Java Project - Currency converter program:

Program goal:

Program will convert currency between USD (United States Dollars) and ILS (Israeli Shekel).

Solution architecture:

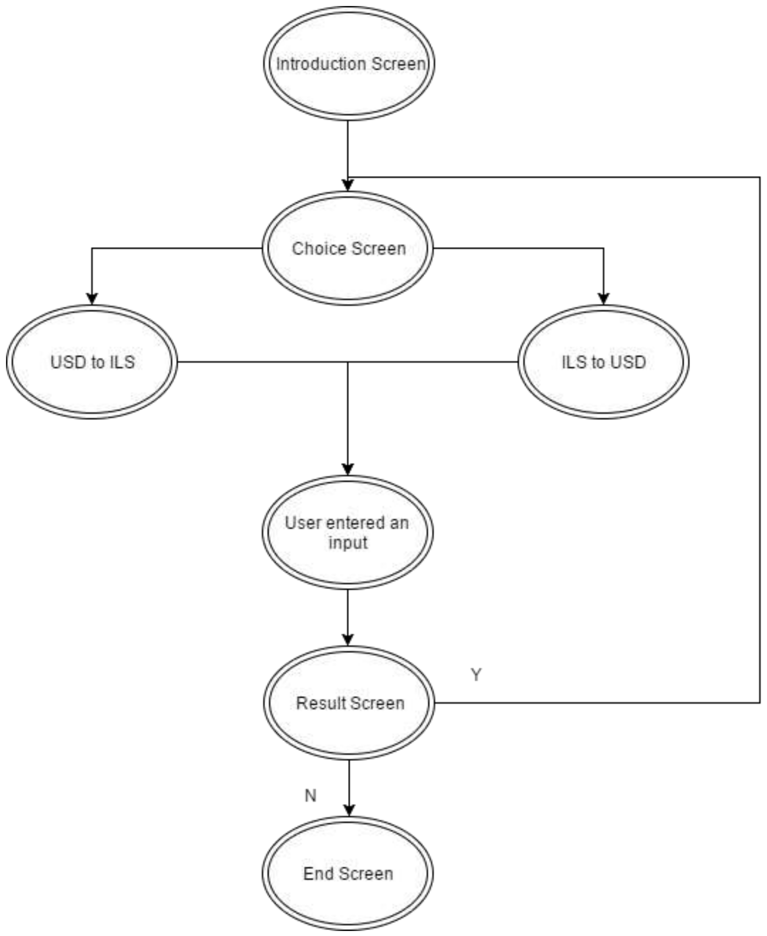
Development platform: program will be developed in Java in Intellij idea IDE.

Distribution type: Public.

Localization: English (US).

Conventions: Java Conventions.

General system diagram:



Guidelines:

General:

1. Where necessary protect code blocks with error handling ways.
2. Use Java "Scanner.class" for receiving input from user.
3. Use proper access modifiers.
4. If user choose an invalid choice, let user choose again and show user the following text: "Invalid Choice, please try again".
5. Each method has to be documented with comments.
6. Stick to this specifications document.
7. Project has to be Maven based.
8. Project has to be uploaded to Github
9. Divide classes into packages for logically division.

Technical:

1. Create an **abstract** class named **Coin** with **abstract** method getValue() which returns a double.
2. Create 2 classes ILS and USD that **extends** **Coin** class, and implement the abstract method getValue().
3. Create a **final** double variable named **value** containing 3.52 in USD class and 0.28 in ILS class.
4. Create an Interface named **ICalcualte** with a method **calculate()** which gets a double argument and returns a double value.
5. Make **Coin** class implement **ICalcualte** and add **calculate()** method.
6. Add **calculate()** to both USD and ILS classes.
7. Call getValue() method from **calculate()** inside both USD and ILS classes to perform the calculation.
8. Use **calculate()** from ILS and USD classes to perform the calculations in your main.
9. Create an enum named Coins with two types ILS and USD.
10. Use factory method to get coins instances (USD and ILS) use the enum types.
11. Both USD and ILS have to be serializable.

Code examples:

```
Class Main…

double input = scanner.nextDouble();

Coin ilsValue = CoinFactory.getCoinInstance(Coins.USD);

double value = usd.calculate(input);
```

```
Class USD extends Coin

private final double value = 3.52;

@Override

public double getValue(){

    return value;

}

@Override

public double calculate(double input) {

    return input * getValue();

}
```

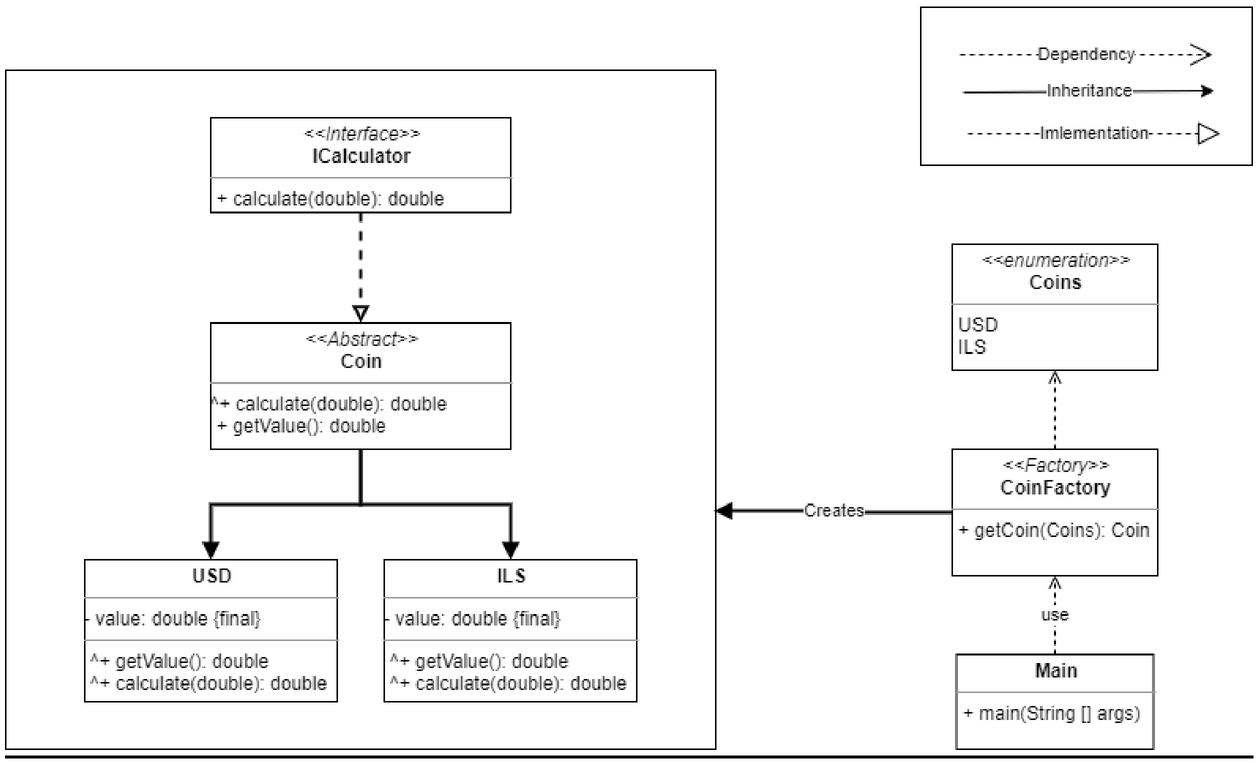
```
public abstract class Coin implements ICalculate… {

    public abstract double getValue();

}
```

```
public enum Coins {

    USD,

    ILS

}
```

```
public class CoinFactory {

    public static Coin getCoinInstance(Coins coin) {

        switch (coin) {

            case ILS:

                return new ILS();

….

    }
```

```
public interface ICalculate {

    double calculate(double value);

}
```

UML diagram:



Screens:

First screen (Welcome Screen) –

Will have following text: “Welcome to currency converter”

“Please choose an option (1/2):”

“1. Dollars to Shekels”

“2. Shekels to Dollars”

(User enters his choice 1/2)

Second screen (Choice Screen) –

Will have following text: “Please enter an amount to convert” (amount will be in double).

(User enters an amount to convert)

Third screen (Result screen) –

1. Will show user the result.
2. Result will be saved in a list.
3. Ask user if to start over Y / N.

Y – Will start cycle again (First screen without “welcome..” text).

N – Will show end screen.

(User choose Y / N - allowing non case sensitive)

Fourth Screen (End Screen) –

1. Will have following text: “Thanks for using our currency converter”.
2. Will print all previous results from results list.
3. Will write all results to results.txt (file)

**** Screen means to show in console**

Bonuses –

1. Write a unit test for your calculator with the following 3 tests:
   - Entering a value to convert
   - Asserting Result is valid
   - Checking the content of the results file.

2. Create an object named **Result** which will hold the result and the conversion flow, for example:
   Result result = new Result(4.27, “USD to ILS”);
   - Use it in your results list (ArrayList<Result> list).

3. Learn about REST Api and:
   - Get the Currency rate using any currency REST Api.
   - Serialize conversion value into Result object.
   - In case API is not available use the hard coded value and print:
     "Could not get rate from API using default rate…"

4. Read about Javadoc:
   https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html
   And add it to your project.

5. Add a third coin EUR with the value of 4.23 and add it as an option of conversion from ILS **only**.

6. Once user decided to exit (choose **n** in result screen), open results file automatically (from code).

7. Create a project using JavaFX / Java Swing which will show the the content of the results file (results.txt).