Exercise 1

The following program uses primitive data type `short`:

```
public class Shortfall
{
  public static void main ( String[] args )
  {
    short value = 32;
    System.out.println("A short: " +  value);
  }
}
```

`value` in this program is a *variable*—a name for a section of memory that holds data using a particular data type. In this case `value` is the name for 16 bits of main memory that uses `short` to represent an integer. (The next chapter says much more about variables.) This program puts the value 32 into `value`. Then it writes out:

```
A short: 32
```

In other words, that line of the program examines the variable and writes out what it finds.

Your Job: Create a file called `Shortfall.java` that contains this program. (Copy and paste will greatly speed this up.) Compile and run the program. Check what it writes onto the screen. Now edit the program so that the 32 is changed to some other small number, say 356. Compile and run the program. Everything should be fine.

Next change the number to 35000 and try to compile and run the program. This number is too large to work with the data type `short` (in other words, it cannot be represented in 16 bits using data type `short`.) What happens?

Now edit the program (don't change the 35000) so that the data type is `int`. Compile and run the program. Is there a difference?

Exercise 2

The following program uses primitive data type `double`:

```
public class DoubleJeopardy
{
  public static void main ( String[] args )
  {
    double value = 32;
    System.out.println("A double: " +  value);
  }
}
```

In this program, `value` is the name for a variable that uses the `double` data type to represent floating point numbers. Recall that this data type uses 64 bits.

It is perfectly OK to use the name `value` in this and in the previous program. A variable name helps describe what you want the program to do. It does not permanently reserve part of computer memory for any particular use.

Compile and run the program. Does its output (what it puts on the screen) differ from the output of the the previous exercise?

Change the 32 to 32.0 and see if that makes a difference when you compile and run the program.

Exercise 3

The following program uses primitive data type `char`:

```
public class CharAssassination
{
  public static void main ( String[] args )
  {
    char ch = 'A' ;
    System.out.println("A char: " +  ch );
  }
}
```

The variable `ch` is 16 bits of main memory that uses the `char` data type to represent characters. The a bit pattern that represents 'A' is placed in it. The program writes:

```
A char: A
```

Do the following:

- Change the `'A'` into `'Z'` and compile and run.
- Change the `'A'` into `'AA'` and try to compile the program.
- Change the `'A'` into `' '` and compile and run the program.
  - Notice carefully: there is a single space between the two ' marks.
- Change the `'A'` into `''` and try to compile.
  - Notice carefully: there is no character between the two ' marks.
- Change the `'A'` into `"A"` and try to compile the program.
  - (The double quotes " signify a *String*, which is something different from a *char*).

Observe and explain what works and what does not work in the above.

Exercise 4

Examine this program (from the chapter):

```
public class Example
{
  public static void main ( String[] args )
  {
    long   hoursWorked = 40;
    double payRate     = 10.0, taxRate = 0.10;
    System.out.println("Hours Worked: " + hoursWorked );
    System.out.println("pay Amount  : " + (hoursWorked * payRate) );
    System.out.println("tax Amount  : " + (hoursWorked * payRate * taxRate)
);
  }
}
```

Modify it so that each variable is declared by itself and is not initialized. Next write three assignment statements to assign a value to each variable. Run the program; examine the output.

Now let's break something: Remove one of the declarations from the program. Can you compile it?

Now remove one of the initializations from the correct program. (For example, delete the characters "= 40" from the first declaration. Try to compile and run the program. When is a problem detected?

Exercise 5

Say that you are interested in the value of the quadradic

$3X^2 - 8X + 4$

for several values of X. Write a program that has a double precision variable X. Assign a value to it. Write statement that computes a value for the quadradic and stores the result in another double precision variable. Finally write out the result, something like:

```
At X = 4.0 the value is 20.0
```

Run the program with several values for X (edit the program for each value of X) and examine the result. Use values with decimal points, large values, small values, negative values, and zero. Solve the equation with paper and pencil (use the quadradic formula.) The quadradic *should* evaluate to zero at X = 2.0 and at X = 2/3. Try these values for X. Are the results exactly correct?

Exercise 6

Write a program that averages the rain fall for three months, April, May, and June. Declare and initialize a variable to the rain fall for each month. Compute the average, and write out the results, something like:

```
Rainfall for April:  12
Rainfall for May  :  14
Rainfall for June:   8
Average rainfall:    11.333333
```

To get the numerical values to line up use the tabulation character `'\t'` as part of the character string in the output statements. Check that your program prints the correct results. There is a beginner's error lurking in this program too! Did you fall victim to it?

Exercise 7

It is sometimes hard to think in terms of radians; we would rather use degrees. Remember (from those dark days of trigonometry class) that there are PI radians per 180 degrees. So to convert an angle given in degrees to radians do this:

```
rad = degrees * Math.PI/180
```

Math.PI gives you an accurate value of PI.
Edit the previous program so that it does the same things, but the angle is 30 degrees (which you will convert into radians.)