CSC102/CPE102

Review and Overview

By Philip Fu, 2010

What we expect you!!!

This is a basic course on programming

- **Lecture slides**:
  - Please read notes and textbook every week!!!
  - Capture -> Understand -> Practice
- **Tutorial questions**:
  - Don’t just listen!!! Do the questions yourself!!!
- **Lab questions**:
  - Work out the problems yourself!!!
  - Do the log book with good documentation!!!

Course Syllabus

- Computer Systems & Java Programming
- Java Program Development
- Data and Operators
- Console Input/Output
- Branching
- Looping
- Methods

Philip

- Arrays
- Classes & Objects
- Strings & Characters
- Class Inheritance (Optional & Non-Examinable)
- Exception Handling
- File Input/Output

Kheng Leong Tan

Today: Two parts

- Review: 1st half
- Overview: 2nd half

Review

- What is Review?
- Go through

  things that you should have learnt!!!

- So… If you found anything that you don't know
  - Now, mark it down!!!
  - And use the recess week to catch up!!!

Review Chapter #1:
Computer Systems & Java Programming

- What is Hardware?
  - Memory, CPU, input and output device
- What is Software?
  - Programs to control the hardware
- Basic Linux commands
- Number systems: Binary, Octal, Decimal, etc.
- What is Computer Programming?
  - Syntax + logic -> instructions to computer
- Java programming
  - General purpose language
  - JVM (Java Virtual Machine)

Review Chapter #2:
Java Program Development

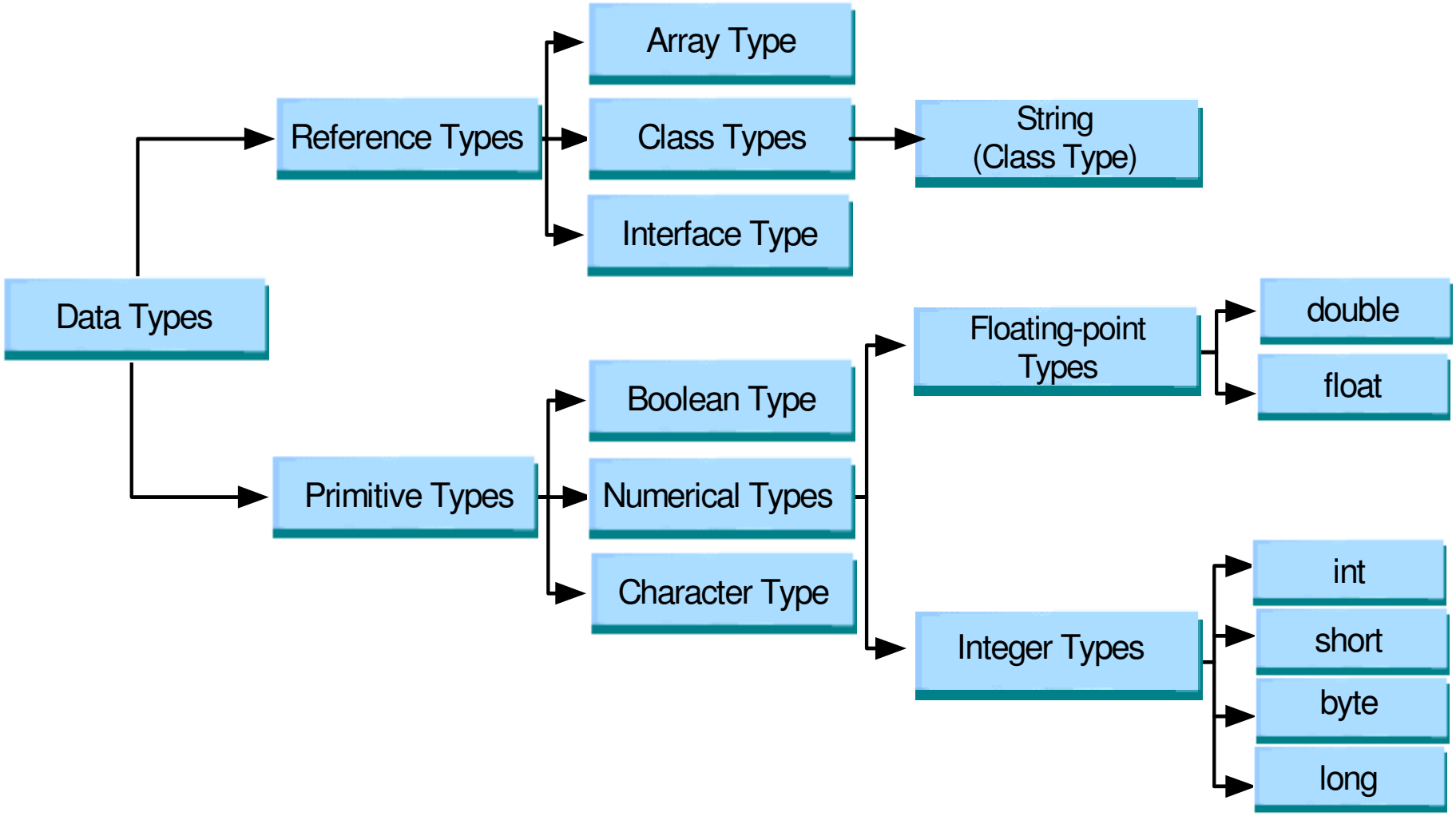
- JDE: Java Development Environment: JGrasp, etc.
- Development of a Java Program
  - Step 1: Write the Java source code (.java)
  - Step 2: Compile -> Java Byte Code (.class)
  - Step 3: Run the compiled code (interpreter: JVM)

  [You should have done all these by yourself in lab. 1]
- Program Development Process (Problem Solving)
  - Problem Specification (e.g., from the tutorial, exam, etc.)
  - Problem Analysis (input, output, formulaes, etc.)
  - Program design (Steps!!!! Flowchart!!!)
  - Implementation
  - Program Testing
- Program Debugging: Syntax Error? Runtime Error?
- Documentation

Review Chapter #3:
Data

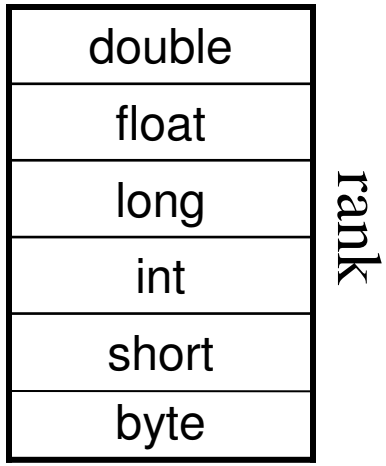
- Data type:
  - Available data types in Java: see the tree on next page!!!
  - Reference types: 2nd part of this course
  - Primitive types: Pick up the right type for your data
    - Boolean: true or false (for conditions) (1 byte storage)
    - Character: ASCII table, 2 bytes, hexadecimal representation, Unicode, escape characters (why and how?)
    - Numerical: integers and floats
      - Representation range and precision
      - Storage size
      - Significant digits (for float and double) – they are not precise reprsentation
- Identifiers, Literals, Constants, and Variables
  - Naming rules for identifiers
  - Literal: plain value in code
  - Constant: its value is fixed when initialized [keyword: final]
  - Variables: value can be changed

Data Types


Data Types
Reference Types
Array Type
Class Types
String
(Class Type)
Interface Type
Primitive Types
Boolean Type
Numerical Types
Character Type
Floating-point
Types
double
float
Integer Types
int
short
byte
long

Review Chapter #5:
Operators

- Operations on one or more pieces of data (e.g., variables)
- Java Operators
  - Fundamental arithmetic operators: +, -, *, /, etc.
  - Assignment operator: =
    - Variable = expression
  - Arithmetic Assignment operators: +=, -=, etc.
  - Increment or decrement operators: ++ or --
    - note: side effect and post/pre (ordering)
  - Concatenation operators: + and String
  - Relational operators (comparison): <, >, ==, etc. (next chapter)
  - Logical operators: not, and, or (next chapter)



- Data conversion: type cast, arithmetic, and assignment
- Precedence of operators: evaluation order
- Programming style: see notes.

Precedence of Operators

This decides the **order** of evaluation for arithmetic expressions containing several operands.

The list of operators with decreasing priority:

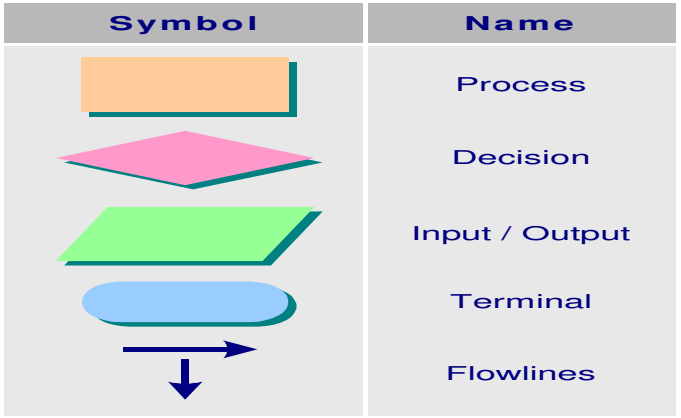
Operator	Meaning	Associativity
`()`	parentheses	left to right
`++, --`	increment, decrement	right to left
`+,-`	unary	right to left
`(Type)`	type cast	right to left
`*, /, %`	multiplication, division, modulus	left to right
`+,-,+`	binary addition, subtraction, String concatenation	left to right
`=,+=,-=,*=,/=`	assignment	right to left

decr

Review Chapter #4:
Console I/O

- Basics: System.in, System.out, and System.err
- Console output
  - System.out.print and println
  - Message Box and Formatted Console Output (self reading)
- Every class in Java belongs to a package, e.g., Scanner
  - import java.util.Scanner; OR import java.util.*;
  - Classes in java.lang are imported automatically
- Console input
  - Scanner: Import, create a scanner object; then just its methods
    int data = sc.nextInt() ; // or nextLine, next(), nextDouble(), etc.
  - The concept of input stream
  - The mechanism of nextLine VS nextInt, next, etc.
  - Dialog Box (self reading)

Review Chapter #6: Branching

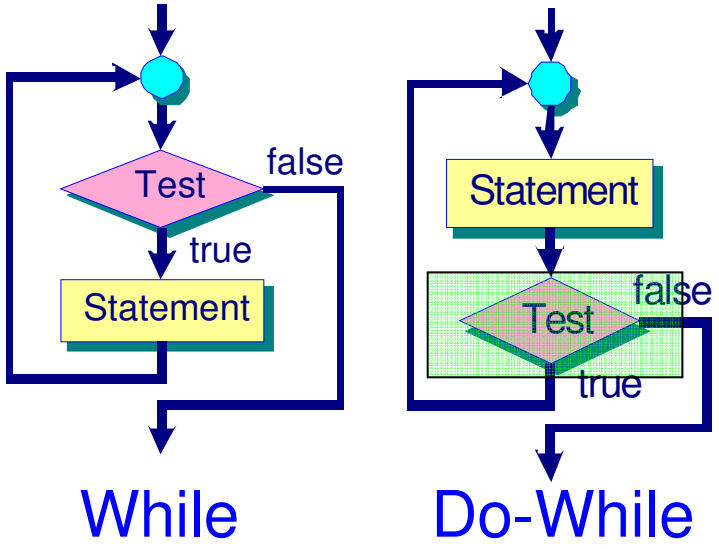


- Why Branching?
- Condition:
  - boolean expression using logical and relational operators
- Control flow: flow of program and flowchart
- Java programming:
  - IF, IF-ELSE, IF-ELSE-IF-ELSE, Nested-IF
    - The power of IF-ELSE (pre-knowledge inside bodies)
    - Note: the program control flow and the LOGIC!!!
    - Use of parenthesis to enclose properly
    - Rule: Else associates with the nearest unresolved IF before it
  - switch
    - Meaning: Nested-IF
    - syntax and rules
  - Conditional operator (it is a shortform)
    - int max = (x > y) ? x : y ;

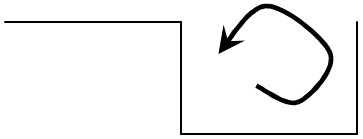
```
if (x > y)
    max = x;
else
    max = y;
```

Review Chapter #7:
Looping

- Why looping? Repeated code!!!
- How?
  - Fundamentally (usually) 4 basic stages:
    Initialize (loop control vari.), Test condition, Loop Body, and Update
  - Counter-controllable and sentinel-controllable
- While:
  - Test condition **first**
  - Body may not be executed
- Do-While:
  - Test condition **last**
  - Body executes at least once

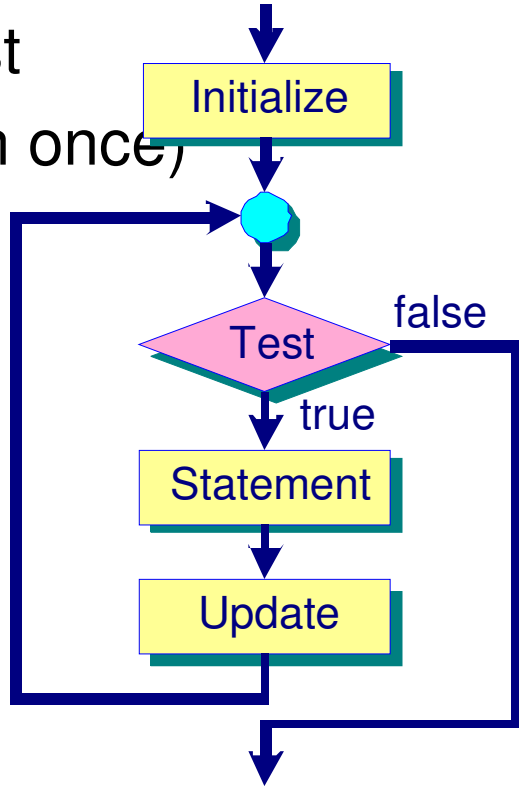


Review Chapter #7:
Looping (continue)

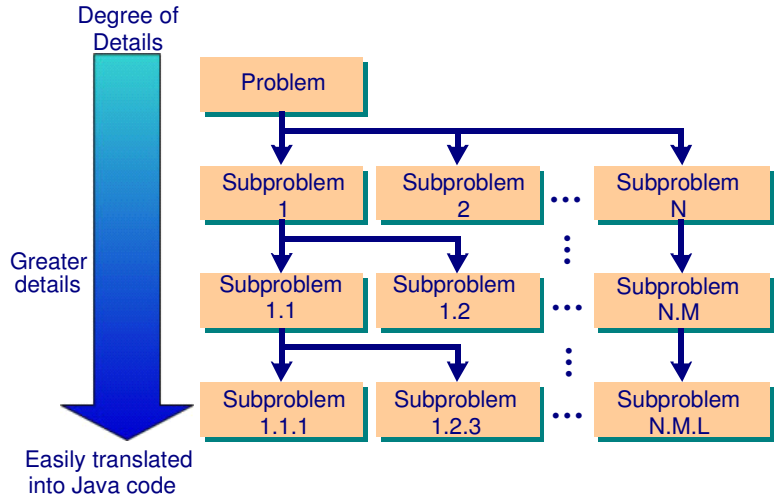


for (Initialize; Test; Update)
Statement;

- For Loop:
  - All four things explicitly
  - Initialize always executed first
  - Loop body (may not run even once)
- break:
  - Control flow: where it goes?
- continue:
  - Control flow: where it goes?
  - Very tricky for *for-loop*!!!
- Infinite loop:
  - while (true) { … }



Review Chapter #8: Methods



- Why Methods?

  Modular Programming!!!
  - Large software: better programming structure
  - Better code readability, code re-use, etc.
- A method: a self-contained unit of code for a specific task
- Defining a method (and syntax)
  - public static Return_Type Method_Name (Parameter_List)
- Calling a method
  - Passing value(s) into the method through the parameter(s)
- Method Overloading: method signature
  - public static int findMin(int num1, int num2)
  - public static double findMin(double num1, double num2)
  - public static int findMin(int num1, int num2 , int num3)

Review Chapter #8:
Methods (continue)

- Scope and Duration of Variables
  - Scope: section of code that can use the variable
    - Block VS Class
  - Duration: how long a variable can exist in memory
    - Static VS automatic
- Math class and methods inside java.lang
- Wrapper Classes
  - wraps a primitive data type into an object like a container
  - Why? E.g., conversion from string
  - Autoboxing and Unboxing
- Enumerated Types (symbolic names, more readable)

  enum Day {Sun, Mon, Tue, Wed, Thu, Fri, Sat};

  Day today = Day.Tue;

  name() and ordinal() methods

Course Syllabus

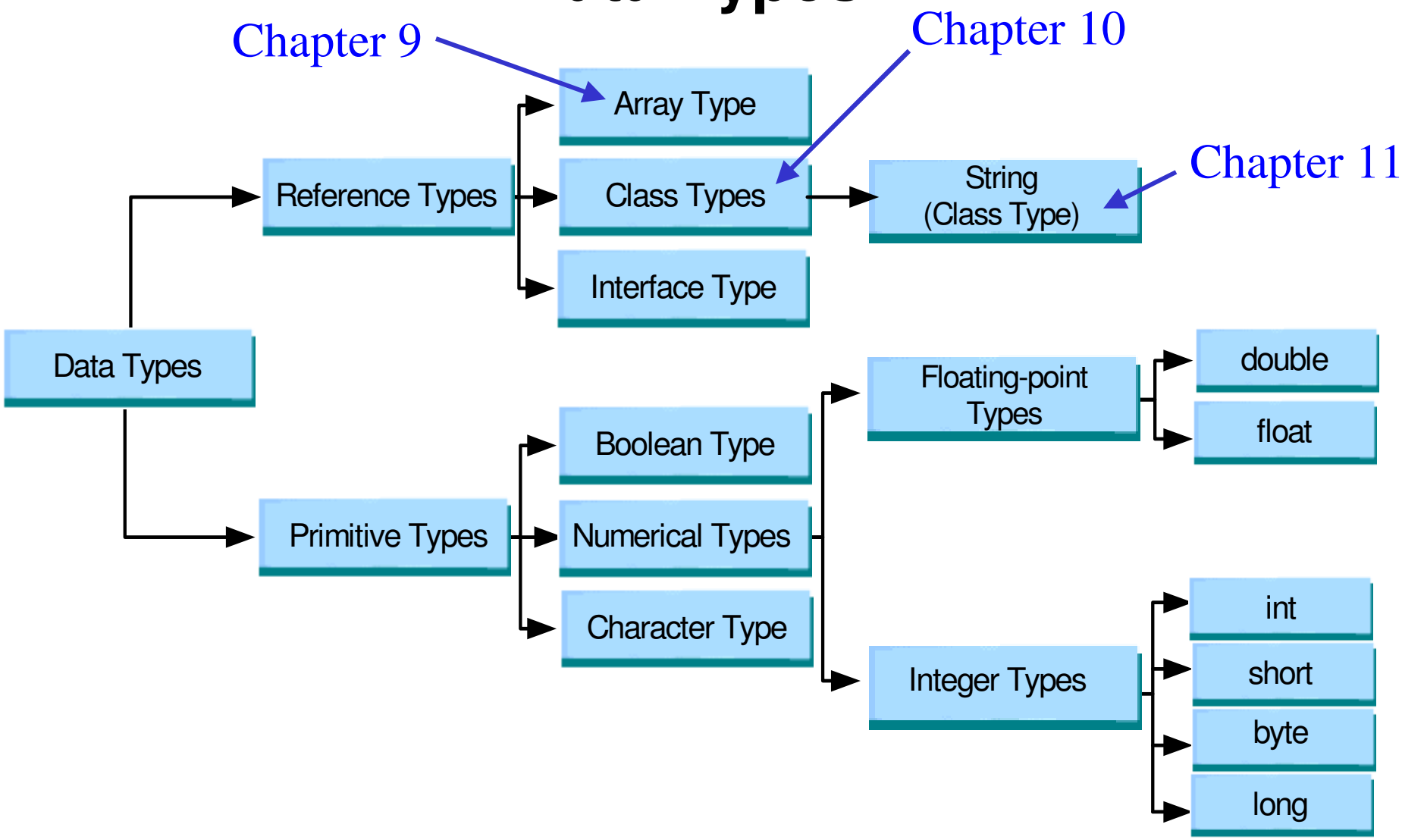
- Computer Systems & Java Programming
- Java Program Development
- Data and Operators
- Console Input/Output
- Branching
- Looping
- Methods

Philip

- Arrays
- Classes & Objects
- Strings & Characters
- Class Inheritance (Optional & Non-Examinable)
- Exception Handling
- File Input/Output

Kheng Leong Tan

Data Types


Chapter 9
Chapter 10
Chapter 11
Array Type
Reference Types
Class Types
String
(Class Type)
Interface Type
Data Types
Floating-point
Types
double
float
Boolean Type
Primitive Types
Numerical Types
Character Type
Integer Types
int
short
byte
long

Again… Don't just listen!!!

- First, make sure you **understand** each concept!
- But… Don't just know what they are!
- You have to know **how to use** the techniques in programming!!! E.g., how to write methods, how to do looping, how to create constants, etc.
- You must have HANDS-ON knowledge in this course!

Make good use of your **recess** week!!!

Last reminder…

- CUP: Capture, Understand, and Practice;
- Computers are the most stupid things on Earth; you must use correct (logic) and precise (syntax) instructions so that it can work for you;
- Fix the program logic before the program syntax;
- Program is not only for computers to compile and run, but also for human to read! Beware of your programming style!
- More space less bugs (indentation, white space, blank lines)
- Copy and paste is the major source of errors (in coding);
- println is a good friend for you in debugging.

Lastly… make good use of the time. :)