CHAPTER 4 : VARIABLES

In this Chapter...

You will learn about:	You will learn how to:
input, output, variables, data types: String, integer, real/double, char; Methods: readString, readLine, readInt, readDouble, readChar, getChar; overwriting, characteristics of a variable, rules and conventions for identifier names	declare and name variables, input data into variables, recognise valid identifier names

A friend of yours and you have decided to write a Java program to print labels for all your school books. You want the label to include your name and the subject's name. It should look something like this:

```
**********
Tina
Geography
**********
```

It would be more useful to use a person's full name but that presents complications that will be dealt with later in this chapter.

- Start a new HSA Boilerplate Application called **MakeLabels**.
- Add your name and today's date as a comment.
- Insert the following code, typing it exactly as shown:


```
c.println("**********");
c.println("Tina");
c.println("Geography");
c.println("**********");
```

- Save and run the program and check that it displays the label on the screen.
- If your computer is connected to a printer which is switched on and online, click on the Print button in the output window to print the label on paper.
- In order to create a label for History, the program needs to be changed. Change the correct line in the program so that Subject: "History" is displayed on the screen under the name "Tina".
- If you are able to, print this new label on paper.

It is going to be very tedious to keep changing the program for each subject. A variable is needed to solve the problem.

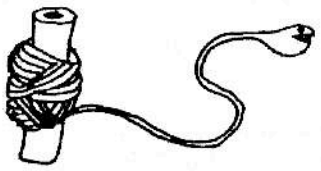
So far, we have only coded programs that displayed output. In the above example, instead of typing in each subject's name, we design one label with a "placeholder" for the subject name. When the program is Run, the user is asked to enter the subject, then the relevant label is displayed. This typing in of data when the program is Run is called input and it makes programs interactive. These types of programs accept input, and then act upon this input to produce output.

The data (in this case, the subject's name) needs to be remembered by the program. A "placeholder" called a variable is used for this purpose.

4.1 Variables

Variables can be thought of as boxes in memory. Each variable (or box) can only store one item of data of one data type. (Later on we will find out about more complex types of variables.) Each variable has a data type. The data types that we have dealt with so far are integers, real numbers and Strings. Each of these types takes up different amount of space in memory, so it is important to tell Java what data type a variable will be, so that it knows how much space in memory to allocate for each variable.

- Open the program **MakeLabels**.



- Change the program so that the main method is as follows:

```
public static void main (String[] args)
{
    c = new Console ();

    String subject;

    c.println("Type in the subject and <Enter>");

    subject = c.readString();
    c.println();
    c.println ("*****************************");
    c.println ("Tina");
    c.println (subject);
    c.println ("*****************************");

} // main method
} // MakeLabels class
```

Declare the variable subject as being of type String

Tell the user to enter a subject. This is called a prompt.

Input the name of the subject

Do not put quote marks around the variable

- Save and then run the program and test it by typing in different subjects when prompted. Print out the labels if you can.

You do NOT have to type in the quote marks when you input a String value into a variable - the computer automatically adds them for you.

Each of the subjects that you have entered are examples of **Strings**.

- Run the program twice more and type in "12345" and "@#$%^&*" as input. What happened?

. . . . . .

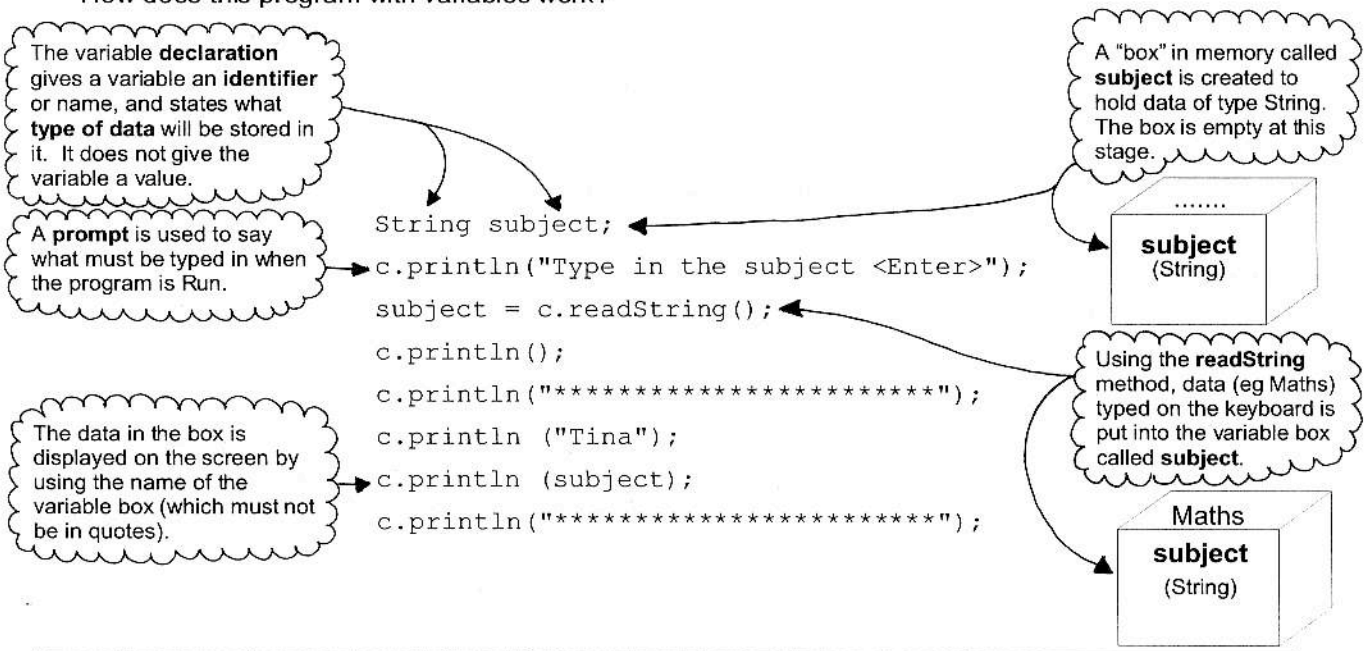
"12345" and "@#$%^&*" are also examples of Strings.

A String in Java is a collection of characters and can be made up of letters, numbers and symbols. Strings are enclosed in double quotes.

Examples of Strings: "Geography",
"This is a string";
"123456";
"876-221";
"#$%^";
" Hello there, how are you?";
"How R U 2?";
"Any collection of characters in double quotes is a String".

Inputting values into variables enables the same program to be used to produce different results. In this example, the same program is Run, but different names are used in the output because different values are input into the variables each time the program is Run.

How does this program with variables work?



The program would be more useful if it could a print a label using any person's name as well as any subject.

▣ Change the program so that it declares another variable called **name** and prints the value stored in the variable in the label.

Your program should look like this:

```
    public static void main (String[] args)
    {
        c = new Console ();

        String subject, name;

        c.println("Type in the person's name and <Enter>");
        name = c.readString();

        c.println("Type in the subject and <Enter>");
        subject = c.readString();

        c.println();
        c.println ("******************************");
        c.println (name);

        c.println (subject);
        c.println ("******************************");

    } // main method
} // MakeLabels class
```

The variable name can be declared in the same statement, because it also stores data of type String.

The **readString** method is used again to read data typed on the keyboard into the variable box called **name**.

The variable's identifier is used, so that whatever is stored in name will be displayed.

- Save and Run the program.
- What are the variables called in this program? .....
- What **data type** are both the variables declared as? .....
- Write down the statements that input the data that the user has typed into the variable.

  .....

  .....

Your friend has decided to change the format of the labels to :

```
**************************
Name : Tina
Subject : Geography
**************************
```

- Change the lines that display the name and subject to:

```
c.println ("Name : " + name);
c.println ("Subject : " + subject);
```

A variable can be joined/concatenated to a String using a + (plus) sign.

It would be better if you could print your first name and surname in a label, so the label appeared as:

```
**************************
Name : Tina Sanders
Subject : Geography
**************************
```

- Run your existing program and write down what happens when you try to enter two names (such as your first name and surname) or a subject that has more than one word in it (such as Computer Studies): .....

  .....

  .....

The `readString()` method does not allow a String of more than one word to be entered. It puts the second word of the String in the next variable to be read, if there is one.

To solve the problem, use the method `readLine()` instead of `readString()`.

- Change your program as indicated in bold on the following page:

```
public static void main (String[] args)
{
   c = new Console ();

   String subject, name;

   c.println("Type in the name and <Enter>");
   name = c.readLine();
   c.println("Type in the subject and <Enter>");
   subject = c.readString();

   c.println();
   c.println ("*****************************");
   c.println ("Name : " + name);
   c.println ("Subject : " + subject);
   c.println ("*****************************");


} // main method
```

Both the `readLine()` and the `readString()` methods read **String** values in from the keyboard.

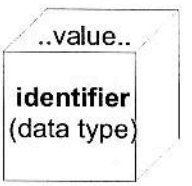
The `readLine()` method allows more than one word in a String, while the `readString()` method only allows a single word in a String.

- ▣ Run the program and check that it works correctly. Test the program with different names such as "Sue Ellen van der Merwe".
- ▣ Change the code so that a subject with more than one word in it can also be typed in when the program is Run.
- ▣ When the program works perfectly, print it and place it in your file.

Remember to use the `readLine()` method and not the `readString()` method when you want to store more than one word, or a group of characters with spaces, in a String variable.

Every variable has 3 characteristics:

- an identifier or name;
- a data type, stating what type of data can be stored in it;
- a value that is stored in it.



In a program, every variable that is used has to be

- **declared** (created) - given an identifier and a data type
- **given a value** - this is either **assigned** (see chapter 5) or **read in from the keyboard when the program is Run**

The **MakeLabels** program uses two variables. The 3 characteristics of these variables are:

variable identifier	name	subject
data type	String	String
value stored	user's input	user's input

The String data type means that any collection of letters, numbers and symbols can be stored in this variable.

EXERCISE 4A

1 Code a Java program which asks for any word to be entered, and then displays a message using the word that was typed in. Follow the steps outlined below:

☐ Start a new class called **InOutWord**

☐ Add your name and today's date as a comment.

☐ In the main method, after the new Console object (c) has been created, type in the following code:

```
String word;

c.println("Type in any word, then <Enter> ");
word = c.readString();

c.println();
c.println("The word that was input is "+ word);
```

The variable `word` is **declared.**

This is called an **input prompt.**

A value is **read in** from the keyboard and **stored** in the variable `word`.

A blank line is **displayed.**

Leave a space before the quote mark.

The message followed by the value stored in the variable `word` is **displayed** on the screen.

Items in a single output statement are **joined together** using plus signs to form a compound output statement.

☐ Save the program and Run it. Type in any word when prompted.

☐ Write down the one line of output (after the blank line): .....

☐ Run the program again and this time, type the word **"computer"** when asked for any word. Write down the output: .....

☐ Answer the following questions:

- How many variables are used in this program? .....
- Write down the variable's identifier .....
- What data type is being stored? .....
- How many variable storage boxes will be used in memory .....

Draw the box(es) here, showing the identifier and data type and the current value stored: →

2 Code a new interactive Java program using the class name **SchoolInfo** to input a person's full name, the name of the school they go to, and the grade they are currently in. Display the information that the user has entered on 3 lines, one below the other, near the centre of the output screen.

KWIKWIZ

1. What does it mean if a program is interactive? .....

2. We work with many computer systems every day. Each system has input and output. Consider the following systems and write down an example of input and output for each system.

   ATM (Automatic Teller Machine) .....

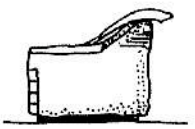
   POS (Point of Sale) i.e. the check out till at a supermarket .....

   Washing Machine .....

3. Is a program of any use if it does not produce output? .....

4. Name some other devices that a program can output to besides a screen.

   .....



5. Name other devices that a program can get input from besides a keyboard.

   .....

6. Name the three characteristics of a variable. .....

7. Why is it important to give a variable a type? .....

8. What is the difference between the methods `readString()` and `readLine()`? .....

9. What is the difference between an input prompt and output - both appear on the output screen? .....

4.2 Inputting data of other types

In the previous section you learned how to input data of the String data type. To do this, the readString() method was used. Numeric data such as whole numbers and real numbers are stored in the computer's memory in different size boxes to Strings. Therefore other methods are required for the input of other data types.

4.2.1 Input of integer values

An integer is a positive or negative whole number, or zero.
Examples of integers: 67, -7657, 0, -1, 5, 62150

- Study the section of code from the **InputRect** class, listed below:

```
c = new Console ();
int length, breadth;
c.println("RECTANGLE PROGRAM");
c.print("Type in the length of the rectangle and <Enter>: ");
length = c.readInt();
c.print("Type in the breadth of the rectangle and <Enter>: ");
breadth = c.readInt();
c.setColor( Color.green );
c.fillRect( 100, 100, length, breadth );
```

Two variables are declared

A value is read into length

A value is read into breadth

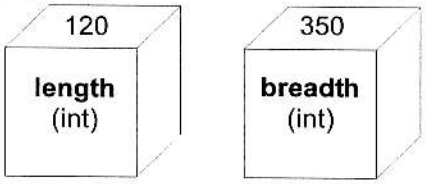
- What are the names (identifiers) of the two variables used in this code? .....
- Write down the **data type** that these variables are declared as .....
- What does the program do? .....
  .....
- Type in the program using the class name **InputRect**.
- Save and Run, inputting **500 for length** and **100 for breadth** when asked to type in the length and breadth of the rectangle.
- If you have no errors and the program runs, what is the output? .....
  .....
- Close the output window, then Run the program again, and this time type in **120 for length** and **350 for breadth**.
- Do you get a different shaped rectangle to the previous one? Yes / No

So now we have a program that will display a rectangle of any dimensions!!

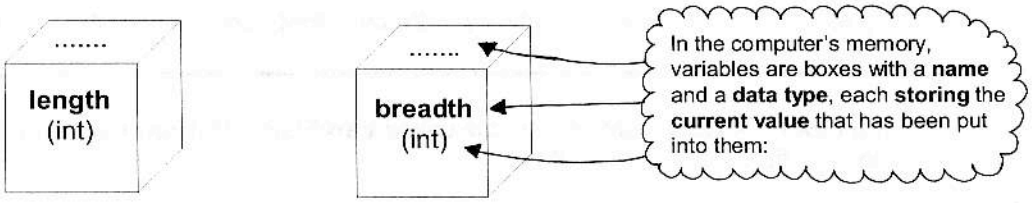
This program uses 2 variables. The characteristics of these 2 variables are listed below:

variable name	length	breadth
data type	int	int
value stored	120	350

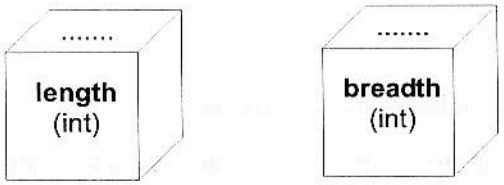
int stands for integer, which means that these variables can only store whole numbers (positive or negative or 0).



- Run the program again, and this time type in **453 for length** and **99 for breadth.**
- What values will now be stored in the variables' boxes in memory? Fill in these values.



- Run the program again, typing in values that will result in a **fairly small square.**
- Write the values you used in the variables' boxes:



- Run the program again, typing in values that will result in a **large square.**
- Write down the value that you used for both **length** and **breadth** . . . . . .

Inputting values into variables enables the same program to be used to produce different results. In this example, the same program is Run, but different rectangles are output because different values are input into the variables.

- Change the program so that the first 2 values in `c.fillRect(100,100,lenght,breadth);` are also declared as variables and their values are read in.

- What happens if you **input real numbers** when you Run the program? Try this and see, by running the program and typing in the value 123.45 when asked for the length: . . . . . .
  Why do you think that this happens? . . . . . .
  . . . . . .

The names or labels given to variables, such as **num1** or **firstName**, are called **identifiers.**

Rules for Variable names/identifiers

- Identifiers can contain numbers, but may not start with a number.
- Spaces are not allowed in a variable identifier.
- No special characters (such as %, @, *, -, ") may be used in an identifier.
- Reserved words (keywords that have a special meaning in Java) may not be used. (See the Appendix for a list of keywords.)

By convention,

- identifiers in Java start with lowercase letters.
- the identifier should reflect the contents of the variable.
- any identifier that contains more than one word must have the words joined together, and the first letter of each word (after the first) is a capital letter - for example, instead of **num1** we could have used the identifier **firstRealNumber.** (This is not a rule, but a convention to make identifiers more readable.)

The following are valid variable identifiers: num2, firstName, fieldWidth, ch

The following are NOT valid variable identifiers: 2num, first name, FieldWidth, char, tel-num

- Start a new HSA Application Boilerplate called **VaryField**. This program will use a variable to change the field width of the output.

- Add your name and today's date as a comment.

- Type in one line of code to display the number 999 in a field of 10 spaces on the first row of the output screen:

```
c.println( 999, 10);
```

```
. . . . . 999
```

- Save and Run your program, checking that it works correctly.

Instead of using 10 as the field width, we will change the 10 to a variable called **fieldWidth** so that we can vary the position of the number on the output screen.

- Change the 10 in your code to the identifier **fieldWidth**:
  c.println(999,fieldWidth);

- Run the program, and write down the error message that you get .....
  .....

A variable has to be **1) declared** and **2) given a value** before it can be used in a program.

- Before your one line of code, type in the following:

  - a line of code to declare the variable **fieldWidth** so that it can store integer values;
  - another line of code just after this to prompt for an integer value to be typed in for the field width;
  - a third line of code to read in the integer value and store it in the variable **fieldWidth**.

- Save and Run your program, typing in the value 25 when prompted for the field width.

- If your program has no errors, the number 999 should be displayed a bit further across the screen than when a field width of 10 was used.

- Check that your code is similar to the lines shown below, correcting any errors you may have:

```
int fieldWidth;
c.print("Type in an integer value for the field width and <Enter> ");
fieldWidth = c.readInt();
c.println(999,fieldWidth);
```

- Run your program again and type in the value 80 when prompted for a field width. Where about on the output screen is the number 999 positioned? .....

- Run your program again and type in a larger value such as 200 when prompted for a field width.

- Run your program again and type in the value 1000 when prompted for a field width.
- What happens to the number 999 when large values are used for the field width? .....
  .....
- Why does this occur? .....
  .....
- Answer the following questions:

- How many variables are used in this program? .....
- Write down the name(s) of the variable(s) .....
- What data type is being stored? .....
- How many variable storage boxes will be used in memory .....

Draw the box(es) here, showing the name and data type and the current value stored:

4.2.2 Input of Real number values

- Start a new class called **InputReals** which will ask for 2 real numbers to be input, and then displays these numbers with decimals.

A real number is a number with a decimal point, a whole number before the decimal point and a fractional number after the decimal point. Eg 54.876

- Add your name and today's date as a comment.
- Add code in the program to declare 2 real number variables, as follows:

```
double num1, num2;
```

In Java, the data type which stores real numbers is called **double**.

- In the same way as you asked for a name to be entered in the previous program, add an input prompt asking for <u>one</u> real number to be typed in.
- Get this real number from the keyboard and store it in the first variable, num1.

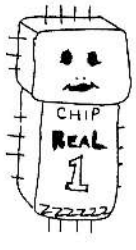
Use `num1 = c.readDouble();`

- Add another input prompt asking for another real number to be typed in.

Use Copy and Paste, then make the necessary changes.

- Get this real number from the keyboard and store it in the second variable, num2
- Display a blank line, using c.println();
- Display both of the numbers, with suitable messages, as follows:

```
c.println("The first real number entered was " + num1);
c.println("The second real number entered was " + num2);
```



- Save, then Run your program, using the real numbers 2.5 and 7.5 when prompted.

- Write down the 2 lines of output that your program produces.

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

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

- Close the output window, then Run your program again, using the real numbers 762.9634125 and -0.3475125 when prompted.

- **Format** the output of the first real number **to 3 decimal places**, separating the one output statement into **2 statements**, as follows:

```
c.print("The first real number entered was ");
c.println(num1,0,3);
```

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

- Also format the output of the second real number to 3 decimal places.

- Save, then Run your program, using the same real numbers 762.9634125 and -0.3475125 when prompted.

- Write down the 2 lines of output: . . . . . .
. . . . . .

- Why are there only 2 lines of output when there are now 4 output statements (each of the previous 2 output statements was separated into 2)? . . . . . .
. . . . . .

- Add a statement at the end of your code to display a blank line.

- Add the following line of code to see the error that occurs when formatting numbers are used in a compound output statement:

```
c.println ("The cost is $" + 123.7567, 0, 2);
```

- Write down the error message that is given in the status bar at the bottom of the screen: . . . .
. . . . . .

The error message is stating that Java does not have a method that enables you to display a String using field widths and decimal places.

- Separate this output statement into 2 separate output statements - with the text message in one, and the real number formatted to 2 decimal places in the next - and ensure that the output of these 2 statements will appear on one line.

- Save and Run the program, checking that it works correctly. (Input any 2 real numbers when prompted.) Print out the program and file it in your file.

EXERCISE 4B

☐ Start a new HSA Boilerplate Application for each of the following problems, using the class name given next to each number. Code programs for each following the specifications given. Save each program onto disk with the same name. Print your programs when they work correctly and file them in your file.

1 **PersonalDetails**
This program will ask for a person's first name, surname, telephone number, year of birth and height in metres to be typed in. The details will then be displayed out, with suitable text messages.

Type in code to do the following:

☐ Declare 5 variables (the first name and the surname must be entered separately): the first 3 variables must be of data type String, the year of birth variable will store integers, and the height variable will store real numbers.

☐ Prompt for the 5 details to be entered, being user-friendly, so that when the program is Run, it is clear what must be typed in.

☐ Read the input values into the appropriate variables (using read*Type*() methods).

☐ Clear the screen using `c.clear();`

☐ Display a heading: "PERSONAL DETAILS"

☐ Display another blank line.

☐ Then display the details as follows (in the order specified):
- The subheading "NAME: " followed by the first name and surname on one line with a space between the names;
- The subheading "YEAR OF BIRTH:" followed by the integer year of birth
- The subheading "HEIGHT(m): " followed by the height displayed using 1 decimal place
- The subheading "TELEPHONE NUMBER: " followed by the telephone number.

☐ Save and Run your program, correcting any errors if necessary. Input your details when prompted.

☐ Check that your output is well spaced out and easy to read. Make changes to make it clearer if necessary.

☐ When you think your program is working well, Run it again and input your friends' details, checking that the output is still clear.

2 **ColoredCircles**
This program will draw one red and one blue circle, using sizes input when the program is Run.

☐ Declare 2 variables that will store the integer sizes of the diameter of each circle.

☐ Display a message stating that your program will draw one red and one blue circle.

☐ Prompt for the size of the red circle to be typed in - suggest an integer between 100 and 300.

☐ Store this integer value in the appropriate variable.

- Prompt for the size of the blue circle to be typed in - suggest an integer between 50 and 150.
- Store this integer value in the appropriate variable.
- Type in the code to draw the red circle near the left hand side of the screen and the blue circle near the right hand side of the screen, using the variables for the sizes of the circles. (Eg. for the red circle: `c.drawOval(100, 200, redSize, redSize);` where `redSize` is the identifier of the variable that stores the size of the red circle.)
- Save and Run your program, inputting the two integer values for the sizes when prompted.
- Run your program several times inputting different sizes each time.
- Write down the sizes you typed in to get the two circles to overlap each other:
  Size for red circle: . . . . . Size of blue circle: . . . . . .
- To make the circles more exciting, create a new color for each circle using the statement:
```
Color col = new Color(red,green,blue);
c.setColor(col);
```
red, green and blue must be declared as variables and values read in from the keyboard.

3 **Prices.**
This program will ask for 3 products and their prices to be input, then will leave a blank line and will display the products and prices in a well lined up list, with a heading.

- Declare 3 String variables and 3 real number variables.
- Type in code to
  - prompt for 3 product names to be typed in, and the price of each product also to be typed in;
  - store the input values in each of the variables
  - display a blank line
  - display a heading in capital letters "SHOPPING LIST"
  - display each product and its price next to it, with the 3 items and prices well lined up, using field widths.
- Save and Run your program, and input the following data to check that your program works properly:

Potatoes	12.99
Beans	3.85
Sweetcorn	8.15

- Run your program again with different data and make sure that the output is still well lined up.

It is important that your programs work for ANY data - not just the data used to test the program.

4.2.3 Input of character values

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

```
char ch1, ch2, ch3;
c.println ("Type in any three characters on the keyboard");
c.println ("Press <Enter> after each.");
ch1 = c.readChar ();
ch2 = c.readChar ();
ch3 = c.readChar ();
c.println ();
c.print ("Together these 3 letters spell: " );
c.println (ch1 + ch2 +  ch3);
```

- Save and Run the program, and when prompted, input the 3 letters c, a, t pressing the <Enter> key after each one.

- Write down the output that this program produces (after the blank line): .....

.....

.....

Unfortunately, characters in Java can be somewhat tricky at times!

Every character on the keyboard has a unique code number, so that the computer will know what key has been pressed

(A = 65, B = 66, the <Enter> key = 10, etc).

Because the 3 characters in this program are displayed in a separate output statement, without being joined onto any text message, they are each converted into the code numbers that the computer uses, then these code numbers are added together.

So in this program,
'c' is taken as the first character and converted into its code number (99), then
the <Enter> key is taken as the second character and converted into its code number (10), then
'a' is taken as the first character and converted into its code number (97).

99 + 10 + 97 = 206 which is the number displayed on the output screen.

- Change the 2 output statements into a single output statement:

```
c.print ("Together these 3 letters spell: " + ch1 + ch2 +  ch3);
```

- Run the program and enter the same 3 characters: c, a, t pressing the <Enter> key after each one.

- Write down the output that this program produces (after the blank line): .....

.....

.....

Because <Enter> is a character, 'c' is stored in the variable `ch1`, <Enter> is stored in the variable `ch2` and 'a' is stored in the variable `ch3`.

This time the actual characters are displayed (the 'c' and the 'a' anyway, one below the other - the <Enter> key is not displayed, but is carried out, causing the 'a' to be one the line below the 'c'.)

If the characters `ch1 + ch2 + ch3` are not enclosed in brackets, but simply joined onto the text message with a plus sign, the computer converts the characters into strings and displays them. If `(ch1 + ch2 + ch3)` are enclosed in brackets, and then joined on to the text message, they are first converted into their code values and added, as before, then changed to a string and displayed.

- Change the 2 output statements into a single output statement:

```
c.print ("Together these 3 letters spell: " + (ch1 + ch2 + ch3));
```

- Run the program and enter the same 3 characters: c, a, t pressing the <Enter> key after each one.

- Write down the output that this program produces (after the blank line): .....

.....

There are 2 ways of making characters "behave":

1) Use the `getChar ();` method instead of `readChar ();` which stores a character in a variable <u>without</u> the <Enter> key having to be pressed.

- Change the code in your program, as follows:
  - Put 2 slashes (//) in front of `c.println ("Press <Enter> after each.");` so that it becomes a comment:
    `//c.println ("Press <Enter> after each.");`
  - change `readChar ();` to `getChar ();` in the 3 assignment statements
  - change the output statement to a single compound statement, with <u>no</u> brackets around `ch1 + ch2 + ch3`:

```
c.println ("Together these 3 letters spell: " + ch1 + ch2 + ch3);
```

- Save and Run the program, and when prompted, input the 3 letters c, a, t. (Do NOT press the <Enter> key after any of the characters.)

- Write down the output that this program produces (after the blank line):

.....

2) Use the **String** data type instead of **char**

> In Java, all strings are in double quotes (for example, "Hello"), while characters (of the **char** data type) are in single quotes (for example, 'a')
>
> However, a single character can also be a string. So "Y" is a valid String.

- Open the **InputChars** program if you do not still have it on the screen. Move the cursor to the beginning of the first line of the program, and use Replace to replace all occurrences of **InputChars** with **InputChars2**.

- Close the Replace dialog box when all 3 occurrences of InputChars have been changed.

- From the File menu, choose Save As and Save this program as **InputChars2**.

- Change the code in the program on the following page so that it becomes:

```
*     String ch1, ch2, ch3;
      c.println ("Type in any three characters on the keyboard");
      c.println ("Press <Enter> after each.");
*     ch1 = c.readString ();
*     ch2 = c.readString ();
*     ch3 = c.readString ();
      c.println ();
      c.println ("Together these 3 letters spell: " + ch1 + ch2 + ch3);
```

(Use `String` instead of `char` in the lines marked with a *)

- Save and Run the program, and when prompted, input the 3 letters d, o, g pressing the <Enter> key after each one.
- Write down the output that this program produces (after the blank line):

  .....

INVESTIGATION

Using character literals such as '\t' and '\n'

- Start a new HSA Application called **CharLiterals.**
- Declare 3 **char** variables called **ch1** and **ch2** and **tab**.
- Add code to prompt for any two characters to be typed in, so that <Enter> does not have to be pressed. Store the characters that are input in **ch1** and **ch2**.
- Declare and assign the **character literal** `'\t'` to the variable tab using the statement:

  ```
  tab = '\t';
  ```

- Clear the screen.
- You have 3 variables in this program. Add an output statement to display the 3 variables next to each other, in this order: ch1, tab, ch2.
- Save and Run the program and write down the output: .....
- Add a text message at the front of the output statement:

  ```
  "The two characters are " + ch1 + tab + ch2
  ```

- Run the program again and write down the output: .....
- Explain the output: .....

  .....

  .....

  .....

  .....

KWIKWIZ

1 What are the four **data types** that you have learned about in this chapter? .....

.....

What types of values can be stored in each type? .....

.....

What are the methods to read in data for each type? .....

.....

2 Can you read an integer value into a variable declared to store real numbers? In other words if the user entered 5 in the program below, would an error be produced? .....

```
double num;

c.println("Enter an number");
num = c.readDouble();
```

3 Can you read a character into a String? .....

A String into a character? .....

A real into a string? .....

To make sure of your answers check them by writing programs.

4 Write down the rules for identifier names .....

.....

.....

.....

5 Which of the following are valid identifier names? Give reasons if an identifier is invalid.

`MyFirstProgram` .....

`myFirstProgram` .....

`1program` .....

`Program1` .....

`Program 1` .....

`Percentage num` .....

`PercentageNum` .....

`%num` .....

`num%` .....

`num 5` .....

6 Why will the following statement cause an error?

```
c.println ("The total is" + total, 0, 2);
```

.....

.....

What type would you expect the variable **total** to be declared as? .....

7 Write down all the problems that are be caused by the program below and explain them in your own words. .....

.....

.....

```
ch1 = c.readChar ();
ch2 = c.readChar ();
ch3 = c.readChar ();
c.println ();
c.print ("Together these 3 letters spell: " );
c.println (ch1 + ch2 +  ch3);
```

Is it a good idea to use `c.readChar ();`? Are there better methods? Explain your answer.

.....

.....

EXERCISE 4C

Open the **PersonalDetails** class which you saved in a previous exercise.

- ▶ Change the code in this program so that the person's full name can be entered in one statement instead of entering the first name and the surname in 2 separate statements.
- ▶ Add code so that the program also prompts for the letter of the class the person is in this year (for example the 'C' in Grade 10C). This must be stored in a **char** type variable called **classLetter**.
- ▶ Add an output statement so that the class letter will be displayed after the subheading "GRADE 10 ".
- ▶ Save and Run the program, checking that it works properly.

SUMMARY

When a value is read into a variable from the keyboard, one of the following methods is used (depending on the type of data being read in):

`readInt(), readDouble(), readChar(), readString(), readLine()`

The syntax used is `variableIdentifier = c.readType();`

For example:
```
surname = c.readString();
```
or
```
realNum = c.readDouble();
```

Assume that the variables have already been declared

Even though there are no parameters in brackets after these methods, by convention, Java requires that empty brackets () be used after the method name.

For input of single characters, the **getChar()** method is preferable to readChar().

So far we have learned to use 4 common data types: **int, double, char** and **String.**

NOTE that the names of all data types except for String start with a lowercase letter. String has an initial capital letter, because, in Java, declaring a variable as a String is actually creating a String object from the String class, and not creating a variable of a primitive data type as when the other data types are used.

The type of data stored in variables is important, because data of different types takes up different amounts of space in memory. Although we have been drawing the memory boxes approximately the same size, in the computer's memory they are not equal in size.

▶ Fill in the missing words, showing what type of data is stored in each Java data type:

Java Data Type	Type of data stored
String	Collections of characters in . . . . . quote marks eg. "hello"
char	One single character in . . . . . quote marks eg. 'k'
int	Integers ie. . . . . numbers eg. 678
double	. . . . . numbers ie. numbers with decimals eg. 4.8765

▶ A variable is given its identifier and its data type when it is declared. For each of the following variable declarations, fill in the identifier and data type:

```
String surname;
int number;
```

Identifier: . . . . . Data Type: . . . . . .
Identifier: . . . . . Data Type: . . . . . .

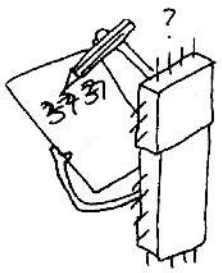
Variable Data Types

The most commonly used data types that are used when variables are declared are listed in the table below:

Data Type	Description of data that can be stored in a variable of this type
int	A whole number in the range -2.14 billion to 2.14 billion
double	A real number with a decimal point
char	A single character in single quotes
String	A group of one or more characters in double quotes
boolean	Either the value **true** or the value **false**

We will not use the **boolean** data type at the moment, however it is good to know that it exists, and you will see how it can be used later on.

4.3 Overwriting



The program below has one variable called **name.** It is of type **String.**

```
// The "Overwrite" class.
import java.awt.*;
import hsa.Console;

public class Overwrite
{
    static Console c;           // The output console

    public static void main (String[] args)
    {
        c = new Console ();
        String name;

        c.println("Type in a name and <Enter>");
*1      name = c.readString();
        c.println("Type in another name and <Enter>");
        name = c.readString();

*2      c.println("The name is " + name);

    } // main method
} // Overwrite class
```

- How many prompts are there in this program? (Remember that a prompt is a message saying what must be typed in when the program is Run) .....

There is only ONE output statement in this program: `c.println("The name is " + name);`

- What output will be displayed on the screen if the user enters "apple" at the first prompt, and then "banana" at the second prompt? .....
  .....
- What is the value that is stored in the variable **name** after the program has executed? .....
- Start a new class called **Overwrite.**
- Type in the program to check your answers.
- What output will be displayed if the line marked *2 is copied straight after the line marked *1?
  .....
- Copy this line in your program, Run it and check your answer.

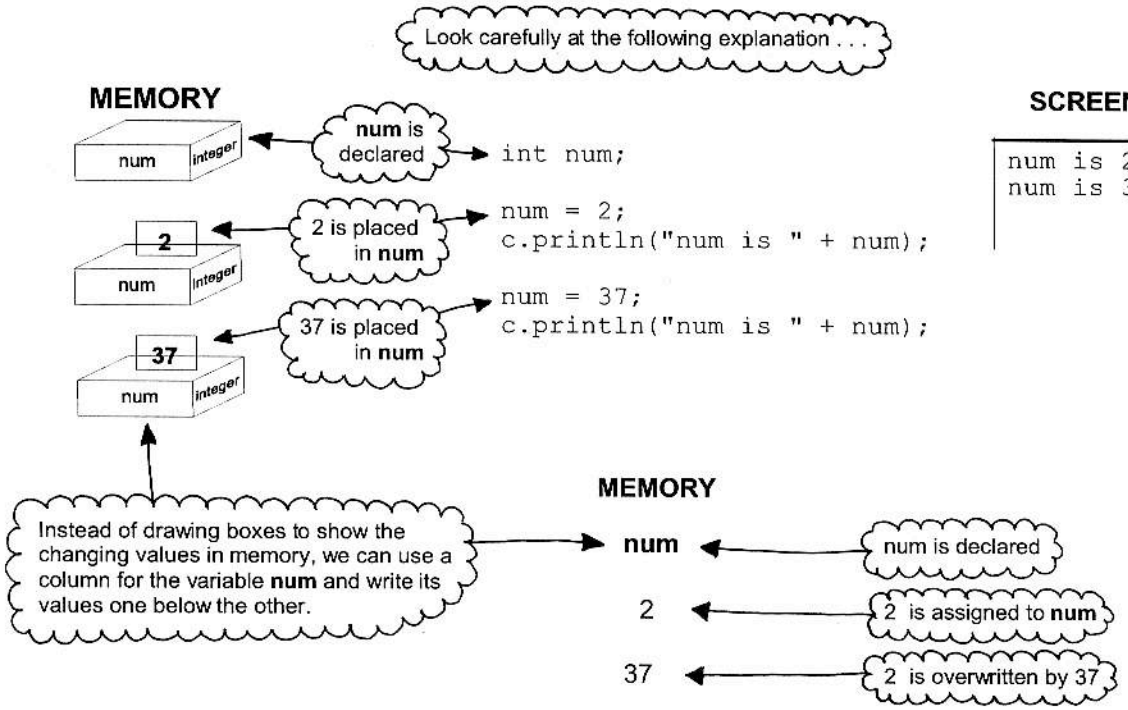
Overwriting occurs when the value in a variable is replaced by another. The original value is lost.

Instead of inputting values into variables when a program is Run, values can be assigned to variables in the program. Eg. If 2 variables called num1 and num2 of data type **int** are declared, they can be given values directly in the program, using the = sign:

```
int num1, num2;
num1 = 7;
num2 = 55;
```

Read the = signs as "gets the value of"
num1 gets the value of 7.
num2 gets the value of 55.

What happens if a variable is assigned more than one value in a program?



EXERCISE 4D

Show value will be stored in the variable **num** in memory and what will be displayed on the screen after each of the following statements has been carried out. Use the outlines provided.

a.

MEMORY

num
4
.....

```
// over_write1;
int num;
num = 4;
c.println("num is " + num);
num = 10;
c.println("num is " + num);
```

SCREEN

b.

MEMORY

num
.....
.....

```
// over_write2;
int num;
num = 4;
num = 10;
c.println("num is " + num);
```

SCREEN

c.

MEMORY

SCREEN

```
// over_write3;
int num;
num = 4;
c.println("num is " + num);
num = 10;
```

d.

MEMORY

SCREEN

```
// over_write4;
int num;
num = 4;
c.println("num is " + num);
c.print("Enter a number ");
num = c.readInt();
c.println("num is " + num);
```

The integer 7 is entered

KWIKWIZ

1. What does it mean if the value in a variable has been overwritten? .....
   .....

2. Tebello has saved a program on disk as **A:Homework**. Her sister uses the same disk and also saves a different program as **A:Homework** on the same disk. Will Tebello have her program if she prints out **A:Homework** the next morning? .....
   Explain your answer. .....
   .....

3. Does overwriting only apply to variables in memory? .....
   .....

4. Write down 2 ways of getting a value into a variable, and give an example of each:
   .....
   .....
   .....
   .....