Read-Compute-Write Pattern: Sample Exercises

**Topics**

Objects/classes: type `int, double`; arithmetic and assignment

class `String` and its assignment operator

Java language: identifiers and variables; types and classes: the first look

```java
console.out.print/println
int x = console.in.demandInt("prompt");
double x = console.in.demandDouble("prompt");
String s = console.in.demandString("prompt");
```

Exercises: read-write

read-compute-write, using `int, double, and String`

**Overview**

At the core of many programs are small sections of code where some given set of input values is transformed through computations into a set of resulting values. In the simplest form only a few arithmetic expressions and assignments are needed. This set of exercises focuses on problems where only a short straightforward calculation is needed to compute the result. Later we will see more complex computational patterns.

These exercises use the *read-compute-write pattern* to solve the problem. All exercises use the JPT Console to read the data and print the results, except for the last exercise that also illustrates the use of Java2D graphics for painting simple colored shapes. The console input functions verify that the input is of the type demanded by the program. The programs use two basic types of variables (`int` and `double`) and also objects in the class `String`. The programmer learns to use the *declare-define-use pattern* for the identifiers representing these entities.

The compute part is also used to learn and practice the rules of both integer and decimal number arithmetic: the precedence rules, the conversion rules, and just the computational rules.

**Design strategies for students**

1. Read the problem carefully. Make a list of all information items specified in the problem.
2. Use a pencil and paper to solve the problem by hand. Create several examples of application of this problem that use different data.
3. Update your first list to include any items you may have omitted. Note which are constants (they are the same for all instances of the problem) and which are the variables that depend on the user input. For example, when converting hours to minutes, 60 minutes in an hour is a constant, while the number of hours depends on the user input and is a variable. The result in minutes is also variable. Give
meaningful names to the variables and for each variable decide whether it should be an integer (whole number), double (a decimal number), or a string (sequence of letters and other characters).

4. Write the statements that will declare these variables.

5. Figure out the program interface: what information you will need from the user and what information will the user receive in return. For example for the hours to minutes conversion, the user needs to supply the number of hours, the program will print the number of minutes in the given number of hours.

6. Write the statements that will perform the reading and writing tasks.

7. Make a list of at least three test cases that specify the input values and the desired outputs for this input set.

8. Write out the sequence of computations. If you saved an intermediate result, you may use another variable to save it. Typically, you may need one or more variables to save the result.

9. Write the code that will perform the computations, paying attention to the rules of arithmetic for the Java language.

10. Test your program with the prepared test cases. Compare the results with your hand calculations and make sure they are the same. If they differ, make sure you know why. If you cannot figure out why, ASK SOMEONE!!!! Add other tests if you are not confident about the result.

11. Clean up your code: add comments, check indentation, add white space, check the variable names.

12. Save you work on a diskette or on a web site and print your program.

**Exercise pedagogy for instructor**

Read with students solutions to the first five problems, and one more. Identify the steps in each program. Ask students to solve in small groups one of the later solved problems. Compare their solutions with the given ones.

Ask students to solve in small groups one of the easier practice exercises. Have them write the solutions on the board (concurrently) and compare the results. Ask each group to pick one of the harder exercises and let those that are done present their solution on the board.

Each student should run all exercises that have been done in the class and the exercise that was started in their group.
Sample Solved Exercises

The first four exercises are straightforward - read the code to become familiar with the syntax of forming the output string, reading in integers or strings, performing simple computations.

**Exercise 1: Product of Two**
Ask the user to type in two numbers (integers). Print the product of these two numbers.

**Exercise 2: Name and Town**
Ask the user her name. Ask the user what town is she from. Type a message "Your name is from your town".

**Exercise 3: Feet to Inches Conversion**
Ask the user to type in a length in feet and inches (whole numbers only). Print the total length in inches.
Note: A simple solution accepts any integer input - including negative values or number of inches greater than 12. We will learn how to filter the input in later exercises.

**Exercise 4: Name and Age**
Ask the user his name and age. Print a message of the type "John Doe is 25 years old."

The next set of exercises requires a modest amount of computation. Make sure you test the program with several sets of input data.

**Exercise 5: Count Change**
Ask the user how many quarters, dimes, nickels, and pennies are in his wallet. Print the total amount of change in dollars and cents.

We need to record four input values: quarters, dimes, nickels, and pennies - all integers. We do not include the code to check whether the values are non-negative. We need two output values - the total of dollars and the total of cents. However, when we think of the calculation we realize that first we compute the total amount in cents. We then we find out the total amount of dollars by dividing this total amount by 100, and compute the remaining amount in cents as a reminder after the division by 100 of the original total amount in cents.

We decided not to use constants to represent the numeric value of the different coins - as the problem is quite straightforward.

Once we understand the computation involved, we prepare several test cases. We make sure the test suite includes cases where one of the inputs is zero (i.e. the user has no nickels in his wallet). We also make sure at least one test case results in zero dollars and at least one test case results in zero cents.
So, the sample test cases may be:

<table>
<thead>
<tr>
<th>Coins</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>quarters</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>dimes</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>nickels</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pennies</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>$0.96</td>
<td>$2.00</td>
<td>$1.24</td>
<td>$2.47</td>
<td>$1.41</td>
</tr>
</tbody>
</table>

Make sure you write down the calculations that need to be done, work out a couple of problems by hand, and generate your suite of test cases with expected results before you try to understand the code.

**Exercise 6: Trip Planning**
You plan to visit a friend in another town. You are given the distance in miles (with possibly fractional part - like 3.5 miles). How long will the trip take if you drive on the average at a given speed. (One assumes, you give the speed in miles per hour.)

This problem seems quite simple - until we start thinking about how to deal with the different measurements. We should start with a couple of straightforward examples. If the distance is 60 miles, it will take us one hour at 60 mph and two hours at 30mph. If the speed is 45 mph, the trip will take 1 hour at 20 minutes. How did we get that? Divide 60 miles by 45 miles per hour and get 1.3333. 0.3333 hour is one third of an hour, or 20 minutes.

We need to figure out how to convert fraction of an hour into whole minutes. Our program multiplies this quantity by 60 and casts the result as an integer.

Work out the computations by hand. Work out a couple of sample problems and record the expected input. Read the code and see how it compares with your solution. Work out your problems by following the given code. Run your test cases to see if you were right.

**Exercise 7: Hike**
You returned from a hike, and wonder why it took so long to cover only a couple of miles. It must have been the climbing that you did, that added the extra time to the trip. Compute your average rate of climb - the number of feet you climbed up per hour. Input the length of the climbing part of your hike, the elevation at the start and at the top.

**Exercise 8: Payroll**
Compute the gross pay for the last week for a worker who gets paid 1.5 times of the regular wages for working overtime (more than 40 hours in a week). You are given the number of hours worked and the hourly pay rate.

**Exercise 9: Race Timer**
Runners in the Boston Marathon - and many other races - want to know what was their average speed. However, they like to be told this information by specifying how long
does it take them to run a mile. The world record for a mile is under 4 minutes, the top marathon runners run in under five minutes miles.

You need to know the time it took to run the race - in hours, minutes, and seconds, and the distance in miles and fractions of a mile.

**Sample Practice Exercises**

Easier Practice Problems:

1. Write the program that will compute the cost of a purchase of several equally priced items in a store. The user types in the number of items and the cost per item.

2. Modify the above program to add 5% sales tax to the total price.

3. Modify the above program so that the user types in also the name of the item. Also, add a message to the client 'Thank you for shopping with us.'.

4. Ask the user to type in how many one dollar, five dollar, ten dollar and twenty dollar bills does she have in her wallet. Print the total amount in the wallet.

5. Ask the user to type in the number of days, hours, and minutes, as well as his heart pulse rate. Print how many times did his heart beat during this time.

6. Write a program that will compute monthly gross pay for a salaried employee. The user types in the yearly salary base and a yearly elected contribution to pre-tax medical account. The program deducts 5% contribution to the retirement fund from the gross pay as well as the appropriate contribution to the medical account and prints the resulting gross pay.

Harder Practice Problems:

7. One pound of flour equals four cups. Write a program for a cook, so she can enter the number of cups needed and the program will print the amount in pounds and ounces. Allow the user to enter 1.5 cups or 0.666 cups. Document your strategy in your program.

8. Write a program that will make change. The user enters number of cents due the customer (<100) and the program will print how many quarters, dimes, nickels, and pennies are needed. At this point you do not need to verify that the amount entered is indeed less than 100 and more than 0. You may want to explore how your program will perform with the erroneous input.

9. Recently a new world record for 400m has been set at 43.18 seconds. Write a program that will read the length of the race in meters and the time in seconds and fractions of seconds and print the average speed of the racer in miles per hour.

10. Write a program that will compute the number of hours and minutes of daylight. The user types in the time of the sunrise (am) and the time of the sunset (pm). Count only the time the sun is up (not the dawn and dusk hours).

11. Foresters compute the volume of the wood in a forest using estimations and computations.
(a) First they compute the height of a tree by measuring the angle from the ground to the top of the tree at a given distance from a tree. Write a program that will let the user type in the distance from the tree and the angle and will print the tree height. You decide what units you will use.

(b) Next, compute the volume of the tree trunk. Assume the tree is a cone of the given height, and with a given diameter of the trunk at the base.

(c) Finally, if you are given the average volume of a tree, the average number of trees per acre, and the size of the plot in acres, compute the volume of the wood in the forest.

12. You are getting ready for a scuba dive. You need to figure out how much oxygen will you need for a dive. You want to compute the volume of oxygen you breathe in during for the length of time you plan to be underwater. You need to know how many times you inhale in a minute and you need to measure your lung capacity. Remember that the air contains 20% oxygen.

Let the user type in the proposed duration of the dive, the respiration rate (inhalations per minute) and the lung capacity of the prospective diver (typically between 0.25 liters and 0.5 liters). Print how much oxygen will be needed.

13. In preparation for a homecoming weekend you plan to paint a float that will represent the earth globe. How much blue and how much yellow paint will you need for the orb of a given diameter? One pint of paint covers 75 square feet area. The earth surface is 2/3 water and 1/3 land. Modify the program to work for paints that cover more or less area.

14. For a home remodeling project you want to tile a floor with new tiles. Write a program that will read the dimensions of the room (assume it is a rectangle) and the size of the square tiles to be used. Compute how many tiles will be needed. Of course, the tiles at two sides may need to be cut to fit.

15. Modify the previous program to compute the cost as well, given the cost of a box of 12 tiles. You cannot buy less than 12 tiles.

16. The computer screen is displayed by a beam that paints pixels as the beam sweeps across the lines of the CRT. The refresh rate tells us how many times per second is each pixel repainted. Compute the total number of pixels-paint operations per second for a given size screen resolution and the given refresh rate.
Study points:

Patterns

Observe and learn how each identifier is declared, defined, and used. The compiler is responsible for defining when the identifier is first created and when it is destroyed (i.e., the name is no longer valid and the space reserved for it has been released back to the free memory pool).

Using variables of the type `int`:

```java
int x;           // declare x as a variable of the type int - i.e. representing one whole number
x = 4;           // initialize the value of the variable x to be 4
int x = 4;       // declare x as variable of the type int and initialize the value to be 4
x = 4 * 3 - 5/(2 + 3); // use variable x; assign a new value to it.
int y = 5 * x;    // declare and define new int variable y; use variable x
```

The compiler destroys `int` variable automatically when appropriate.

Using objects in the class `String`:

```java
String s;       //declare s to represent a character string
s = "Hello World!";   //assigns the initial value to the string s
String a = "Hello";
String b = "World";
String s = a + " " + b + "!"; // declares s as string,
                              // initialized the initial value
                              // using previously defined strings a and b
```

Again, we do not have to worry about the destroy step.