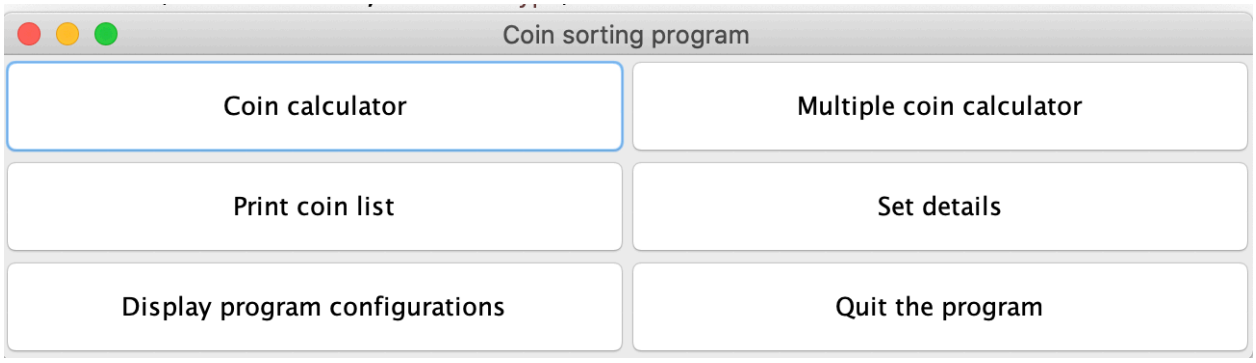
Coin Sorting Program

John Akhilomen

Evidence:

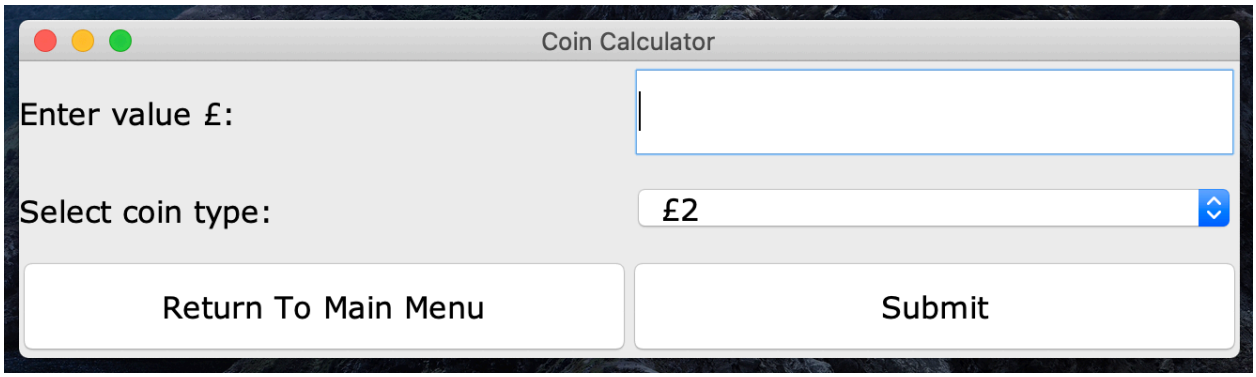
The application's main window shows the main menu; Coin calculator, Multiple coin calculator, Print coin list, Set details, Display program configuration and Quit the program. The title of the main window is set to the name of the application, Coin sorting program.

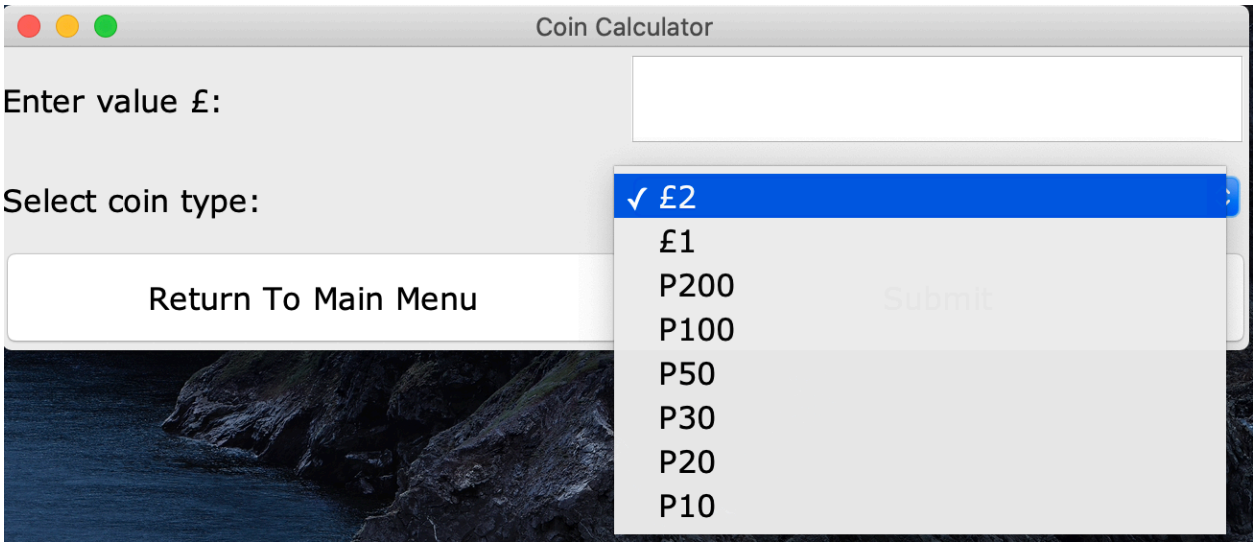


1. *Coin calculator menu:*

The user can exchange from Pound sterling to a particular coin type in circulation. Additionally, the user can exchange from pennies to another coin type in circulation The coin types in circulation in the scope of this program are:

- £2 (or 200p)
- £1 (or 100p)
- 50p
- 30p
- 20p
- 10p

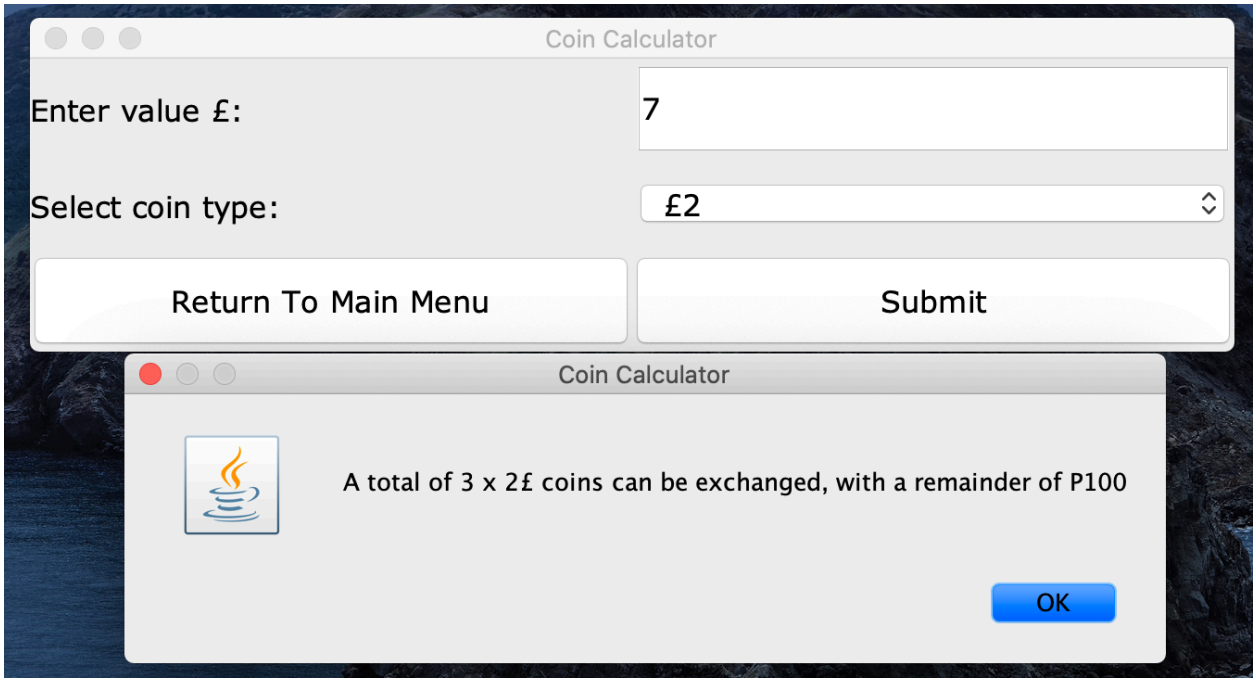




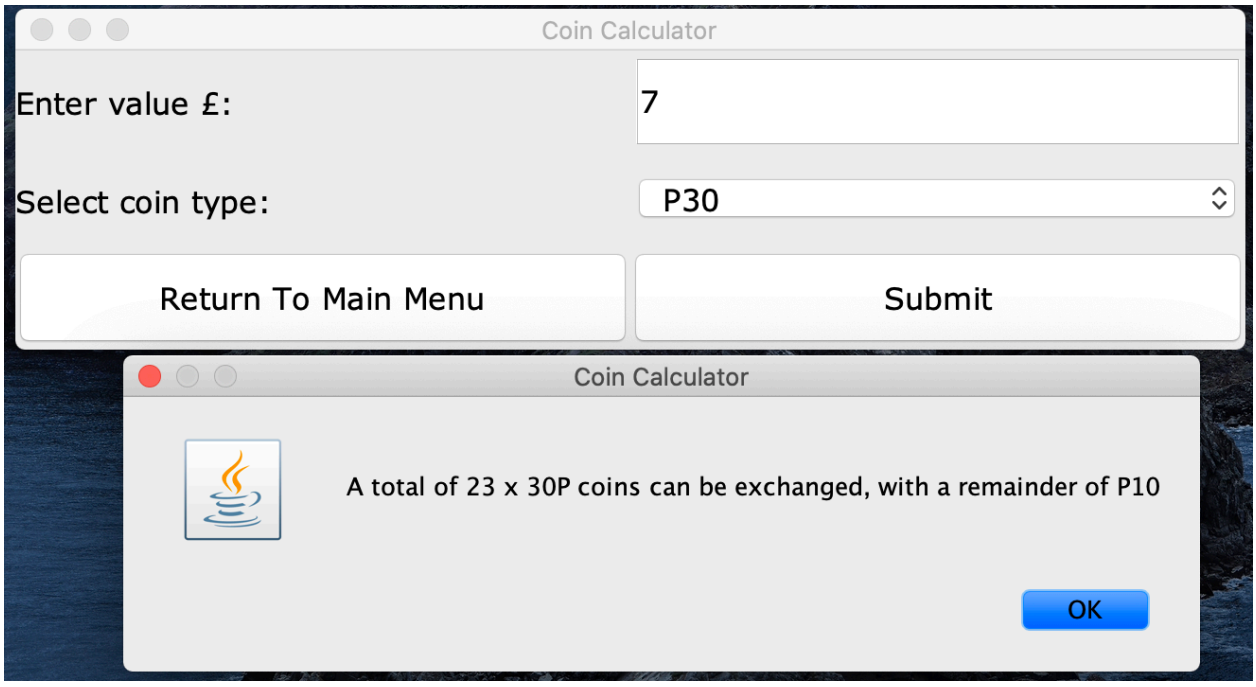
The text field “Enter value £“ accepts an input for total value for exchange. The combo-box displays the coin types in circulation for the user to select one.

Test Scenarios:

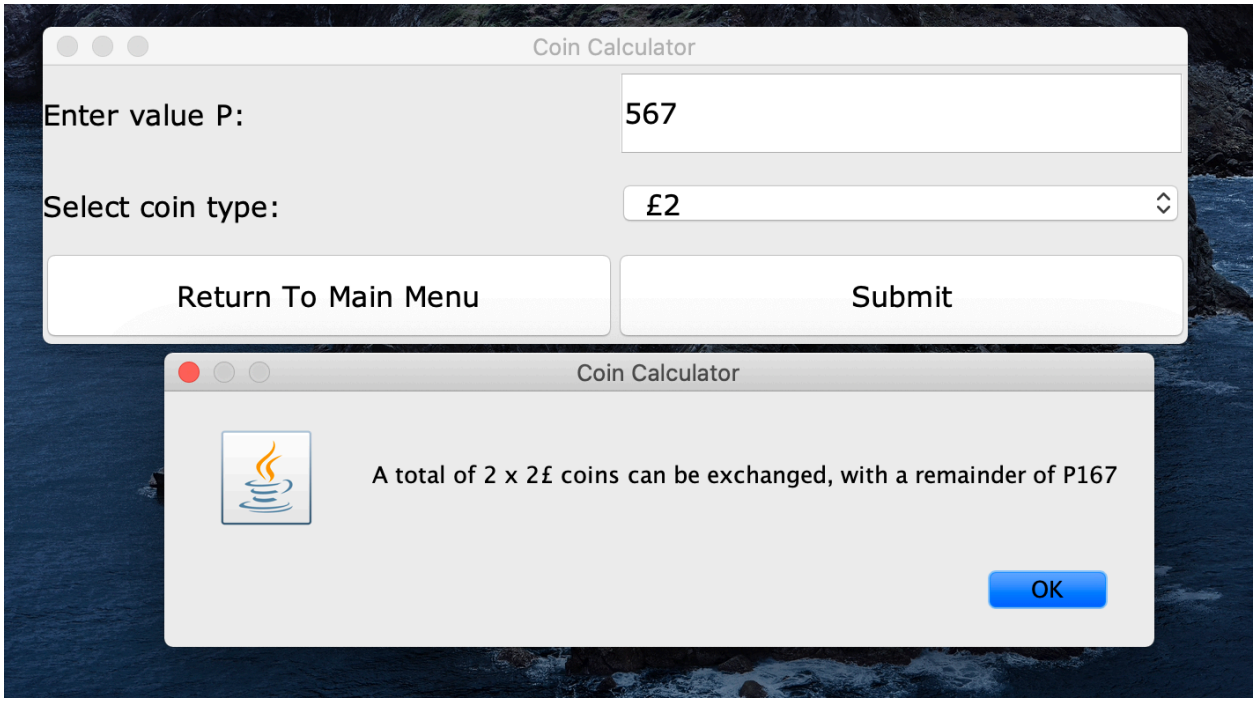
- The first test scenario makes an exchange of £7 to £2(a coin type in circulation). For £7, a total of 3 x £2 can be exchanged with a remainder of P100. The program displays the information using a message box.



- The second test scenario makes an exchange of £7 to P30 (a coin type in circulation). For £7, a total of 3 x £2 can be exchanged with a remainder of P100.



- The third test scenario makes an exchange of 567 pennies to £2. For P567, a total of 2 x £2 can be exchanged with a remainder of P167.

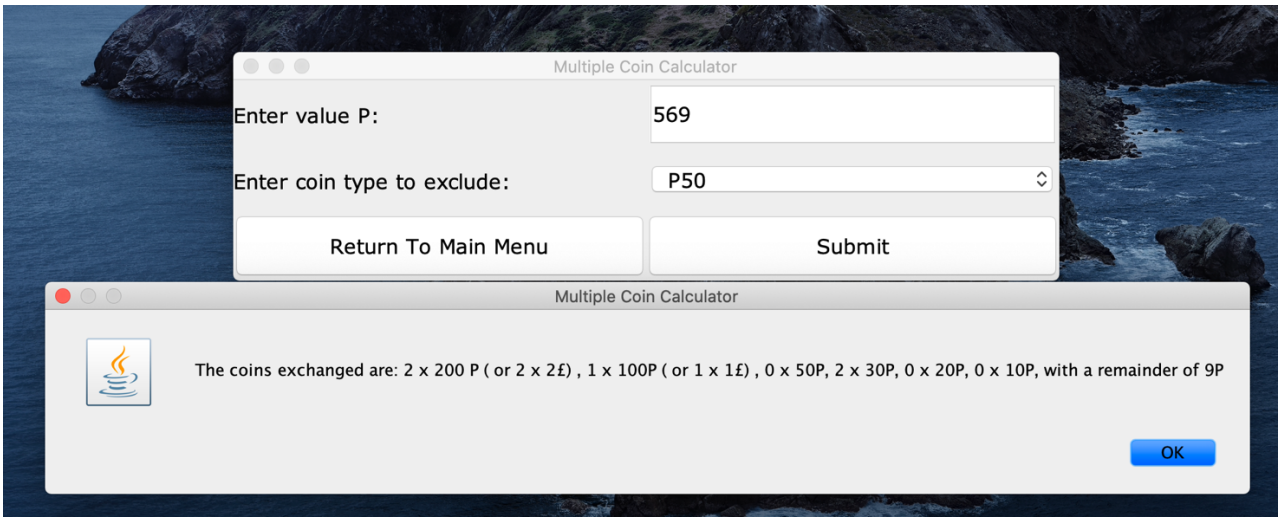


2. *Multiple coin calculator menu*

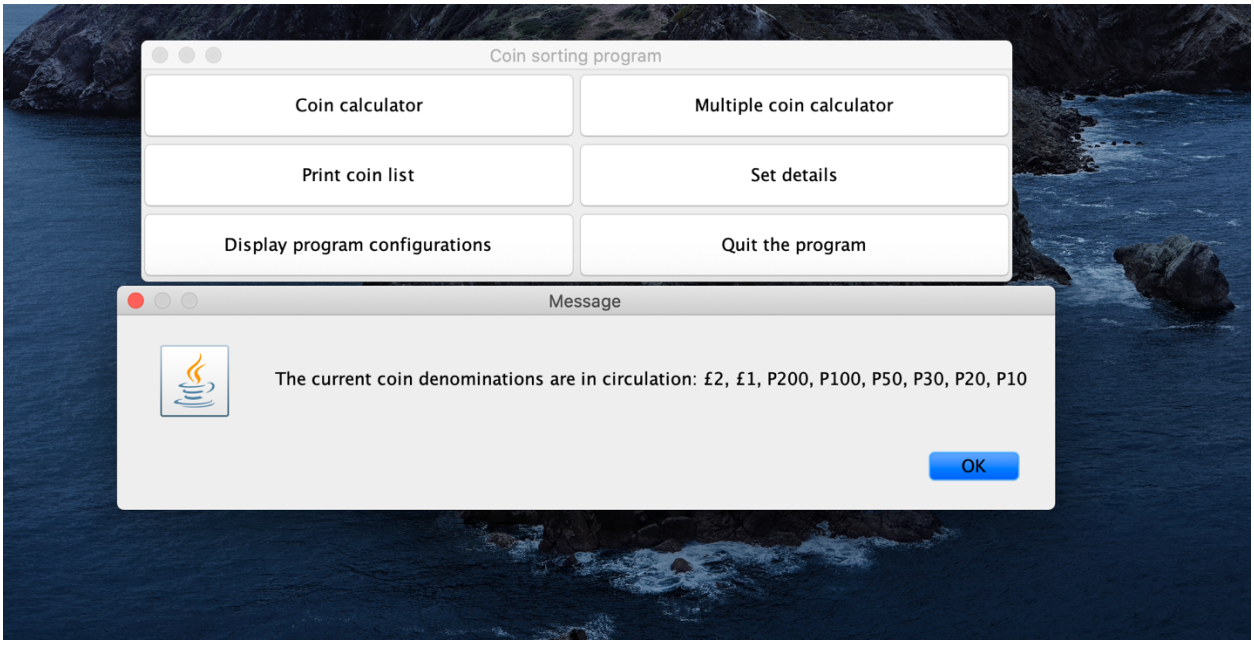
The user can exchange from pennies to a particular multiple coins in circulation and excluding a particular coin type.

Test scenarios:

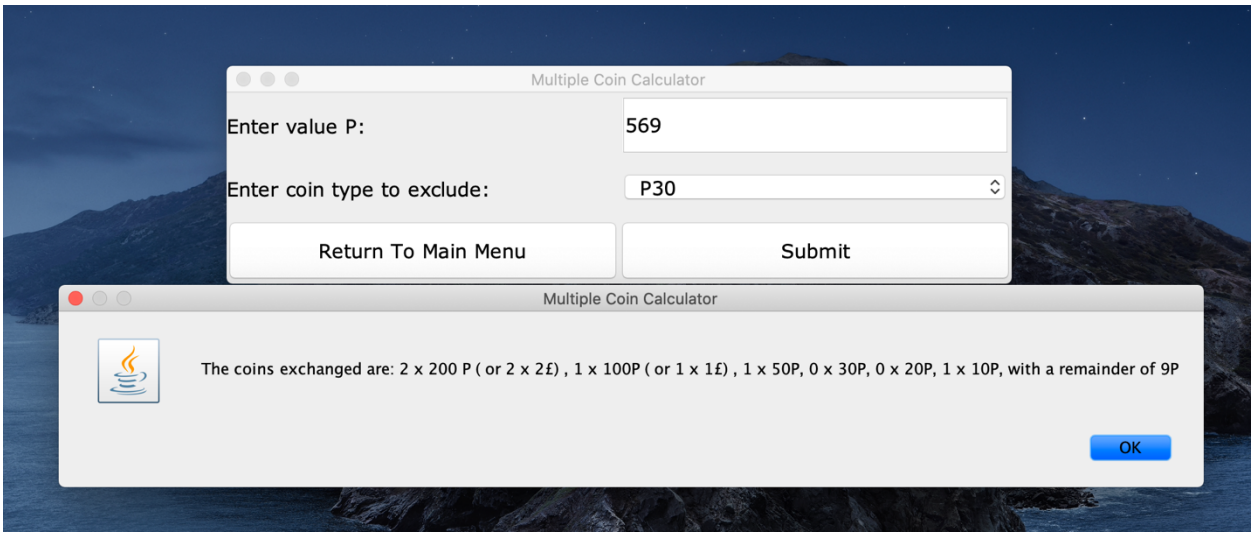
- The first scenario makes an exchange of 569 pennies to multiple coins and excluding P50.



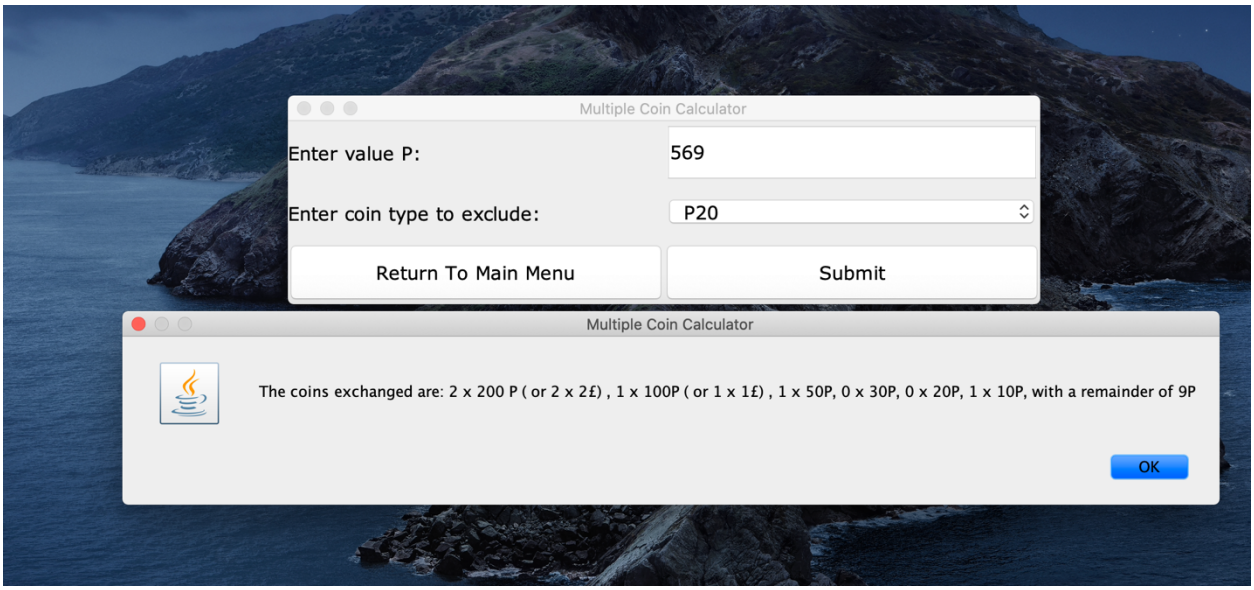
- The second scenario makes an exchange of 569 pennies to multiple coins and excluding P100.



- The third scenario makes an exchange of 569 pennies to multiple coins and excluding P30.

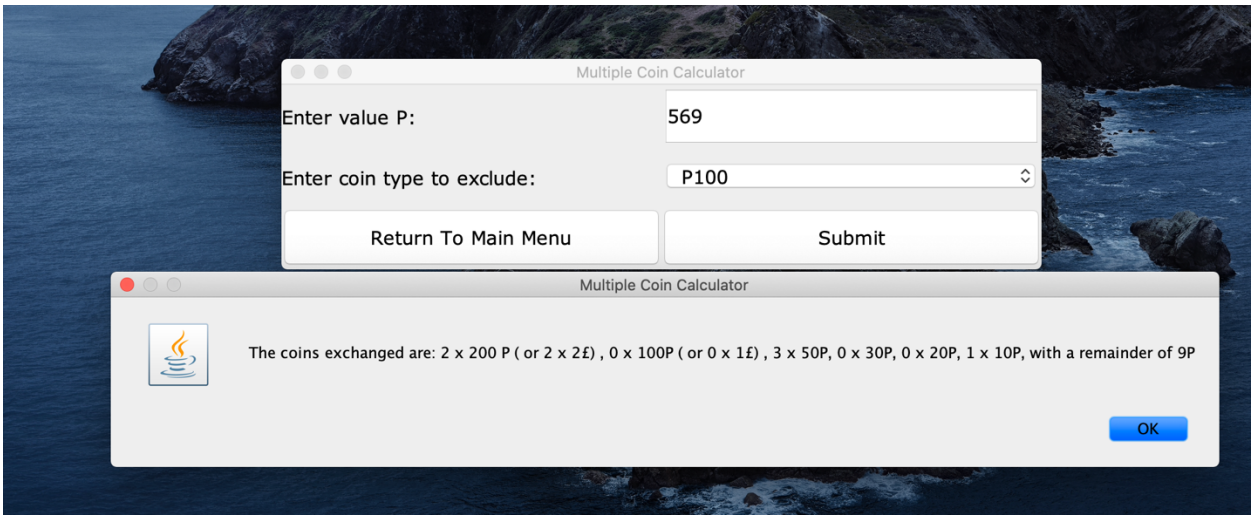


- The fourth scenario makes an exchange of 569 pennies to multiple coins and excluding P20.

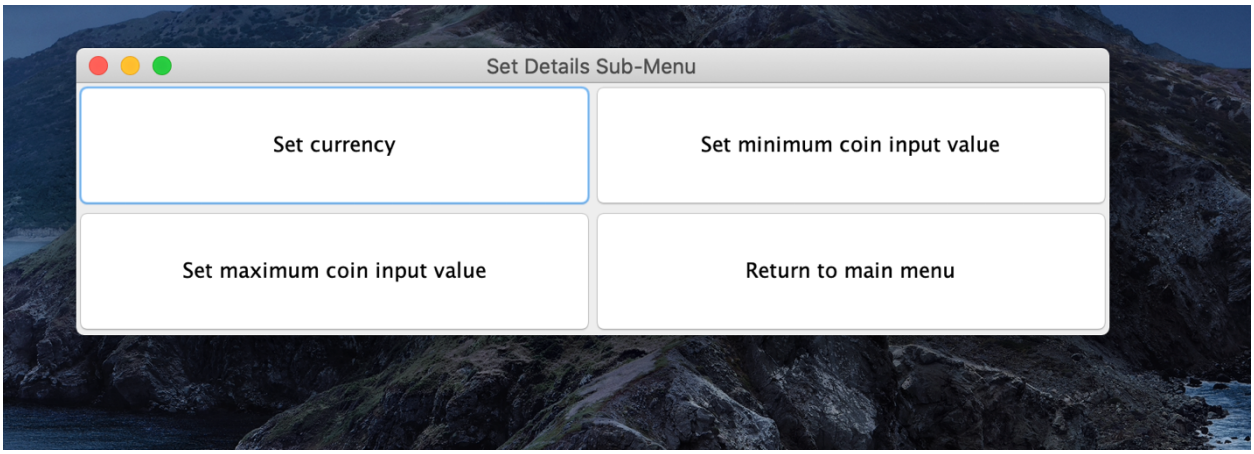


3. *Print Coin list menu*

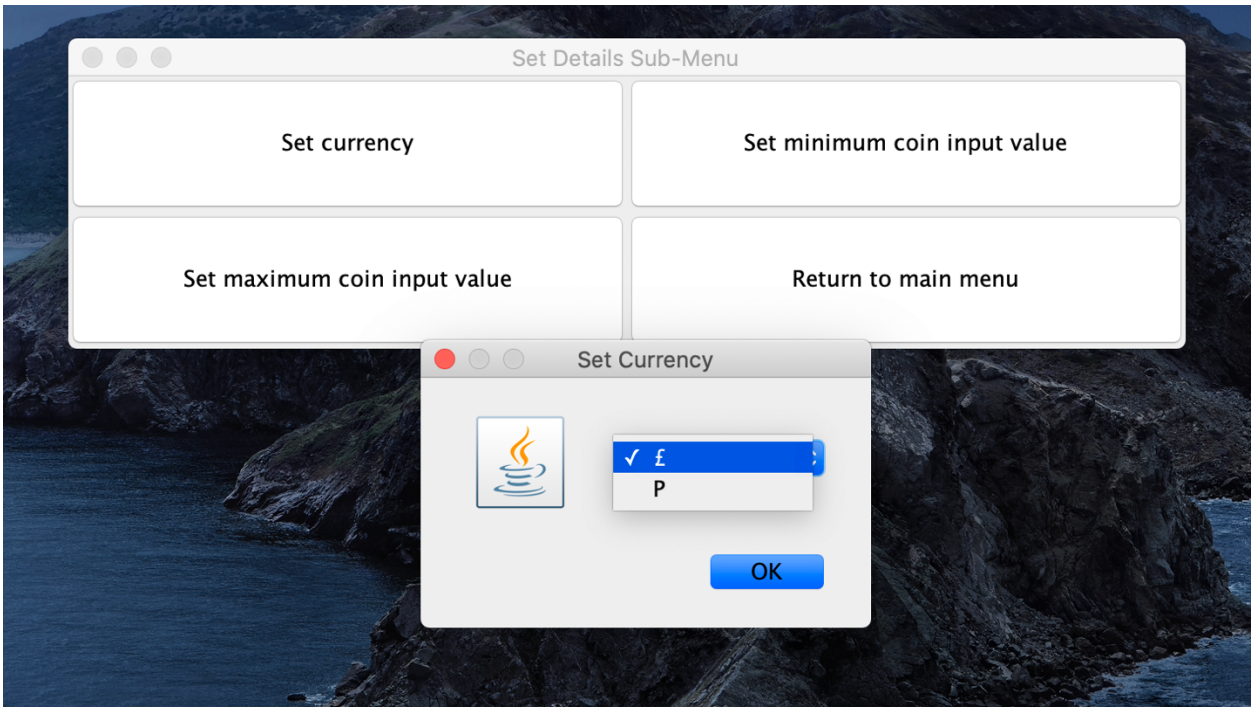
This will display the coin list in circulation



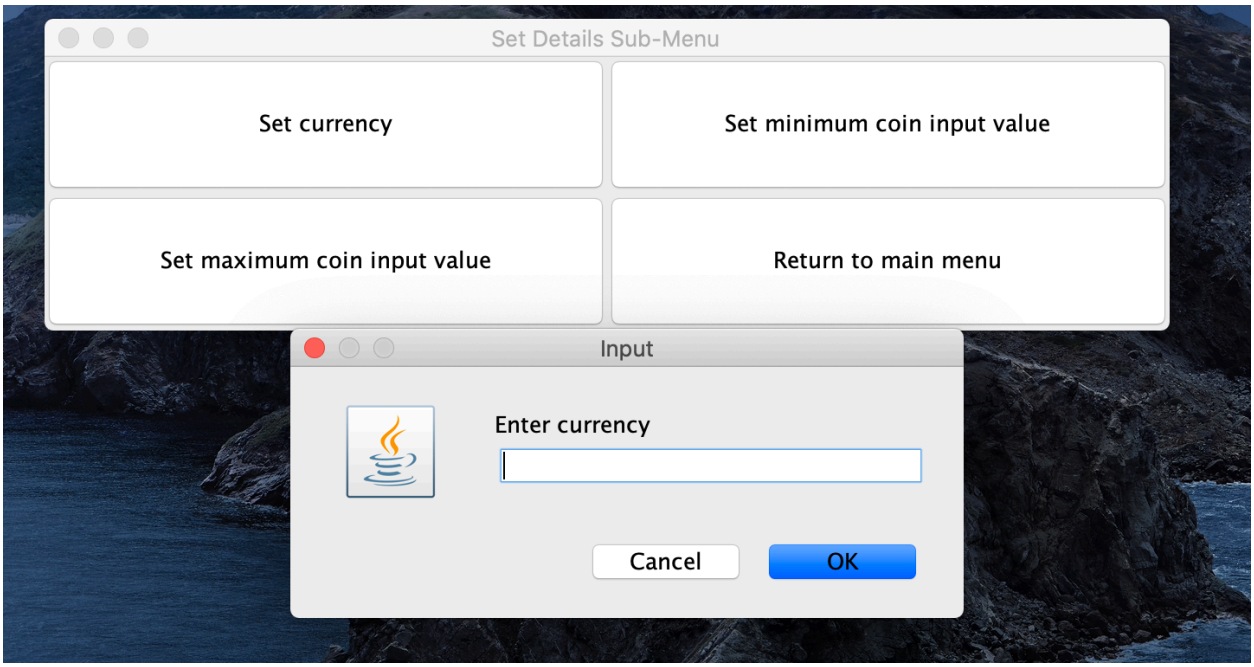
4. *Set Details Menu*

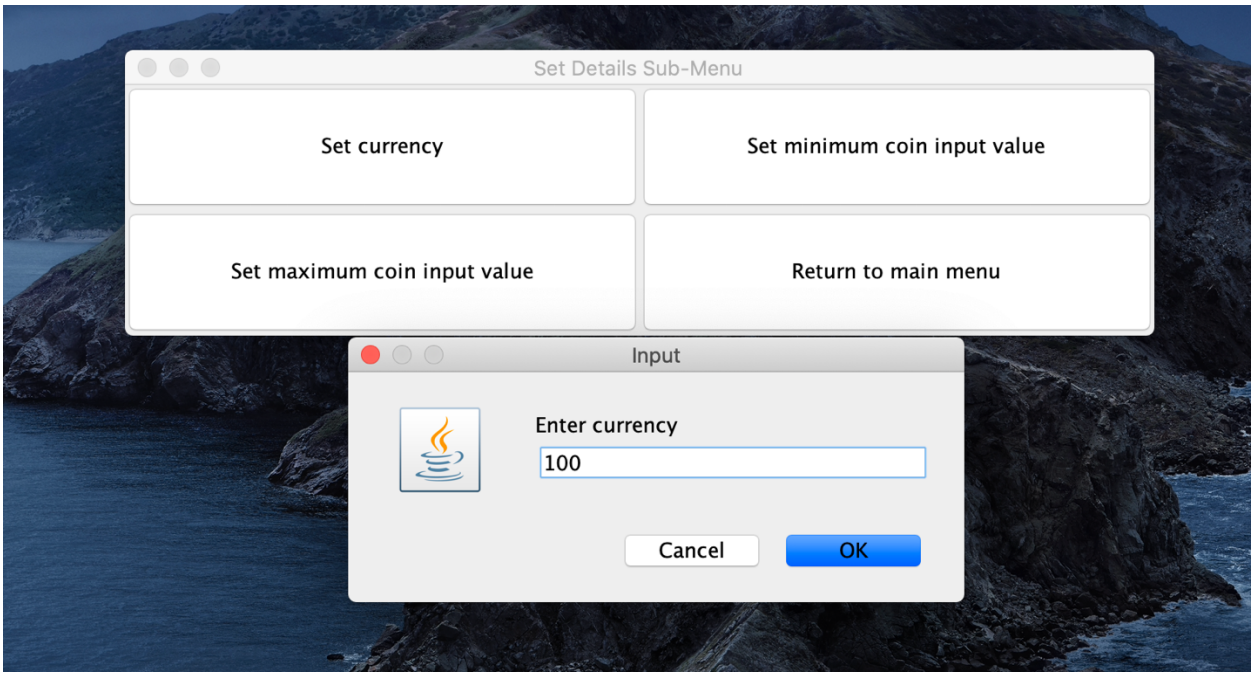


- Set currency:

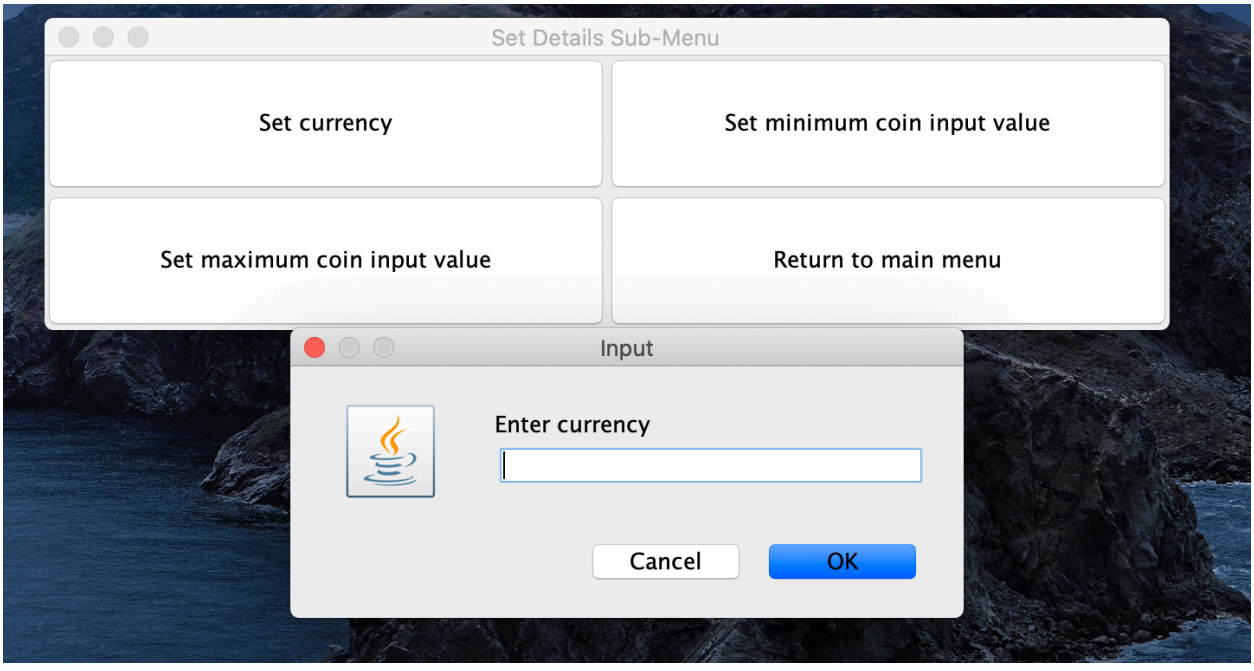


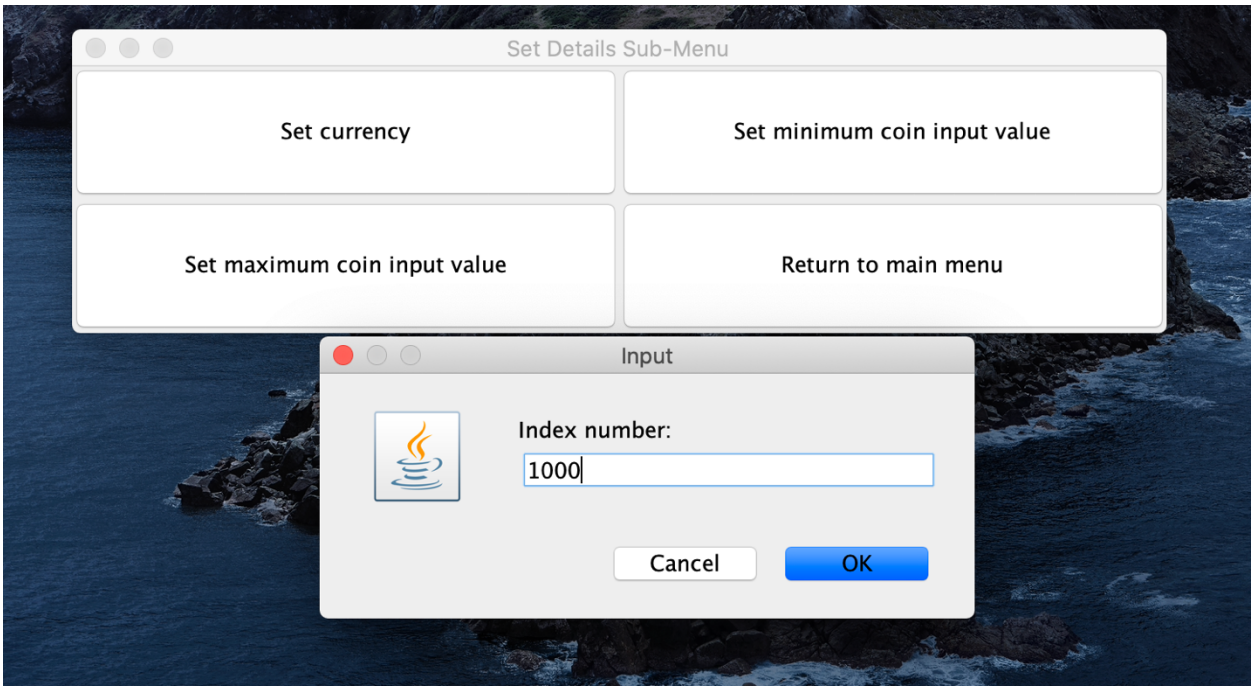
- Set minimum coin input value





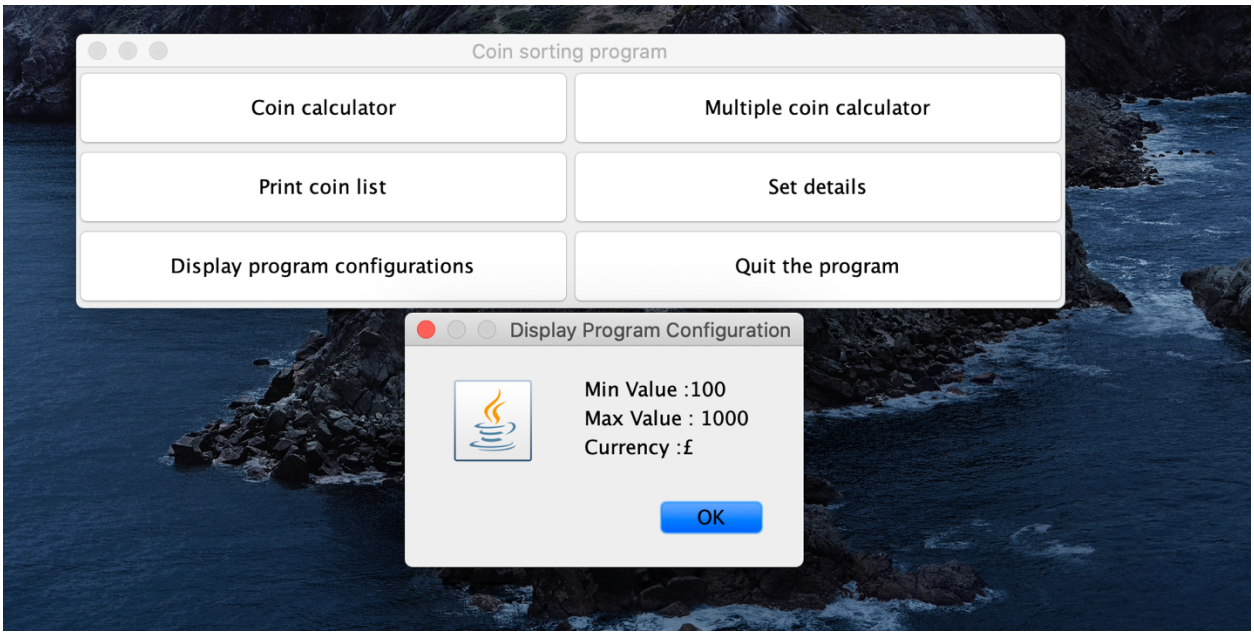
- Set maximum coin input value





5. *Display program configurations menu*

As displayed below are the configurations settings made for minimum input value (100), maximum input value (1000), and currency £.



6. *Quit the program will exit the window*

Source code:

CoinSorter.java:

```
package com.coinsort.johnakhilomen;

import java.util.ArrayList;
import java.util.Arrays;


public class CoinSorter {

	private String _currency;
	private int _minCoinIn;
	private int _maxCoinIn;
	private ArrayList<String> _coinList;
	public ArrayList<Integer> createMultiCoinList = new ArrayList<Integer>();

	public CoinSorter(String currency, int minCoinIn, int maxCoinIn,
ArrayList<String> coinList)
	{
		_currency = currency;
		_minCoinIn = minCoinIn;
		_maxCoinIn = maxCoinIn;
		_coinList = coinList;
	}

	public CoinSorter()
	{

	}

	//setters
	public void setCurrency(String currency)
	{
		_currency = currency;
	}
	public void setMinCoinIn(int minCoinIn)
```

```
	{
		_minCoinIn = minCoinIn;
	}
	public void setMaxCoinIn(int maxCoinIn)
	{
		_maxCoinIn = maxCoinIn;
	}
	public void setCoinList(ArrayList<String> coinList)
	{
		_coinList = coinList;
	}
	//getters
	public String getCurrency()
	{
		return _currency;
	}
	public int getMinCoinIn()
	{
		return _minCoinIn;
	}
	public int getMaxCoinIn()
	{
		return _maxCoinIn;
	}
	public ArrayList<String> getCoinList()
	{
		return _coinList;
	}

	public int ConvertFromPoundToPenny(int pound)
	{
		return pound * 100;
	}

	public String printCoinList()
	{
		return "The current coin denominations are in circulation: "+String.join(", ",
_coinList);
	}

	public String coinCalculator(int totalValue, int coinType)
```

```
	{
		if(getCurrency()=="£")
		{
			totalValue = ConvertFromPoundToPenny(totalValue);
			setCurrency("P");
		}
		int numberOfCoins = (int) Math.floor(totalValue / coinType);
		int numberOfCoinsRemainder = totalValue % coinType;
		if (getCurrency()=="P" && (coinType == 2 || coinType == 1))
		{
			numberOfCoins = (int) Math.floor(totalValue / (coinType*100));
			numberOfCoinsRemainder = totalValue % (coinType*100);
			setCurrency("£");
		}
		String msg = "A total of "+numberOfCoins +" x " +coinType+
getCurrency()+" coins can be exchanged";
		return numberOfCoinsRemainder == 0 ? msg : msg+", with a remainder of
"+"P"+numberOfCoinsRemainder;
	}

	public String multiCoinCalculator(int totalValue, int coinTypeToExclude)
	{
		int p200 = 0;
		int p100 = 0;
		int p50 = 0;
		int p30 = 0;
		int p20 = 0;
		int p10 = 0;

		ArrayList<Integer> newList = new ArrayList<Integer>();
		for (int i = 0; i < _coinList.size(); i++)
		{
			int coinT = Integer.parseInt(_coinList.get(i).substring(1));
			newList.add(coinT);
		}

		for (int i = 0; i < newList.size(); i++)
		{
			if(newList.get(i) == coinTypeToExclude)
			{
				newList.remove(i);
```

```
                break;
        }
        else
        {
                continue;
        }
}
if(getCurrency()=="£")
{
        totalValue = ConvertFromPoundToPenny(totalValue);
        setCurrency("P");
}


processMultiCoinList(totalValue, newList);
var yy = createMultiCoinList;
for (int i = 0; i < yy.size(); i++)
{
        System.out.println(i);
}
for (int i = 0; i < yy.size(); i++)
{
        if(yy.get(i)==200)
    {
      p200++;
    }
    else if(yy.get(i)==100)
    {
      p100++;
    }
    else if(yy.get(i)==50)
    {
      p50++;
    }
    else if(yy.get(i)==30)
    {
      p30++;
    }
    else if(yy.get(i)==20)
    {
      p20++;
```

```
                }
                else if(yy.get(i)==10)
                {
                  p10++;
                }
            }

            String msg = "The coins exchanged are: "+p200+" x 200
"+getCurrency()+" ( or "+p200+" x 2£) , "+p100+" x 100"+getCurrency()+
                                " ( or "+p100+" x 1£) , "+p50+" x 50"+getCurrency()+",
"+p30+" x 30"+getCurrency()+", "
                                +p20+" x 20"+getCurrency()+", "+p10+" x 10"+getCurrency();
            return remainderValue == 0 ? msg : msg+", with a remainder of
"+remainderValue+"P";

        }

        public String displayProgramConfigs()
        {
            return "";
        }

        public int remainderValue = 0;
        public ArrayList<Integer> processMultiCoinList(int n, ArrayList<Integer> arr)
        {
            if(n == 0 || arr.size() == 0)
            {
               return arr;
            }
            else
            {
              if(n >= arr.get(0))
              {
              int t = n - arr.get(0);
              createMultiCoinList.add(arr.get(0));
              return processMultiCoinList(t,arr);
              }

            else
            {
             if(n < 10)
```

```
                {
                        remainderValue = n;
                        return arr;
                }
                arr.remove(0);
                return processMultiCoinList(n,arr);

                }

                }

        }

}
```

DisplayProgramConfigurationsPanel.java

```
package com.coinsort.johnakhilomen;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class DisplayProgramConfigurationsPanel extends JPanel{

        private JComboBox currencyCombo;
        private String[] _buttonNames;
        public DisplayProgramConfigurationsPanel(String[] buttonNames)
        {
                _buttonNames = buttonNames;
                setLayout(new GridLayout(2,2));
                setupButtons();
```

```
	}

	private void setLayout()
	{
		setLayout(new GridLayout(2,2));
	}

	public void setupButtons()
	{
		for (int i = 0; i< _buttonNames.length; i++)
		{
			JButton button = new JButton(_buttonNames[i]);
			button.addActionListener(new ActionListener(){
				@Override
				public void actionPerformed(ActionEvent e) {

					if(e.getActionCommand() == "Set currency")
					{
						setCurrency();

					}
					else if(e.getActionCommand() == "Set minimum coin
input value")
					{
						setMinCoinInput();
					}
					else if(e.getActionCommand() == "Set maximum coin
input value")
					{
						setMaxCoinInput();
					}
					else if(e.getActionCommand() == "Return to main
menu")
					{
						returnToMainMenu();
						System.out.println("Return to main menu");
					}

				}
			});
			add(button);
```

```
		}

	}

	protected void setMaxCoinInput() {
		String input= JOptionPane.showInputDialog("Index number: ");
		TestCoinSorter.coinSorter.setMaxCoinIn(Integer.parseInt(input));



	}

	protected void setMinCoinInput() {
		String minInput = JOptionPane.showInputDialog("Enter currency");
		TestCoinSorter.coinSorter.setMinCoinIn(Integer.parseInt(minInput));

	}

	protected void setCurrency() {
		String currencies[] = {"£", "P"};
		currencyCombo = new JComboBox(currencies);
		currencyCombo.addItemListener(new ItemListener(){

			@Override
			public void itemStateChanged(ItemEvent e) {
				if (e.getSource() == currencyCombo) {


TestCoinSorter.coinSorter.setCurrency(currencyCombo.getSelectedItem().toString());

				}

			}

			});
		JOptionPane.showMessageDialog(null, currencyCombo, "Set Currency",
JOptionPane.INFORMATION_MESSAGE);

	}

	protected void returnToMainMenu() {
		MainPanel.subMenudialog.setVisible(false);
```

```
                TestCoinSorter.mainDialog.setVisible(true);
        }

}
```

FrameWindow.java:

```
package com.coinsort.johnakhilomen;

import java.awt.GridLayout;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class FrameWindow extends JFrame
{
        public FrameWindow(String title, JPanel panel)
        {
                setSize(650, 180);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setTitle(title);
                this.add(panel);
        }

}
```

InputDenominationCombo:

```
package com.coinsort.johnakhilomen;

import javax.swing.JComboBox;

public class InputDenominationCombo{
```

```
	public JComboBox getJComoBox(String[] denominations) {
		JComboBox inputDenomination = new JComboBox(denominations);
		inputDenomination.setFont(MainPanel.font);
		return inputDenomination;
	}

}
```

MainPanel.java

```
package com.coinsort.johnakhilomen;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Arrays;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class MainPanel extends JPanel{
	public static Font font = new Font("Verdana", Font.PLAIN, 16);
	private static FrameWindow coinCalcFrame, multcoinCalcFrame;
	private String[] _buttonNames;
	/*public static JLabel _currencyLabel = new JLabel();
	public static JLabel _minInputLabel = new JLabel();
	public static JLabel _maxInputLabel = new JLabel();*/
	public static JFrame subMenudialog;
	public MainPanel(String[] buttonNames)
	{
		_buttonNames = buttonNames;
		setLayout(new GridLayout(3,3));
```

```
		//setupLabels();
		setupButtons();
	}

	/*private void setupLabels()
	{
		_currencyLabel.setText("Currecncy:
"+TestCoinSorter.coinSorter.getCurrency());
		_currencyLabel.setForeground(Color.red);
		_minInputLabel.setText("Min Input:
"+TestCoinSorter.coinSorter.getMinCoinIn());
		_minInputLabel.setForeground(Color.red);
		_maxInputLabel.setText("Max Input:
"+TestCoinSorter.coinSorter.getMaxCoinIn());
		_maxInputLabel.setForeground(Color.red);
		add(_currencyLabel);
		add(_minInputLabel);
		add(_maxInputLabel);
	}*/

	public void setupButtons()
	{

		for (int i = 0; i< _buttonNames.length; i++)
		{
			JButton button = new JButton(_buttonNames[i]);
			button.addActionListener(new ActionListener(){
				@Override
				public void actionPerformed(ActionEvent e) {
					if(e.getActionCommand() == "Coin calculator")
					{
						System.out.println("Coin calculator...");
						coinCalculator();

					}
					else if(e.getActionCommand() == "Multiple coin
calculator")
					{
						System.out.println("Multiple coin
calculators...");
						multipleCoinCalculator();
```

```
                                        }
                                        else if(e.getActionCommand() == "Print coin list")
                                        {
                                                System.out.println("Print coin lists...");
                                                printCoinList();

                                        }
                                        else if(e.getActionCommand() == "Set details")
                                        {
                                                System.out.println("Set details");
                                                setDetails();
                                        }
                                        else if(e.getActionCommand() == "Display program
configurations")
                                        {
                                                displayProgramConfigurations();
                                        }
                                        else if(e.getActionCommand() == "Quit the program")
                                        {
                                                System.out.println("Quit the program");
                                                quitTheProgram();
                                        }
                                }
                        });
                        add(button);
                }

        }

        protected void quitTheProgram() {
        System.exit(0);
        }

        protected void displayProgramConfigurations() {
                String info = "Min Value :" + TestCoinSorter.coinSorter.getMinCoinIn()
+"\n"
                                        +"Max Value : "+
TestCoinSorter.coinSorter.getMaxCoinIn()+"\n"
                                        +"Currency :" + TestCoinSorter.coinSorter.getCurrency();
```

```
                JOptionPane.showMessageDialog(null, info, "Display Program
Configuration", JOptionPane.INFORMATION_MESSAGE);
        }

        protected void setDetails() {
                String[] subMenuButtonNames = new String[] {"Set currency", "Set
minimum coin input value", "Set maximum coin input value",
                "Return to main menu"};
                subMenudialog = new FrameWindow("Set Details Sub-Menu", new
DisplayProgramConfigurationsPanel(subMenuButtonNames));
                TestCoinSorter.mainDialog.setVisible(false);
                subMenudialog.setVisible(true);
        }

        protected void printCoinList() {
                JOptionPane.showMessageDialog(null,
TestCoinSorter.coinSorter.printCoinList());
        }

        protected void multipleCoinCalculator() {

                var multcoinCalcPanel = new JPanel();
                multcoinCalcPanel.setLayout(new GridLayout(3,3));
                JLabel multcoinValue = new JLabel("Enter value
"+TestCoinSorter.coinSorter.getCurrency()+": ");
                multcoinValue.setFont(font);
                JTextField multcoinValueTextField = new JTextField();
                multcoinValueTextField.setFont(font);
                JLabel multcoinType = new JLabel("Enter coin type to exclude: ");
                multcoinType.setFont(font);

                var inputDenomination = new
InputDenominationCombo().getJComoBox(TestCoinSorter.inputDenominationsMultiplec
oins);
                JButton submit = new JButton("Submit");
                submit.setFont(font);
                submit.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {

        //System.out.println(testCoinSorter.coinSorter.getCurrency());
```

```
                                if (
!validateInputValueRange(Integer.parseInt(multcoinValueTextField.getText())))
                                {
                                        JOptionPane.showMessageDialog(null, "Invalid input
- Value must be in the range of Min and Max values set!");
                                        return;
                                }
                                else if(!validateInput(multcoinValueTextField.getText()))
                                {
                                        JOptionPane.showMessageDialog(null, "Invalid input
for Coin Value - Only integers are allowed!");
                                        return;
                                }
                                else if (TestCoinSorter.coinSorter.getCurrency() == null)
                                {
                                        JOptionPane.showMessageDialog(null, "Please set
the currency in \n Display Program Configuration menu!");
                                        return;
                                }
                                else
                                {
                                        var inputDenominationStr =
inputDenomination.getSelectedItem().toString().substring(1);
                                        TestCoinSorter.coinList = new
ArrayList<String>(Arrays.asList(TestCoinSorter.inputDenominationsMultiplecoins));
                                        var coinSorter = new
CoinSorter(TestCoinSorter.coinSorter.getCurrency(), TestCoinSorter.minValue,
TestCoinSorter.maxValue, TestCoinSorter.coinList);
                                        String result =
coinSorter.multiCoinCalculator(Integer.parseInt(multcoinValueTextField.getText()),
Integer.parseInt(inputDenominationStr) );
                                        JOptionPane.showMessageDialog(null, result,
"Multiple Coin Calculator", JOptionPane.INFORMATION_MESSAGE);
                                }

                        }
                });

                JButton returnToMainMenu = new JButton("Return To Main Menu");
                returnToMainMenu.setFont(font);
                returnToMainMenu.addActionListener(new ActionListener(){
                        @Override
```

```
                public void actionPerformed(ActionEvent e) {
                        multcoinCalcFrame.setVisible(false);
                        TestCoinSorter.mainDialog.setVisible(true);
                }
        });

        multcoinCalcPanel.add(multcoinValue);
        multcoinCalcPanel.add(multcoinValueTextField);
        multcoinCalcPanel.add(multcoinType);
        multcoinCalcPanel.add(inputDenomination);
        multcoinCalcPanel.add(returnToMainMenu);
        multcoinCalcPanel.add(submit);

        multcoinCalcFrame = new FrameWindow("Multiple Coin Calculator",
multcoinCalcPanel);
        multcoinCalcFrame.setVisible(true);
        TestCoinSorter.mainDialog.setVisible(false);
        multcoinCalcFrame.setLocationRelativeTo(TestCoinSorter.mainDialog);


    }

    protected void coinCalculator() {
        var coinCalcPanel = new JPanel();
        coinCalcPanel.setLayout(new GridLayout(3,3));
        JLabel coinValue = new JLabel("Enter value
"+TestCoinSorter.coinSorter.getCurrency()+": ");
        coinValue.setFont(font);
        JTextField coinValueTextField = new JTextField();
        coinValueTextField.setFont(font);
        JLabel coinType = new JLabel("Select coin type: ");
        coinType.setFont(font);
        var inputDenomination = new
InputDenominationCombo().getJComoBox(TestCoinSorter.inputDenominations);
        JButton submit = new JButton("Submit");
        submit.setFont(font);
        submit.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                        System.out.println(TestCoinSorter.coinSorter.getCurrency());
```

```
                                if (
!validateInputValueRange(Integer.parseInt(coinValueTextField.getText())))
                                {
                                        JOptionPane.showMessageDialog(null, "Invalid input
- Value must be in the range of Min and Max values set!");
                                        return;
                                }
                                else if(!validateInput(coinValueTextField.getText()))
                                {
                                        JOptionPane.showMessageDialog(null, "Invalid input
for Coin Value - Only integers are allowed!");
                                        return;
                                }
                                else if (TestCoinSorter.coinSorter.getCurrency() == null)
                                {
                                        JOptionPane.showMessageDialog(null, "Please set
the currency in \n Display Program Configuration menu!");
                                        return;
                                }
                                else
                                {
                                        var inputDenominationStr =
inputDenomination.getSelectedItem().toString().substring(1);
                                        var coinSorter = new
CoinSorter(TestCoinSorter.coinSorter.getCurrency(), TestCoinSorter.minValue,
TestCoinSorter.maxValue, TestCoinSorter.coinList);
                                        String result =
coinSorter.coinCalculator(Integer.parseInt(coinValueTextField.getText()),
Integer.parseInt(inputDenominationStr) );
                                        JOptionPane.showMessageDialog(null, result, "Coin
Calculator", JOptionPane.INFORMATION_MESSAGE);

                                }

                        }
                });

                JButton returnToMainMenu = new JButton("Return To Main Menu");
                returnToMainMenu.setFont(font);
                returnToMainMenu.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
```

```
                        coinCalcFrame.setVisible(false);
                        TestCoinSorter.mainDialog.setVisible(true);
                    }
                });

                coinCalcPanel.add(coinValue);
                coinCalcPanel.add(coinValueTextField);
                coinCalcPanel.add(coinType);
                coinCalcPanel.add(inputDenomination);
                coinCalcPanel.add(returnToMainMenu);
                coinCalcPanel.add(submit);

                coinCalcFrame = new FrameWindow("Coin Calculator", coinCalcPanel);
                coinCalcFrame.setVisible(true);
                TestCoinSorter.mainDialog.setVisible(false);
                coinCalcFrame.setLocationRelativeTo(TestCoinSorter.mainDialog);

        }

        private boolean validateInput(String input)
        {
                try {
            Integer num = Integer.valueOf(input);
            return true;
           } catch (NumberFormatException e) {
            return false;
           }
        }

        private boolean validateInputValueRange(int value)
        {
                if ( !(value >= TestCoinSorter.coinSorter.getMinCoinIn() && (value <=
TestCoinSorter.coinSorter.getMaxCoinIn()) ) )
                {
                        return false;
                }
                return true;
        }
}
```

TestCoinSorter.java

```
package com.coinsort.johnakhilomen;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Arrays;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;


public class TestCoinSorter {

	public static MainPanel mainPanel;
	public static JFrame mainDialog;
	public static CoinSorter coinSorter;
	public static int minValue = 0;
	public static int maxValue = 10000;
	public static ArrayList<String> coinList;
	public static  String[] inputDenominationsMultiplecoins = new String[]
{"P200","P100", "P50", "P30","P20","P10"};
	public static  String[] inputDenominations = new String[] {"£2", "£1",
"P200","P100", "P50", "P30","P20","P10"};
	public static void main(String[] args) {
		String[] buttonNames = new String[] {"Coin calculator", "Multiple coin
calculator", "Print coin list", "Set details",
				"Display program configurations", "Quit the program"};
		coinList = new ArrayList<String>(Arrays.asList(inputDenominations));
		coinSorter = new CoinSorter("£", 0, 10000, coinList);
		mainPanel = new MainPanel(buttonNames);
		mainDialog = new FrameWindow("Coin sorting program", mainPanel);
		mainDialog.setVisible(true);

	}

}
```