Methods Provided by Scanner Objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nextByte()</code></td>
<td>reads an integer of the <code>byte</code> type.</td>
</tr>
<tr>
<td><code>nextShort()</code></td>
<td>reads an integer of the <code>short</code> type.</td>
</tr>
<tr>
<td><code>nextInt()</code></td>
<td>reads an integer of the <code>int</code> type.</td>
</tr>
<tr>
<td><code>nextLong()</code></td>
<td>reads an integer of the <code>long</code> type.</td>
</tr>
<tr>
<td><code>nextFloat()</code></td>
<td>reads a number of the <code>float</code> type.</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>reads a number of the <code>double</code> type.</td>
</tr>
<tr>
<td><code>next()</code></td>
<td>reads a string that ends before a whitespace character.</td>
</tr>
<tr>
<td><code>nextLine()</code></td>
<td>reads a line of text (i.e., a string ending with the <code>Enter</code> key pressed).</td>
</tr>
</tbody>
</table>

\[1\]See Table 2-1 in YDL, p. 38.
Example: Mean and Standard Deviation

Write a program which takes 3 numbers as user input and calculates the mean and the standard deviation for them.

- The mean of 3 numbers is given by \( \overline{x} = \left( \sum_{i=1}^{3} x_i \right) / 3. \)
- Also, the resulting standard deviation is given by

\[
S = \sqrt{\frac{\sum_{i=1}^{3} (x_i - \overline{x})^2}{3}}.
\]

- You may use these two methods:
  - Math.pow(double x, double y) for \( x^y \).
  - Math.sqrt(double x) for \( \sqrt{x} \).
- See more methods within Math class.
Scanner input = new Scanner(System.in);
System.out.println("a = ?");
double a = input.nextDouble();
System.out.println("b = ?");
double b = input.nextDouble();
System.out.println("c = ?");
double c = input.nextDouble();
input.close();

double mean = (a + b + c) / 3;
double std = Math.sqrt((Math.pow(a - mean, 2) +
    Math.pow(b - mean, 2) +
    Math.pow(c - mean, 2)) / 3);

System.out.println("mean = " + mean);
System.out.println("std = " + std);
class Lecture3 {
    "Selections"
}

// Keywords
if, else, else if, switch, case, default
Flow Controls

- **Sequential statements**: execute instructions in order.
- **Selection**: first check if the predetermined condition is satisfied, then execute the corresponding instruction.
- **Repetition**: repeat the execution of some instructions until the criterion fails.
• Note that they are involved with each other generally.
• For example, recall how to find the maximum in the input list?
Selections

- One-way if statements.
- Two-way if-else statements.
- Nested if statements.
- Multi-way if-else if-else statements.
- switch-case statements.
- Conditional operators.
A one-way if statement executes an action if and only if the condition is true.
The keyword `if` is followed by the parenthesized condition.

The condition should be a boolean expression or a boolean value.

If the condition is `true`, then the statements in the selection body will be executed once.

If not, then the program won’t enter the selection body and skip the whole selection body.

Note that the braces can be omitted if the block contains only single statement.
Example

Write a program which receives a nonnegative number as input for the radius of a circle, and determines the area of the circle.

```java
...  
  double area;
  if (r > 0) {
    area = r * r * 3.14;
    System.out.println(area);
  }
  ...
…
```

• However, the world is not well-defined.
A two-way if-else statement decides which statements to execute based on whether the condition is true or false.

```java
... if (condition) {
    // body for the true case
} else {
    // body for the false case
}
...```

Two-Way if-else Statements
Flowchart:

- **boolean-expression**
  - **true**
    - Statement(s) for the true case
  - **false**
    - Statement(s) for the false case
Write a program which receives a number as input for the radius of a circle. If the number is nonnegative, then determine the area of the circle; otherwise, output “Not a circle.”

```java
double area;
if (r > 0) {
    area = r * r * 3.14;
    System.out.println(area);
} else {
    System.out.println("Not a circle.");
}
input.close();
```
Nested if Statements

• For example,

```java
... if (score >= 90) 
    System.out.println("A");
else {
    if (score >= 80) 
        System.out.println("B");
    else {
        if (score >= 70) 
            System.out.println("C");
        else {
            if (score >= 60) 
                System.out.println("D");
            else 
                System.out.println("F");
        }
    }
}
...```
Multi-Way if-else

• Redo the previous problem:

```java
... if (score >= 90) 
    System.out.println("A"); 
else if (score >= 80) 
    System.out.println("B"); 
else if (score >= 70) 
    System.out.println("C"); 
else if (score >= 60) 
    System.out.println("D"); 
else 
    System.out.println("F"); 
...
```

• An if-elseif-else statement is a preferred format for multiple alternatives, in order to avoid deep indentation and make the program easy to read.
The order of conditions may be relevant. (Why?)

```java
... if (score >= 90 && score <= 100)
  else if (score >= 80 && score < 90)
    ...
  else
    ...
```

The performance may degrade due to the order of conditions. (Why?)
Common Errors

Don’t add a semicolon to the condition (in Line 3).

• If you do so in Line 3, this statement is not effective (useless).

• Multiple conditional statements should be grouped by braces.
Write a program which shows a math question, say sum of two random integers ranging from 0 to 9, and asks the user to answer.

- For example, the program shows $2 + 5 = ?$
- If the user types 7, then the program reports “Correct.”
- Otherwise, the program reports “Wrong answer. The correct answer is 7.”
- You may use `Math.random()` for a random value between 0.0 and 1.0, excluding themselves.\(^2\)

\(^2\)You may see PRNG in [https://en.wikipedia.org/wiki/Pseudorandom_number_generator](https://en.wikipedia.org/wiki/Pseudorandom_number_generator)
Can you extend this program for all arithmetic expressions (i.e., + − × ÷)?
“Exploring the unknown requires tolerating uncertainty.”

– Brian Greene

“I can live with doubt, and uncertainty, and not knowing. I think it is much more interesting to live not knowing than have answers which might be wrong.”

– Richard Feynman
Write a program which determines the maximum value in 3 random integers ranging from 1 to 100.

- How many variables do we need?
- How to compare?
- How to keep the maximum value?
• For any number of data, this program is limited.
• We need arrays and loops for general cases.
A switch-case structure takes actions depending on the target variable.

```java
... switch (target) {
    case v1:
        // statements
        break;
    case v2:
        .
        .
        case vk:
            // statements
            break;
    default:
        // statements
}
...```

Zheng-Liang Lu  Java Programming
• A switch-case statement is convenient for multiple discrete conditions.

• The variable target, always enclosed in parentheses, must yield a value of char, byte, short, int, or String type.

• The type of \( v_1, \ldots, \) and \( v_k \) must be the same as target.

• For each case, a break statement is needed to leave the construct.\(^3\)

• The default case, which is optional, can be used to perform actions when none of the specified cases matches target.
  • Counterpart to else statements.

\(^3\)If not, there will be a fall-through behavior.
...  
  // We define the traffic lights as follows:  
  // RED: 0  
  // YELLOW: 1  
  // GREEN: 2  

  int trafficLight = (int) (Math.random() * 3);

  switch (trafficLight) {
    case 0:
      System.out.println("Stop!!!");
      break;
    case 1:
      System.out.println("Slow down!!");
      break;
    case 2:
      System.out.println("Go!");
  }
...
Conditional Operators

A conditional expression evaluates an expression based on the specified condition and returns a value accordingly.

```java
... var = A ? B : C; ...
```

- This is the only ternary operator in Java.
- If A is `true`, then return the result of B; otherwise, return the result of C.
• For example,

```java
... 
    if (num1 > num2) 
        max = num1; 
    else 
        max = num2;
...
```

• Alternatively, you can do this:

```java
... 
    max = num1 > num2 ? num1 : num2;
...
```
class Lecture4 {

    "Loops"
}

// keywords:
while, do, for, break, continue
A loop can be used to make a program execute statements repeatedly without having to code the same statements.

• For example, output “Hello, Java.” for 100 times.

```java
...  
System.out.println("Hello, Java.");
System.out.println("Hello, Java.");
.
  // copy and paste for 100 times
  .
System.out.println("Hello, Java.");
...  
```

This is a toy example to show the power of loops.

In practice, any routine which repeats couples of times can be done by folding them into a loop.

\[\text{int cnt = 0;}
\text{while (cnt < 100) {}
\text{System.out.println("Hello, Java.");}
\text{cnt++;}
\text{}}\]
Loops provide substantial computational power.
Loops bring an efficient way of programming.
Loops could consume a lot of time.\footnote{We will introduce the analysis of algorithms soon.}
while Loops

A while loop executes statements repeatedly while the condition is true.

```

... while (condition) {
    // loop body
}
...
```

- The condition should be a boolean expression which determines whether or not the execution of the body occurs.
- If true, the loop body is executed and check the condition again.
- Otherwise, the entire loop terminates.
loop-continuation-condition?

true

Statement(s) (loop body)

false

count = 0;

(count < 100)?

true

System.out.println("Welcome to Java!"); count++;

false
Example

Write a program which sums up all integers from 1 to 100.

• In math, the question can be written as:

\[ \text{sum} = 1 + 2 + \cdots + 100. \]

• But this form is not doable in the machine.\(^7\)

\(^7\)We need to develop computational thinking. Read http://rsta.royalsocietypublishing.org/content/366/1881/3717.full or http://blog.orangeapple.tw/posts/what-is-computational-thinking/.
• Normally, the machine executes the instructions **sequentially**.
• So one needs to decompose the math equation into several steps, like:

```java
int sum = 0;
sum = sum + 1;
sum = sum + 2;
sum = sum + 100;
```

• It is obvious that many similar statements can be found.
Using a `while` loop, the program can be rearranged as follows:

```java
...  
int sum = 0;  
int i = 1;  
while (i <= 100) {
    sum = sum + i;  
    ++i;  
}  
...  
```

- You should guarantee that the loop will terminate as expected.
- In practice, the number of loop steps (iterations) is unknown until the input data is given.
Malfunctioned Loops

• It is really easy to make an infinite loop.

1 \quad \ldots
2 \quad \textbf{while} \ (\textbf{true});
3 \quad \ldots

• The common errors of the loops are:
  • never start
  • never stop
  • not complete
  • exceed the expected number of iterations
Example

Write a program which asks the sum of two random integers and lets the user repeatedly enter a new answer until correct.

```java
Scanner input = new Scanner(System.in);
int x = (int) (Math.random() * 10);
int y = (int) (Math.random() * 10);
int ans = x + y;

System.out.println(x + " + " + y + " = ? ");
int z = input.nextInt();

while (z != ans) {
    System.out.println("Try again? ");
    z = input.nextInt();
}
System.out.println("Correct.");
input.close();
```
Loop Design Strategy

- Writing a correct loop is not an easy task for novice programmers.
- Consider 3 steps when writing a loop:
  - **Find the pattern**: identify the statements that need to be repeated.
  - **Wrap by loops**: put these statements in the loop.
  - **Set the continuation condition**: translate the criterion from the real world problem into computational conditions.\(^8\)

\(^8\)Not unique.
Another common technique for controlling a loop is to designate a special value when reading and processing a set of values.

- This special input value, known as a sentinel value, signifies the end of the loop.
- For example, the operating systems and the GUI apps.
Example: Cashier Problem

Write a program which sums over positive integers from consecutive inputs and then outputs the sum when the input is nonpositive.

```java
... int total = 0, price = 0;
Scanner input = new Scanner(System.in);

System.out.println("Enter price?");
price = input.nextInt();
while (price > 0) {
    total += price;
    System.out.println("Enter price?");
    price = input.nextInt();
    // These two lines above repeat Line 5 and 6?!
}

System.out.println("Total = " + total);
input.close();
...