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.

```
... 
  do {
    // loop body
  } while (condition); // Do not miss the semicolon!
...
```

- Note that there must be a semicolon at the end of do-while loops.
- The do-while loops are also called posttest loops, in contrast to while loops, which are pretest loops.
Example (Revisited)

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);

do {
    total += price;
    System.out.println("Enter price?");
    price = input.nextInt();
} while (price > 0);

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

A for loop generally uses a variable to control how many times the loop body is executed.

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

- **init-action**: declare and initialize a variable.
- **condition**: for loop continuation.
- **increment**: how the loop variable changes after each iteration.
- Note that these terms are separated by semicolons.
Write a program which sums from 1 up to 100.

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

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

loop-continuation-condition?

Statement(s) (loop body)

action-after-each-iteration

false

ture
Exercise

Write a program which displays all even numbers between 1 and 100.

- 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?
Numerical Example: Monte Carlo Simulation

• Let $m$ be the number of sample points falling in the region of the quarter circle shown in the next page, $n$ be the total number of sample points.
  • Simply use `Math.random()` to generate a value between 0 and 1 (exclusive).
• Write a program which estimates $\pi$ by

$$\hat{\pi} = 4 \times \frac{m}{n}.$$

• Cute and sweet!
• Note that $\hat{\pi} \to \pi$ as $n \to \infty$ by the law of large numbers (LLN).

---

Numerical Example: Bisection Method for Root-Finding

- Assume that $f(x) = x^3 - x - 2$.
- Consider to find a root between $[a, b] = [1, 2]$ as initial guess.
- Write a program to calculate the approximate root $\hat{r}$ by using the bisection method.
- Note that we set an error tolerance, say $\epsilon = 1e-9$, to strike a balance between efficiency and accuracy.

---

2For most of numerical algorithms, say Newton’s method, an initial guess is a must. Even more, the solution is severely sensitive to the initial guess for some cases.

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 Test

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");
}
• 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;
    total += price;
  }
  System.out.println("Total = " + total);
...`

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

• Let \( \text{balance} \) be the current amount, \( \text{goal} \) be the goal of this investment, and \( r \) be the annual interest rate.
• Recall that the compounding formula is given by

\[
\text{balance} = \text{balance} \times (1 + \frac{r}{100}).
\]

• Then this investment should take at least \( n \) years so that the balance of the investment can double its value.
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.

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

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 pi = 3.1415926;
System.out.printf("pi = %4.2f", pi); // output 3.14
...
```

% 4 . 2f

- format specifier
- field width
- conversion code
- precision
By default, a floating-point value is displayed with 6 digits after the decimal point.

<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>
Multiple Items to Print

```
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.
- By default, the output is right justified.
- If an item requires more spaces than the specified width, the width is automatically increased.
- 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) {
        // In row i, output each j
        for (int j = 1; j <= 9; ++j) {
            System.out.printf("%3d", i * j);
        }
        System.out.println();
    }
}

- For each $i$, the inner loop goes from $j = 1$ to $j = 9$.
- As an analog, $i$ acts like the hour hand of the clock, while $j$ acts like the minute hand.
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
}