class Lecture3 {
    "Selections"
}

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

The basic algorithm (and program) is constituted by the following operations:

- **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 \texttt{if} statements
- Two-way \texttt{if-else} statements
- Nested \texttt{if} statements
- Multiway \texttt{if-else if-else} statements
- \texttt{switch-case} statements
- Conditional operators
One-Way \textbf{if} Statements

A one-way \textbf{if} statement executes an action \textbf{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
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

• Let’s redo the previous problem.

...  
  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

```java
... double area;
if (r > 0);
    area = r * r * 3.14;
System.out.println(area);
...
```
Generating random numbers

Write a program which generates 2 random integers and asks the user to answer the math expression.

- For example, the program shows $2 + 5 = ?$
- If the user answers 7, then the program reports “Correct.” and terminates.
- Otherwise, the program reports “Wrong answer. The correct answer is 7.” for this case.
- You may use `Math.random()` for a random value between 0.0 and 1.0, excluding themselves.
```java
int x = (int) (Math.random() * 10); // integers 0 ~ 9
int y = (int) (Math.random() * 10);
int answer = x + y;

System.out.println(x + " + " + y + " = 
Wrong. Answer: " + answer);
input.close();
```

- Can you extend this program for all arithmetic expressions (i.e., $+, -, \times, \div$)?
**Exercise**

**Find Max**

Write a program which determines the maximum value in 3 random integers whose range from 0 to 99.

- How many variables do we need?
- How to compare?
- How to keep the maximum value?
In this case, a **scalar** variable is not convenient. (Why?)

So we need **arrays** and **loops**.
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
}
...
- A **switch-case** statement is more convenient than an if statement for multiple **discrete** conditions.
- The variable *target*, always enclosed in parentheses, must yield a value of *char*, *byte*, *short*, *int*, or **String** type.
- The value \( v_1, \ldots, \) and \( v_k \) must have the same data type as the variable *target*.
- In each case, a **break** statement is a must.\(^1\)
  - **break** is used to break a construct!
- The **default** case, which is optional, can be used to perform actions when none of the specified cases matches *target*.
  - Counterpart to **else** statements.

\(^1\)If not, there will be a fall-through behavior.
... 

// 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.

```
... someVar = booleanExpr ? exprA : exprB;
... 
```

- This is the only ternary operator in Java.
- If the `boolean` expression is evaluated `true`, then return `expr A`; otherwise, `expr B`.

Zheng-Liang Lu

Java Programming
• For example,

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

• Alternatively, one can use a conditional expression like this:

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

    "Loops"

}

// keywords:
while, do, for, break, continue
Loops

A loop can be used to make a program execute statements repeatedly without having to code the same statements.

• For example, a program outputs “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 simple example to show the power of loops.

In practice, any routine which repeats couples of times\(^2\) can be done by folding them into a loop.

\(^2\)I prefer to call them “patterns.”
Loops provide substantial computational power.
Loops bring an efficient way of programming.
Loops could consume a lot of time.\(^3\)

\(^3\)We will visit the analysis of algorithms in the end of this lecture.
while Loops

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

```java
...
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?

false
true

Statement(s) (loop body)

false
true

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

count = 0;

(count < 100)?

false
true

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.\(^4\)

\(^4\)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.

```java
... while (true);
...
```

• 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.\(^5\)

\(^5\)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.

```
...  
    int sum = 0;
Scanner input = new Scanner(System.in);
System.out.println("Enter price?");
int price = input.nextInt();
while (price > 0) {
    sum += price;
    System.out.println("Enter price?");
    price = input.nextInt();
}
System.out.println("Total = " + sum);
input.close();
...
```

- Line 8 and 9 are the recurrence of Line 3 and 4?!
do-while Loops

A do-while loop is similar to a while loop except that it does execute the loop body first and then checks the loop continuation condition.

```java
1. ...
2. do {
3.     // loop body
4. } while (condition); // Do not miss the semicolon!
5. ...
```

- Note that there is a semicolon at the end of the do-while loop.
- The do-while loops are also called posttest loops, in contrast to while loops, which are pretest loops.
Write a program which sums over positive integers from consecutive inputs and then outputs the sum when the input is nonpositive.

```java
...
int sum = 0, price = 0;
Scanner input = new Scanner(System.in);
do {
    sum += price;
    System.out.println("Enter price?"FOUND IN TEXT);
    price = input.nextInt();
} while (price > 0);
System.out.println("Total = " + sum);
input.close();
...
A for loop generally uses a variable to control how many times the loop body is executed.

```
... for (init_action; condition; increment) {
    // loop body
}
...```

- **init-action**: declare and initialize a variable
- **condition**: set a criterion for loop continuation
- **increment**: how the variable changes after each iteration
- Note that these terms are separated by semicolons.
Example

Sum from 1 to 100

Write a program which sums from 1 to 100.

```java
...  
int sum = 0;
for (int i = 1; i <= 100; ++i)
    sum = sum + i;
...
```

- Compared to the while version,

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

Write a program which displays all even numbers smaller than 100.

- An even number is an integer of the form $x = 2k$, where $k$ is an integer.
• You may use the modular operator (%).

```java
... 
for (int i = 1; i <= 100; i++) {
    if (i % 2 == 0) System.out.println(i);
}
...
```

• Also consider this alternative:

```java
... 
for (int i = 2; i <= 100; i += 2) {
    System.out.println(i);
}
...
```

• How about odd numbers?
Example: Monte Carlo Simulation\textsuperscript{6}

- Write a program which conducts a Monte Carlo simulation to estimate $\pi$.

\textsuperscript{6}See https://en.wikipedia.org/wiki/Monte_Carlo_method.
Jump Statements

The keyword `break` and `continue` are often used in repetition structures to provide additional controls.

- **break**: the loop is terminated right after a `break` statement is executed.
- **continue**: the loop skips this iteration right after a `continue` statement is executed.
- In practice, jump statements in loops should be conditioned.
Example: Primality

Write a program which determines if the input integer is a prime number.

• Let $x > 1$ be any natural number.
• Then $x$ is said to be a prime number if $x$ has no positive divisors other than 1 and itself.
• It is then straightforward to check if it is prime by dividing $x$ by all natural numbers smaller than $x$.
• For speedup, you can divide $x$ by only numbers smaller than $\sqrt{x}$. (Why?)
Scanner input = new Scanner(System.in);

System.out.println("Enter x > 2?");

int x = input.nextInt();

boolean isPrime = true;
input.close();

double upperBd = Math.sqrt(x);

for (int y = 2; y < upperBd; y++) {
    if (x % y == 0) {
        isPrime = false;
        break;
    }
}

if (isPrime) {
    System.out.println("Prime");
} else {
    System.out.println("Composite");
}
Exercise (Revisited)

- Redo the cashier problem by using an infinite loop with a break statement.

```java
... while (true) {
    System.out.println("Enter price?");
    price = input.nextInt();
    if (price <= 0) break;
    sum += price;
}
System.out.println("Total = " + sum);
...
```
Another Example: Compounding

Write a program which determines the holding years for an investment doubling its value.

- Let $balance$ be the current amount, $goal$ be the goal of this investment, and $r$ be the annual interest rate.
- Then this investment should take at least $n$ years so that the balance of the investment can double its value.
- Recall that the compounding formula is given by

$$balance = balance \times (1 + r/100).$$
... 

```java
int r = 18; // 18%
int balance = 100;
int goal = 200;

int years = 0;
while (balance <= goal) {
    balance *= (1 + r / 100.0);
    years++;
}

System.out.println("Balance = " + balance);
System.out.println("Years = " + years);
```
...

A for loop can be an infinite loop by setting true or simply leaving empty in the condition statement.

An infinite for loop with an if-break statement is equivalent to a normal while loop.
In general, a for loop may be used if the number of repetitions is known in advance. If not, a while loop is preferred.
Nested Loops

A loop can be nested inside another loop.

- Nested loops consist of an outer loop and one or more inner loops.
- Each time the outer loop is repeated, the inner loops are reentered, and started anew.
Example

Multiplycation table

Write a program which displays the multiplication table.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
</tr>
</tbody>
</table>
You can use `System.out.printf()` to display formatted output on the console.

```java
... double amount = 1234.601;
double interestRate = 0.00528;
double interest = amount * interestRate;
System.out.printf("Interest = %4.2f", interest);
...
```
<table>
<thead>
<tr>
<th>Format Specifier</th>
<th>Output</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%b</td>
<td>a Boolean value</td>
<td>true or false</td>
</tr>
<tr>
<td>%c</td>
<td>a character</td>
<td>‘a’</td>
</tr>
<tr>
<td>%d</td>
<td>a decimal integer</td>
<td>200</td>
</tr>
<tr>
<td>%f</td>
<td>a floating-point number</td>
<td>45.460000</td>
</tr>
<tr>
<td>%e</td>
<td>a number in standard scientific notation</td>
<td>4.556000e+01</td>
</tr>
<tr>
<td>%s</td>
<td>a string</td>
<td>“Java is cool”</td>
</tr>
</tbody>
</table>

- By default, a floating-point value is displayed with 6 digits after the decimal point.
Multiple Items to Print

```java
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);
```

display:
```
count is 5 and amount is 45.560000
```

- Items must match the format specifiers in order, in number, and in exact type.
- If an item requires more spaces than the specified width, the width is automatically increased.
- By default, the output is right justified.
- You may try the plus sign (+), the minus sign (-), and 0 in the middle of format specifiers.
  - Say % + 8.2f, % – 8.2f, and %08.2f.
public static void main(String[] args) {
    for (int i = 1; i <= 9; ++i) {
        for (int j = 1; j <= 9; ++j) {
            System.out.printf("%3d", i * j);
        }
        System.out.println();
    }
}
Exercise: Coupled Loops

```
*     ********     *     ********
**    *****      **    *****
***   ***        ***   ***
****   **        ****   **
*****   *        *****   *
********   *      ********   *
```

(a)  (b)  (c)  (d)
public class PrintStarsDemo {
    public static void main(String[] args) {
        // case (a)
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.printf("*");  
            }
            System.out.println();
        }
        // case (b), (c), (d)
        // your work here
    }
}