Chapter 2: Review Exercise Solutions

R2.1

The value of `mystery` is equal to 0 after the statements are executed. In the first statement (line 1), `mystery` is initialized to a value of 1. In the assignment statement on line 2, `mystery` is set to -1. Finally, `mystery` is set to 0 in line 3.

R2.2

The variable `mystery` is being declared twice, first in line 1 and then again in line 2. A variable can only be initialized once. (If you remove the reserved word `int` on line 3, the statements will work just fine, and will set `mystery` to -3.)

R2.3

```
s = s0 + v0 * t + g * Math.pow(t, 2) / 2.0;
G = 4.0 * Math.pow(Math.PI,2))* Math.pow(a,3) / (Math.pow(p,2) * (m1 + m2));
FV = PV * Math.pow((1.0 + INT / 100.0), YRS);
c = Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2) - 2 * a * b * Math.cos(gamma));
```

R2.4

a) $dm = \mathrm{m}\left(\frac{\sqrt{1+\frac{v}{c}}}{\sqrt{1-\frac{v}{c}}} - 1\right)$

b) $volume = \pi r^2 h$

c) $volume = 4\pi\frac{r^3}{3}$

d) $z = \sqrt{x^2 + y^2}$

R2.5

a) `6.25`
b) `6`
c) `12.5`
d) `-3`
e) `1.4142135623730951`

R2.6

a) `8`

b) `1`
c) `17`
d) `17.5`
e) `17`
f) `18`

R2.7

a) `10`
b) `e`
c) `llo`
d) `HelloWorld`
e) `WorldHello`

R2.8

1) There is an extraneous semicolon after `main()`.
2) The message in the first print statement is not surrounded by quotation marks like a string should be.
3) The variables `x` and `y` are not declared.
4) The variable `in` is not declared.
5) The method `readDouble` doesn't exist (`nextDouble` does).
6) If `readDouble` were a method, it should be followed by `()`.
7) In the last line the method `println` is incorrectly spelled `printline`.

R2.9

1) The scanner should have been constructed as `Scanner in = new Scanner(System.in);`
2) The second integer should have been read as `y = in.readInt();`
3) The sum should have been printed as `"The sum is " + (x + y)`, so that the integer `x` is not concatenated with the string `"The sum is"`.

R2.10

The given output is printed in a raw format up to the range of a `double` data type. Users can use *format specifiers* (`printf` with `%`) to format the output as they require.

R2.11

The type of `2.0` is a double; specifically the Java programming language recognizes it as the real number 2.0. On the other hand, Java treats `"2.0"` as a string. Its value is literally the three characters `2`, `.`, and `0`.

R2.12

a) This statement doubles the value of x. Given the initial value of x as 2, the result is 4.
b) This statement concatenates the strings `"2"` together. The result is `"22"`.

R2.13

Read input into a variable called word.
Print first character in word followed by a space.
Print character at length -1 in word followed by a space.
Print substring between 1st and last character (length -1) in word.

R2.14

Read first name into variable called first.
Read middle name into variable called middle.
Read last name into variable called last.

Print first character in first.
Print first character in middle.
Print first character in last.

R2.15

// Input is stored in variable called num
exponent = integer part of log10 of num
first digit = integer part of num / 10^exponent
last digit = num % 10
Print first digit.
Print last digit.

R2.16

change due = 100 x bill value – item price in pennies
quarters = change due / 25 (without remainder)
change due = change due % 25
dimes = change due / 10 (without remainder)
amount due = change due % 10
nickels = change due / 5 (without remainder)

// Change is now in quarters, dimes, and nickels

R2.17

First, compute the total volume by hand using the formula in Self Check 25. Let's assume the following values for the three sections:

- Conic Section 1 (top of bottle) has first radius ($r1_1$) 0.5 inches, second radius ($r2_1$) 0.75 inches, and height (h_1) 1.5 inches.
- Conic Section 2 (middle) has first radius ($r1_2$) 0.75 inches, second radius ($r2_2$) 1.5 inches, and height (h_2) 0.5 inches.
- Conic Section 3 (bottom of bottle) has first radius ($r1_3$) 1.5 inches, second radius ($r2_3$) 0.75 inches, and height (h_3) 6.0 inches.

So, the total volume of the bottle, V_t, will equal the volume of conic section 1 (V_1) plus the volume of conic section 2 (V_2) plus the volume of the conic section 3 (V_3):

$V_t = V_1 + V_2 + V_3$

Using the equation given in Self Check 25, we calculate the three volumes to be (rounded to two decimal places):

$V_1 = 1.87$ cubic inches
$V_2 = 2.06$ cubic inches
$V_3 = 24.74$ cubic inches

So, the total volume for the cocktail shaker in our example is:

$V_t = V_1 + V_2 + V_3 = 28.67$ cubic inches

The algorithm we use is very similar to the calculation we did above:

volume 1 = $\pi \times (r1_1^2 + r1_1 \times r2_1 + r2_1^2) \times h1 / 3$
volume 2 = $\pi \times (r1_2^2 + r1_2 \times r2_2 + r2_2^2) \times h2 / 3$
volume 3 = $\pi \times (r1_3^2 + r1_3 \times r2_3 + r2_3^2) \times h3 / 3$
total volume = volume 1 + volume 2 + volume 3

R2.18

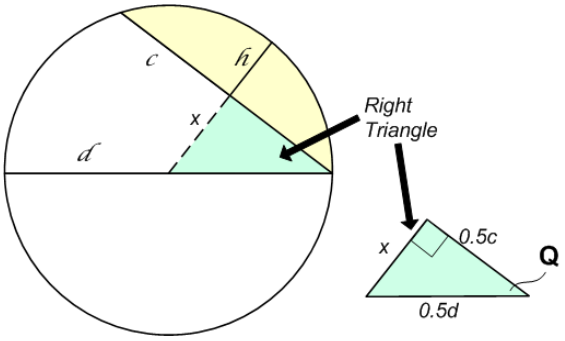
Because we are given the diameter of the circle, d, and not the height of the circular sector, h, we need to develop an algorithm to find h based on the value of d.

If you look at the figure on the right, you see that a right triangle can be made bounded by the chord, the diameter of the circle, and the radius of the circle drawn through h.

Because we know the length of the radius (half the diameter), we can set up an equation to find the value of h:

h = 0.5d - x

where x is one side of the right triangle. We can solve for x by using trigonometry on the right triangle, and then use that value to solve for the value of h, which we need to compute the area of the circular sector (the oddly-shaped piece of pie). In the right triangle, the side labeled x is considered to be the "opposite" side. We know that the hypotenuse of the right triangle is half the diameter, or 0.5d, and the "adjacent" side of the right triangle is half the chord length, or 0.5c. Therefore, we can use the following formulas for a right triangle to determine what we need for the problem:



Facts:
The radius of the circle, which is half the diameter, is equal to h + x (so h = 0.5d – x).
Adjacent side of triangle is half the chord length (0.5c).
Hypotenuse is half the diameter (0.5d).
Solve for x using trigonometry, then use x and d to solve for h.

`sin θ = opposite/hypotenuse  cos θ = adjacent/hypotenuse`

Here is the general algorithm (see figure for definition of the variables):

Ask the user to enter the diameter of the circle, store the value in d.
Ask the user to enter the length of the chord, store the value in c.
Compute the angle θ with formula $\theta = \cos^{-1}(0.5c / 0.5d)$.
Compute the length of x with the formula x = sin θ * 0.5d.
Determine h using the formula h = 0.5d – x.
Calculate the area of the pie piece, A, using the formula.
A = 2.0 / 3 * c * h + pow(h, 3) / (2 * c).

We can use this algorithm to solve for the specific example of a pie with diameter 12 inches and a chord length of 10 inches:

We know that d = 12, c = 10.
Therefore, $\theta = \cos^{-1}(0.5c/0.5d) = \cos^{-1}(5/6) = 33.56$ degrees
x = sin θ * 0.5d = sin 33.56 * 6 = 3.32 inches
h = 0.5d – x = 6 – 3.32 = 2.68 inches
$A = 2/3 * c * h + h^3/(2*c) = 2/3 * 10 * 2.68 + 2.68^3 / (2 * 10) = 18.83$ square inches

R2.19

Starting position is 3 x 4 = 12 and length of substring is 3.

S	u	n	M	o	n	T	u	e	W	e	d	T	h	u	F	r	i	S	a	T
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

We can see this will extract the substring `Thu`.

R2.20

Character at position `i` (2)

```
G  a  t  e  w  a  y
0  1  2  3  4  5  6
```

Character at position `j` (4)

```
G  a  t  e  w  a  y
0  1  2  3  4  5  6
```

first:

```
G  a  t  e  w  a  y
0  1  2  3  4  5  6
```

middle:

```
G  a  t  e  w  a  y
0  1  2  3  4  5  6
```

last:

```
G  a  t  e  w  a  y
0  1  2  3  4  5  6
```

Generate a new string: first + character at j + middle + character at i + last

```
G  a  w  e  t  a  y
0  1  2  3  4  5  6
```

R2.21

You can read characters from a string with the `charAt` method. For the first character, pass a position of 0 to `charAt`. For the last character pass the position that is equal to the length of the string -1 to `charAt`.

Strings in Java are immutable, so they cannot be directly changed. Thus, to "remove" the first character from a string, we can take a substring of the original string that contains the entire thing minus the first character. If our string is called `s`, then this works: `s.substring(1, s.length());`. The last character can be obtained by extracting the substring that contains the entire string minus the last character, like this: `s.substring(0, s.length()-1);`.

R2.22

This program:

```
public class R222
```



```
{
   public static void main(String[] args)
   {
      System.out.println(3 * 1000 * 1000 * 1000);
      System.out.println(3.0 * 1000 * 1000 * 1000);
   }
}
```

Prints out the following:

```
-1294967296
3.0E9
```

The reason the first number is negative is because we have exceeded the limited range of an `int` and the number overflowed to a negative. The second number's result is correct but displayed in scientific notation because it is a floating-point type from the `3.0` in the calculation.

R2.23

- Variable names are in lowercase with occasional uppercase characters in the middle.
- Constant names are in uppercase with the occasional underscore.
- No magic numbers may be used.

