

Imperative Programming Part II

1



Agenda

- We've seen the basic ideas of sequence, iteration and selection.
- Now let's focus on how they are supported in Java and what we need to add to write useful programs.

2



Displaying Things

```
System.out.println("Hello world") ;
```

- We have been using this to display text on the computer screen. How does it work?

3



How does it work?

```
System.out.println("Hello world") ;
```

- *out* is another kind of *object* - a character stream object.
- A character stream is a sequence of characters, with a *source* and a *sink*.
- With *out*, the sink is connected to the computer display.

4

Department of Computer Science



println and print

```
System.out.println("Hello world") ;
```

- Display message, followed by a newline.
 - next message appears on a next line.

```
System.out.print("Hello world") ;
```

- Just display message.
 - next message appears on the same line as the last.

5

Department of Computer Science



\n

- \n is the character representation of newline.

```
System.out.print("Hello World\n") ;
```

has the same result as:

```
System.out.println("Hello World") ;
```

```
System.out.println("Hello World\n") ;
```

will result in two newlines.

6

Department of Computer Science



Displaying messages using a loop

```
// This is real Java syntax
while (true)
{
    System.out.println("Hello");
}
```

7

Department of Computer Science



Counting

- How do we display our message just 10 times?
- We obviously need to count 1 to 10, then stop.
- Can be done with a sequence ... but then, how do we display our message 100 times?
- Better with a loop and a counter! How?

8

Department of Computer Science



A counter

- We need a container to hold a counter, which can be systematically incremented.
- We will call the container a *Variable*.
- The variable can hold an integer value we can count with.

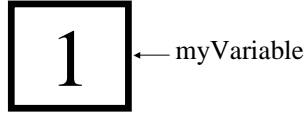
9

Department of Computer Science



Let's think variable

- A variable is a container:



- It can hold a *value*,
- and needs a *name or identifier*.
- In Java, Identifiers are written with Unicode (letters or digit) plus '_' and '\$' (although try not to use '\$').

10

Department of Computer Science



What is a value?

- Values are things like an integer or floating point number, or a character, or text...
- Values themselves are *abstract, intangible*.
- So we use *representations* of values in order to work with them.

11

Department of Computer Science



Representations

- 1, I, One, one, one, ONE - all representations.
- In the computer, integers are represented by binary numbers (e.g., 32-bit 2s complement binary numbers).

12

Department of Computer Science



Other representations

- Floating point numbers are represented by IEEE 754 format.
- Characters are represented by Unicode binary character codes.
- Text by a sequence of characters.
- Boolean by binary zero or one.

13

Department of Computer Science



Why talk about representations?

- Representations represent *finite ranges*.
 - 32-bit integer ranges from -2147483648 to 2147483647
- A variable holding an integer representation cannot *have a value* outside the range.

14

Department of Computer Science



Type

- A variable container is very specific about the kind of values it can hold.
- A *type* defines which kind of value.
- To use a variable you *have* to state what type of value it can hold.
- We typically say “a variable has a type”.

15

Department of Computer Science



Common types

- boolean - true or false
 - int - 32-bit 2's complement integer
 - long - 64-bit 2's complement integer
 - char - 16-bit unsigned Unicode character code
 - float - 32-bit floating point
 - double - 64-bit floating point
- (Check book for more detail)

16

Department of Computer Science



Just checking - What has type?

- A value has a type which defines what kind of value it is.
- A variable is associated with a type which defines what kind of values it can hold.
- To store a value (or really its representation) in a variable, the types must match.

17

Department of Computer Science



Shape?

- It may help to think of a type denoting a shape.
- Only values of the right shape can fit in a variable of a given type.

18

Department of Computer Science



Type int

- *int* is the name of the integer type (32-bit 2's complement).
- A variable named *x* of type *int* can be *declared* like this:

```
int x ;
```
- You have to *declare* a variable before you can use it.

19

Department of Computer Science



Declaration?

- Anything you name (such as a variable) must be introduced first, in order to know what is being named.
- The introduction, or *declaration*, gives the name and type of the thing being named.
- Once declared a name can be used but not before.

20

Department of Computer Science



Definition?

- With Java, *Definition* is effectively an alias for *Declaration*.
- Defining a variable is the same thing as declaring it.
- Other programming languages make an important distinction between the two.

21

Department of Computer Science



Missing declaration...

- If you use an undeclared name the Java compiler will complain!

```
compiling: T1.java
T1.java:5: Undefined variable: counter
    counter = 10 ;
    ^
1 error
```

22

Department of Computer Science



Primitive types

- The types listed earlier (and a few others) are *primitive* types.
- Why? They are directly represented by typical processors (and, hence, the JVM).
- They are the most efficient.

23

Department of Computer Science



Non-primitive types?

- Yes, they not only exist but will be very important.
- Every kind of value we use must have a type: Address, BankAccount, Date, Book,...
- Non-primitive types are abstractions, constructed from primitive types.

24

Department of Computer Science



String

- String is the type of a line of text:
 - “This is a String”
- It is a non-primitive type that is very widely used.

25

Department of Computer Science



Integer variables

- What can you do with them?
- First *declare* your variable:
 - `int myInt ;`
- What is the value of this variable?
- It hasn't got one - you *must* give it one before you can use it.

26

Department of Computer Science



Initialising

```
int myInt = 10 ;
```

- Declare `myInt` *and* give it an initial value.
- Always, ALWAYS, *initialise* a variable.
- The Java compiler will make sure you do.

27

Department of Computer Science



10?

- 10 is a *literal* value of type `int`.
- All primitive types have literal values that can be used directly in a program.
- 3.141 is a floating point literal of type `double`.
- `true` and `false` are boolean literals.

28

Department of Computer Science



More initialising

- `double d = 1.23456789 ;`
- `float f = 1.234F ;`
- `boolean b = false ;`
- `char c = 'a' ;`
- `int x = 0xff ;`
- `String s = "Hello" ;`
(Many more examples in book.)

29

Department of Computer Science



Integers

- Type `int` (32-bit). Can be
 - Decimal:
 - `int d = -10;`
 - Hexadecimal: start with `0x`
 - `int h = 0xffed;`
 - Octal: start with `0`
 - `int o = 0777;`
- Type `long` (64-bit)
 - `Long l = 0777l;`

30

Department of Computer Science



Floating point numbers

- Double (64-bit)
 - double d = -23.456;
 - double dd = -23.456d;
- Float (32-bit)
 - float f = -23.456f;

31

Department of Computer Science



Changing a variable's value

- A variable is changed by an *assignment expression*.

```
x = 20 ;
```

- The value 20 (or really its representation) is stored into the variable container.
- The old value is overwritten and lost.

32

Department of Computer Science



= OR =

- Note that we have now used = for *two* things.

```
int x = 5 ;  
x = 20 ;
```

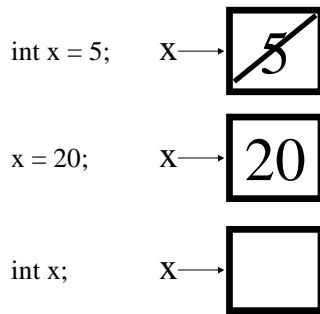
- Initialisation v. assignment.
- Subtle but different. Why?

33

Department of Computer Science



Initialisation vs. assignment



34

Department of Computer Science



State

- The state of a program is given by:
 - the Java Virtual Machine
 - a collection of variables
- The basic idea of computation is to transform the initial state to the final state.
- Each program instruction clicks the state forward one step.

35

Department of Computer Science



Wrong state?

- A computation can fail if any invalid state is reached (e.g., a variable has the wrong value).
- A typical computation may proceed through billions of states...

36

Department of Computer Science



Operators and expressions

- An operator applies an operation to values!
- +, -, /, *

```
x = 2 + 3 ;  
y = 3.2 * 2.4 ;
```

- We can combine variables, operators and literals to write *expressions*.

37

Department of Computer Science



Comparison

- There are also *boolean operators* to compare values:

```
<, >, <=, >=, ==, !=
```

```
boolean b1 = (x < 5) ;  
boolean b2 = (y == 6) ;
```

38

Department of Computer Science



Precedence

- How do you know the meaning of:
 $2 + 3 * 5 / 8$
- You use *precedence rules* – these determine which operators are applied first.
- *High precedence* operators are applied before *low precedence*.

39

Department of Computer Science



Use brackets

- We can bracket the sub-expressions to make the meaning clear:

$$2 + ((3*5) / 8)$$

40

Department of Computer Science 

Lots of operators

- See the book for the full list!!
- Understand the difference between unary and binary operators.
- Check the precedence table.

41

Department of Computer Science 

Types and operators

- A type determines exactly which operators can be applied to a value.
- No other operators can be applied.

$$x = 2 \# 3 ;$$

- Meaningless as # is not an operator.
- Won't compile.

42

Department of Computer Science 

So a type is?

- A type defines
 - the set of values belonging to the type.
 - the set of operations that can be applied to the values.
- In our programs, values of a type are given concrete (and finite) representations.

43

Department of Computer Science



Interesting...

- Given assignment and operators we can write:

$x = x + 1 ;$

- Mathematicians panic now...
- But, of course, we are not writing a mathematical formula.
- This is a program *statement*.

44

Department of Computer Science



++ (and --)

$x = x + 1 ; x = x - 1 ;$

- Increment or decrement a variable.
- Can use the ++ or -- operators:

$x++ ;$

$x-- ;$

- Or $x += 1 ; x -= 1 ;$

45

Department of Computer Science



Oh, yes...

- This all started as we wanted a counter for our program.
- We now have the bits, let's put them together.

46

Department of Computer Science



Counting

```
int counter = 0 ; // We count from zero
while (counter < 10)
{
    System.out.println("Hello") ;
    counter++ ;
}

    Done!!
```

47

Department of Computer Science



Result...

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello

    Hello is printed 10 times.
```

48

Department of Computer Science



A bit more...

```
int counter = 0 ; // We count from zero
while (counter < 10)
{
    System.out.print("Hello ");
    System.out.println(counter) ;
    counter++ ;
}
```

49

Department of Computer Science



Result...

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
```

Programmers always count from zero!
Zero is a number!!

50

Department of Computer Science



A bit more compact...

```
int counter = 0 ; // We count from zero
while (counter ++ < 10)
{
    System.out.println("Hello " + counter) ;
}
```

51

Department of Computer Science



Result...

Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10

Different result.
What needs to be done to
get the same result as before
(printing from 0 to 9)?

52

Department of Computer Science



Are there other kinds of loop?

- Yes!
- We have:
 - while loops
 - do loops
 - for loops

53

Department of Computer Science



While loops

- Seen them already:

```
while (condition)
{
    // Do something
}
```
- The loop body will be executed *zero* or more times.

54

Department of Computer Science



Do loops

- A variant:
do
{
 // Do something
}
while (condition) ;
- The loop body will be executed *one* or more times.

55

Department of Computer Science



For loops

- Often used for counting:
for (start ; limit ; increment/decrement)
{
 // Do something
}
- Count from start to limit by a step size.

56

Department of Computer Science



For Loop Example

```
for (int counter = 0 ; counter < 10 ; counter++)  
{  
  System.out.println("Hello " + counter) ;  
}
```

- Start at zero, then count up by one, while less than 10.

57

Department of Computer Science



Evens

```
for (int counter = 0 ; counter < 10 ; counter = counter + 2)
{
    System.out.println("Hello " + counter) ;
}
```

- Count up 0,2,4,6,8

58

Department of Computer Science



While v. For

- Q. Is a for loop a while loop in fancy dress?
- A. Yes!

- But it often gives a neater solution than a while loop, especially if counting.

59

Department of Computer Science



While, do, for - which to use?

- Many problems can be solved using any kind of loop.
- However, often one kind of loop gives a better (more elegant) solution.
- We shall return to this.

60

Department of Computer Science



Ex: Rewrite a for loop into a while loop

```
for (int counter = 0 ; counter < 10 ; counter = counter + 2)
{
    System.out.println("Hello " + counter);
}
```

61

Department of Computer Science



Remember selection?

- The if statement
if (boolean-exp) // Must have the brackets
{
 // Statement sequence
}
else ←
{
 // Statement sequence
}

Look! You can write a *else!*

62

Department of Computer Science



Short-cut?

- You can write:
if (boolean-exp)
 statement ; // No braces
else
 another-statement;
next-statement ;
- In fact, you can do the same with loops.

63

Department of Computer Science



Attention!!!

Originally write:

```
if (x > 10)
  x = 10 ; // Limit x
z = x * y ; // Use x
```

But then change:

```
if (x > 10)
{ x = 10 ; // Limit x
  y = 1 ; // and update y
}
z = x * y ; // Use x
```

Uh oh, this was meant to be executed only if x > 10...

Correct writing?

64

Department of Computer Science



Moral

Always put the braces in, even when the if statement (or loop) body only contains a single statement.

65

Department of Computer Science



Defensive programming

- Anticipate the kinds of programming errors you might make.
- Write the code in a style that prevents mistakes happening or, at least, makes them stand out.
- Code layout, indentation, use of blank space, use of compound statements all help.

66

Department of Computer Science



More selection?

- Yes.
- Check out the *switch* statement.
- Look at the conditional operator (a ternary operator).

All in the book!

67

Department of Computer Science



Statements

- We've been using this bit of jargon - let's just be clear what it means.
- A statement is a complete command terminated by a semi-colon.

```
a = b * c * d ; // A statement
```

In fact, an assignment statement.

68

Department of Computer Science



Expression?

- An expression is a sub-part of a statement:

```
1 * 2
```

```
a + b / c
```

- A full statement can be constructed from a number of expressions.

```
int a = y * (p + q) - (r / s) ;
```

69

Department of Computer Science



The Compound Statement

- All the loop bodies have statements bracketed by braces: { }
- This kind of bracketed sequence of statements is called a *compound statement*.
- A sequence of statements is a kind of statement!

70

Combining statements

- Any kind of statement, including a compound statement, can be used where a statement is expected.

```
while (x++ < 10)
{
  if (x == 2)
  {
    System.out.println("It's 2!");
  }
}
```

71

Summary

- Programs need to work with values.
- We use variables, assignment and operators.
- Variables have types.
- We can write expressions and statements.
- We can do selection and iteration.

72
