**CSPS 1100 Lab 4**

In this chapter, we are going to work with the concepts of using mathematical calculations and reading input from the user. At this point, it is assumed that you know how to create the plumbing class which contains a main and serves as the tester program. Also, you know how to create another class. If you feel that you cannot create classes, then review lab 3.

1. We are going to begin with some simple math, reading input, and formatted output.
   1. Create a class called MyCalculator.
   2. You will not have an instance variable.
   3. Since you have no instance variables to initialize, you do not need a constructor. Do NOT create a constructor.
   4. Create a method named add. This method should have two arguments, X and Y (Both of these should be integers). The method should add X and Y together and return the result. Remember the return type needs to be an int.
   5. Create a method named add. This method should have two arguments, X and Y (Both of these should be floating points). The method should add X and Y together and return the result. Remember the return type need to be a double. This is an example of an overload for a method. The compiler will choose which method to use based on the values being sent.
   6. Repeat steps d and e for subtract, divide, and multiply.
   7. Create a MyCalculatorTester class. Remember this class has the main and no methods.
   8. Import java.util.Scanner;
   9. Create an object of type Scanner. I suggest you use the name in or input so that you have an idea of what you are doing.
   10. Write the code to prompt the user to input the value for X. Use something like this.

System.out.println(“Please enter the value for X as an integer: “);

* 1. Create a variable of type int named X to receive the input. There is an example in your book on page 145 in the box named Syntax 4.3.
  2. Write the code to prompt the user to input the value for Y using step j as an example.
  3. Repeat step k except you want to name the variables Y.
  4. Repeat steps j – m asking for doubles rather than integers. You will need to change the names of each of the variables. Perhaps XD and YD would work. (After this step, you should have 4 variables, one for an X integer value, one for a Y integer value, one for an X double value, and one for a Y double value.
  5. Create an object of type MyCalculator. This would look very much like the object you created for BankAccountusing the default constructor. Like this:

BankAccountharryAccount = new BankAccount();

* 1. Using the object which you created call each of the methods and print the results. You have 2 options at this point, you may create new variables to hold the results such as int total = calc.add(X,Y) and then print total. You may also print straight from the call like System.out.println(“The sum of X and Y is: “ +calc.add(x,y)); A couple on notes at this point. The word calc is the name of the object. Your name may be different. Where I put int total, the int would only apply to the integer values and you can only declare a type once. What that means is that if I were creating a total and using it for integer add, subtract, divide, and multiply, I would only put int total one time. After that I would simply say total = . Also I would need a new name for the doubles. An example here might be double totalD = calc.add(XD, YD);)

1. We are going to redo our calculator a little to allow it to calculate more than two numbers.
   1. Copy both your MyCalculator and MyCalculatorTester. Change the names to MyCalculatorAdvanced and MyCalculatorAdvancedTester.
   2. Open the MyCalculatorAdvanced.
   3. Add an instance variable called total. This will hold your integer totals.
   4. Add an instance variable called totalD. This will hold your double totals.
   5. Add a constructor with no parameters that sets the value of total and totalDto 0.
   6. Change the return type of each of your methods to void.
   7. Remove the return statement from each of your methods.
   8. Use the total to hold the values. For example total = X + Y.
   9. Create a method called getTotal() similar to getBalance(). Remember this will return an integer.
   10. Create a method called getTotalD(). This will return the double.
   11. Create a method called clearTotal(). This method will set total to 0. Do not return anything.
   12. Create a method called clearTotalD(). This method will set totalD to 0. Do not return anything.
   13. Open your MyCalculatorAdvancedTester.
   14. Everytime it says MyCalculator change to MyCalculatorAdvanced.
   15. Add another prompt. Ask the use to enter another integer number. You will need another integer variable for this.
   16. Add another prompt to ask the user to enter another double number. You will need another double variable for this.
   17. Calculate X + Y – Z. If you did not create a variable named total in the previous problem, do so now. You will need a total.
   18. Call add using x and y;
   19. Call getTotal();
   20. Call subtract using the total from step s. calc.subtract(total,z);
   21. Call getTotal() again to get the new result.
   22. Print the result. Be sure and print the formula with the result.
   23. Call the clearTotal(). Remember to use the object name.
   24. If you did not create a variable named total in the previous problem, do so now. You will need a totalD.
   25. Repeat steps q-v for doubles instead of integers.
   26. Repeat steps r-y for the following formula. X \* Y + Z for both integers and doubles.
2. I hope this program is a little fun and that it will cause you to think. This program will simulate the growth and decrease of bugs based on breeding the bugs and spraying the bugs. This is a program where you will work most on your own without a lot of guidance. Ask questions where you need help.
   1. Create a Bug class
   2. Create an instance variable to hold the number of bugs.
   3. Create a constructor that sets the original number of bugs to an amount that is provided through the argument. Hint: BankAccount with an initial balance.
   4. Create a method call breedBugs. This method will not return anything and will not have an argument. The breedBugs method should double the number of bugs each time it is invoked or called
   5. Create a method called sprayBugs. This method will not return anything and will not have an argument. The sprayBugs should decrease the number of bugs by 1/4 each time it is invoked.Multiply the total bugs by .75. Your bugs should be integers because you do not have partial bugs. However, because we are dividing your result may be a double. To take care of this problem, put (int)(your formula) in your method. This is a cast. Be sure and put your formula in ().
   6. Create a method to return the number of bugs.
   7. Create a BugTester class
   8. Create a main
   9. Create a Bug object and begin with 10 bugs.Hint: Remember how you created a BankAccount Object with an initial balance.
   10. Call / invoke the breedBugsmethod . Hint: Call deposit but no parameters.
   11. Call the sprayBugs method 3 .
   12. Print the number of bugs (The result should be an integer – no half dead bugs)
   13. Repeat j – l two additional times. This means that you will breed, spray and print 3 times.
3. **This was one of the problems from the first assignment. Correct Change:** Write a program to assist a cashier with determining correct change.
   1. Create a Change Class.
   2. Create 4 integer instance variable, one for quarters, dimes, nickels, pennies.
   3. Create a constructor that does not have any parameters and sets each of the instance variables to 0.
   4. Create 4 integer constant variables. One for quarters (ie. private static final intQUARTER\_VALUE = 25. Do the same thing for dimes, nickels, and pennies.
   5. Create a calculateQuarters method. This method should accept a single integer (amount) to serves as the argument as used as input to the method. Calculate the number of quarters by dividing amount by the value of quarters and assign it to the quarters instance variable. Use the QUARTER\_VALUEvariable in your calculations see page 137. Using modulus (see page 138-139 of the text) determine the remainder. The remainder should be returned to the calling program.
   6. Create 3 more methods like the calculateQuarters for dimes, nickels and pennies.
   7. Create 4 get methods. One to return the number of quarters, one for dimes, one for nickels, and one for pennies.
   8. Create a ChangeTester
   9. Import Scanner.
   10. Create a Scanner object.
   11. Prompt the user to enter the amount.
   12. Create an integer variable named change and using Scanner get the input from the user.
   13. Create an object of type Change.
   14. Create an integer variable called remainder and set it to 0.
   15. Call the calculateQuarters method and put the return value in the variable called remainder. Example: remainer = myObject.calculateQuaters(change);
   16. Call the calculateDimes method and put the return value in the variable called remainder. In this case, when you call calculateDimes use remainder as the parameter.
   17. Repeat sept p for nickels and pennies.
   18. Call and print getQuarters(), getDimes(), getNickels(), and getPennies(). When printing identify each number printed. Example: Quarters: 2
4. In this program we are going to work with strings. This will be a fun one I hope.
   1. Create a class called NameTester. (You will not need a class. This will have the main method in it).
   2. Import Scanner
   3. Create a Scanner object
   4. Prompt for your first name.
   5. Create a variable for your first and get the user input for your first name.
   6. Prompt for your last name.
   7. Create a variable for your last name and get the input for your last name.
   8. Create a variable for a blank.
   9. Create a variable for your full name and assign it your first and last name and a blank using the + operator.
   10. Using the variable for your full name, use the substring method (use the table on page 159) as a guide) print out your name with one letter on each line. For example:

M

S

.

W

**Extra Credit:**

1. Work out the programming project P4.2 on page 170
2. Work out the programming project P4.7 on page 171