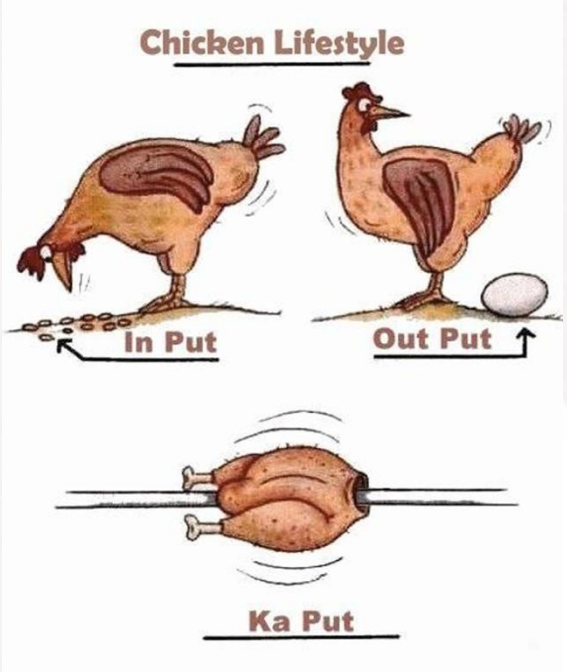
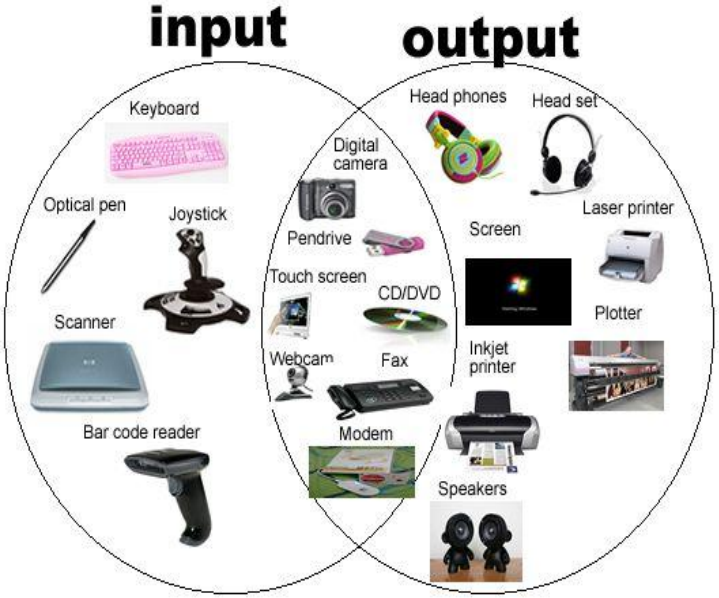
Chapter 2 – Console Input & Output





Printf format

- printf, it is a special printing that allow programmer to product a specific format

Syntax:
printf("[width] [.precision][Conversion character]", VariableName);

Width: How long to set it, if too short it will ignore
Precision: number after the decimal
Conversion char: Type of data, ie: s for String

Conversion character for printf

Display 2.1 **Format Specifiers for** `System.out.printf`

CONVERSION CHARACTER	TYPE OF OUTPUT	EXAMPLES
d	Decimal (ordinary) integer	%5d %d
f	Fixed-point (everyday notation) floating point	%6.2f %f
e	E-notation floating point	%8.3e %e
g	General floating point (Java decides whether to use E-notation or not)	%8.3g %g
s	String	%12s %s
c	Character	%2c %c

- %n is used for new line
- \\ is used for backlash

Example of printf

Code	Result
printf("'%-10s'", "Hello")	'Hello '
printf("'%10s'", "Hello");	' Hello'
printf("'%.2f'", 10.3456);	'10.35'
printf("'%8.4f'", 10.3456);	' 10.3456'
printf("'%08.2f'", 10.3456);	'00010.35'
printf("Hello\tworld");	Hello world
printf("Hello\nworld");	Hello world

Importing packages & classes

- All libraries in Java are called “packages”
- package: is a collection of classes already created
- To use the classes, we need to import the library
- Examples:
- import Java.util.Scanner;
- import Java.util.*; Import all classes from util into program

Console input using Scanner

- We can use Scanner for keyboard input
- to use it, we need to first import it and declare a variable

Example:

```
import Java.util.Scanner;

Scanner keyboardInput = new Scanner(System.in);

Int userInput = keyboardInput.nextInt();

System.out.println(userInput);
```

Create a scanner Object

Get the input from user

Print input from user

More data type for Scanner object

Display 2.8 Methods of the Scanner Class

Scannner_Object_Name.nextLong()

Returns the next value of type long that is typed on the keyboard.

Scannner_Object_Name.nextByte()

Returns the next value of type byte that is typed on the keyboard.

Scannner_Object_Name.nextShort()

Returns the next value of type short that is typed on the keyboard.

Scannner_Object_Name.nextDouble()

Returns the next value of type double that is typed on the keyboard.

Scannner_Object_Name.nextFloat()

Returns the next value of type float that is typed on the keyboard.

Display 2.8 Methods of the Scanner Class

Scannner_Object_Name.next()

Returns the String value consisting of the next keyboard characters up to, but not including, the first delimiter character. The default delimiters are whitespace characters.

Scannner_Object_Name.nextBoolean()

Returns the next value of type boolean that is typed on the keyboard. The values of true and false are entered as the strings "true" and "false". Any combination of upper- and/or lowercase letters is allowed in spelling "true" and "false".

Scanner_Object_Name.nextLine()

Reads the rest of the current keyboard input line and returns the characters read as a value of type String. Note that the line terminator '\n' is read and discarded; it is not included in the string returned.

Scanner_Object_Name.useDelimiter(*New_Delimiter*);

Changes the delimiter for keyboard input with *Scanner_Object_Name*. The *New_Delimiter* is a value of type String. After this statement is executed, *New_Delimiter* is the only delimiter that separates words or numbers. See the subsection "Other Input Delimiters" for details.

next vs nextLine

- next: will read only one string that is non-whitespace, as soon there a space it stop
- Example:

User Input	Stored in next
I love Obama	I

- nextLine: will read an entire line, meaning it will read all inputed until user hit enter
- Example:

User Input	Stored in nextLine
I hate my life	I hate my life

Delimiter

- Delimiter are used to separate word, it basically chop word into what value you want to separate. It will go to a new line once it is chopped

Syntax:
Object.useDelimiter(“CharToChop“);

Delimiter - Example

Display 2.10 **Changing the Input Delimiter**

```
import java.util.Scanner;

public class DelimiterDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard1 = new Scanner(System.in);
        Scanner keyboard2 = new Scanner(System.in);
        keyboard2.useDelimiter("##");
        //Delimiter for keyboard1 is whitespace.
        //Delimiter for keyboard2 is ##.
```

Specify what you want to chop

(continued)

Delimiter - Example

Display 2.10 Changing the Input Delimiter

```
        String word1, word2;
        System.out.println("Enter a line of text:");
        word1 = keyboard1.next();
        word2 = keyboard1.next();
        System.out.println("For keyboard1 the two words read are:");
        System.out.println(word1);
        System.out.println(word2);
        String junk = keyboard1.nextLine(); //To get rid of rest of line.

        System.out.println("Reenter the same line of text:");
        word1 = keyboard2.next();
        word2 = keyboard2.next();
        System.out.println("For keyboard2 the two words read are:");
        System.out.println(word1);
        System.out.println(word2);
    }
}
```

Get user input

Print what the user entered

(continued)

Delimiter - Example

SAMPLE DIALOGUE

```
Enter a line of text:
one two##three##
For keyboard1 the two words read are:
one
two##three##
Reenter the same line of text:
one two##three##
For keyboard2 the two words read are:
one two
three
```

Decimal Format

Decimal format are useful if you always need to print in a certain format, we can create a decimalFormat object to always print a specific pattern

DecimalFormat Class

Display 2.5 **The DecimalFormat Class**

```
import java.text.DecimalFormat;

public class DecimalFormatDemo
{
    public static void main(String[] args)
    {
        DecimalFormat pattern00dot000 = new DecimalFormat("00.000");
        DecimalFormat pattern0dot00 = new DecimalFormat("0.00");

        double d = 12.3456789;
        System.out.println("Pattern 00.000");
        System.out.println(pattern00dot000.format(d));
        System.out.println("Pattern 0.00");
        System.out.println(pattern0dot00.format(d));

        double money = 19.8;
        System.out.println("Pattern 0.00");
        System.out.println("$" + pattern0dot00.format(money));

        DecimalFormat percent = new DecimalFormat("0.00%");
```

Create a DecimalFormat object, with a specific pattern as argument

Use to pattern object and call the .format with d as argument

(continued)

DecimalFormat Class

Display 2.5 The DecimalFormat Class

```
        System.out.println("Pattern 0.00%");
        System.out.println(percent.format(0.308));

        DecimalFormat eNotation1 =
           new DecimalFormat("#0.###E0");//1 or 2 digits before point
        DecimalFormat eNotation2 =
           new DecimalFormat("00.###E0");//2 digits before point

        System.out.println("Pattern #0.###E0");
        System.out.println(eNotation1.format(123.456));
        System.out.println("Pattern 00.###E0");
        System.out.println(eNotation2.format(123.456));

        double smallNumber = 0.0000123456;
        System.out.println("Pattern #0.###E0");
        System.out.println(eNotation1.format(smallNumber));
        System.out.println("Pattern 00.###E0");
        System.out.println(eNotation2.format(smallNumber));
    }
}
```

(continued)

DecimalFormat Class

SAMPLE DIALOGUE

```
Pattern 00.000
12.346
Pattern 0.00
12.35
Pattern 0.00
$19.80
Pattern 0.00%
30.80%
Pattern #0.###E0
1.2346E2
Pattern 00.###E0
12.346E1
Pattern #0.###E0
12.346E-6
Pattern 00.###E0
12.346E-6
```

The number is always given, even if this requires violating the format pattern.