Object Oriented Programming Practicum

ICT2132

Introduction to Java

P.H.P. Nuwan Laksiri
Department of ICT
Faculty of Technology
University of Ruhuna

Lesson 01 – Part 2

Recap

- Programming and OOP
- JAVA – History
- JAVA – Features
- JAVA – How It Works
- JAVA - Platform
- JAVA – Installation
- JAVA – Writing Your First Program
- JAVA – Keywords
- JAVA – Statements
- JAVA – White Spaces
- JAVA – Blocks
- JAVA – Identifiers
- JAVA - Comments
- JAVA – Data Types
- JAVA – Variables
- JAVA – User Input
- Exercises

Outline

- JAVA – Primitive Types
- JAVA – Reference Types
- Primitive vs Reference
- JAVA – Wrapper Classes for Primitives
- JAVA – Literals
  - Integer Literals
  - Floating point Literals
  - String Literals
  - Character Literals
  - Boolean Literals
- JAVA – Constants
- JAVA –Operators
- JAVA - Type Casting

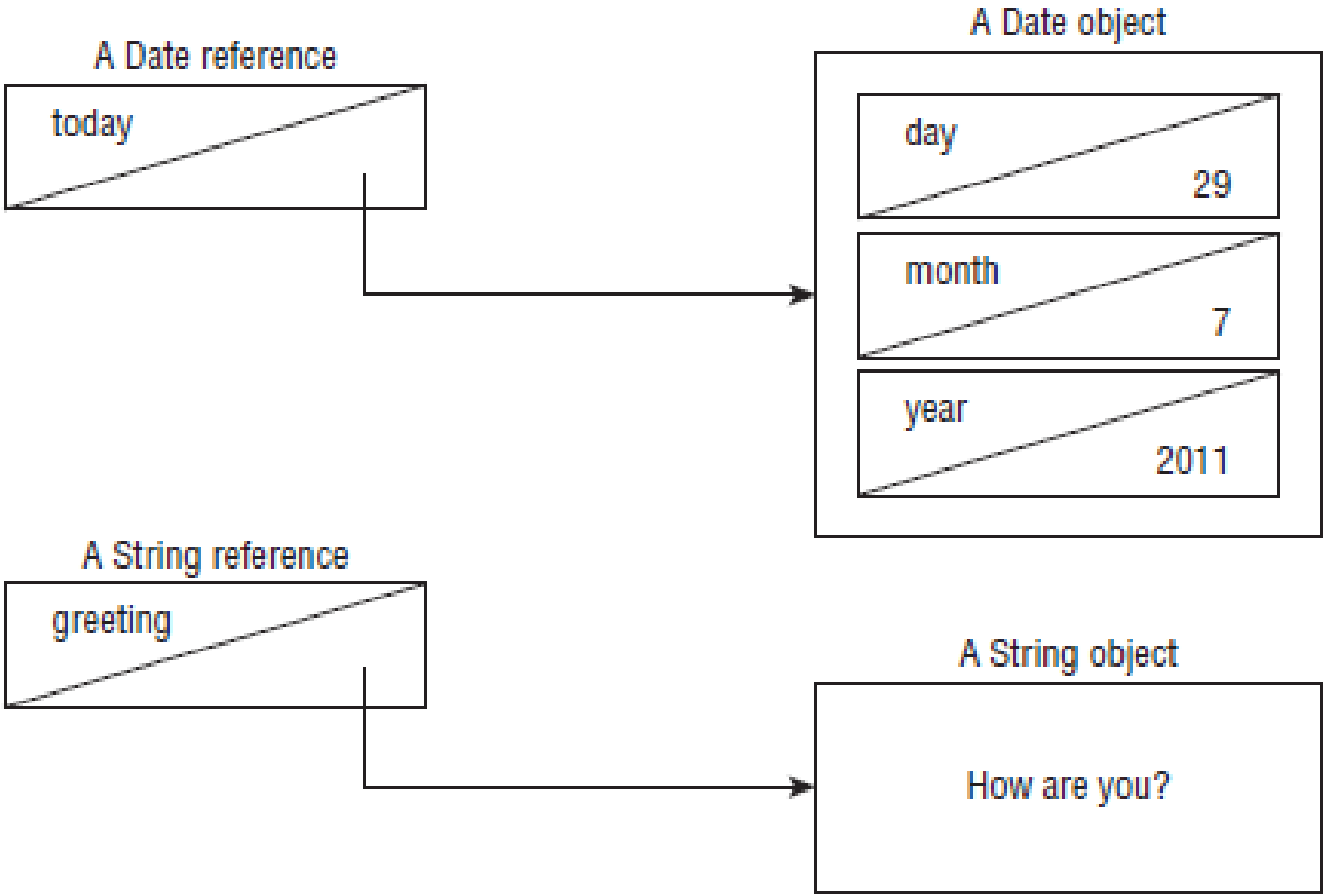
Primitive Types vs Reference Types

- **Primitive types** are the data types defined by the language itself.
- **Reference types** are types defined by classes in the Java application programming interface (API) or by classes you create rather than by the language itself.

JAVA – Primitive Types

Keyword	Type	Example
`boolean`	true or false	`true`
`byte`	8-bit integral value	`123`
`short`	16-bit integral value	`123`
`int`	32-bit integral value	`123`
`long`	64-bit integral value	`123`
`float`	32-bit floating-point value	`123.45f`
`double`	64-bit floating-point value	`123.456`
`char`	16-bit Unicode value	`'a'`

JAVA – Reference Types



Primitive Types vs Reference Types

- Reference types can be assigned null, which means they do not currently refer to an object.
  - Primitive types will give you a compiler error if you attempt to assign them null.
- Reference types can be used to call methods when they do not point to null.
  - Primitives do not have methods declared on them.
- All the primitive types have lowercase type names.
  - All classes that come with Java and designed by you are begin with uppercase.

JAVA – Wrapper Classes for Primitive Types

- Every primitive type has a corresponding class defined in the Java API class library.
  - make the primitive type look and behave like an object.

Primitive Type	*Wrapper Class*
`int`	`Integer`
`short`	`Short`
`long`	`Long`
`byte`	`Byte`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

JAVA - Type Casting

- Type casting is when you assign a value of one primitive data type to another type.
- In Java, there are two types of casting:

- **Automatic conversions/Widening Casting** (automatically) - converting a smaller type to a larger type size
  - byte -> short -> char -> int -> long -> float -> double

- **Narrowing Casting** (manually) - converting a larger type to a smaller size type
  - double -> float -> long -> int -> char -> short -> byte

JAVA - Literals

- Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables.
- Five major types of literals.
  - Integer Literals
  - Floating point Literals
  - String Literals
  - Character Literals
  - Boolean Literals

Integer Literals

- An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int.
- Integer literals can be expressed by three number systems:
  - Decimal:
    - Base 10, whose digits consists of the numbers 0 through 9; this is the number system you use every day
    - // The number 26, in decimal

```
int decVal = 26;
```

Integer Literals

- Hexadecimal:
  - Base 16, whose digits consist of the numbers 0 through 9 and the letters A through F
  - // The number 26, in hexadecimal

```
int hexVal = 0x1a;
```

- Binary:
  - Base 2, whose digits consists of the numbers 0 and 1 (you can create binary literals in Java SE 7 and later)
  - // The number 26, in binary

```
int binVal = 0b11010;
```

Floating Point Literals

- A floating-point literal is of type float if it ends with the letter F or f
- Otherwise, its type is double, and it can optionally end with the letter D or d.
- Can also be expressed using E or e (for scientific notation), F or f (32-bit float literal) and D or d (64-bit double literal).
  - double d1 = 123.4;
  - // same value as d1, but in scientific notation
  - double d2 = 1.234e2;
  - float f1 = 123.4f;

Character & String Literals

- Use 'single quotes' for char literals and "double quotes" for String literals
- Java supports a few special escape sequences for char and String literals:
  - \b (backspace)
  - \t (tab)
  - \n (line feed)
  - \f (form feed)
  - \r (carriage return)
  - \" (double quote)
  - \' (single quote)
  - \\ (backslash)

String Literals

- The set of characters in represented as String literals in Java.
- Always use "double quotes" for String literals.
- There are few methods provided in Java to combine strings, modify strings and to know whether to strings have the same values.
  - “”- empty string
  - “\”

Boolean Literals

- The values true and false are treated as literals in Java programming
- When we assign a value to a boolean variable, we can only use these two values.
- Unlike C, we can't presume that the value of 1 is equivalent to true and 0 is equivalent to false in Java.
- We have to use the values true and false to represent a Boolean value

JAVA - Constants

- A constant in Java is used to map an exact and unchanging value to a variable name.

- Constants are used in programming to make code a bit more robust and human readable

JAVA Constants Naming Convention

- The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").
- (ANSI constants should be avoided, for ease of debugging.)

  - static final int MIN_WIDTH = 4;
  - static final int MAX_WIDTH = 999;
  - static final int GET_THE_CPU = 1;

JAVA Constants

Without Constants

```
public class AreasAndVolumes
{
  public double volumnOfSphere (double radius)

  {
    return (4/3) * Math.pow(3.14159 * radius, 3);
  }

  public double volumeOfCylinder (double radius, double height)
  {
    return Math.pow(radius * 3.14159, 2) * height;

  }

  public double areaOfCircle (double radius)
  {
    return Math.pow(radius * 3.14159, 2);
  }
}
```

With constants

```
public class AreasAndVolumes
{
//  assign the value of 3.14159 to this new variable.
  private static final double PI = 3.14159;

  public double volumnOfSphere (double radius)
  {
    return (4/3) * Math.pow(PI * radius, 3);
  }

  public double volumeOfCylinder (double radius, double height)
  {
    return Math.pow(radius * PI, 2) * height;
  }

  public double areaOfCircle (double radius)
  {
    return Math.pow(radius * PI, 2);
  }
}
```

JAVA - Operators

- Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.
- Operators are categorized according to number of operands
  - *Unary - one operand*
  - *Binary - two operands*
  - *Turnery - three operands*
- Each operator has a precedence.
- Operators with higher precedence are evaluated before operators with relatively lower precedence.

Assignment Operator

- It assigns the value on its right to the operand on its left:

```
int noOfWheels = 4;
int speed = 0;
int gear = 1;
```

Arithmetic Operators

- Operators that perform addition, subtraction, multiplication, and division
  - + additive operator (also used for String concatenation)
  - - subtraction operator
  - * multiplication operator
  - / division operator
  - % remainder operator

Unary Operators

- The unary operators require only one operand
- They perform various operations such as
  - Incrementing/decrementing a value by one
  - Negating an expression
  - Inverting the value of a boolean
- + Unary plus operator
- - Unary minus operator
- ++ Increment operator
  - increments a value by 1
- -- Decrement operator
  - decrements a value by 1
- ! Logical complement operator
  - inverts the value of a boolean

Equality & Relational Operators

- Determines if one operand is greater than, less than, equal to, or not equal to another operand
  - == equal to
  - != not equal to
  - > greater than
  - >= greater than or equal to
  - < less than
  - <= less than or equal to

Conditional Operators

- && Conditional-AND
- || Conditional-OR
- ?: shorthand for an if-then-else statement

```
if (a > b)
        { max = a; }
else
        { max = b; }

max = (a > b) ? a : b;
```

Instanceof Operators

instanceof

- used to test whether the object is an instance of the specified type

    ex: obj1 instanceof Student

Bitwise & Bit shift Operators

- The unary bitwise complement operator "~" inverts a bit pattern;
- The signed left shift operator "<<" shifts a bit pattern to the left.
- The signed right shift operator ">>" shifts a bit pattern to the right.
- The unsigned right shift operator ">>>"
  - For positive number, >> and >>> works same,
  - For negative number, >>> changes parity bit (MSB) to 0
- The bitwise & operator performs a bitwise AND operation.
- The bitwise ^ operator performs a bitwise exclusive OR operation.
- The bitwise | operator performs a bitwise inclusive OR operation.

Summary

- JAVA – Primitive Types
- JAVA – Reference Types
- Primitive vs Reference
- JAVA – Wrapper Classes for Primitives
- JAVA – Literals
  - Integer Literals
  - Floating point Literals
  - String Literals
  - Character Literals
  - Boolean Literals
- JAVA – Constants
- JAVA –Operators
- JAVA - Type Casting

References

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

- How To Program (Early Objects)
  - By H .Deitel and P. Deitel
- Headfirst Java
  - By Kathy Sierra and Bert Bates

Questions ???

Thank You