Unary Operator

• An operator that takes only a single operand
• Plus: +
• Minus: –
• Cast: (type). E.g. (double)
# Compound Assignment Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>x += y</td>
<td>x = x + y</td>
</tr>
<tr>
<td>-=</td>
<td>x -= y</td>
<td>x = x - y</td>
</tr>
<tr>
<td>*=</td>
<td>x *= y</td>
<td>x = x * y</td>
</tr>
<tr>
<td>/=</td>
<td>x /= y</td>
<td>x = x / y</td>
</tr>
<tr>
<td>%=</td>
<td>x %= y</td>
<td>x = x % y</td>
</tr>
</tbody>
</table>
Increment & Decrement Operator

- **Prefix increment ++** e.g. ++i
  Increase i by 1, then use the new value of i to evaluate the expression that i resides
- **Postfix increment ++** e.g. i++
  Use the current value of i to evaluate the expression that i resides, then increase i by 1
- **Prefix decrement --** e.g. --i
  Decrease i by 1, then use the new value of i to evaluate the expression that i resides
- **Postfix decrement --** e.g. I--
  Use the current value of i to evaluate the expression that i resides, then decrease i by 1
# Operator Precedence

<table>
<thead>
<tr>
<th>Operator</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>++  --</td>
<td>Unary postfix</td>
</tr>
<tr>
<td>++  --  +  -</td>
<td>Unary prefix</td>
</tr>
<tr>
<td>*  /  %</td>
<td>multiplicative</td>
</tr>
<tr>
<td>+  -</td>
<td>additive</td>
</tr>
<tr>
<td>&lt;  &lt;=  &gt;  &gt;=</td>
<td>relational</td>
</tr>
<tr>
<td>==  !=</td>
<td>equality</td>
</tr>
<tr>
<td>? :</td>
<td>conditional</td>
</tr>
<tr>
<td>=  +=  -=  *=  /=  %=</td>
<td>assignment</td>
</tr>
</tbody>
</table>
Pseudo Code

- Pseudo code is informal language that helps programmer THINK and develop algorithm without worrying about strict Java language.
- Pseudo code can be written in plain English.
- Pseudo code doesn’t need to execute on a computer.
Pseudo code and real code

- **Pseudo code**
  If test score is higher than or equal to 60, 
  Print out the message to console, “I passed”. 
  If the test score is lower than 60, 
  Print out the message to console, “I failed”.
- **Java code**
  ```java
  if (testScore >= 60) 
      System.out.println("I passed");
  else 
      System.out.println("I failed");
  ```
switch statement

• Multiple selection statement.
• Perform a different action dependent on the value of a variable or expression.
• The value of the variable or expression can only be any of data type of byte, short, int, or char.
Example of switch statement

switch (temperature/10)
{
    case 9:
        System.out.println("The temperature is higher than 90.");
        break; //Cannot omit this
    case 8:
        System.out.println("The temperature is between 80 and 90.");
        break;
    default:
        System.out.println("The temperature is between 70 and 80.");
        break;
}
break Statement

• The “break” statement, when executed in a “switch”, “while”, “for”, “do…while” causes immediate exit from that statement.
• Execution continues with the first statement after the control statement.
• Common uses of the break statement are to skip the remainder of a switch.
• Without a break statements, each time a match occurs in the switch, the statements for that case and subsequent cases execute until a break statement or the end of the switch is encountered. “Falling through” to the statements in subsequent cases.
• It is a good practice to have “break” statement at the end of each “case”, so that accidental falling through will not happen.
while loop

• A repetition statement (loop) allows a block of code between {} to repeat while some condition remains true.

• Braces can be omitted if the block statement is a single statement.

• `int i = 10;`
  ```
  while (i < 100)
    i = i + 10;
  ```
Counter-controlled Repetition

• Use a variable called a counter (or control variable) to control the number of repetition.
• Definite repetition. The number of repetition is known before the loop begins executing.
Example: WhileStatement
Nested Control Structure

• Control statement can be nested with each other to create a more power control structure.

```java
int j = 5;
    while (j > 0)
    {
        if (j > 2)
            System.out.println("i is "+ j + " and is greater than 2");
        else
            System.out.println("i is "+ j + " and is less than 2");
        j = j - 1;
    }
```
do...while repetition statement

- Similar to while repetition statement
- In while statement, the program will evaluate loop-continuation condition first before executing any statement.
- In do...while statement, the program will execute the statement at least once before evaluating loop-continuation condition
Compare do...while and while

- **do...while**
  int k = -1;
  do
  {
      System.out.println("k="+k); //This statement execute
  } while (k > 0);

- **While**
  int l = -1;
  while (l > 0)
  {
      System.out.println("l="+l); //This statement will not execute
  }
Demo do…while
Essential of Counter-controlled Repetition

- A control variable (loop counter)
- The initial value of control variable
- The increment (or decrement) by which the control variable is modified each time through the loop (each iteration of the loop).
- The loop-continuation condition that determines whether looping should continue.
while/do..while generalization

• While statement
Initialization;
while (loopContinuationCondition)
{
    statement;
    increment;
}

• Do..while statement
Initialization;
Do
{
    statement;
    increment;
} while (loopContinuationCondition)
Reading Assignment

• “Java 2” p78-83, 92-97, p100-106, p112-113