Chapter 2: Variables and Data Types

Learning Outcomes:

- ✓ Identify the importance of variables
- ✓ Identify the importance of data types
- ✓ Implement variables in a program
- ✓ Implement a range of data types in a program
- ✓ Convert data between data types

Prerequisite Knowledge:

- ✓ Complete Chapter 1
- ✓ Be familiar with the maths operators used in Java to perform calculations (`+ - * /`)
- ✓ Be familiar with the term 'integer'
- ✓ Be familiar with computing capacity, i.e. bits

Keywords: Variable, Data Type, String, Integer, Expression

2.1 The Theory: Variables and Data Types

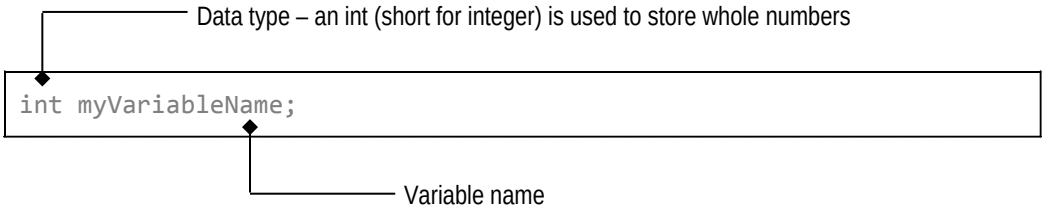
In the previous steps values were outputted directly to the command-line and were not retained in memory. To retain information, a programmer must use a variable. Using a variable allows data to be stored for later use; for example, storing the result(s) of a calculation prior to using it.

What is a Variable?

A variable is an allocation in memory that can be used to store values (such as numbers and letters) or references to other objects (reference variables are examined in Chapter 7). A good analogy might be to consider a variable similar to a labelled cardboard box – a box that stores things for later use and is labelled with a unique name so it is easily found. Variables are a fundamental part of all programming languages; they allow useful values to be stored ready for future manipulation.

What are Data Types?

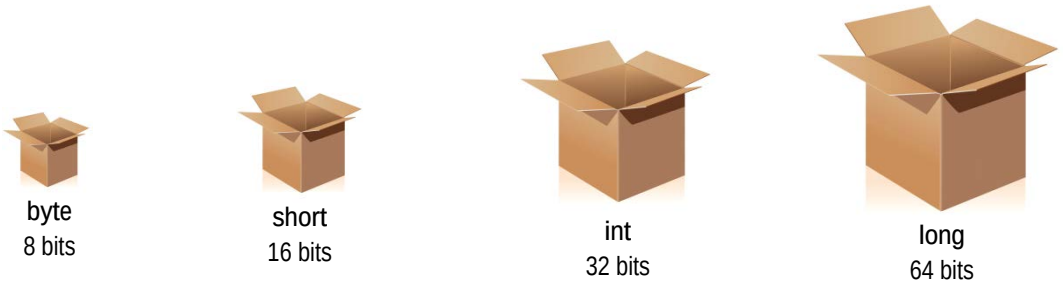
Variables must have a data type. A data type is simply a declaration of what the variable will contain; for example, 'I will hold a number' or 'I will hold letters'. Data types are important as they enable programs to be more efficient, as well as protecting their integrity; for example, by stopping letters being placed into a variable that is intended to hold numbers. In Java, variables must be declared with their data type and name before they can be used in a program. The statement declaration is show below:



Java Data Types

Any variable that stores a value is referred to as primitive. There is a selection of primitive data types available in Java, and the one to use will depend on what the variable is intended to hold. For example, if a programmer requires a variable to store a whole number, then an integer data type would be declared. However, there are also several sizes of integer available, and the one to use will depend on the approximate size of the value to be stored. For example, if the value will be less than 127, then 'short' is the most appropriate integer type to use.

The larger the bits, the larger the box!



An outline of some of the primitive data types (and their purposes) available in Java are shown in the table below:

Data Type	Number of Bits (Capacity)	Value Range	Purpose
boolean	-	true or false	Simple true or false condition
char	16	-	Single unicode character (i.e. a letter)
String	-	-	Text values
double	64	-	Default choice for decimal values
Integer (whole number)			
byte	8	-128 to 127	Tiny integer
short	16	-32768 to 32767	Small integer
int	32	-2147483648 to 2147483647	Medium integer
long	64	Massive to massive	Big integer

String with a Capital?

String has a capital S; this is not a mistake. Java is a case-sensitive language and all primitives start with lower-case letters. String, however, (although it can be used similarly to other primitives) is in fact not a true primitive; it is a class. There are some special advantages to the `String` class which will be examined later on.

Reserved Words

Variables can be named almost anything, providing that the name does not begin with a number and is not a keyword already reserved by the Java language. Some of the reserved keywords that are not permitted are shown in the list below:

```
public    private    protected   abstract   final
static    volatile   if          else       do
while     switch     case        default    for
break     class      extends     continue   implements
import    interface  new         package    super
this      catch      finally     try        throw
void      return     goto        enum       const
boolean   char       double      byte       short
```


2.2 Practical: Simple Variables

My Variable Program

An assignment statement is used to assign a value to a variable; in Java the equals symbols (=) is used to achieve this. Variables can be declared with or without an assigned value; see `x` and `y` in the program below. Be aware that integers have a default value of 0 and not null (empty).

Activity 2.1

Write a program that calculates the area of a 9 cm square and outputs the answer to the command-line. Use variables to store the length, width and area values.

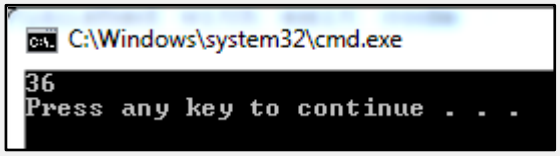
Try the code

```
public class MyVariableProgram{

	public static void main(String[] args){
		int x = 5;
		int y;
		y = 4;
		System.out.println((x+y)*4);
	}
}
```

Step by step

1. Type out the above code
2. Save the program as MyVariableProgram.java
3. Compile the program
4. Run the MyVariableProgram.class file
5. Check the results of the program



There are many expressions that can be assigned to a variable, including the value of other variables (of the same data type) and calculations. All of the examples below are legal in Java:

```
int x = (5*4)/10;
```

```
int x = 4;
int y = x;
```

```
int x = 4;
int y = 5;
int z = (x+y)/3;
```

Programming Tips!

Using naming conventions for variables is good practice, especially when working as part of a team. In Java, the following naming conventions are used:

- Start names with lower-case letters, not numbers, underscores or the dollar symbol
- Although it is legal for subsequent letters to be numbers, underscores or the dollar symbol, it is not encouraged practice so try to avoid it
- Try to name variables with meaningful, full names; avoid using cryptic abbreviations and do not use reserved keywords
- If the variable name is one word, then spell the word in all lower-case letters. If the variable name consists of more than one word, capitalise the initial letter of each subsequent word, but keep the first word in all lower-case, for example 'myVariable'

Variables and Concatenation

The example below uses a `String` variable to store the programmer's name and a `short` variable to store the programmer's age. The example below also demonstrates the use of concatenation (the `+` symbol) to join literal text to the variable value, before outputting both to the command-line. Notice that text values are placed inside quotations, whereas numeric values are not; failing to do this will cause a compile error.

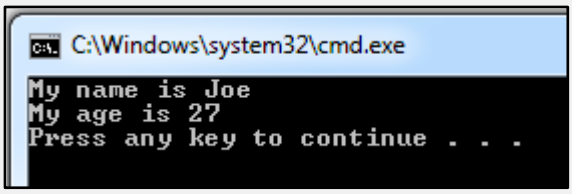
Try the code

```
public class MyVariableProgram2{

	public static void main(String[] args){
		String myName = "Joe";
		short myAge = 27;
		System.out.println("My name is " + myName);
		System.out.println("My age is " + myAge);
	}
}
```

Step by step

1. Type out the above code
2. Save the program as MyVariableProgram2.java
3. Compile the program
4. Run the MyVariableProgram2.class file
5. Check the results of the program



Activity 2.2

Write a program that has the two variables `length` and `width`, with the values 23 and 30 respectively. Use the variables to calculate the perimeter of the rectangle and output the answer to the command-line.

Activity 2.3

Write a program that calculates how many seconds there are in a single year and displays the answer to the command-line.

Use the following variables:

```
secondsPerHour
secondsPerDay
secondsPerYear
```

Activity 2.4

Write a program that has the following variables:

```
double annualRate = 3.4;
short monthsRemaining = 12;
double amountToRepay = 20000;
```

The program should calculate both the total amount to be repaid, by including the annual interest rate, and the amount to be repaid per month. Display both answers (with appropriate on-screen text) to the command-line.

2.3 Practical: Using Strings

String and its Methods

Class (non-primitive) data types, such as `String`, have their own pre-built methods that are available for programmers to manipulate the data type; a full list of these methods can be found online in the Java documentation. The `String` data type has many useful methods; a small sample of those methods is shown below.

```
String myText = "hello";
myText = myText.toUpperCase();
```

The [`toUpperCase()`] method will convert all characters in a `String` to upper case.

```
String myText = "HELLO";
myText = myText.toLowerCase();
```

The [`toLowerCase()`] method will convert all characters in a `String` to lower case.

```
String myText = "hello";
int textLength = myText.length();
```

The [`length()`] method will return the number of characters in a `String`, including spaces. Note that the returned value is an integer and thus is stored into an integer variable.

Activity 2.5

Write a program that performs the following actions, before displaying the results (with appropriate on-screen text) to the command-line:

- Displays your first name in upper-case (use the `toUpperCase()` method)
- Displays your surname in lower-case (use the `toLowerCase()` method)
- Displays the number of characters in your surname (use the `length()` method)
- Displays the sentence '*this is how to use "double quotes" in Java*'

As stated earlier, all strings must be inside double quotations for a program to compile, but what if a programmer wants to use double quotes inside of a sentence?

```
String myText = "the man said "Hello" today";
```

Unfortunately, the above code will not compile! Therefore if a programmer requires the use of double quotations inside of a sentence, then the double quotes must be excused. In Java, this can be achieved using the backslash character (`\`).

```
String myText = "the man said \"Hello\" today";
```

The above code will now compile, allowing double quotations to be used mid-sentence.

Programming Tips!

If a variable's value is unlikely to change, for example VAT or Pi, then the variable can be marked as a constant. A constant ensures no changes will be made to its value by a program. In Java, this is achieved using the keyword [`final`] before declaring the variable's data type:

```
final double Pi = 3.14;
```


2.4 Practical: Converting Data Types

Conversion Methods and Casting Data Types

Earlier examples showed how the compiler permitted the values of variables to be assigned to other variables, providing that the data types matched. However, in programming there are times when a programmer will need to convert between different data types, i.e. a string to integer. Unfortunately, code such as the code below, although it may seem logical, will in fact cause an error when compiled. The compiler will say 'a string will not fit into an integer' and that will be the end of the program.

```
String myText = "7";
int myNumber = myText;
```

```
int X = 7;
int Y = X;
```

Instead, if a programmer needs to convert between data types, they can either: use specific methods of the data type (if they are available), or use casting. The table below shows legal examples of how to convert between data types in Java:

Integer to String	
`int myNumber = 10;` `String newNumber = Integer.toString(myNumber);`	Integer is a class; one of its methods is the [`toString()`] method which can be used to convert an integer value to type string. Notice that the `myNumber` variable is placed between the brackets.
String to Integer	
`String myNumber = "23";` `int newNumber = Integer.parseInt(myNumber);`	Another useful method of the Integer class is the [`parseInt()`] method which can be used to convert a string value to an integer; however, if the string value contains letters this will cause an error.
Integer to Double	
`int myNumber = 3;` `double newNumber = (double) myNumber;`	The casting technique can be used to convert between number types. This example converts between the integer and double data types; the value of the `newNumber` variable will be 3.0.
Double to Integer	
`double myNumber = 3.999;` `int newNumber = (int) myNumber;`	This example converts between double and integer; the value of the `newNumber` variable will be 3. As integer is a whole number, any decimals will be discarded.

Activity 2.6

Write a program that practises the examples shown in the table above. Ensure that the program converts data between string, integer and double data types. Output the results to the command-line.

Programming Tips!

When converting a string to an integer, ensure that the string contains only numeric content, otherwise this will cause a run-time error. Remember, once a number is converted to type string it has no value; it is considered like a letter unless converted back.

