

CHAPTER 5 : OPERATORS AND EXPRESSIONS



In this Chapter...

You will learn about:

arithmetic operators namely + - * / (integer and real division) and % (modulus), incrementing (count++) and decrementing (count--), order of arithmetic operations, assignment statements and the length method from the String class

You will learn how to:

write programs with arithmetic operations, evaluate arithmetic expressions and determine the resulting type, assign values or expressions to variables, declare variables, find the number of characters in a String

5.1 The plus (+) operator

You are familiar with the + operator:

When using numbers, the + operator adds the numbers together.

Eg. $5 + 7 = 12$

When using strings or characters, the + operator joins (or concatenates) the strings or characters together. Eg. "sun" + "shine" = "sunshine"

In compound output statements, consisting of more than one item joined using plus signs, the items are converted to strings, and then joined together.

Eg. The output statement `c.println("The cost of the shirt is " + 75.99);` will result in the real number 75.99 being converted to a string, then joined onto the text message so that the line of output displayed will be the string "The cost of the shirt is 75.99".

- ▶ Open the class called **InputReals** which you saved to disk in Chapter 4.
- ▶ Add a line of code after the existing output statements, to display the sum of the 2 numbers, as shown below. (Ignore any formatting that you added previously.)

```
c.println("The first real number entered was " + num1);  
c.println("The second real number entered was " + num2);  
c.println("The sum of the 2 numbers is " + num1 + num2);
```

← Add this line

- ▶ Save, then Run your program, using the real numbers 2.5 and 7.5 when prompted.
- ▶ Write down the 3 lines of output that your program produces.

Do not write down the input prompts - they are not output.

.....
.....
.....

- ▶ Is the sum of the 2 numbers correct? YES / NO

NOTE: Values in a single output statement are joined together using plus signs.
(Not added, unless you more specifically instruct the computer to add them.)

- ▶ Change the last output statement by using brackets around `num1 + num2` to make the computer add the numbers first and then convert the sum to a String:

```
c.println("The sum of the 2 numbers is " + (num1 + num2));
```

- ▶ Save, then Run your program, again using the numbers 2.5 and 7.5 when prompted.
- ▶ Write down the 3 lines of output that your program now produces:

.....

- ▶ Is the sum now correct? YES / NO
- ▶ Run the program using other pairs of real numbers and just write down their sum:
 2.375 + 7645.2345
 17.99 + 432.8

5.2 Subtraction using the - operator and multiplication using the * operator

Continue using the **InputReals** class, or open it if you do not still have it on your screen.

- ▶ Add another output statement to subtract the 2 numbers and to display their difference with a suitable message. Remember to use brackets in the compound output statement to make the computer subtract first, before joining the answer to the text string.
- ▶ Save and Run the program, inputting the numbers 954.53266 and 1072.876
- ▶ Write down the output your program produces:

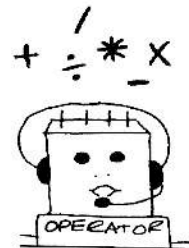
.....

- ▶ Add another output statement to multiply the 2 numbers, using the * operator, and to display their product, with a suitable message.

- ▶ Save and Run the program, inputting the numbers 7.5 and 0.75

- ▶ Write down the output your program produces:

.....



- ▶ **Format** the output of the sum to **1 decimal place**, using **2 separate statements**, as follows:

```
c.print("The sum of the 2 numbers is ");  
c.println((num1 + num2),0,1);
```

If you try to put formatting numbers in a compound output statement, together with a text message, you will get an error.

- ▶ Format the output of the difference and the product as well, also to 1 decimal place.

- Save and Run the program again using the following input values and write down the output produced by your program:

20.375 and 98.7638

.....

As well as addition(+), subtraction(-) and multiplication(*), there are a few more operators that can be used when doing arithmetic.

5.3 Slash (/) division with INTEGERS

The / operator is used for division. This can give unexpected results with integers.



- Start a new HSA Application called **Division**.
- Add a heading and a blank line after it:


```
c.println("INTEGER DIVISION");
c.println();
```
- Declare 2 integer type variables with identifiers `num1` and `num2`.
- Prompt for integer values to be input via the keyboard into each of these variables, and store the input values in `num1` and `num2` respectively.
- Leave a blank line, then type in the following statement:


```
c.println("The answer to " + num1 + " divided by " + num2 + " is " + {num1/num2});
```
- Make sure that you have left spaces inside the quote marks so that the output is readable when you run the program.
- Save the program and Run it repeatedly. Input the following 2 integers each time, and write down the output produced by `num1/num2`:

Input these 2 integers

Result obtained by `num1/num2`

7	and	5
2	and	5
2	and	2
3	and	4
15	and	4
....	and
....	and
....	and

Use values
of your own
here

- Check the answers - do you think that they are correct? YES / NO

If both numbers being divided are integers, the result will be truncated to the nearest whole number. This operation is called DIV in other programming languages.

5.4 The modulus (%) operator

This operator is used with **integers** to find the **remainder** after the first number has been divided by the second. The modulus operator is also known as **mod**.

- ▶ Start a new HSA Application called **Remainder**.
- ▶ Add a heading and a blank line after it:

```
c.println("FINDING THE REMAINDER");  
c.println();
```
- ▶ Declare 2 integer type variables with identifiers `num1` and `num2`.
- ▶ Prompt for integer values to be input via the keyboard into each of these variables, and store the input values in `num1` and `num2` respectively.
- ▶ Leave a blank line, then type in the following statement:

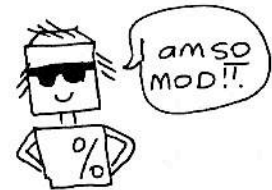
```
c.println("The remainder when " + num1 + " is divided by " + num2  
        + " is " + (num1%num2));
```
- ▶ Make sure that you have left spaces inside the quote marks so that the output is readable when you run the program.
- ▶ Save the program and Run it repeatedly. Input the following 2 integers each time, and write down the **remainder** produced by `num1%num2`:

Input these 2 integers

7	and	5
2	and	5
2	and	2
3	and	4
0	and	4
15	and	4
....	and
....	and
....	and

Result obtained by `num1%num2`

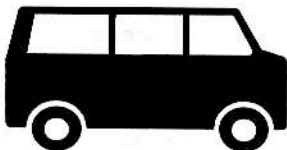
.....
.....
.....
.....
.....
.....
.....
.....
.....



Use values
of your own
here

Integer division and **modulus** are operations that are done on **integers** and produce an **integer** as a result.

Integer division and modulus can be described using a taxi scenario.



A group of people are waiting for a taxi. Each taxi can only hold 12 people and a taxi only leaves if it is full. If there are 15 people altogether, 12 people will be able to fit into one taxi so that it will be full, and 3 will be left over to wait for the next taxi to get filled up.

In order to work out how many full taxis are needed, use **integer division**.

$$\text{eg. } 15 / 12 = 1$$

The **modulus** operator is used to work out how many people will be left over.

$$\text{eg. } 15 \% 12 = 3$$

15 and 12 are
integers so **integer
division** is performed

Integer division and **modulus** are special integer operators

$$15 / 12 = 1 \quad \text{and} \quad 15 \% 12 = 3$$

At a different taxi rank, each taxi can hold 15 people. If there are 35 people waiting, 2 taxis will be filled ($15 * 2 = 30$) and there will be $35 - 30 = 5$ people left over.

ie. $35 / 15 = 2$ and $35 \% 15 = 5$

x / y where x and y are integers means "divide x by y and throw away the remainder"

$x \% y$ means "the remainder when x is divided by y "

More examples:

$$16 / 3 = 5$$

$$22 / 8 = 2$$

$$14 / 2 = 7$$

$$4 / 5 = 0$$

$$-10 / 3 = -3$$

$$9 / -4 = -2$$

$$16 \% 3 = 1$$

$$22 \% 8 = 6$$

$$14 \% 2 = 0$$

$$4 \% 5 = 4$$

$$-10 \% 3 = -1$$

$$9 \% -4 = 1$$

-	/	+	→	-
+	/	-	→	-
-	%	+	→	-
+	%	-	→	+

EXERCISE 5A

Use integer division and the modulus operator to complete these:

$$43 / 5 = \dots\dots\dots$$

$$43 \% 5 = \dots\dots\dots$$

$$63 / 9 = \dots\dots\dots$$

$$63 \% 9 = \dots\dots\dots$$

$$12 / 15 = \dots\dots\dots$$

$$12 \% 15 = \dots\dots\dots$$

$$-14 / 8 = \dots\dots\dots$$

$$-14 \% 8 = \dots\dots\dots$$

$$7 / 7 = \dots\dots\dots$$

$$7 \% 7 = \dots\dots\dots$$

$$10 / -3 = \dots\dots\dots$$

$$10 \% -3 = \dots\dots\dots$$

Integer division works with integers, and the answer is always an integer.

Modulus can work with either integer or real numbers.

Integers are positive and negative whole numbers and zero with no decimal places. They are different from **real numbers**.

Examples of integers:

-55 7 0 -1

Examples of real numbers:

2.5 -0.000879 6523.4 -77.99

In Mathematics, the set of real numbers includes the set of integers. In programming, we separate the two types of numbers because the computer deals with them in different ways.



INVESTIGATION

The modulus operator can also be used with real numbers.

- ▣ Load the program **Remainder** if you do not already have it loaded.
- ▣ Change the type of `num1` and `num2` to `double`.
- ▣ Change the input statements for `num1` and `num2` to input real numbers.
- ▣ Run the program with the following data:

Input these 2 numbers	Result obtained by <code>num1 % num2</code>
5 and 2.5
5 and 3.5
5.5 and 2
5.5 and 3.5
5 and -2.5
5 and -3.5
-5.5 and 2.5
-5.5 and 3.5

EXERCISE 5B

- 1 Complete these examples with a calculator using the following algorithm :

Algorithm to find `x % y` for integers or real numbers

1. Let **ans** = `x / y`
2. **ans2** = all the decimals removed from **ans**
3. **remainder** = `x - ans2 * y`

Example: Find `17.6 % 4`

1. **ans** = `17.6 / 4 = 4.4`
2. **ans2** = `4`
3. **remainder** = `17.6 - 4 * 4`
= `1.6`

<code>17.6 % 4</code>	=	<code>17 % 4.6</code>	=
<code>63.2 % 9</code>	=	<code>63 % 9.2</code>	=
<code>-12.1 % 15</code>	=	<code>12 % -15.1</code>	=
<code>-14.8 % 8</code>	=	<code>-14 % 8.8</code>	=
<code>7.7 % 7.7</code>	=	<code>-7.7 % -7.7</code>	=
<code>10.25 % -3</code>	=	<code>10 % -3.25</code>	=

You will find that the answers given are not entirely accurate. This is caused by the way that the real numbers are stored in memory, with accuracy errors occurring during the calculations. You possibly have experienced this when using your calculator for big calculations.

- 2 Test your answers using the program **Remainder**.

5.5 Real Division

- ▶ Load the program **Division**.
- ▶ Change your program so that it uses real numbers, declaring the variables as `double` instead of `int`, and using `readDouble` instead of `readInt`. Also change the heading to "REAL NUMBER DIVISION" and change the input prompts, if necessary.
- ▶ Save and Run the program repeatedly. Input the following 2 numbers each time, and write down the output produced by `num1/num2`:

Input these 2 numbers

7 and 5
2 and 5
2 and 2
3 and 4.1
15.0 and 4
15.4 and 0.6
.... and
.... and
.... and

Result obtained by `num1/num2`

.....
.....
.....
.....
.....
.....
.....
.....
.....

Use values
of your own
here

Some of the results are given with many decimal places.

- ▶ Format the result of the division in your program so that it produces **3 decimal places** each time.
- ▶ Run your program and write down the result of dividing each of the following:

Input these 2 numbers

123.45 and 76.54
8.99 and 1.2
43.5 and 0.875

Result obtained by `num1/num2`

.....
.....
.....

As long as one of the numbers in the division sum is declared as a real number, the result will be real. The answer to `4 / 2` is 2.0 if either 4 or 2 are stored in a real number variable.

- ▶ Change your program so that the first variable is an integer and the second variable is a real number. Change the corresponding `readDouble()` method to `readInt()` and change the input prompts, so that one asks for an integer and the other asks for a real number to be typed in.
- ▶ Run the program and enter the integer 17 and the real number 2.5 when prompted.
- ▶ Write down the output:

- ▶ Test your program with several other values and fill in the results here:

The answer to divided by is

The answer to divided by is

The answer to divided by is

5.6 Incrementing and decrementing the values of variables

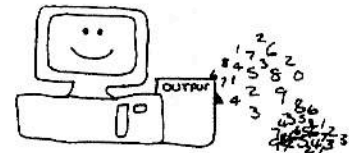
In any programming language it is useful to increase (or decrease) the value of a variable. This is called incrementing (or decrementing).

For example, if a variable `count` exists in a program to count the number of times something happens, it may be useful to increase it by 1. To do this, the code is `count = count + 1`;

- ▶ Start a new class called **Incrementing**.
- ▶ Declare 2 integer variables called `num1` and `num2`, and assign the values 7 and 10 to `num1` and `num2` respectively.

- ▶ Add output statements which display the values that are stored in the 2 variables:

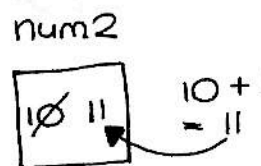
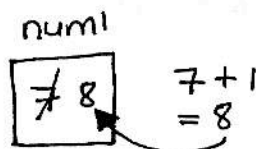
```
c.println("The first integer is " + num1);  
c.println("The second integer is " + num2);
```



- ▶ After these 2 output statements for `num1` and `num2`, add the incrementing statements on the next 2 lines:

```
num1 = num1 + 1;  
num2 = num2 + 1;
```

Read these statements like this:
The variable (`num1` or `num2`) is now assigned its old value plus 1.
ie. The previous value has been overwritten.



- ▶ Add output statements which display the values that are NOW stored in the 2 variables:

```
c.println("After incrementing:");  
c.println("The first integer is now " + num1);  
c.println("The second integer is now " + num2);
```

- ▶ Save and Run

Write down the output produced:

```
The first integer is .....  
The second integer is .....  
After incrementing:  
The first integer is now .....  
The second integer is now .....
```


Instead of typing the code `num = num + 1;` the shorter version `num++` can be used.

- ▣ Change the incrementing statements in the above program so that they both use the shorter form.
(ie. in place of the statement `num1 = num1 + 1;` type `num1++` instead.)
- ▣ Save and Run your program, checking that it works correctly.

In a similar way, the values stored in variables can be decremented (decreased) by 1, using the code `num = num - 1;` or the shorter version `num--`

Incrementing (and decrementing) will be used a great deal later on when you learn how to code programs with loops (statements that are repeated).

SUMMARY

Common **arithmetic operators** used with numbers in Java are listed in the table below, showing what operation they perform:

Operator	Arithmetic Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (used to find the Remainder)
++	Increment
--	Decrement

ORDER OF ARITHMETIC OPERATORS

The order in which arithmetic operations are carried out is important. The computer works out arithmetic expressions in this order:

FIRST	Brackets
SECOND	*, /, (both integer and real division), %
THIRD	+, -

Operations with the same importance are evaluated from left to right, as they appear in the expression.

Example 1

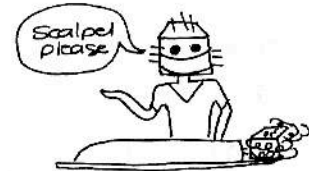
```
4 + 5 * 3 - 12 mod 5
= 4 + 15 - 2
= 19 - 2
= 17 (Integer)
```

Example 2

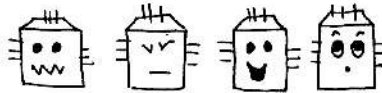
```
(10 + 4) / (25 - 2 * 11)
= 14 / (25 - 22)
= 14 / 3 (Integer Division)
= 4 (Integer)
```

Example 3

```
(10 + 4) / (24 - 2.2 * 11)
= 14 / (24 - 24.2)
= 14 / -0.2 (Real Division)
= -70.0 (Real)
```



EXERCISE 5C



- 1 Work out the answers to the following expressions, as the computer would work them out. Remember that the / and % are performed before + and -. (Show all steps in the working.)
 - a. 32 / 3
 - b. 32 % 3
 - c. 444 % 2
 - d. 27 / 5 + 27 % 5
 - e. 43 / 12 - 16 / 7
 - f. 65 / 9 + 38 % 6 - 76 % 8
 - g. 1 + 15 / 4
 - h. 8 + 5 / 7 - 3 % 4
- ▣ Type the expressions in 1.a-h in output statements in a program, then compile, save and run the program to check your answers.
eg. `c.println(32 / 3);`
- 2 Work out the answers to each of the following, showing each step. Also say whether the final result will be **real** or **integer**: (Remember that / division depends on the types of numbers on either side of it (integer or real).)
 - a. 19 - 15 / 2 + 1
 - b. 7 * 3 + 7 % 3
 - c. 8 - (27 / 6) * 2
 - d. 19 - (26 % 3) * 3 + 10.0 / 4

e. $32 / 4 + 27 / 10 / 2 - 16 / 7$

f. $64.2 + 5 \% 7$

g. $(7 + 2 * 5) / (64 - 6 * 10 + 2) + 35 / 5 * 2 + 15 \% 11$

KWIKWIZ

1. Why are the brackets around `num1 + num2` necessary in the statement:
`c.println("The sum of the 2 numbers is " + (num1 + num2));`

2. Explain why $5 / 2 = 2$ and $5.0 / 2 = 2.5$ in Java.
3. Write down the algorithm to determine the modulus `%` of 2 real numbers `x` and `y`:

4. Why are the answers given by the computer to some calculations innaccurate?
5. What is the shortened form of `num = num + 1` and `num = num - 1` ?
6. List the 5 operators that we have studied in this chapter
7. Write down the order in which these operators will be evaluated:

8. Explain why the output of `c.println(20 % 4)` is zero.

9. a. In what order will the computer evaluate this arithmetic expression?
 $15 + (7 * 2) \% 4 - 13 / 8 + 1$

 b. What will the answer be?

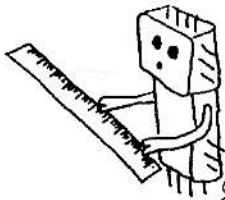


You have used several methods to do with input and output on the Console output screen (named `c` in all the programs we have used). Some of the methods you have learned are **print**, **setColor**, **readInt**.

- Write down a few other methods from the Console class that you have used in your programs:

All of these methods are from the HSA Console class, and because `c` is the name of the Console output window (or screen) object in each program, each method is joined on to the object `c` using dot notation. So the code in the Java programs we have created is `c.print()` or `c.setColor(...)` or `c.readInt()`.

5.7 The `length()` method of the String class



Not all methods belong to the Console class. There are many other classes built in to the Java language that have methods belonging to them. One of these is the String class. The String class has many useful methods, most of which you will learn about at a later stage.

One very useful method of the String class is the **length()** method, which returns the number of characters in a String.

Because this method is not of the Console class, we do not use `c.length()`. Instead, whatever Strings variables are declared in a program can use dot notation to join the String variable identifier to the **length()** method. For example if `name`, `word` and `sentence` are declared as strings, then `name.length()`, or `word.length()`, or `sentence.length()` will find the number of characters in each string.

The **length()** method has no parameters in brackets, but Java requires that any method name must be followed by an opening and closing bracket.

It is sometimes useful to know how many characters there are in a String. The **length()** method is used to do this.

The **length()** method is used with **String** data, not numeric data (of type `int` or `double`).

```
c.println( "ETHNIK AFRIKA".length() );
```

will output the integer 13

The blank space also counts as one character.

```
c.println( "Hi there!".length() );
```

will output the integer 9

- Start a new HSA Console Application Boilerplate called **StringLengths** and type in the following code:

```
String name1, name2, message;
```

```
name1 = "Jane";
```

```
name2 = "Sam Nkomo";
```

```
message = "Computers are great fun!";
```

```
c.println (name1 + " has " + name1.length () + " characters in it ");
```

```
c.println ();
```


```
c.println (name2 + " has " + name2.length () + " characters in it ");
```

```
c.println ();
```

```
c.println (message + " has " + message.length () + " characters in it ");
```

The **length()** method counts the number of characters in a String.

- ▶ Save and Run this program and view the output.
- ▶ Count the number of characters in each String, including spaces, to see how the computer applies the **length()** method.
- ▶ Copy the output into the blocks below.



- ▶ Change the code to allow a user to input 2 names and a message.
- ▶ Run the program and check that the output is correct.
- ▶ Print the program and file it. Also print the output, if you are able to.

EXERCISE 5D

- 1 Start a new class called **ThreeWords**. Add code which asks a user to enter 3 words which must be stored in 3 separate variables. Determine the total number of characters in the 3 words together. Display each word, with the number of characters in each next to it, and the total number of characters below this, with a suitable message.
- 2
 - a. Start a new class called **NumberOfDigits**. Add code which asks for any integer to be typed in. Store this number in a variable of type **String**. Display the number and the quantity of digits in the number. Eg. The number 76465 has 5 digits in it.
 - b. Add code to declare a variable called **wholeNum** of type **int**. Prompt for any integer to be typed in, and store this in **wholeNum**.

Can you determine and display the number of digits in **wholeNum**?

- You cannot use the **length()** method to do this! Why not?
- You have not learned enough skills at this stage to solve this problem.

5.7.1 Lining up output in Columns

It is often difficult to position data neatly in a tabular form in columns and rows. The **length()** method helps to line up this kind of output, and can be used together with field widths.

- ▶ Start a new Application called **Marks**.
- ▶ Type in the following code, making sure you use `print` and `println` as shown:

```
c.println( "12345678901234567890" );
c.print( "Refiloe");
c.println( 82, 10 );
c.print( "Anna");
c.println( 67, 10 );
c.print( "Timothy");
c.println( 75, 10 );
```

These statements display pupils' names and the marks they obtained in a Computer Studies test.

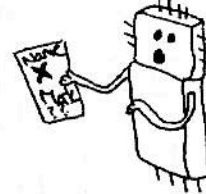
- Run this program, and view the output.

The pupils' marks are not lined up, because the pupils' names are all different lengths.

- Edit the program so that the statements are as follows:

```
c.println( "123456789012345678901234567890" );
c.print( "Refiloe");
c.println( 82, 25 - "Refiloe".length() );
c.print( "Anna");
c.println( 67, 25 - "Anna".length() );
c.print( "Timothy");
c.println( 75, 25 - "Timothy".length() );
```

Add more column nos.



This tells the computer to display the output ending in column 25, first subtracting the number of columns already used by each name. Because the exact amount of characters that have already been displayed from the left margin is taken away each time, all the marks will end in column 25.

Eg. "Timothy" has 7 characters and displaying this name has used 7 columns, so to get his mark (75) to end in column 25, the computer uses a field width of 18, because:

$$\begin{aligned} & 25 - \text{"Timothy"}.length() \\ &= 25 - (\text{number of characters in "Timothy"}) \\ &= 25 - 7 \\ &= 18. \end{aligned}$$

- Run the program again, and view the output. The marks should be lined up.
- Change the program so that any 3 pupils' full names and their marks can be entered when the program is run.
- Check that the output is well lined up for any names and marks that are typed in.
- Save and print the program and the output, if possible.

EXERCISE 5E

- Start a new class called **ShoppingList**. Add code to the program to display a shopping list such as the one shown below. Use field widths and the **length()** method to make sure that the figures are properly aligned.

```
123456789012345678901234567890
SHOPPING LIST
```

```
Box of disks      43.95
Computer          4999.99
Printer           899.50
MS Works          450.00
```

```
TOTAL
```

The computer must calculate the total amount and display it to 2 decimal places.

- Start a new class called **Stock**. Type in code to display the following output in 3 columns, using the column numbers indicated for the first letter of each heading. Make sure that column 1 is left justified and columns 2 and 3 are right justified.

Col 1	Col 23	Col 41
ITEM	ITEM CODE	QUANTITY
T-shirt	537	150
Dress	7231	80
Earrings	7732	500

5.7.2 Centring Text in the middle of a row

Say you wanted to display a heading such as "SALE! ALL GOODS HALF PRICE TODAY" in the centre of a row on the output screen.

If you know how many characters are in the String that you want to display, then you can display half the number of characters on the one side of the middle of the screen, and the other half of the characters on the other side of the middle!

The text output screen has 80 columns across it, therefore the middle of the screen is at column 40.

- ▣ Start a new class called **CentreDisplay**.
- ▣ Type in code which prompts for any sentence to be input. Store this in a variable called **sent**.
- ▣ Add a statement to display a blank line.
- ▣ Add a statement to display 80 column numbers across the screen:

```
c.println("1234567890123456789012345678901234567890123456789012345678901234567890");
```

- ▣ Type in the following statement to display the sentence in the centre of the next row:

```
c.print( " ", 40 - sent.length()/2 );  
c.println( sent );
```

Because Strings are left justified in fields, we display a space in a field of (40 minus half the number of characters in the String), and then display the String stored in **sent**.

EXERCISE 5F

- 1 You have been asked to create a flyer to advertise a stall at a flea market.

- ▣ Start a new class called **Flyer**.
- ▣ Add statements so that the following output is achieved:

```
/////////  
                ETHNIK  
                AFRIKA  
            Ethnic gear at low prices  
/////////  
WEDNESDAY                      Fleamarket  
SATURDAY                      Bruma Lake  
SUNDAY                        Stand 43  
/////////
```

- ▣ Make sure that the first 3 lines are centred on the pattern, the day names are left justified and the address details right justified.
- 2 Design a menu, using as many computer terms as you can. Centre the items and space them out well so that the menu is easy to read. On the next page are some ideas to get you started. Add your own ideas ...

MEGABYTE MENU

STARTERS

Hot keys - hot Mexican chilli chips

Data - be surprised and try this starter

MAIN COURSE

Root directory - vegetarian platter

RAM - grilled to perfection in BBQ sauce

DRINKS

Tabs - Caffeine-free, sugar-free cola

Copy - imitation fruit juice

- ▣ Use the class name **ComputerMenu**.
- ▣ Print your program, file it and print the output on hard copy.

5.8 More on Declaring variables and storing values in variables

Declaring a variable means naming the variable and stating the variable's data type. In Java, the data type is put in front of the variable identifier. Eg: `int num;` which means that the variable `num` stores data of type integer.

Variables can be declared at the beginning of a method, or they can be declared the first time they are used in a program. It is useful to declare all the variables at the beginning of a method as it gives a summary of the variables that are going to be used in the method. Later on we will discuss advantages of declaring variables as they are used.

- ▣ Start a new class called **VarValues**.

- ▣ Add the following code:

```
// Declaring variables
int num1;
int num2;
int num3;

// Assigning values to variables
num1 = 7;
num2 = 15;

// Getting a value into a variable from the keyboard
c.print("Type in any integer and <Enter>: ");
num3 = c.readInt();
```

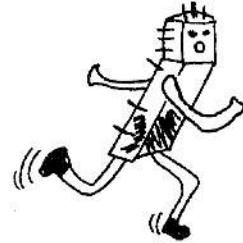
The comments have been put in to show what is happening in each section of the program

Variables can get values by assignment or using keyboard input.

```
// Displaying output
c.println();
c.println("The first integer is " + num1);
c.println("The second integer is " + num2);
c.println("The third integer is " + num3);

// Declaring and assigning in one statement
int sum = num1 + num2 + num3;

// More output
c.println("The sum of the 3 integers is " + sum);
```



- ▣ Save and Run, entering any integer when asked (eg: 17)

NOTE that the variable **sum** is declared and assigned the first time it is used (in the second last line of the program). This is acceptable in Java, however, it is often more useful to declare most variables at the beginning of a class, where they can be easily found.

- ▣ Add another output statement that declares and assigns the product of the 3 integers to a variable called **product**.
- ▣ Also add an output statement to display this product, with a meaningful message.

EXERCISE 5G

- ▣ Add code to programs for each of the following. Use the class name next to each question number. Save them, then Run them, and print the programs when they are correct.

1 Eggs

A farmer supplies the local shop with a quantity of eggs in a large cardboard box. The shop pays the farmer by the dozen. Write a program to display how many full dozen eggs are in the box. The farmer must type in the number of eggs in the box.

Write down the answer if there are 376 eggs in the box: full dozen.

2 Date100

A year is entered in the form YYYY eg. 1977. Use the / and % operators to separate the first 2 digits (19) from the last 2 (77). Store these 2 numbers in intermediate variables. Add 1 onto the first (so that 19 becomes 20), and then display the result together with the second, to give the date in 100 years' time (2077).

There is a much easier way to calculate the year number 100 years from now.

Write it down