B.Sc. In Software Development. Year 3.
Applications Programming.
Data Types & Operations

LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
Department of Information Technology

Identifiers

- Just as every entity in the real world has a real name, you need to choose names for the things you will refer to in the program.
  - An identifier .
- An identifier must start with a letter, an underscore, or a dollar sign.
- An identifier cannot contain operators, such as +, -, and so on.
- An identifier cannot be a reserved word.
- An identifier cannot be `true`, `false`, or `null`.
- An identifier can be of any length.

Variables

- They are used to store data – input, output or intermediate.
- In this program, radius and area are variables of double precision floating point.
- Variables are used to represent many different types of data.
- A variable declaration involves you telling the compiler the name of the variable as well as what type of data it represents.
- Declaring a variable tells the compiler to allocate appropriate memory space for the variable based on its type.

Declaring Variables

Datatype variableName

```
int x;          // Declare x to be an
                // integer variable;
double radius;  // Declare radius to
                // be a double variable;
char a;         // Declare a to be a
                // character variable;
```

Assignment Statements

- After a variable is declared, you can assign a value to it by using an assignment statement. The syntax for such a statement is as follows:

```
Variable = expression
```

```
x = 1;             // Assign 1 to x;
radius = 1.0;      // Assign 1.0 to radius;
a = 'A';           // Assign 'A' to a;
```

Assignment Statements

- In Java, an assignment statement can be treated as an expression that evaluates to the value being assigned to the variable on the left-hand side of the assignment statement.
- For example, the following statement is correct:

```
System.out.println(x  = 1);
```

- Which is equivalent to:

```
x = 1;
System.out.println(x);
```

- The following statement is also correct:

```
i = j = k = 1;
```

- This is equivalent to:

```
i = 1;
j = 1;
k = 1;
```

Declaring & Initializing in One Step

- Variables often have initial values.
- Variables can be declared and initialised in one step.

```
int x = 1;
```

- Which is equivalent to:

```
int x;
x = 1;
```

Constants

- The values of a variable may change during the execution of a program.
- A constant represents permanent data which never changes.
- The word final is a Java keyword which means that the constant cannot be changed,

```
final datatype CONSTANTNAME = VALUE;
```

```
final double PI = 3.14159;
final int SIZE = 3;
```

Primitive Data Types

Type	Size in bits
`boolean`	
`char`	16
`byte`	8
`short`	16
`int`	32
`long`	64
`float`	32
`double`	64

Strings are not classified as primitive types – they are treated as reference types

Numerical Data Types

- Every data type has a range of values.
- The compiler allocates memory space to store each variable or constant according to its data type.
- Java has six numeric types: four for integers and two for floating point.
- Numeric operations include:
  - Addition.(+).
  - Subtraction.(-).
  - Multiplication.(*).
  - Division.(/).
  - Modulus. (%).

A word on modulus

- Very useful in programming.
- For example, an even number % 2 is always 0 and an odd number % 2 is always 1.
- Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days? You can find that day is Tuesday using the following expression:

Saturday is the 6th day in a week

A week has 7 days

`(6 + 10) % 7 is 2`

The 2nd day in a week is Tuesday

After 10 days

Number literals

- A literal is a primitive type value that appears directly in the program.
- In the following example, 34 is a literal.

```
int i = 34;
```

- An integer literal can be assigned to an integer variable as long as it can fit into the variable.
- A compilation error would be if the literal were too large for the variable to hold.
- For example, the statement

```
byte b = 1000;
```

would cause a compilation error, because 1000 cannot be stored in a variable of the byte type.

Shortcut Operators

Operator	*Example*	*Equivalent*
`+=`	`i+=8`	`i = i+8`
`-=`	`f-=8.0`	`f = f-8.0`
`*=`	`i*=8`	`i = i*8`
`/=`	`i/=8`	`i = i/8`
`%=`	`i%=8`	`i = i%8`

Increment & Decrement Operators

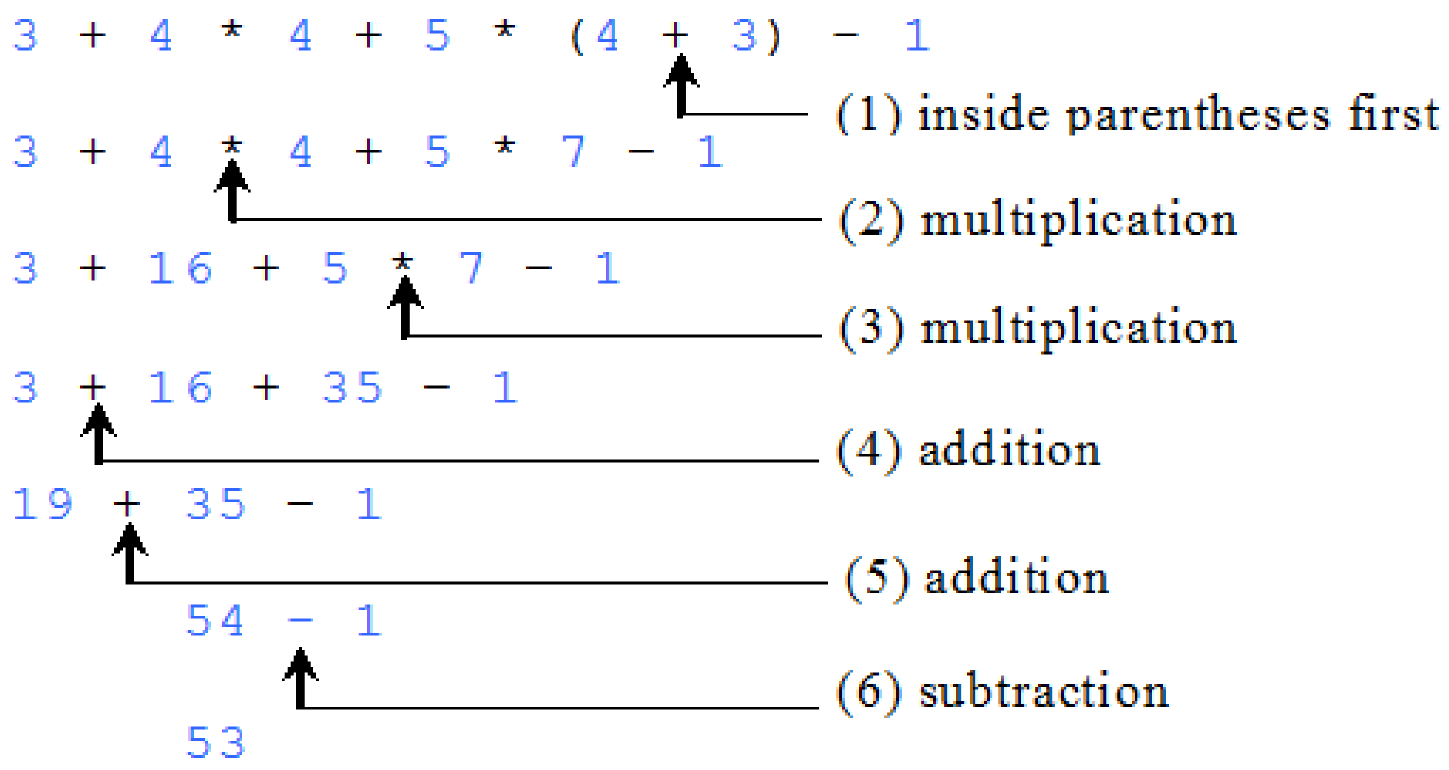
```
x = 1;
y = 1 + x++;
y = 1 + ++x;

y = 1 + x--;
y = 1 + --x;
```

Using increment and decrement operators makes expressions short, but it also makes them complex and difficult to read. Avoid using these operators in expressions that modify multiple variables, or the same variable for multiple times such as this: int k = ++i + I;

Operator Precedence

- You can safely apply the arithmetic rules for evaluating a Java expression.
  - Operations contained within brackets are evaluated first.
    - Brackets can be nested, in which case the expression in the inner brackets are evaluated first.
  - When more than one operator is used in an expression, the following operator precedence rule is used to determine the order of evaluation.
    - Multiplication, division and modulus operators are performed first. If an expression contains several multiplications, division and modulus operators they are applied from left to right.
    - Addition and subtraction are applied last. If an expression contains several addition and subtraction operators they are applied left to right.

Operator Precedence



```
3 + 4 * 4 + 5 * (4 + 3) - 1
                       (1) inside parentheses first
3 + 4 * 4 + 5 * 7 - 1
                       (2) multiplication
3 + 16 + 5 * 7 - 1
                       (3) multiplication
3 + 16 + 35 - 1
                       (4) addition
19 + 35 - 1
                       (5) addition
    54 - 1
                       (6) subtraction
    53
```

Type Casting

- Casting is the process of converting a primitive data type value into another primitive type or converting an object of one type into another type.

- Care should be taken when casting values of primitive types , because this can often result in the loss of information/precision.

- For example, when casting from an int to a double.

"TYPE CASTING"
NEXT!
SESAME STREET CASTING DEPT.
Bob
© 2011 BOB DOWNS · EVILCRAYON.COM

Type Casting

Implicit casting

`double d = 3;` (type widening)

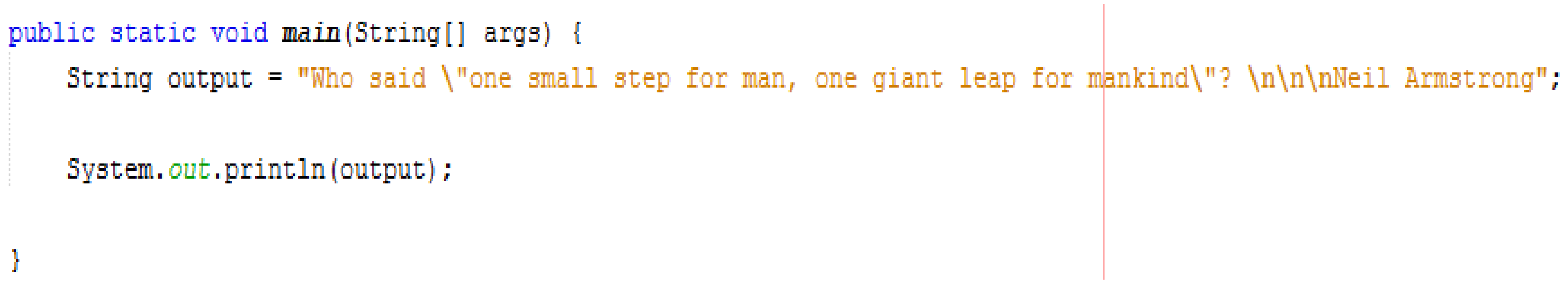
Explicit casting

`int i = (int)3.0;` (type narrowing)

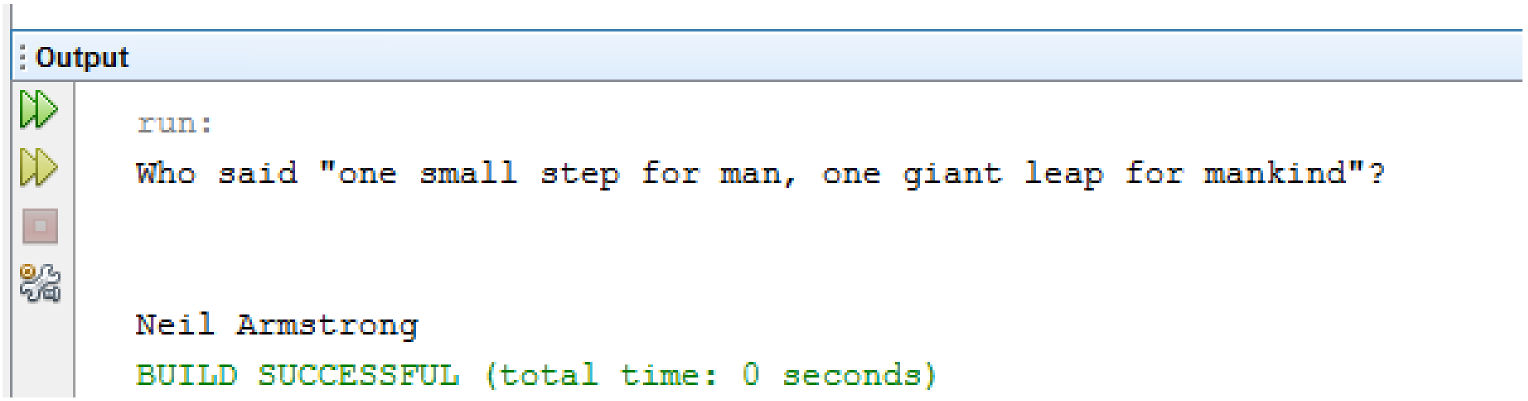
Special Characters

Escape Sequence	*Description*
`\b`	Backspace
`\t`	Tab
`\n`	Linefeed
`\r`	Carriage Return
`\\`	Backslash
`\”`	Double Quote
`\’`	Single Quote

Special Characters



```java
public static void main(String[] args) {
    String output = "Who said \"one small step for man, one giant leap for mankind\"? \n\n\nNeil Armstrong";

    System.out.println(output);

}
```



```
run:
Who said "one small step for man, one giant leap for mankind"?



Neil Armstrong
BUILD SUCCESSFUL (total time: 0 seconds)
```

The `boolean` Type and Operators

```
boolean lightsOn = true;

boolean lightsOn = false;
```

Operator	*Meaning*	*Example*
&&	And	(1 < x) && (x < 100)
||	OR	(lightsOn) || (isDayTime)
!	not	!(isStopped)

Exercise 1

Write a program (call it Reverse) that accepts as input a number in the range 101 – 199. The program should then reverse the number that the user entered.

Hint: Use the % operator to extract digits and use the / operator to remove the extracted digit

For example: 123 % 10 = 3

123 / 10 = 12

Exercise 2

Write a Java program that lets the user enter an amount in decimal format which represents an amount of money. Your program should output a report listing the minimum number of €2, €1, 50c, 20c, 10c, 5c, 2c and 1c coins in the original amount.

```
For example: €3.45 =

1 €1 coins

1 €2 coins

0 50cent coins

2 20cent coins

0 10cent coins

1 5cent coins

0 2cent coins

0 1cent coins
```

References

Joel Murach 2015, *Murach's Beginning Java with NetBeans* Mike Murach & Associates.
ISBN-13 9781890774844 (Link)

Paul Deitel 2017, *Java How To Program (Early Objects) (11th Edition).* Prentice Hall.
ISBN-13 9780134800271 (Link)

Daniel Liang 2017, *Introduction to Java Programming and Data Structures, Comprehensive Version (11th Edition). Pearson.*
ISBN-13 978-0134670942 (Link)