Programming Fundamentals

Java Tokens

Week 2| Iresh Bandara

Learning Outcomes

- Covers part of LO1 for Module
- On completion of this lecture, students are expected to be able to:
  - Classify primitive data types and literals used in java.
  - Declare, assign and initialize variables in java.
  - Use constants in places where necessary.
  - Demonstrate competence in using Java to solve problems.
  - Construct Java applications that can accept arguments from the command line.

What is a Token?

- Java program is a collection of tokens, comments and white spaces.
- 5 types of tokens.
  - Reserved Keywords
  - Identifiers
  - Literals
  - Operators
  - Separators

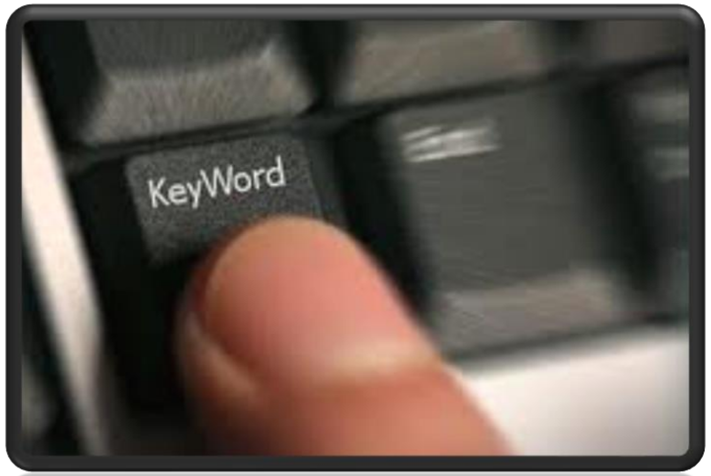
Reserved Keywords

- This include:
  - primitive data types: int, double, char, boolean, etc.
  - storage modifiers: public, private, static, final, etc.
  - control statements: if, else, switch, while, for, etc.
  - built-in constants: true, false, null

- There are about 60 reserved words total.

Reserved Keywords

- Have standard, pre-defined meanings in Java.
- Cannot be used as programmer-defined identifiers.
- Reserved words use only lowercase letters.



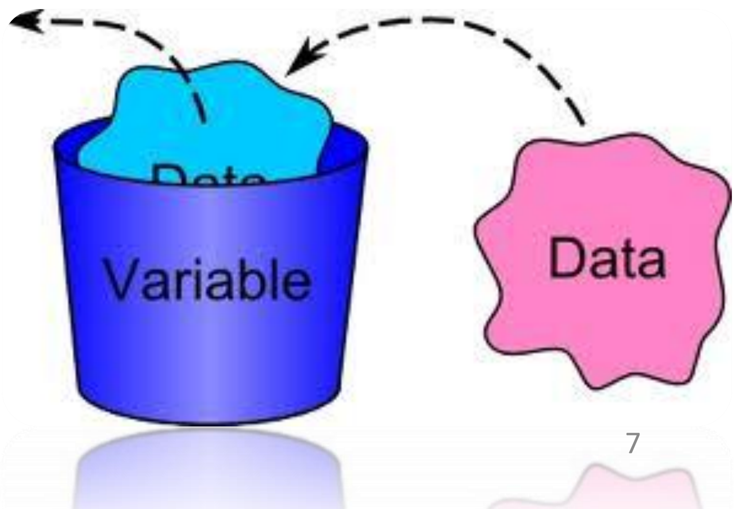
Identifiers

- Programmer-defined names that are given to various program elements.
  - i.e. variables, methods, arrays, classes, objects, packages, interfaces.
- Identifiers consist of alphabets, digits, underscore and dollar sign.
- Must not begin with a digit.
- Case sensitive.
- No white spaces.
- Can be of any length.

Variables

- An identifier that has locations in the memory that can hold values.
- Names should be descriptive to improve readability.
  - Descriptive but not too long
- It is OK to use standard short names for temporary “throwaway” variables:

  i, k, x, y, str



Task 1

What are the valid variable names?

average

height

123

pass_mark

(area)

totalMark

sum1

25th

%

What are the valid variable names?

average
height
123
pass_mark
(area)
totalMark
sum1
25th
%

Different kinds of variables

1. Instance Variables (Non-Static Fields)
    - objects store their individual states
    - their values are unique to each *instance (object)* of a class
    - fields declared without the **`static`** keyword
2. Class Variables (Static Fields)
    - fields declared with the **`static`** modifier
    - there is exactly one copy of this variable

Different kinds of variables

3. Local Variables
   - methods store its temporary state
   - only visible to the methods
   - not accessible from the rest of the class
4. Parameters
   - Used in method signature (when declaring)

both local variables and parameters are always classified as "variables" (not "fields")

Variable declaration

- All variables have four important attributes.
  - Name (identifier)
  - Data type
  - Size
  - Value
- Java is a Strongly-typed language. That means, every variable must be declared as a type.

Primitive Data Types in Java

- There are 8 primitive types: 6 of those refer to numbers.
  - 4 for integers types,
  - 2 for floating-point types,

- Character type is used for characters in Unicode encoding.
- Boolean type for true or false values.

Primitive Data Types in Java

Type	Size	(from) Range	(to)
boolean	1 bit	-	-
char	16 bits Unicode	‘\u0000' (or 0)	'\uffff' (or 65,535 inclusive)
byte	1 byte	-128	127
short	2 bytes	-32,768	32, 767
int	4 bytes	-2,147, 483,648	2,147, 483,647
long	8 bytes	-9,223,372,036,854,775, 808	9,223,372,036, 854, 775,807
float	4 bytes	3.4e-038	3.4e+038
double	8 bytes	1.7e-308	1.7e+308

Variable Initialization

- Fields (static/non-static) that are declared but not initialized will be set to a reasonable default by the compiler.

- Local variables are slightly different; the compiler never assigns a default value to an uninitialized local variable.

Default values for Data Types

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0D
char	'\u0000'
boolean	false

Literals

- There are five basic types of literals in Java.
  - Integer literals
  - Floating point literals
  - Character literals
  - String literals
  - Boolean literals

Integral literals: byte, short, int, long

- Can be expressed using,
  - Decimal: Base 10, digits 0 - 9
  - Octal: Base 8, digits 0 – 7
  - Hexadecimal: Base 16, digits 0 - 9 & A – F

- Examples

```
int decVal = 26; //The number 26,in decimal
int octVal = 032; //The number 26,in octal
int hexVal = 0x1a;//The number 26,inhexadecimal
```

The prefix 0 indicates octal, whereas 0x indicates hexadecimal

Floating point literals: float, double

- Can be expressed even in scientific notation.

```
    double d1 = 123.4;
    double d2 = 1.234e2; //same value as  d1,
but in scientific notation
    float f1 = 123.4f;
```

- double is the default, not float, therefore, special character is optional.

Character Literals: char

- Any Unicode (UTF-16) character.
- Use 'single quotes' for char literals.

```
      'A'      'x'      '3'      '?'      ' '
```

'H' is a char constant.

"H" is a String that happens to only contain a single character--it is not a char.

String Literals: String

- Any Unicode (UTF-16) character.
- Use "double quotes" for String literals.

```
“green”    “Washington, D.C. 2005”    “270-32-3456”

“$19.99”   “THE CORRECT ANSWER IS”    “2*(I+3)/j”

 “   ”           “”
```

Escape Sequences

Character	***Escape Sequence***
Backspace	\b
Tab	\t
New line (line feed)	\n
Form feed	\f
Carriage return	\r
Quotation quote (“)	\”
Single quote (‘)	\’
Backslash (\)	\\

Non printing characters, Always begins with a backward slash

Boolean Literals: boolean

- Either `true` or `false.`

- In contrast to C, in Java 0 and 1 cannot stand in for true or false. Similar to C++ bool data type.

Variable Declaration, Assignment and Initialization.

- To declare a variable in Java,

  `var_type` *list variables* ;

  eg:
  ```
  int c;
  float x, y, z;
  ```

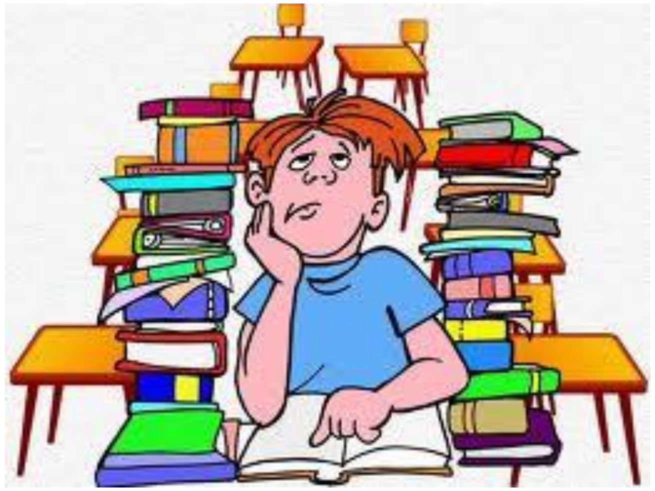
- To initialize/assign values to variables;

  eg:
  ```
  int c = 12;
  ```
  initialize

  ```
  c = 456;
  ```
  assign

Task 3

- What data types would you use to represent the following items? Write suitable declarations for variables to hold the given data.
  - The number of students in a class integer
  - The grade (a letter) attained by a student in the class char
  - The average mark in a class double
  - The distance between two points double
  - The population of a city long or int
  - The weight of a postage stamp double
  - The registration number of a car string



Mixing Data Types

```
float value1, value2, average;
value1 = 1.4f;
value2 = 2.6f;
average = (value1 + value2)/2;
```

- The above expression mixes float variables and is acceptable to the compiler.
  - `value1 + value2` is carried out as a `float` operation.
  - `2` is converted to a `float` value and the division carried out as a `float` operation.

Mixing Data Types

```
float value1, value2, average;
value1 = 1.4f;
value2 = 2.6f;
average = (value1 + value2)/2.0;
```

- By default `2.0` is a double value.
- Forces the division to be carried out as a `double` operation.
  - Hence the result is also of type double.
  - This may not be assigned to a float variable!

Automatic Type Conversion

- The rule for mixed-type expressions is:
    - "The type of each value is converted to type of the *higher* type in the expression"
- This uses the hierarchy of types:
    - `double` high
    - `float`
    - `long`
    - `int`
    - `short`
    - `byte` low

Assigning Values

- An assignment is unacceptable if the result of the expression on the right-hand side is of a higher type to that of the variable on the left-hand side.

```
int i=4;
float f = 4.0f;
f = i;
i = f;
f = 10.0;
f = 10;
```

Check these….

Type Casting

- Occasionally we want to force a calculation to take place against the implicit type conversion rules.
  - we use a cast operator to achieve this – ( )

- Casting - process of explicitly converting one data type to another between two incompatible types.

```
Type variable1 = (type) variable2;
```

```
eg: float a = 100.001f;
    int b = (int)a; // Explicit cast, the
  float could lose info
```

Issues in Type Casting

- Allows conversion from any higher type to a lower.

- But note that casting does have issues:
  - When a floating-point number is converted to an integer the fractional part gets lost by truncating it. (is chopped off)
  - The value of the cast number must not be too large for the new type.

Possible Type Casting

From	To
byte	short, char, int, long, float, double
short	int, long, float, double
int	long, float, double
long	float, double
float	double
char	int, long, float, double

Constraints in Type Casting

- Boolean any type ✗
  OR
  any type Boolean

- Smaller Larger ✓
  Larger Smaller : Loss of data

Symbolic Constants

- Constants are values that do not change while the program is running.

- Benefits,
  - Easy to modify the program
  - Easy to understand the program

- `final` – indicate no further alterations can be made

Symbolic Constant Declaration

```
final type symbolic-name =value;
```

eg:
```
final int PASS_MARK = 50;
final float PI = 3.14159;
```

Symbolic Constant constraints

- Similar to variable names BUT, written in CAPITALS
- Values must be provided at the time of declaration.
- After declaration assigning values again is ILLEGAL

Task 4

- Write a program to calculate and display the area of a circle.
- Area of a circle = πr^2 and take the value of π *as* 3.14159.
- Declare constants for the fixed parameter to improve your code.
- Your output should take the form:

"The area of a circle of radius ... units is .... units."

```
public class Main {
    public static void main(String[] args) {
        final double PI = 3.14159;  // Constant value of Pi
        double radius = 5.0;  // Radius of the circle
        double area = PI * radius * radius;  // Calculate area
        System.out.println("The area of a circle of radius " + radius + " units is " + area + " square units.");
    }
}
```

Task 4

- Write a program to calculate and display the area of a circle.
- Area of a circle = πr^2 and take the value of π *as* 3.14159.
- Declare constants for the fixed parameter to improve your code.
- Your output should take the form:

"The area of a circle of radius ... units is .... units."

Scope of Variables

- The place of declaration decides the scope of the variable
- Instance & class variables declared inside a class
- Local variables & parameters declared & used inside methods

Output Methods

- Java print methods to send results to the screen are,
  - print( ) method - Print & Wait
  - println( ) method - Print a line & move to next

Input Methods

1. Command line arguments
   - Parameters/Input provided at the time of execution.
2. Read method
   - Get values interactively through keyboard
   - Use `nextLine( )` method: reads input from keyboard as a string

Command Line Arguments

- A Java application can accept any number of arguments from the command line.
- This allows the user to specify configuration information when the application is launched.
- When an application is launched, the runtime system passes the command-line arguments to the application's main method via an array of Strings.

Example

```
public class Echo {
   public static void main (String[] args) {
            System.out.println(args[0]);
            System.out.println(args[1]);
            System.out.println(args[2]);
 } }
```

Execute command - java Echo Drink Hot Java

```
Drink
Hot
Java
```

Task 5

- Write a small JAVA program to read a day, City and temperature in Fahrenheit of a particular day as command line arguments. Then it should convert the temperature to Celsius and display the details as follows.

“Hello Colombo citizens !!“

Today is Tuesday

Temperature = 31C

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter temperature in celsius");
        double celsius = scanner.nextDouble();
        double farenheiht = (9.0/5) * celsius + 32;
        System.out.println(celsius + "°C is equivalent to " +
farenheiht + "°F");
```

Note:

- Celsius = (Fahrenheit– 32)/1.8

Read method

- To retrieve a value, you have to go through different classes.
- When you type a value in a program, to retrieve it, you can the **in** object of the **System** package:

  System.in

- After getting that value, you must first store it somewhere.
- One of the classes you can use is called **Scanner**.
- Before using the Scanner class, you must import the **java.util.Scanner** package into your program.

Read method

- This would be done by writing the following in the top section of the file:

```
import java.util.Scanner;
```

- To use the **Scanner** class to retrieve a value, use the following formula:

```
Scanner VariableName = new Scanner(System.in);
```

- After declaring a **Scanner** class, its variable is ready to receive the value.

```
VariableName = ScannerVariable.nextLine()
```

Task 6

- Rewrite the program in Task six, as if you read the values form the key board.

Summary

- Java program is a collection of tokens, comments and white space.
- There are 8 primitive types: 6 of those refer to numbers.
- There are five basic types of literals in Java.
- Variable Declaration, Assignment and Initialization is important.
- Mixing Data Types & Possible Type Casting is part of Java
- Constants are values that do not change while the program is running.
- A Java application can accept any number of arguments from the command line.

Thank you

ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY