Introduction to Computer Programming

CAMT
College of Arts, Media and Technology
Chiang Mai University

Agenda

- Data Representation
- Java programming

Data Representation

Data Representation: How do computers represent data digitally?

- Data is defined as the symbols that represent things, people, events and ideas
- Computers store data in *digital* format as a series of 1s and 0s (known as *binary* code)
  - Each 1 and 0 is called a *bit*
  - Eight bits is called a *byte*

Data Representation

- The term bit comes from “*bi*nary digi*t*”

- Bytes are used to represent one character – a letter, number, or punctuation mark
  - For example, the letter H is represented in binary code as 01001000
  - An exclamation point (!) is 001000001

Data Representation

- *Digital* data is made up of discrete numbers, with each bit being either a 1 or a 0 – it's either on or off, nowhere in between
- *Analog* data is made up of a continuous wave of information, with varying degrees in between
- For example:
    - A digital clock changes it's digital display once every minute to show the time
    - An analog clock is continually moving it's second, minute and hour hands to show the time

Data Representation

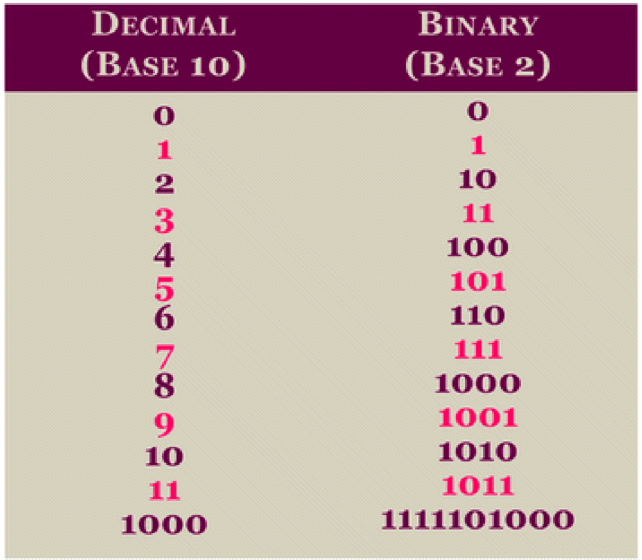
- Another example is a light fixture
  - A standard light switch is similar to digital
    - It is either on or off – 1 or 0
  - A dimmer light switch is similar to analog
    - It's rotating dial can be turned to many different positions to make the light varying degrees of bright or dim

Data Representation

- *Data representation* makes it possible to convert letters, sounds, and images into electrical signals
- *Digital electronics* makes it possible for computer to manipulate simple “on” and “off” signals to perform complex tasks
    - A computer’s circuits have only two states: on and off
    - A binary 1 represents “on”
    - A binary 0 represents “off”

How can a computer represent numbers?

- Unlike the decimal system (base 10), the *binary number system* (base 2) uses only two digits: 0 and 1
- The following table lists some decimal numbers and their binary equivalent:

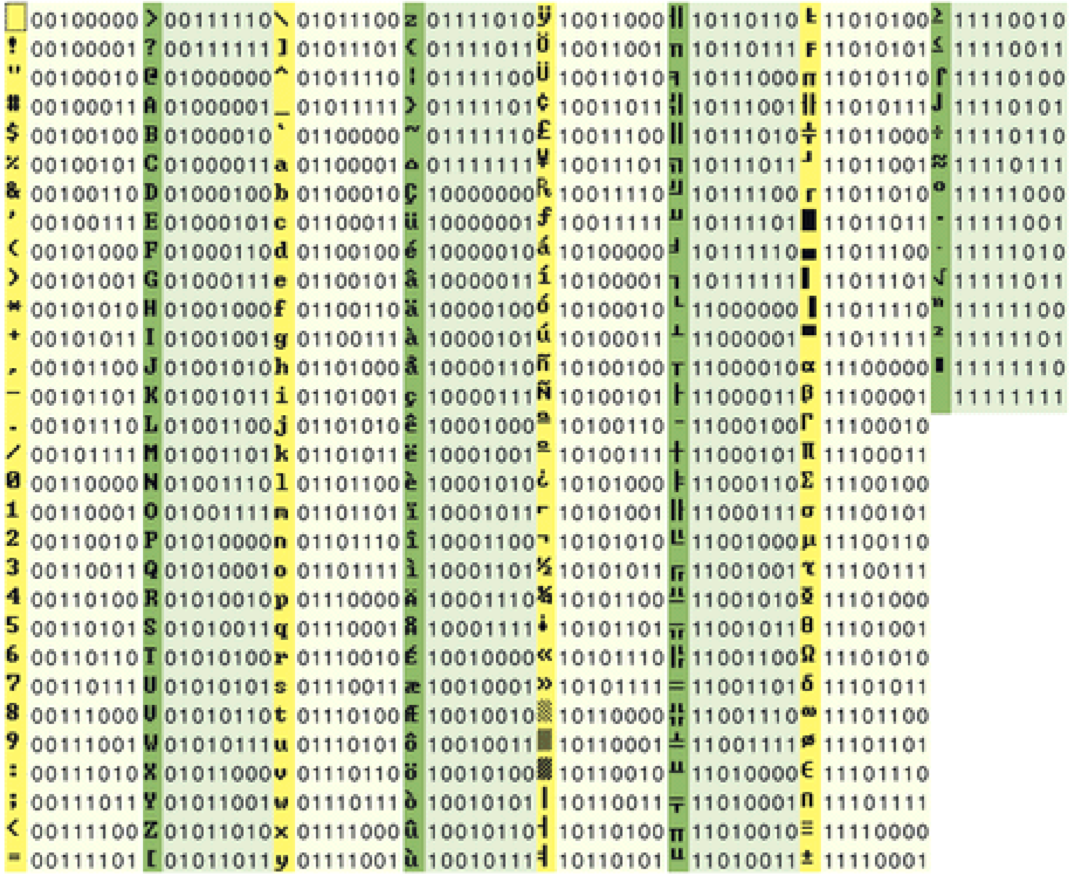


Decimal (Base 10)	Binary (Base 2)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
1000	1111101000

How can a computer represent words and letters using bits?

- *Character data* is composed of letters, symbols, and numbers that will not be used in arithmetic operations
  - *Numeric data* is used in arithmetic calculations, and is encoded differently
- *ASCII* (American Standard Code for Information Interchange) requires only 7 bits for each character
- *Extended ASCII* uses 8 bits for each character. Used in most personal computers
  - See the code on the next slide

How can a computer represent words and letters using bits?



	00100000	>	00111110	\	01011100	z	01111010	ÿ	10011000	╢	10110110	╘	11010100	≥	11110010
!	00100001	?	00111111	]	01011101	{	01111011	Ö	10011001	╖	10110111	╒	11010101	≤	11110011
"	00100010	@	01000000	^	01011110	|	01111100	Ü	10011010	╕	10111000	╓	11010110	⌠	11110100
#	00100011	A	01000001	_	01011111	}	01111101	¢	10011011	╣	10111001	╫	11010111	⌡	11110101
$	00100100	B	01000010	`	01100000	~	01111110	£	10011100	║	10111010	╪	11011000	÷	11110110
%	00100101	C	01000011	a	01100001	⌂	01111111	¥	10011101	╗	10111011	┘	11011001	≈	11110111
&	00100110	D	01000100	b	01100010	Ç	10000000	₧	10011110	╝	10111100	┌	11011010	°	11111000
'	00100111	E	01000101	c	01100011	ü	10000001	ƒ	10011111	╜	10111101	█	11011011	∙	11111001
(	00101000	F	01000110	d	01100100	é	10000010	á	10100000	╛	10111110	▄	11011100	·	11111010
)	00101001	G	01000111	e	01100101	â	10000011	í	10100001	┐	10111111	▌	11011101	√	11111011
*	00101010	H	01001000	f	01100110	ä	10000100	ó	10100010	└	11000000	▐	11011110	ⁿ	11111100
+	00101011	I	01001001	g	01100111	à	10000101	ú	10100011	┴	11000001	▀	11011111	²	11111101
,	00101100	J	01001010	h	01101000	å	10000110	ñ	10100100	┬	11000010	α	11100000	■	11111110
-	00101101	K	01001011	i	01101001	ç	10000111	Ñ	10100101	├	11000011	ß	11100001		11111111
.	00101110	L	01001100	j	01101010	ê	10001000	ª	10100110	─	11000100	Γ	11100010		
/	00101111	M	01001101	k	01101011	ë	10001001	º	10100111	┼	11000101	π	11100011		
0	00110000	N	01001110	l	01101100	è	10001010	¿	10101000	╞	11000110	Σ	11100100		
1	00110001	O	01001111	m	01101101	ï	10001011	⌐	10101001	╟	11000111	σ	11100101		
2	00110010	P	01010000	n	01101110	î	10001100	¬	10101010	╚	11001000	µ	11100110		
3	00110011	Q	01010001	o	01101111	ì	10001101	½	10101011	╔	11001001	τ	11100111		
4	00110100	R	01010010	p	01110000	Ä	10001110	¼	10101100	╩	11001010	Φ	11101000		
5	00110101	S	01010011	q	01110001	Å	10001111	¡	10101101	╦	11001011	Θ	11101001		
6	00110110	T	01010100	r	01110010	É	10010000	«	10101110	╠	11001100	Ω	11101010		
7	00110111	U	01010101	s	01110011	æ	10010001	»	10101111	═	11001101	δ	11101011		
8	00111000	V	01010110	t	01110100	Æ	10010010	░	10110000	╬	11001110	∞	11101100		
9	00111001	W	01010111	u	01110101	ô	10010011	▒	10110001	╧	11001111	φ	11101101		
:	00111010	X	01011000	v	01110110	ö	10010100	▓	10110010	╨	11010000	ε	11101110		
;	00111011	Y	01011001	w	01110111	ò	10010101	│	10110011	╤	11010001	∩	11101111		
<	00111100	Z	01011010	x	01111000	û	10010110	┤	10110100	╥	11010010	≡	11110000		
=	00111101	[	01011011	y	01111001	ù	10010111	╡	10110101	╙	11010011	±	11110001		

How can a computer represent words and letters using bits?

- *EBCDIC* (Extended Binary-Coded Decimal Interchange Code) is an alternative 8-bit used by older IBM systems
- *Unicode* uses 16 bits and provides codes for 65,000 characters – a bonus for representing alphabets of multiple languages
  - Used for foreign language support

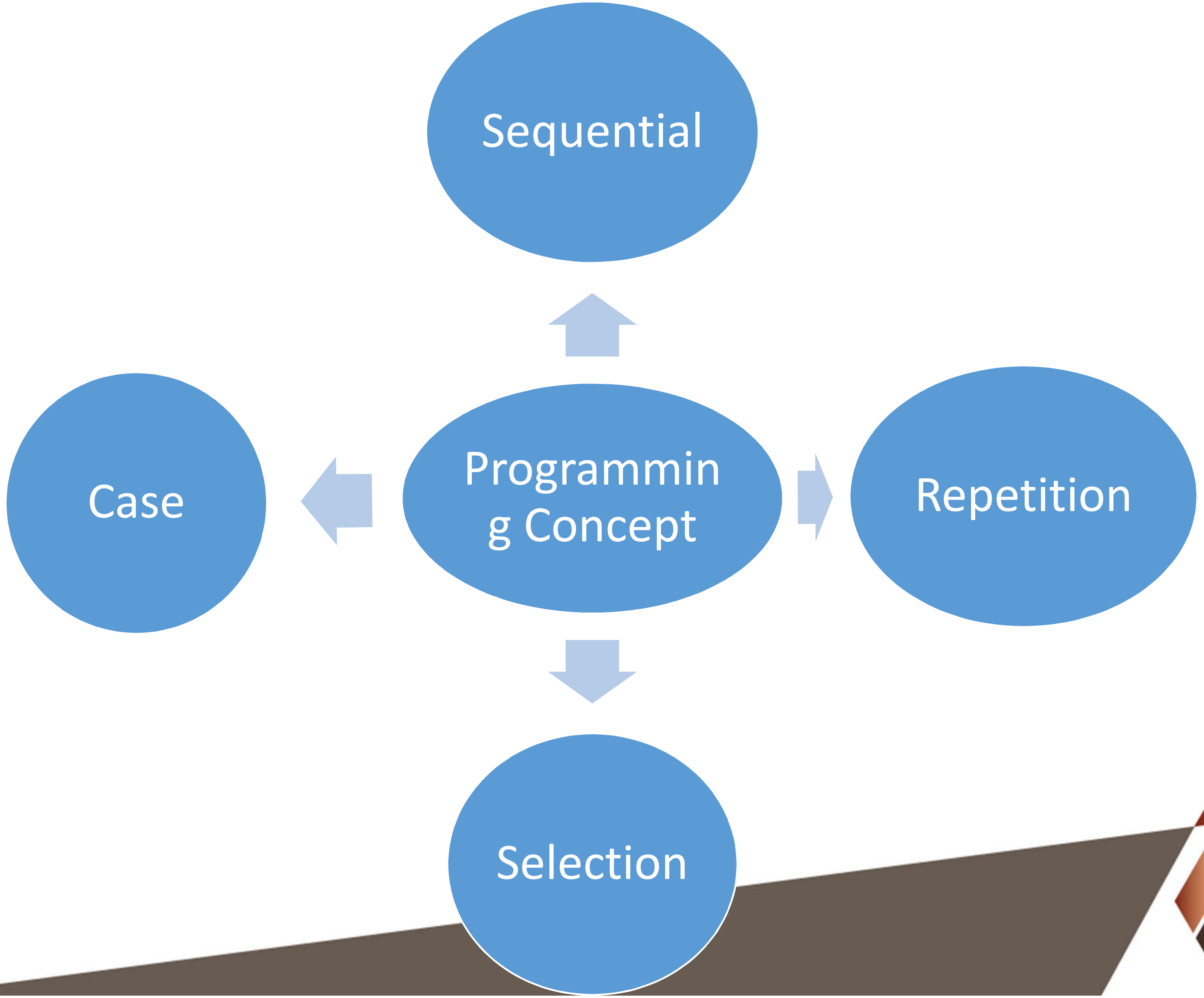
How does a computer convert sounds and pictures into codes?

- Sounds and pictures must be transformed into a format the computer can understand
- A computer must *digitize* colors, notes, and instrument sounds into 1s and 0s
- For example, a red dot on your screen might be represented by 1100, a green dot by 1101

How does a computer store all these codes?

- Data is stored on a computer in a *file*
  - *Data files* might contain the text of a document, the numbers for a calculation, the contents of a web page, or the notes of a music clip as binary code
  - *Executable files* contain the programs or instructions that tell the computer how to perform a specific task. For example, how to display and print text

Computer Programing



Programmin g Concept

Sequential

Repetition

Selection

Case

Sequential Programming

- The program works in order.
- One statement completes before move to the next statement.
- Today !!!!!

What is a computer program?

- A computer program or application is a set of instructions, written in a programming language, that enables a computer to perform some specified task.

A First Program

Problem statement

Write program that displays the line of text:

“Hello World”

Java Solution

```
// This application prints “HelloWorld" on the screen

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

Output :

```
Hello World
```

Line 1

1. // This application prints “Hello World" on the screen

- Line 1 is a comment.
- A comment explains or clarifies the meaning of some section of a program.
- A single line comment begins with the compound symbol // (two forward slashes)
- Once the compiler recognizes the beginning of a single line comment, the compiler skips all subsequent text on that line.

Line 2

2. public class FirstProgram

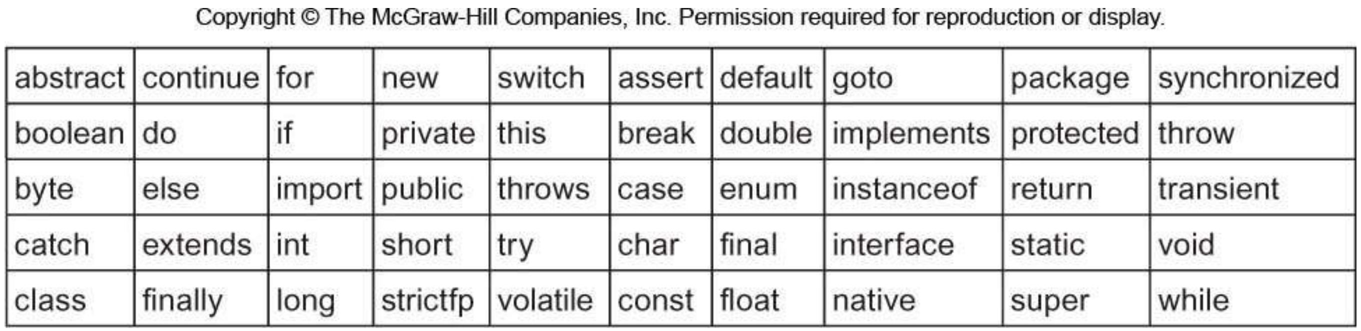
- Line 2 begins with two special words – *public* and *class*.
  For now, a *class* is an application or program and every class begins with the words *public* and *class* followed by the class name.
- FirstProgram is the name of the class. The class name must be a valid Java identifier.
- A valid Java identifier is a "word" of arbitrary length composed of letters and/or digits and/or two special characters $ (dollar sign) and _ (underscore), where the first character must be a letter.
- Examples: R2D2 or HarryPotter

Line 2

- Java is case sensitive. The name FirstProgram is considered different than FIRSTProgram.
- Keywords or reserved words are words that may not be used as Java identifiers
- A class name begins with an uppercase letter
- Spaces may not be part of a name. Uppercase letters are commonly used to separate "words"

Keyword

- Words which are used by the grammar for a certain function



abstract	continue	for	new	switch	assert	default	goto	package	synchronized
boolean	do	if	private	this	break	double	implements	protected	throw
byte	else	import	public	throws	case	enum	instanceof	return	transient
catch	extends	int	short	try	char	final	interface	static	void
class	finally	long	strictfp	volatile	const	float	native	super	while

Lines 3 and 8 {.....}

- The curly braces “{“ and “}” on lines 3 and 8 mark the beginning and the end of the HelloWorld class that comprises our application.
- A group of statements or instructions enclosed by curly braces is called a *block*.
- The body or executable section of a class is contained within these matching braces.

Lines 3 and 8

- The general structure of a class is:

```
public class ClassName
{
    // class body of block

}
```

Lines 4 - 7

```
4. public static void main(String[] args)
5. {
6.     System.out.println (“Hello World");
7. }
```

- The line “public static void main (String[] args)” is the first line or the heading of the class’s *main method*.

- A method is a named section of a class that performs a task.

Lines 4, 5, and 7

- The main method is automatically executed when a program runs.
- The statements of the main method are executed first.
- The main method is the starting point of every program.
- Every application must have a main method. Every main method begins with the same first line.
- The curly braces of lines 5 and 7 mark the beginning and the end of the main method.
- The actions that the main method performs are included between these curly braces.

Lines 4, 5, and 7

- A simple Java program has the following skeletal format:

```
public class ProgramName
{
    public static void main (String args [])
    {
       // executable statements go here
     }
}
```

Line 6

```
6.    System.out.println (“Hello World");
```

- This line instructs the computer to print Hello World on the screen.
- The quoted text (“Hello World") is called a *string literal* or more simply a string.
- A string literal must be contained on a single line.

Line 6

- The quotation marks are not part of the string literal.
- The quotation marks indicate the beginning and the end of the string.
- This line instructs the computer to display the string literal on the screen.
- The statement also prints the newline character, that is, it advances the cursor to the next line.

Line 6

- All statements are terminated with a semi-colon (;)
- The program must be saved in a file named FirstProgram.java.

Variation

```
// This program prints “Hello World" on the screen
public class SecondProgram
{
    public static void main( String[] args)
    {
        System.out.print (“Hello ");
                    // print NOT println
        System.out.print (“World");
    }
}
```

Output

```
Hello World
```

Output using *print* instead of *println* does not include the newline character.

Data Types and Expressions

Problem Statement

Write an application that calculates the number of minutes in a leap year using the fact that there are 525,600 minutes in a 365 day year.

Solution

```
//Calculates the number of minutes in a leap year
// Uses the fact that there are 525,600 minutes in a 365 day year
public class LeapYearMinutes
{
    public static void main(String[] args)
    {
        System.out.print( "The number of minutes in a leap year is ");
        System.out.println( 60*24 +525600);
                          // 60 min/hr  times 24 hr/day + 525600 min
    }
}
```

Output:

```
The number of minutes in a leap year is 527040.
```

Discussion

- The argument supplied to the println method is a numerical expression: 60* 24 + 525600.
- An expression is sequence of symbols that denotes, represents, or signifies a value.
- The value of the expression 60* 24 + 525600 is 527040.
- 60 and 24 are multiplied and the product is added to 525600.
- 527040, is given to the println method.
- * and + are called operators and the numbers 60, 24, and 525600 are called operands.

Data Types

A *data type* is a set of values together with an associated collection of operators for manipulating those values.

Type int

- The values associated with the data type *int* are integers in the range:

  –2,147,483,648 to 2,147,483,647

- The associated operators that manipulate integers are:

  + -- addition

  – -- subtraction

  * -- multiplication

  / -- division

  % -- modulus

- The keyword is the “int”

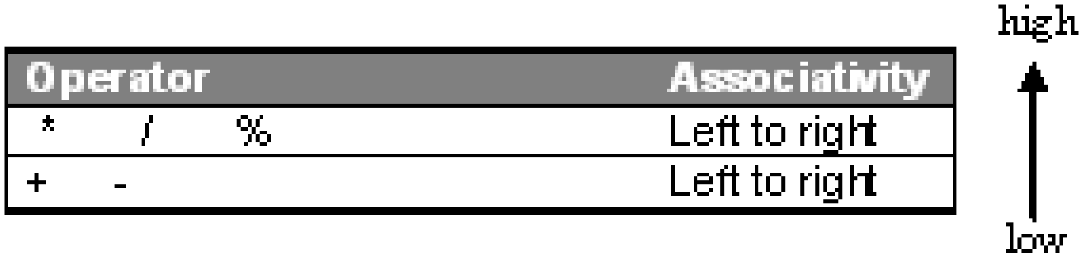
Type int

- The / operator denotes integer division
- a/b evaluates to a divided by b, discarding any remainder
- 5/2 evaluates to 2
- −23/6 evaluates to −3
- 4/43 has the value 0
- The expression a%b evaluates to the remainder of a divided by b
- 5%2 has the value 1
- −23%3 has the value −2

Operator Precedence

- The order in which operations are performed is the same as in ordinary arithmetic.

- For integer expressions, operations are performed according to the following precedence (priority) rules:



Operator	Associativity
* / %	Left to right
+ -	Left to right

Operator Precedence

- *, /, and % have the highest precedence
- *, /, and % are equal in precedence
- + and − are equal in precedence but lower than *, / and %
- Operations of equal precedence have left to right associativity

Operator Precedence

- You may explicitly change the order of operations by inserting parentheses into an expression.

- An expression enclosed by parentheses must always be fully evaluated before it can be used in a larger expression.

Operator Precedence

The day of the week for any date can be found with the following formula:

Day of the week =

((day +(13*((month+9)%12+1)-1)/5 + year%100+year%100/4+ year/400–2*(year/100))%7+7)%7 +1

- Day of the week is a number between 1 and 7 (Sunday= 1, Monday =2 …, Saturday = 7),
- Day is the day of the month (1...31),
- Month is encoded as January = 1, February = 2...December = 12
- Year is the four-digit year in question.

Type double

- The values are decimal numbers in the range -1.7×10^{308} … 1.7×10^{308} with 14 significant digits of accuracy.
- The operators associated with type double are

  + -- addition

  - -- subtraction

  * -- multiplication

  / -- division
- The division operator (/) denotes decimal or floating point division rather than integer division; so 5.0/2.0 has the value 2.5 but 5/2 has the value 2

- The keyword is *double*.

Type char

- Type *char* is the set of all characters found on the standard keyboard (in addition to thousands of other characters that are used for displaying text in languages that do not use the English alphabet).
- A value of type char is enclosed in single quotes:

  '5' is a value of type char,

  "5" is a string literal, and

  5 is an integer.

Type char

- The ASCII code assigns a non-negative integer between 0 and 127 to each character

- These values are stored as binary numbers, typically a leading 0 followed by a 7-bit code number between 0 to 127 inclusive

- ASCII values can be stored using a single byte of memory; that is, one character requires just one byte of storage

Type char

- Java uses the Unicode character set and allocates two bytes of memory for each character.

- Using two bytes allows 65,536 characters.

- Unicode set includes English characters and also characters for many other languages.

- By design, the ASCII character set is a subset of Unicode.

Type char

- Type type char includes several special characters that are represented by an escape sequence or escape character:

\n newline

\t tab

\b backspace

\r carriage return

\' single quote

\\ backslash

Type string

- Sequence of character

- The keyword is “String.”

Activities

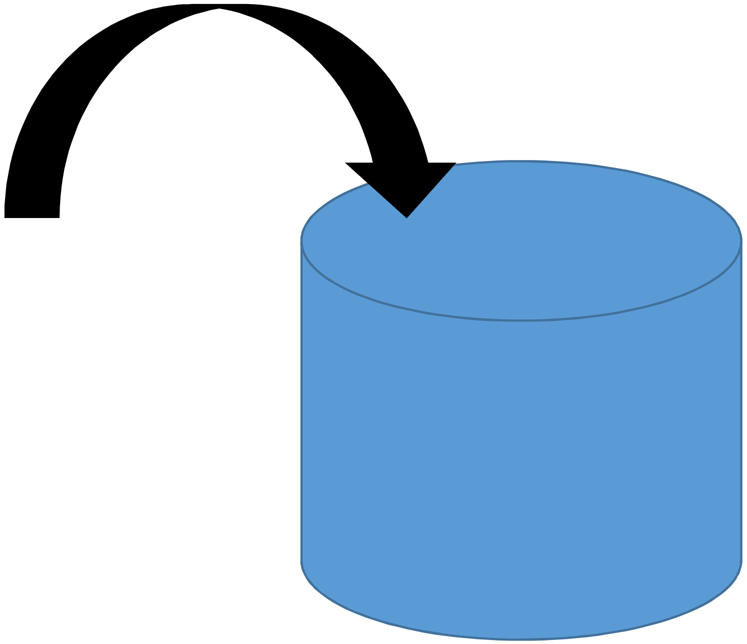
- Calculate the value
- 2+3
- 2+3*6
- (2+3*6)/3
- 2*((2+3*6)/3)
- 2*(2+3*6)/3
- 2*(2+3*6)/3.0

Activities
-Answer

- Calculate the value
- 2+3 = 5
- 2+3*6 = 20
- (2+3*6)/3 = 6
- 2*((2+3*6)/3) = 12
- 2*(2+3*6)/3 = 13
- 2*(2+3*6)/3.0 = 13.33

Variable

- The variable is a place in the memory to store value.



Variable declaration

- Is a process to create the variable in the program.

```
[variable type]  [variable name];
```

```
int number;
```

```
String firstName;
```

```
double salary;
```

Variable declaration

- The variable must start by a-z, A-Z,$,_.
- The a-z,A-Z,$,_,0-9 can be used as a part of the variable name.
- The uppercase and the lowercase are different.

```
public class FirstProgram
{
        public  static  void  main(String[] args)
        {
                int num = 0;
        }
}
```

Variable Usage

- Is the process to access the value in the variable
- Use the name of the variable to access the value in the source code

```java
public class FirstProgram
{
        public  static  void  main(String[] args)
       {
               int num = 0;
               System.out.println(num);
        }
}
```

Variable Assignment

- Is the process to give a new value to a variable.

- Denote by “ = “
- The value from the right hand side to the variable in the left hand side.

[variable name]= [expression];

```
public class FirstProgram
{
        public  static  void  main(String[] args)
        {
                int num = 0;
                System.out.println(num);
                num = 10;
                System.out.println(num);

        }
}
```

```
public class FirstProgram
{
        public  static  void  main(String[] args)
        {
                int num=100;
                System.out.println(num);
                num = 10;
                System.out.println(num);

        }
}
```

Activities - Data type Selection

- Answer the question with the reason
- What is the data type you will use for ?
  - Money in your pocket
  - Your firstname
  - Your GPA
  - Your gender
  - Amount of product

Example

- **Problem Statement**

Write a program that exchanges the values in two variables.

- The first value is your name in a variable named *myName*.
- The second value is the name of your friend in your left side in a variable named *friendName*.

Example

- **Problem Statement**

Write a program that displays 5 Fibonacci number.

Example

- **Problem Statement**

Write a program that calculate the area of a circle with radius of 5.0

.

Example

- **Problem Statement**

Write a program that calculate tax of a product.

- The product is 100 baht.
- The tax rate is 7%.

Obtaining Data from Outside a Program

- A very simple mechanism available for interactive input, a **Scanner** object.

Example

- According to the Farmer’s Almanac, you can estimate air temperature by
  - counting the number of times per minute that a cricket chirps
  - To compute the air temperature (Celsius), divide the number of chirps/minute by 6.6 and add 4.

Problem Statement

Write an application that calculates the air temperature given the number of cricket chirps per minute. A user supplies the number of chirps per minute.

Solution

```java
//calculates the air temperature (Celsius) from the number of cricket chirps/minute
import java.util.*;
public class Cricket
{
    public static void main (String[] args)
    {
        int chirps;         //chirps per minute
        double temperature;     // Celsius
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of chirps/minute:");
        chirps = input.nextInt();
        temperature = chirps/6.6 + 4;
        System.out.println("The temperature is “+temperature+"C");
    }
}
```

Output

```
Enter the number of chirps/minute: 99

The temperature is 19.0C
```

Discussion

Line 7: int chirps;

- Declare an integer variable, chirps.

Line 8: double temperature;

- The variable temperature holds the air temperature
- As the computation of the temperature requires division by 6.6
- Temperature is declared as double.

Line 9: Scanner input = new Scanner(System.in) ;

- The name input refers to a “Scanner object.”

Discussion

Line 10: System.out.print("Enter the number of chirps/minute: ");

- An output statement that prompts the user for data.

Line 11: chirps = input.nextInt();

- The Scanner object, input, accepts or reads one integer from the keyboard
- Once the user supplies an integer
  - That number is assigned to the variable chirps.
- The Scanner object, input, expects an integer (`input.nextInt()`).
- A Scanner object skips leading whitespace

-

Discussion

Line 12: temperature = chirps/6.6 + 4;

- The value stored in chirps is used to compute the air temperature
- The result of the computation is assigned to the variable temperature.

Line 13: System.out.println("The temperature is "+temperature+"C");

- The program displays the value stored in temperature
- along with some explanatory text

A Scanner Object for Interactive Input

- Before using a Scanner object for input you must:
  - Include the import statement: `import java.util.*;`
  - Declare a Scanner object as

```
Scanner name = new Scanner(System.in)
```

where `name` is a valid Java identifier

such as input or keyboardReader.

A Scanner Object for Interactive Input

- Once a Scanner has been declared you can use the following methods to read data:
  - `name.nextInt()`
  - `name.nextShort()`
  - `name.nextLong()`
  - `name.nextDouble()`
  - `name.nextFloat()`
  - `name.nextBoolean()`

  where `name` is the declared name of the Scanner

Example

- **Problem Statement**

Write a program that receive the first name, the last name, the age , the birthday and the school from user and display on the screen.

Example

- **Problem Statement**

Write a program that receive 2 integers and displays the multiplication of the number.

Example

- **Problem Statement**

Write a program that calculate the area of a circle with radius that user inputs.

Example

- **Problem Statement**

  Write a program that calculate tax of a product.

  - User inputs the product price.
  - The tax rate is 7%.

Example

- **Problem Statement**

Write a program that receives the grade of 5 subjects from user and displays the GPA.

  - The subjects are Thai, English, Society, Math and Chemistry.