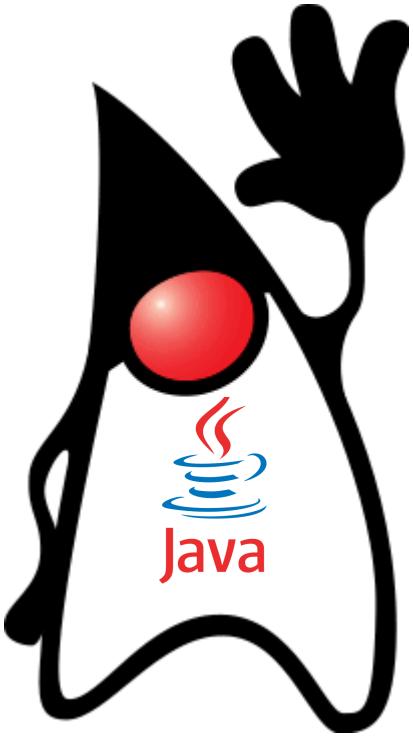
Introduction to java data types and expression



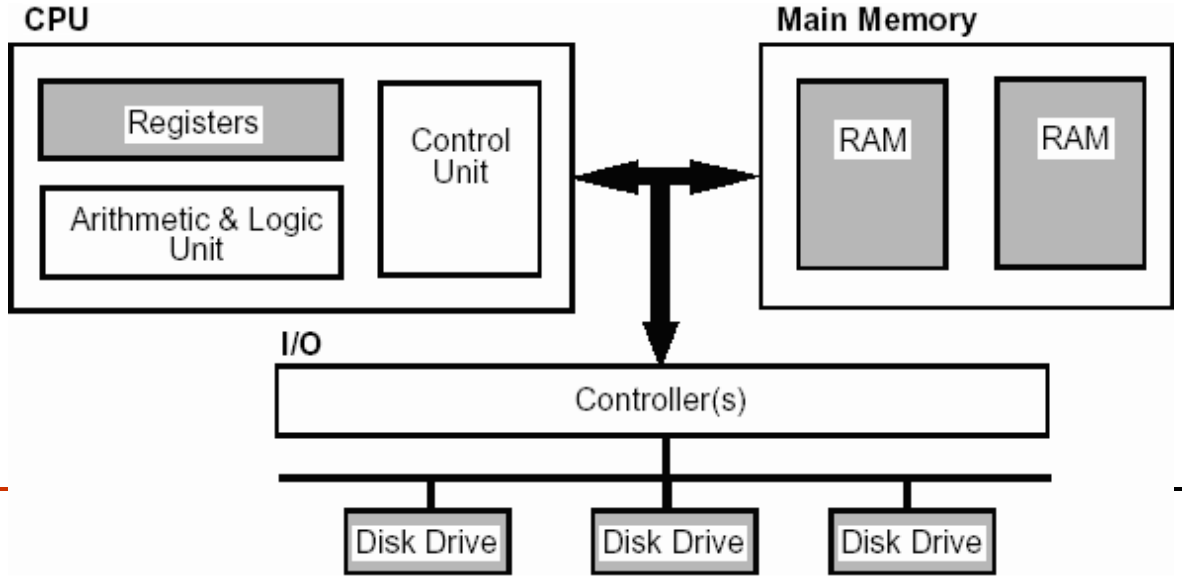
Dr. Praisan Padungweang

Learning outcome

- Describe how classes and objects can be used to build a simple program
- Select appropriate built-in data types and library data structures (abstract data types) to model, represent, and process program data

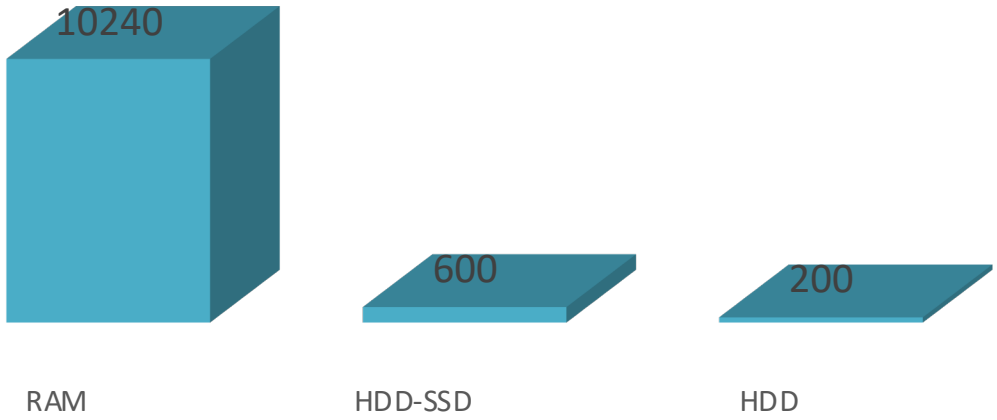
Processor (CPU)

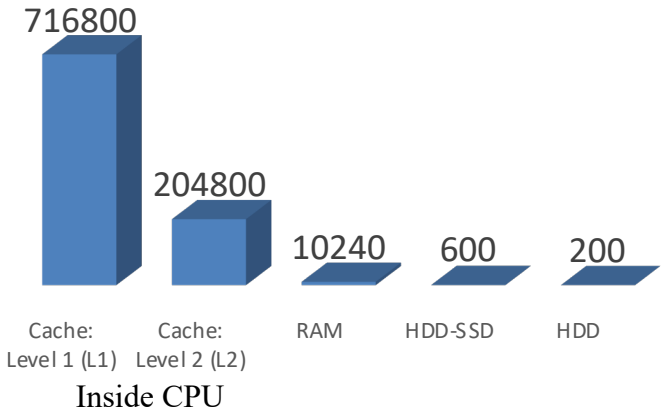
- The ***C***entral ***P***rocessing ***U***nit- the computer's brain that performs most calculations and makes decisions. It consists of two components:
  - ***C***ontrol ***U***nit (**CU**) – control and coordinates the action of the ALU component and the movement of data between the CPU and RAM.
  - ***A***rithmetic and ***L***ogic ***U***nit (**ALU**) – performs numeric operations (+, -, *, /) and logical operations (comparisons such as AND, OR, NOT).



https://www.doc.ic.ac.uk/~eedwards/compsys/memory/index.html

Main Memory Vs Secondary Memory





Price

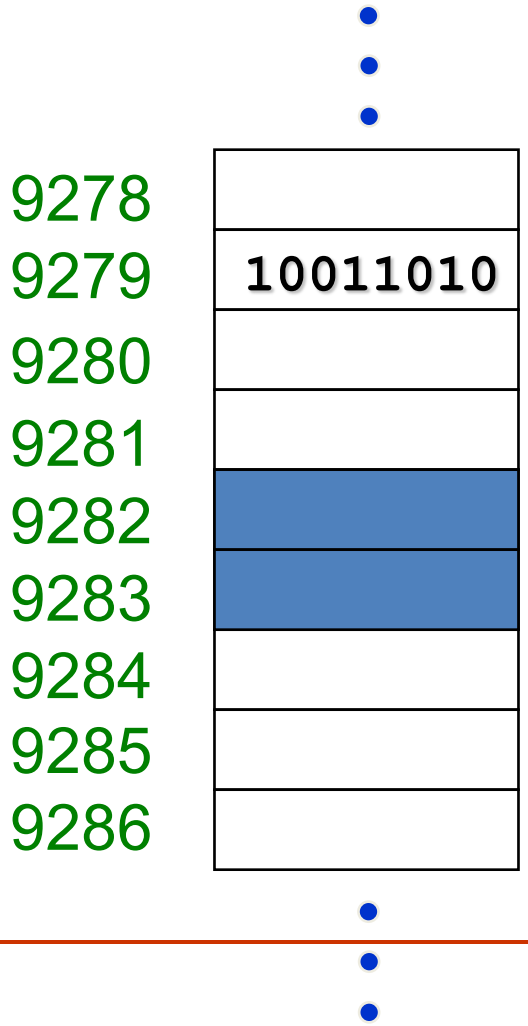
	Observed Highest Speed (2013)	
Cache: Level 1 (L1)	700 GB/s	716,800 MB/s
Cache: Level 2 (L2)	200 GB/s	204,800 MB/s
RAM	10 GB/s	10,240 MB/s
HDD-SSD	0.6 GB/s	600 MB/s
HDD	0.2 GB/s	200 MB/s

https://en.wikipedia.org/wiki/Memory_hierarchy

HDD Charts 2013 (Read Throughput Average)
SSD Charts 2014 (Random Read)

Main Memory (Random Access Memory: RAM)

Main memory is divided into many memory locations (or *cells*)



Each memory cell has a numeric *address*, which uniquely identifies it

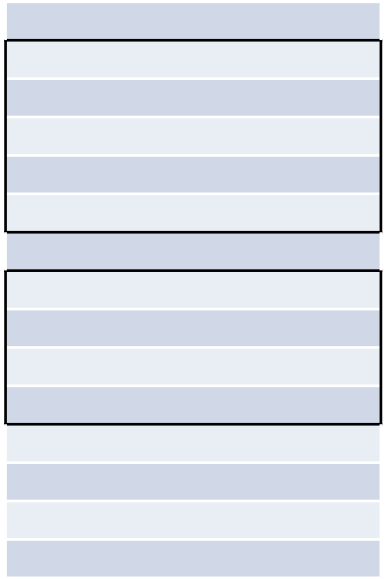
Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

Large values are stored in consecutive memory locations

Program Execution

Instruction code

Data

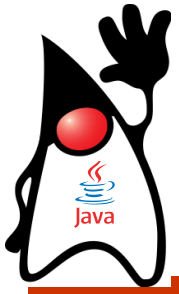


unchanged

reuse / change

Units

- **bit**
  - Short for binary digit, the smallest unit of information on a machine.
  - A single bit can hold only one of two values: 0 or 1.
- **byte**
  - A unit of storage capable of holding a single character. On almost all modern computers, a byte is equal to 8 bits.
  - Large amounts of memory are indicated in terms of kilobytes (1,024 bytes), megabytes (1,048,576 bytes), gigabytes (1,073,741,824 bytes).



Introduction to Java

Compile Once, Run Anywhere

- The Java interpreter is part of the JVM
- The Java Virtual Machine (JVM) is a piece of software (not a piece of hardware) that interprets ***Java byte code*** into machine code.
- Once, you compile a program into byte code, it can be run on any machine with the JVM installed.

Java Program Structure

```
// comments can be placed almost anywhere
// comments about the class
public class MyProgram
{


}
```

class header

class body

Java Program Structure

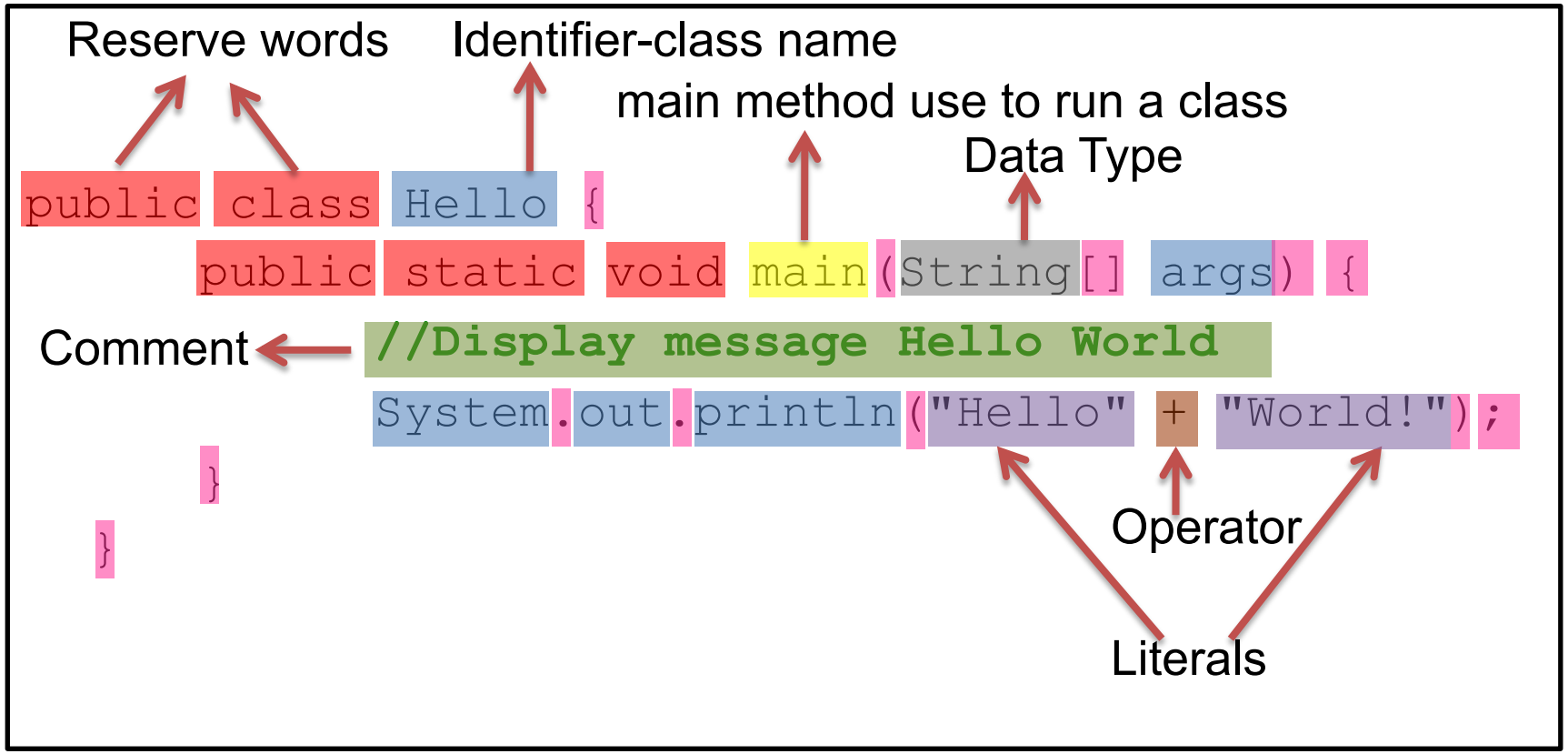
```
//  comments about the class
public class MyProgram {

    // comments about the method
    public static void main (String[] args)
    {
        System.out.println("Java Program");
    }
}
```

method header

method body

Program Structures



reserve words | main method | literals | separators

identifiers | comment | operator | data type

Java Program Layout

```
public class Hello{

    public static void main(String[] args){

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

    }

}
```

Good Layout

```
class
{
    public static void
    main(String[] args
        )
    {
            System.out.println
    ("Hello
     World!");
    }
      }
```

Bad Layout

Create A Java Program

- Write and save a file "`Hello.java`" that defines class "`Hello`"

- If your program use a "public" keyword in front of class name, you must give a **filename** in the **same as your class name**
- File and class name are **case sensitive** and must match exactly

Compile and Run

- Compile Java Program

```
javac  <filename.java>
javac Hello.java
```

Compiler

- This step creates a file extension "Hello.class"

bytecode

- Run Java Program

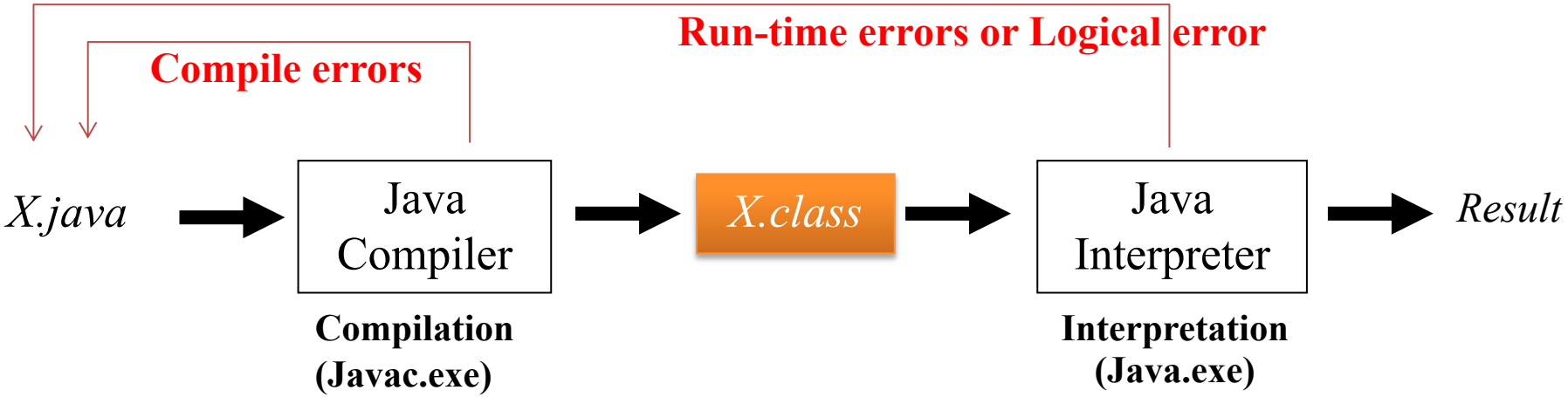
```
java  <classname>
java Hello
```

Interpreter (on JVM)

- Output

```
Hello World!
```

Java Program Development Process



Hello.java

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello Word!");
    }
}
```

Hello.class

```
.class public hello
.super java/lang/Object

.method public static main : ([Ljava/lang/String;)V
    .limit stack 10
    .limit locals 10

    getstatic java/lang/System out Ljava/io/PrintStream;
    ldc "Hello World!"
    invokevirtual java/io/PrintStream println (Ljava/lang/Object;)V
    return
.end method
```

Display on the console

Hello World!

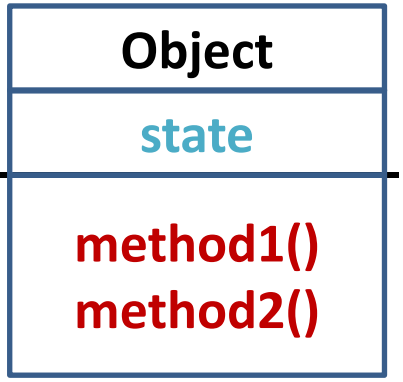
Example simple program

```
import javax.swing.JOptionPane;

public class HelloDialogBox {

    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Welcome to Java!");
    }
}
```

An Object



- Object is an entity that
  - Have a **state** (data/information),
  - Can behave according to the **messages** received.
    - All possible messages that an object can receive are pre-defined: **methods.**
    - Upon receiving a message, object may change its state.

Car (Object – Instance)

odometer
speed

- unlockDoor()
- lockDoor()
- openDoor()
- closeDoor()

doorLockStatus
doorOpenStatus

- startEngine()
- stopEngine()
- changeGear()
- accelerate()

engineStatus
gearStatus

- turnWheel()
- break()

wheelPosition

- turnOnAirConditioner()
- turnOffAirConditioner()
- setAirConditionerTemperature()
- setAirConditionerFanSpeed()

airConOnOffStatus
airConTemperature
airConFanSpeed

- turnOnRadio()
- turnOffRadio()
- setRadioVolume()
- setRadioChannel()

radioOnOffStatus
radioVolumeLevel
radioChannel

Car (Object –> Collection of Objects)

speed
odometer

- **Door System**
  - unlock()
  - lock()
  - open()
  - close()

lockStatus

openStatus

- **Engine**
  - start()
  - stop ()
  - changeGear()
  - accelerate()

engineStatus

gearStatus

- **Wheel System**
  - turnWheel()
  - break()

wheelPosition

- **Air Conditioner**
  - turnOn()
  - turnOff()
  - setTemperature()
  - setFanSpeed()

onOffStatus

temperature

fanSpeed

- **Radio**
  - turnOn()
  - turnOff()
  - setVolume()
  - setChannel()

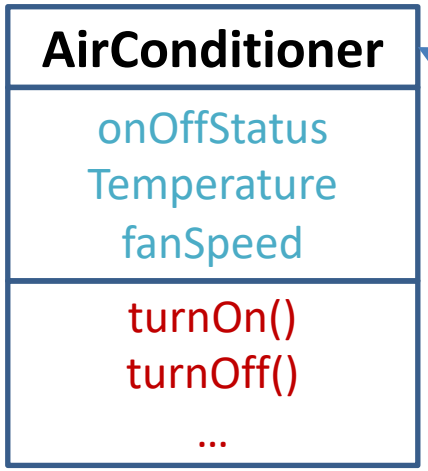
onOffStatus

volumeLevel

channel

Car (Object – Instance)

- Door System
- Engine
- Wheel System
- Air Conditioner
- Radio



- turnOnAirConditioner()
- turnOffAirConditioner()
- setAirConditionerTemperature()
- setAirConditionerFanSpeed()

Air Conditioner

turnOn()

turnOff()

setTemperature()

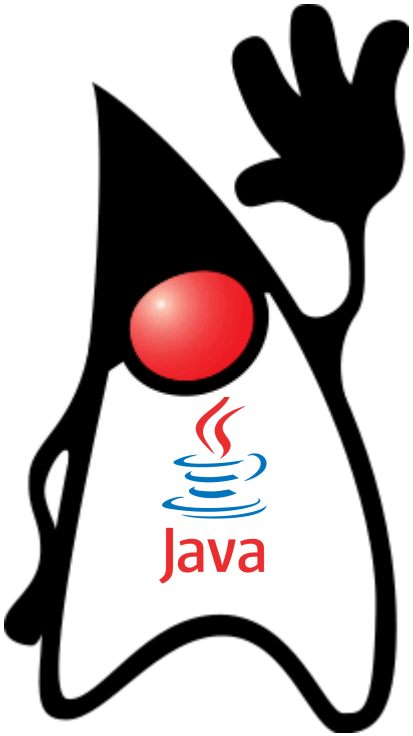
setFanSpeed()

onOffstatus

temperature

fanSpeed

Data Types and Expressions



Primitive Data Types

There are eight primitive data types in Java

- Four of them represent integers:
  - `byte`, `short`, `int`, `long`
- Two of them represent floating point numbers:
  - `float`, `double`
- One of them represents characters:
  - `char`
- One of them represents boolean values:
  - `boolean`

Primitive Data Type

Type	Size	Default	Value Ranges	Contains
boolean	16 bits	false		true or false
byte	8 bits	0	-128 to 127	Signed integer
char	16 bits	\u0000		Unicode character
short	16 bits	0	-32,768 to 32,767	Signed integer
int	32 bits	0	-2,147,483,648 to 2,147,483,647	Signed integer
long	64 bits	0	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed integer
float	32 bits	0.0	Approximately -3.4E+38 to 3.4E+38 with 7 significant digits	IEEE754 floating point
double	64 bits	0.0	Approximately -1.7E+308 to 1.7E+308 with 15 significant digits	IEEE754 floating point

Literals

- *Literal* is a constant value that appears in a program

- **integer**
  - 20 ***(default int)*** 20L 20l
  - 010 (octal) 0Xf0 (hex) 0xffL

- **floating point**
  - 3.14 ***(default double)*** 3.1E12 2.0e-2
  - 2.0F 2.5f

- **character**
  - 'A' '\n' '\udddd' (hex - \udddd)
  - '\\' ' \101' (octal - \ddd)

- **boolean**
  - true
  - false

Character Sets

- A *character set* is an ordered list of characters, with each character corresponding to a unique number

A `char` variable in Java => *Unicode character set* (16-bits)

  - The Unicode character set allows for 65,536 unique characters
  - It is an international character set, containing symbols and characters from many world languages

ASCII Character

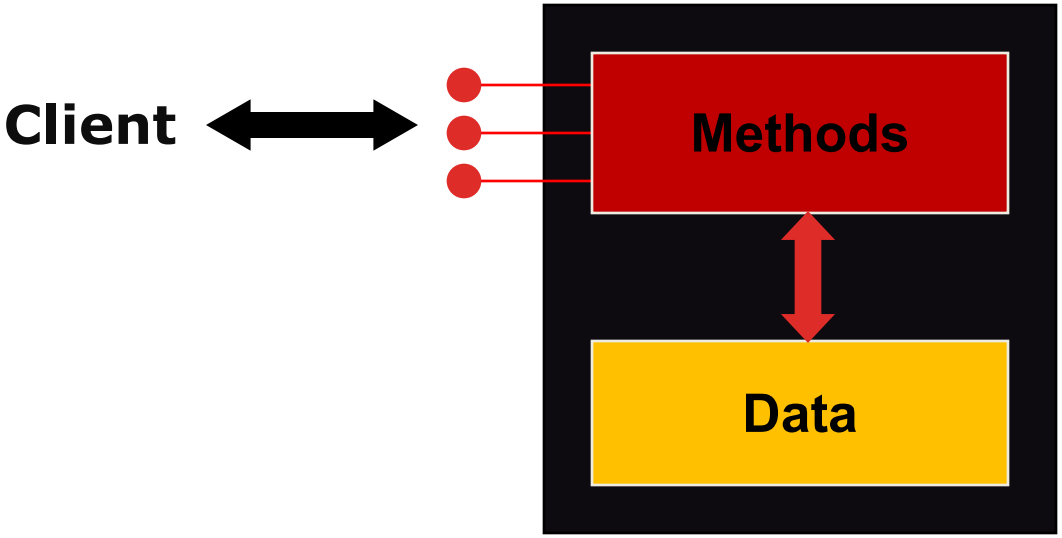
- The ASCII character set is older and smaller than Unicode (7-bits), but is still quite popular
- The ASCII characters are a subset of the Unicode character set, including:
  - **Uppercase letters** - A, B, C,…
  - **Lowercase letters** - a, b, c,…
  - **Digits** - 0, 1, 2,…
  - **Symbols** - &, |, \,:, ;…
  - **Control characters** - carriage return, tab,…

ADT: Abstract Data Type

- Abstract Data Type (ADT) is a collection of data and the particular operations that are allowed on that data.
- ADT whose data representation is hidden with private access modifier and define interface as operations having public access modifier.
- There are two parts
  - an element of ADT data
    - Attributes
  - an implementation of ADT operation
    - Methods

Encapsulation

- An encapsulated object can be thought of as a black box -- its inner workings are hidden from the client
- The client invokes the interface methods of the object, which manages the instance data



String

- A string of characters can be represented as a *string literal* by putting double quotes around the text.
- Note the distinction between a primitive character variable, which holds only one character, and a String object, which can hold multiple characters
- Examples:

```
"This is a string example."
"SIT, KMUTT"
"A"
"Java\nProgramming"
```

String Concatenation

- `" Java " + " Programming"`
- The + operator is evaluated left to right, but parentheses can be used to force the order

```
System.out.println ("learning" + 60 + " hours per course");

System.out.println ("24 and 45 concatenated: " + 24 + 45);
System.out.println ("24 and 45 added: " + (24 + 45));
```

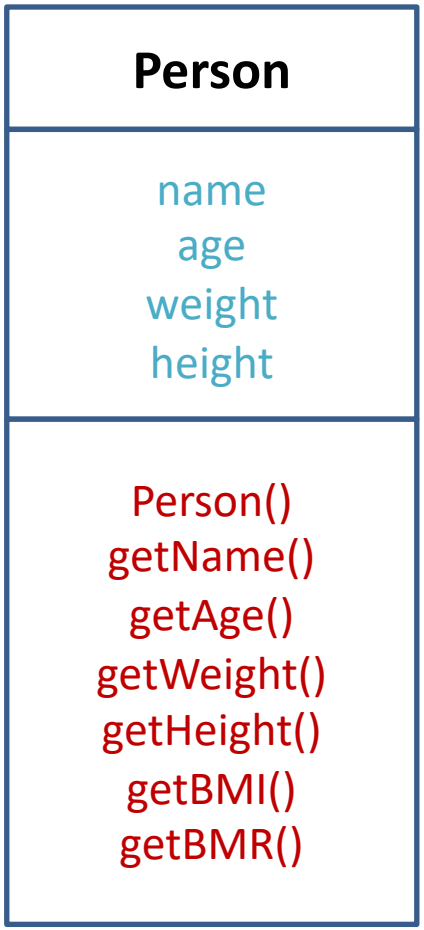
Escape Sequences

- Some Java escape sequences:

Escape Sequence	Name
`\b`	backspace
`\t`	tab
`\n`	newline
`\r`	carriage return
`\"`	double quote
`\'`	single quote
`\\`	backslash

```
System.out.println ("I said \"Hello\" \nto you.");
```

Hand on



BMI:
https://en.wikipedia.org/wiki/Body_mass_index

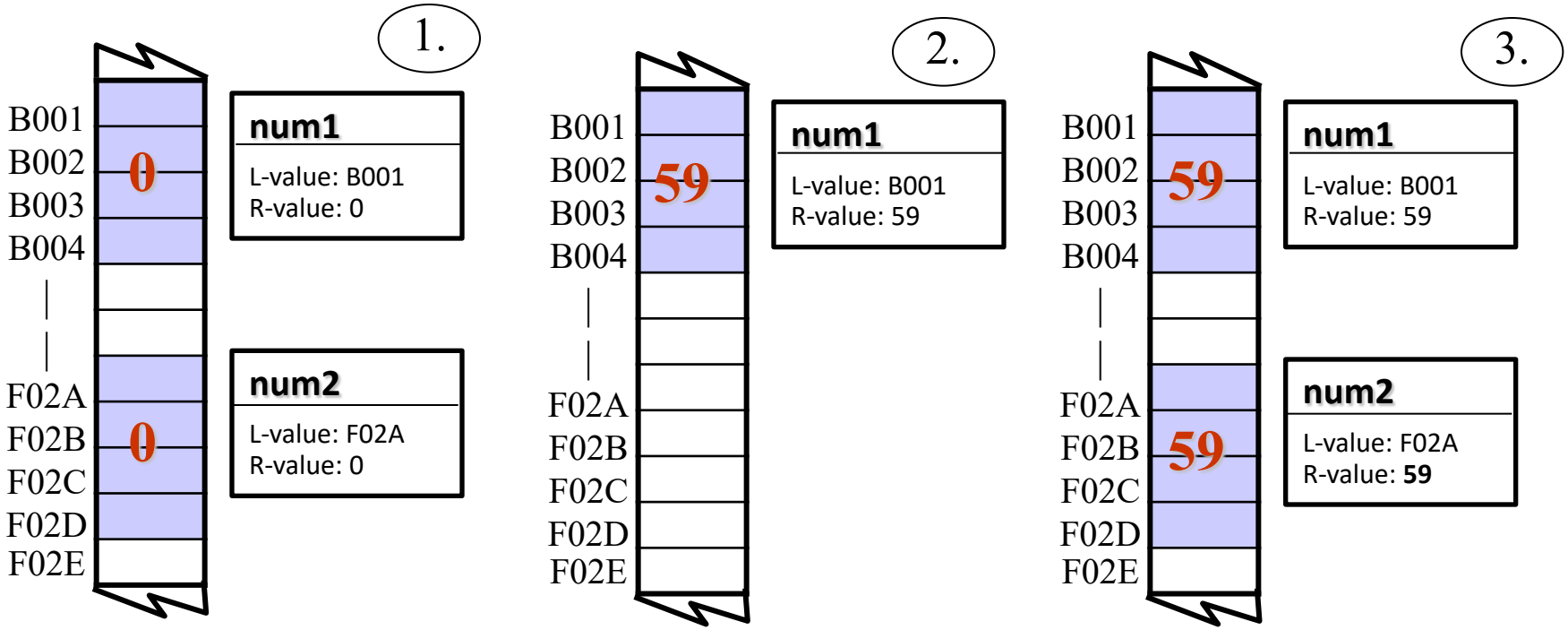
BMR:
https://en.wikipedia.org/wiki/Basal_metabolic_rate#:~:text=Basal%20metabolic%20rate%20(BMR)%20is,by%20endothermic%20animals%20at%20rest.&text=Metabolism%20comprises%20the%20processes%20that,the%20body%20functioning%20at%20rest.

Primitive & Reference Variables

- Each variable has its own L-value and R-value
  1. The L-value is its address
  2. The R-value is its value

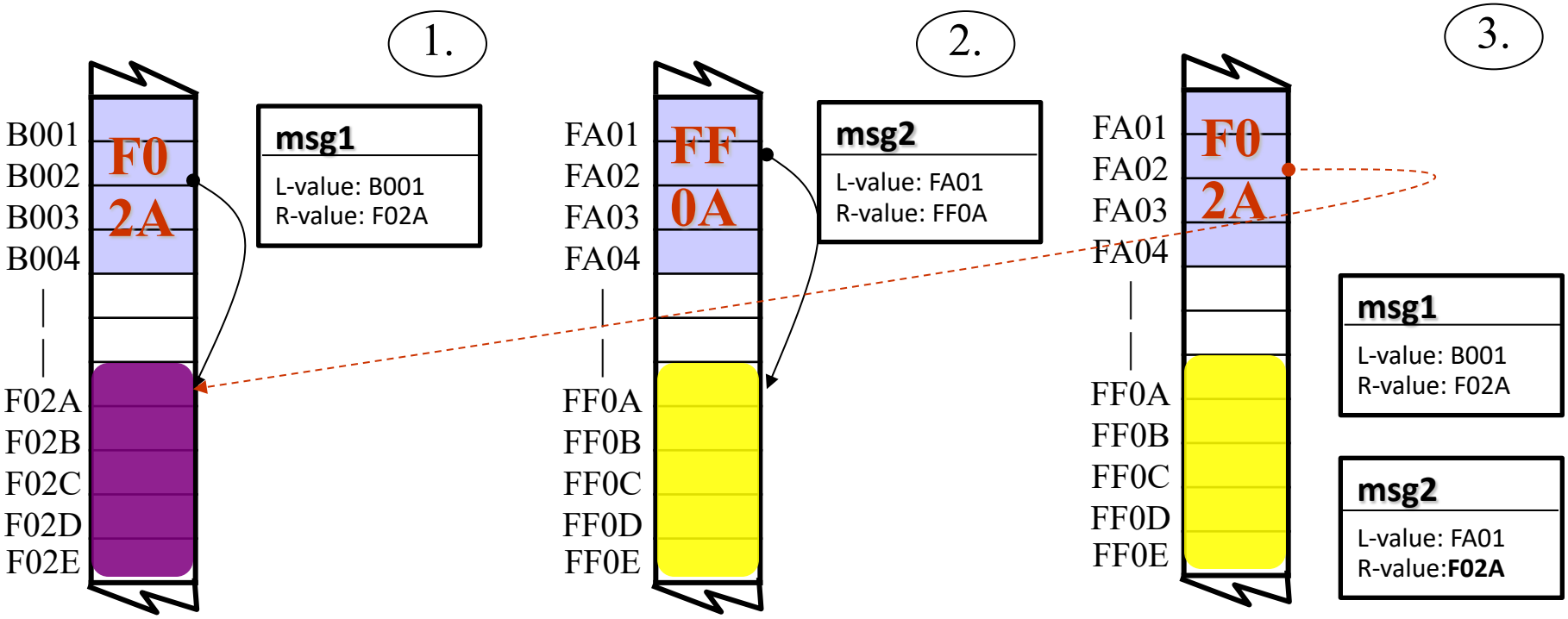
Primitive Assignment

1. int num1, int num2;
2. int num1=59;
3. num2=num1; (R-value Copy)



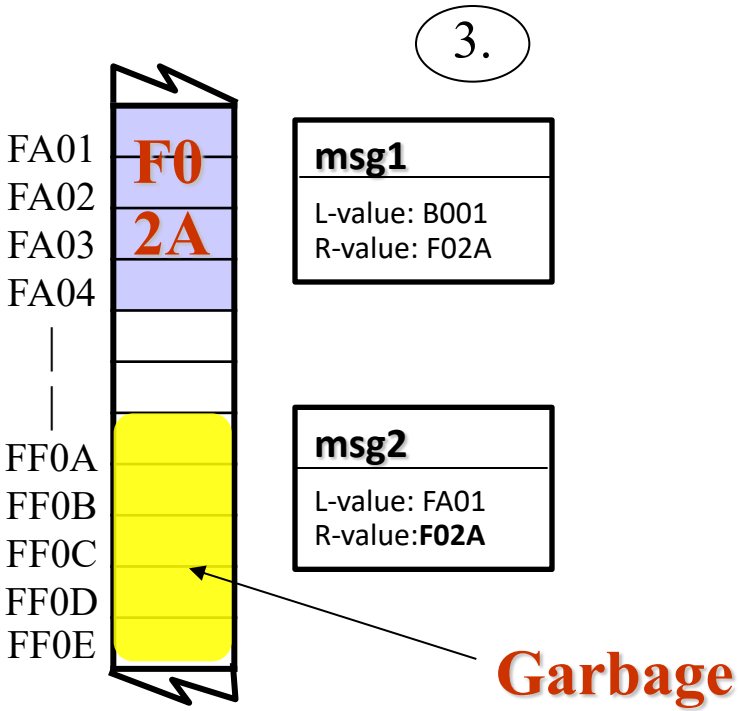
Reference Assignment

1. DogG2 msg1=new DogG2(); msg1.setOwnerName("Sunji");
2. DogG2 msg2=new DogG2(); msg2.setOwnerName("Luffy");
3. DogG2 msg2=msg1; (R-value Copy)



Object without Reference

- We can use an object only if we have a reference to it
- The object without reference is called “Garbage”



Expressions

- An *expression* is a combination of one or more operators and operands
- *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**
Remainder	**%**

Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

```
14 / 3      equals      4

8 / 12      equals      0
```

The remainder operator (%) returns the remainder after dividing the second operand into the first

```
14 % 3      equals      2

8 % 12      equals      8
```

Operator Precedence

- Operators can be combined into complex expressions

```
result  =  total + count / max - offset;
```

Operators

- **Arithmetic**

  + - * / %

- **Relational**

  < <= > >= == !=

- **Logical**

  ! && & || |

Precedence Level	Operator	Operation	Associativity
1	Method()	Method Invocation	L to R
	.	Object Member Reference	
	[]	Array Indexing	
	++, --	**Post-Increment, Post-Decrement**	
2	**++, --**	**Pre-Increment, Pre-Decrement**	R to L
	+, -	**Unary Plus , Unary Minus**	
	!	Logical Not	
3	new	Object Instantiation	R to L
	(<type>)	**Cast (Type Conversion)**	
4	***, /, %**	**Arithmetic Operators**	L to R
5	**+, -**	**Arithmetic Operators**	L to R
	+	**String Concatenation**	
6	<, <=, >, >=	Relational Operators	L to R
7	==, !=	Equality Operators	L to R
8	&	Logical AND	L to R
9	|	Logical OR	L to R
10	&&	Short-Circuit AND	L to R
11	||	Short-Circuit OR	L to R
12	?:	Conditional Operator	R to L
13	**=, *=, /=, %=, +=, -=**	**Assignment with operation**	R to L

Order of Evaluation

- When the Java interpreter evaluates an expression, it performs the various operations in an order specified by
  - the parentheses in the expression
  - the precedence of the operators
  - the associativity of the operators.

Order of Evaluation Example

- Before any operation is performed, however, the interpreter first evaluates the operands of the operator (from left to right).

```
int a = 2;
int result = ++a + ++a * ++a;
```

the expression evaluates 3+4*5, or 23.

Operator Precedence

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e
  1   2   3   4
```

```
a + b * c - d / e
  3   1   4   2
```

```
a / (b + c) - d % e
  2    1     4   3
```

```
a / (b * (c + (d - e)))
  4    3    2    1
```

Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators

First the expression on the right hand side of the = operator is evaluated

```
answer  =  sum / 4 + MAX * lowest;
```

4 1 3 2

Then the result is stored in the variable on the left hand side

Check Point

```
3 + 4 * 4 + 5 * (4 + 3) - 1
```

Check Point

How would you write the following arithmetic expression in Java?

1. $3 + 2^5$

2. $5.5 \times (r + 2.5)^{2.5+t}$

3. $\frac{3 + 4x}{5} - \frac{10(y - 5)(a + b + c)}{x} + 9\left(\frac{4}{x} + \frac{9 + x}{y}\right)$

Increment and Decrement

The increment and decrement adds/subtracts one to its operand

postfix form: (use it then increment/decrement) `count++, count--`

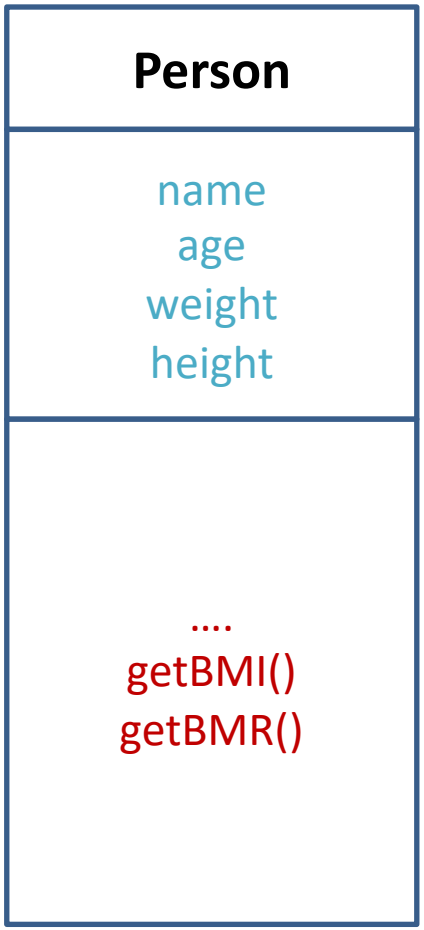
```
total = total + count++;  // is equivalent to
total = total + count;
count = count+1;
```

prefix form: (increment/decrement it then use) `++count, --count`

```
total = total + ++count; // is equivalent to
count = count+1;
total = total + count;
```

should be used with care

Hand on



BMI:
https://en.wikipedia.org/wiki/Body_mass_index

BMR:
https://en.wikipedia.org/wiki/Basal_metabolic_rate#:~:text=Basal%20metabolic%20rate%20(BMR)%20is,by%20endothermic%20animals%20at%20rest.&text=Metabolism%20comprises%20the%20processes%20that,the%20body%20functioning%20at%20rest.

Assignment Operators

- The right hand side of an assignment operator can be a complex expression
- The entire right-hand expression is evaluated first, then the result is combined with the original variable
- Therefore

```
result /= (total-MIN) % num;
```

is equivalent to

```
result = result / ((total-MIN) % num);
```

Java Widening Conversions

Type	Convert to
byte	short, int, long, float, double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

Note that When converting int or long to float or from long to double, some of the significant digits may be lost precision

Java Narrowing Conversions

Type	Convert to
byte	char
short	byte or char
char	byte or short
int	byte, short, or char
long	byte, short, char, or int
float	byte, short, char, int, or long
double	byte, short, char, int, long, or float

Note that boolean values cannot be converted to any other primitive type and vice versa

Data Conversions

- In Java, data conversions can occur in three ways:
  - Casting conversion
  - Assignment conversion
  - Numeric promotion

Casting Conversions

- *Casting conversion* is the most powerful, and dangerous, technique for conversion

- For example,
  if `total` and `count` are integers,
  but we want a floating point result when dividing them, we can cast `total`:

```
result = (float) total / count;
```

Project

- Regtangle
- Circle
- Triangle