application config values from this menu

1: Set Currency
2: Set Minimum Coin Input Value
3: Set Maximum Coin Input Value
4: Return to Main Menu

Command 5 Display Program Configs


Display Program Config

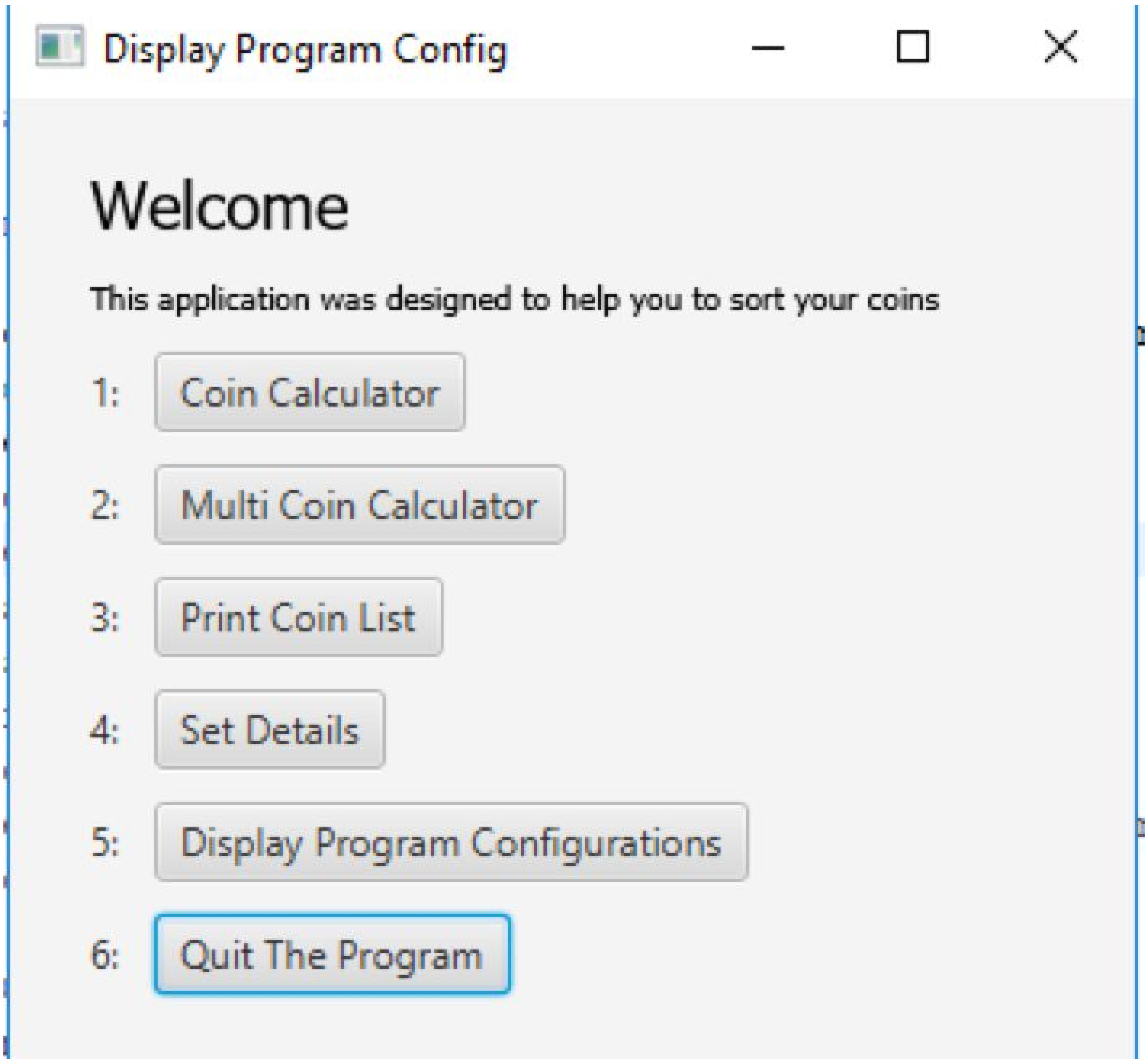
Program Configs

Currency : sterlin

Minimum Coin : 0

Maximum Coin : 10000

Command 6 Quit The Program


Display Program Config
Welcome
This application was designed to help you to sort your coins
1:
Coin Calculator
2:
Multi Coin Calculator
3:
Print Coin List
4:
Set Details
5:
Display Program Configurations
6:
Quit The Program