# Money Changer Exercise

This exercise that will show you how to write a program that will take a total number of cents and decide how many dollars, quarters, dimes, nickels, and pennies should be given out. We will use modulus to accomplish this. This program could be adapted to give change in a vending machine, etc. Our program will just print the answers instead of disbursing coins.

1. From the **File** menu, choose **New Project…**

2. Choose the **Java folder** on the left hand side of the window, then choose **Java Application.**

3. Click **Next**. Give your project the name **MoneyChanger**, select the **location** of where you would like to save your project (Chapter2Exercises folder). Make sure that the **Create Main Class** box is checked and in the textbox change the name to **MoneyChanger** instead of moneychanger.MoneyChanger. Click the **Finish** button.

4. The first thing to do in all Java programs is to put some comments about the program at the beginning of the program. There should be a comment line for name of program, author, date, and the JDK used. These lines should look something like this:

```
/*
 * Your Name and Date
 * This program will break change into corrects coins
 * JDK Version
 */
```

5. Inside of the main method, the variables to be used should be declared first. In this program, we will have several variables. All of the variables will be whole numbers so they will all be int. We will declare the following variables as integers:

```
int cents=393;
int centsLeft;  //temporary variable for storing remaining cents
int dollars;
int quarters;
int dimes;
int nickels;
int pennies;
```

6. The formula for figuring out the number of coins associated with the cents is a complex series of statements. The general logic of it is to begin with the largest denomination which is dollars. Once we know how many dollars are needed, then it must be determined how much money is left to still figure out. The variable that will contain the cents remaining to still have coins dispensed will be called *centsLeft*. The Java statements that will successively figure out dollars, then quarters, then dimes, etc. is given below:

```
dollars = cents/100;
centsLeft = cents %100;
quarters = centsLeft/25;
centsLeft = centsLeft %25;
dimes = centsLeft/10;
centsLeft = centsLeft %10;
nickels = centsLeft/5;
centsLeft = centsLeft %5;
pennies = centsLeft;
```

7. **What do the above calculations do?** We begin with 393 cents which is $3.93 is contained in the variable called *cents*. So ***cents*** contains the total number of cents we have to work with. To figure the dollars we divide by 100 first because there are 100 cents in a dollar. 393/100 will do integer arithmetic because both *cents* and 100 are ints. Integer arithmetic gives us just the integer part as an answer and thus we will get an answer of 3. So the variable dollars will contain the integer 3 which is exactly right!! We now need to know how much money we still need to figure out. We know to give the person 3 dollars in change... but how many quarters, nickels, etc. should they get? To figure out how much money is left, we take cents which is 393 and do a modulus 100. Modulus means to divide but keep the remainder instead of the answer. Dividing 393 by 100 gives you a quotient of 3 (which we don't want at this time) and a remainder of 93 so we put the remainder of 93 (393-300) into *centsLeft*. We are using the variable *centsLeft* to tell us how much change we still need to figure out. In the next formula, we calculate quarters by taking the *centsLeft* of 93 and dividing by 25. This will give us 3 quarters because of integer arithmetic. We are not done so we will figure out *centsLeft* again by doing a modulus of 25 and it gives us a remainder (93-75) of 18 cents. To figure out dimes, we divide the *centsLeft* of 18 by 10 and get 1 dime. The next line calculates *centsLeft* with modulus and gets a remainder (18-10) of 8 cents. Nickels are determined by dividing by 5 and getting 1 nickel. Using modulus again, the *centsLeft* is calculated (8-5) as 3 cents. At this point, all that remains is pennies so the centsLeft is placed into the variable called pennies.

8. The computer has calculated the various denominations of change, but the results are stored in memory and not shown on screen. Each denomination will be placed on a separate line. Instead of using the println method for each line, we can use the "\n" to print a new line after each denomination as follows:

   **System.out.println("Total Cents: " + cents**
   **+ "\nDollars: " + dollars**
   **+ "\nQuarters: " + quarters**
   **+ "\nDimes: " + dimes**
   **+ "\nNickels: " + nickels**
   **+ "\nPennies: " + pennies);**

9. The above lines print the total cents, dollars, etc. on separate lines. Each line consists of words to print such as "Total cents " and the actual number which is in the corresponding variable such as *cents*. The + symbol used here does not mean arithmetic but means to concatenate or "join" the words and the numbers.

10. Compile this program and then correct any errors you have in syntax. Execute your Java program. Your screen should display:

    *Total Cents: 393*
    *Dollars: 3*
    *Quarters: 3*
    *Dimes: 1*
    *Nickels: 1*
    *Pennies: 3*

11. The program should now be:

```
/*
 * Your Name and Date
 * This program will break change into corrects coins
 * JDK Version
 */

public class MoneyChanger {

    public static void main(String[] args) {
        //declaring variables
        int cents = 393;
        int centsLeft; //temporary variable for storing remaining cents
        int dollars;
        int quarters;
        int dimes;
        int nickels;
        int pennies;

        //begin calculations
        dollars = cents / 100;
        centsLeft = cents % 100;
        quarters = centsLeft / 25;
        centsLeft = centsLeft % 25;
        dimes = centsLeft / 10;
        centsLeft = centsLeft % 10;
        nickels = centsLeft / 5;
        centsLeft = centsLeft % 5;
        pennies = centsLeft;

        //printing results
        System.out.println("Total Cents: " + cents
                + "\nDollars: " + dollars
                + "\nQuarters: " + quarters
                + "\nDimes: " + dimes
                + "\nNickels: " + nickels
                + "\nPennies: " + pennies);
    }
}
```

12. If you want the computer to calculate what change to give for 549 cents, what would you change in your program? YES, you would change the line that says 393 to be 549

    **int cents = 549;**

13. Recompile the program with cents being initialized to 549. Execute the program. The formulas for calculating the change are generic formulas and will work no matter what cents you give it.

14. Try your own initialization of cents to some number not tried so far and see if it calculates each of the money denominations correctly.

15. Let's adjust this program to allow the user to input the cents via a dialog box. We will add the JOptionPane input dialog as the first line of your main method and change the cents variable to parse the answer variable.

    **String answer = JOptionPane.showInputDialog(null, "Enter total cents");**
    **int cents = Integer.parseInt(answer);**

1. Because you are using the JOptionPane class in this program, you will need an import statement after your beginning comments as follows:

    **import javax.swing.JOptionPane;**

2. Compile and execute this program. When the dialog box appears requesting cents, enter 393 and click on OK button. The output should look the same as it did before since the *println* statements have not been adjusted.

3. To combine all the *println* statements into one dialog box that appears with all the money change, one very long continuous JOptionPane.showMessageDialog line will be developed with the usage of the escape code \n to display multiple lines as follows:

```
JOptionPane.showMessageDialog(null, "Total Cents: " + cents
        + "\nDollars: " + dollars
        + "\nQuarters: " + quarters
        + "\nDimes: " + dimes
        + "\nNickels: " + nickels
        + "\nPennies: " + pennies);
```

4. Notice that not only the *System.out.println* lines are eliminated; they are replaced by one JoptionPane statement. The above says to display one dialog box on multiple lines (caused by the \n). Everything is concatenated with + signs so that it really represents one very long String.

5. Compile and execute the program. Enter 393 in the dialog box when it asks for the total cents. Make sure that the output is correct.

6. It seems a bit awkward to ask a user to enter in “total cents”. It would make more sense for the user to be asked enter in a decimal amount representing change. Therefore, adjust those two lines early in the program to be:

   **String answer = JOptionPane.showInputDialog(null, "Enter amount of change");**
   **int cents = (int) (Double.parseDouble(answer) * 100);**

7. The above line takes the decimal number that the user enters and saves it as a string in the String variable called answer. It then converts that string to a double, then multiplies by 100, and then takes the integer of that and places that final result into the integer variable called cents.

8. Compile and execute the program. Enter 3.93 when asked to enter the amount of change. It should work fine. Execute the program again entering 4.58 to see if it works for that amount.

9. The final program should look as follows:

```java
/*
 * Your name and date
 * This program will break change into correct coins
 * JDK version
 */
import javax.swing.JOptionPane;

public class MoneyChanger {

    public static void main(String[] args) {
        //declaring variables
        String answer =JOptionPane.showInputDialog(null,
                "Enter amount of change");
        int  cents = (int)(Double.parseDouble(answer) * 100);
        int centsLeft; //temporary variable
        int dollars;
        int quarters;
        int dimes;
        int nickels;
        int pennies;

        //begins calculations
        dollars = cents / 100;
        centsLeft = cents % 100;
        quarters = centsLeft / 25;
        centsLeft = centsLeft % 25;
        dimes = centsLeft / 10;
        centsLeft = centsLeft % 10;
        nickels = centsLeft / 5;
        centsLeft = centsLeft % 5;
        pennies = centsLeft;

        //printing results
        JOptionPane.showMessageDialog(null,
                "Total Cents: " + cents
                + "\nDollars: " + dollars
                + "\nQuarters: " + quarters
                + "\nDimes: " + dimes
                + "\nNickels: " + nickels
                + "\nPennies: " + pennies);

    }
}
```