Chapter 5
Loops
COSC1046

Ping Luo

Loop

- Loop **>** Data types, variables, constants, assignments, Characters, and Strings, etc.

- Loop can solve complicated problems. Many algorithms are based on Loops.

Loop

- **loop** controls how many times an operation or a sequence of operations is performed in succession.


```
100 times  ───►   System.out.println("Programming is fun");
                  System.out.println("Programming is fun");
                  ...
                  System.out.println("Programming is fun");
```
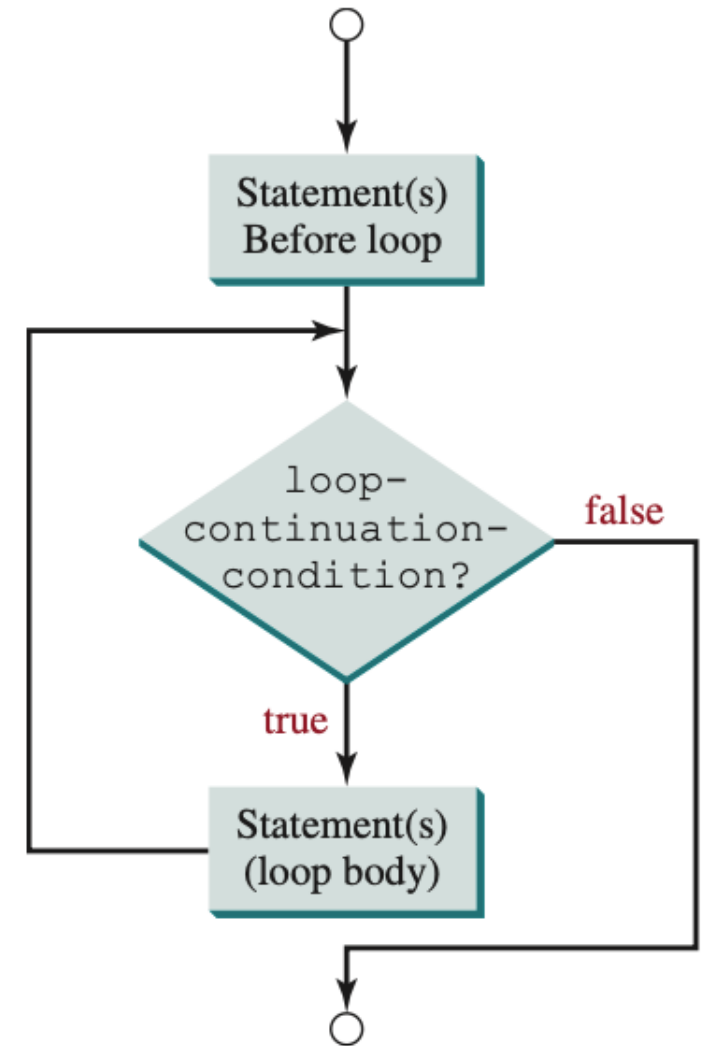
while loop

```java
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java!");
    count++;
}
```
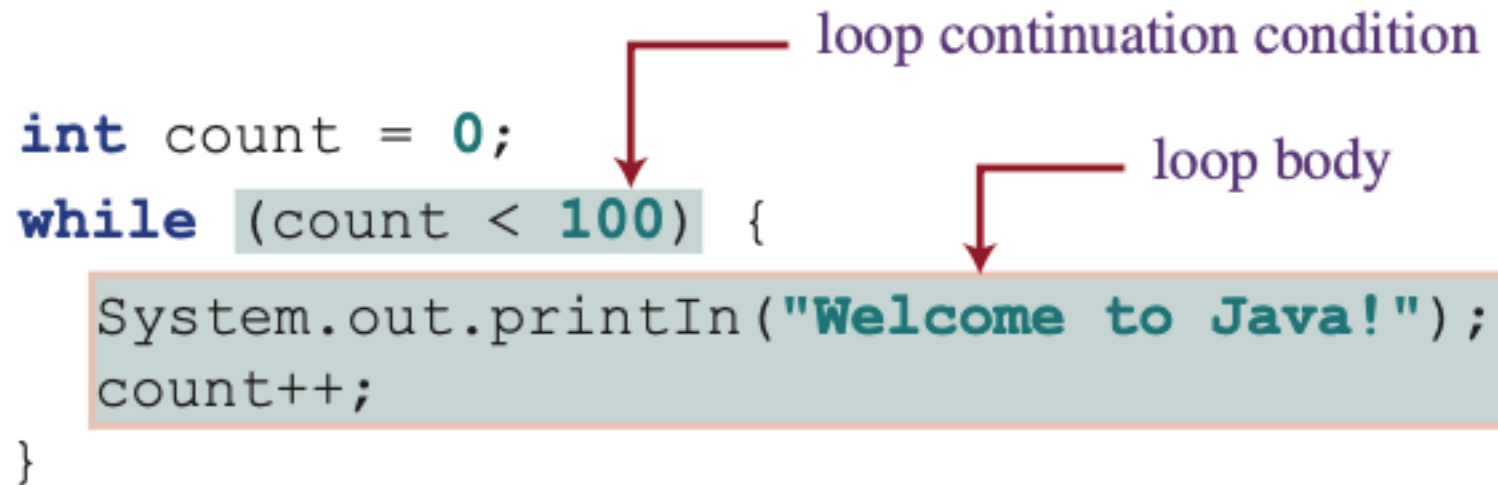
while loop

The syntax for the **while** loop is as follows:

```
while (loop-continuation-condition) {
    // Loop body
    Statement(s);
}
```

Counter-Controlled loop

```
int count = 0;
while (count < 100) {
    System.out.printIn("Welcome to Java!");
    count++;
}
```

loop continuation condition

loop body

while loop

sum = 1+2+3+...+9

```java
int sum = 0, i = 1;
while (i < 10) {
    sum = sum + i;
    i++;
}
System.out.println("sum is " + sum);
```

while loop

what's the problem of the following code?

```
int sum = 0, i = 1;
while (i < 10) {
    sum = sum + i;
}
```

Make sure that the loop can stop at certain condition!

Case Study: Guessing Numbers

Math.random()

- Write a program that randomly generates an integer between 0 and 100, inclusive.

input

- The program prompts the user to enter a number continuously until the number matches the randomly generated number.

- For each user input, the program tells the user whether the input is too low or too high, so the user can make the next guess intelligently.

loop continuation condition

if-else statement

Case Study: Guessing Numbers

input

> guess = sc.nextInt();

computation

> number = (int) (Math.random()*101)
> while (number != guess)

output

> if (guess == number)
> else if (guess < number)
> else

```java
import java.util.Scanner;

public class GuessNumber {
  public static void main(String[] args) {
    // Generate a random number to be guessed
    int number = (int)(Math.random() * 101);

    Scanner input = new Scanner(System.in);
    System.out.println("Guess a magic number between 0 and 100");

    int guess = -1;
    while (guess != number) {
      // Prompt the user to guess the number
      System.out.print("\nEnter your guess: ");
      guess = input.nextInt();

      if (guess == number)
        System.out.println("Yes, the number is " + number);
      else if (guess > number)
        System.out.println("Your guess is too high");
      else
        System.out.println("Your guess is too low");
    } // End of loop
  }
}
```

Loop Design Strategies

- *The key to designing a loop is to identify the code that needs to be repeated and write a condition for terminating the loop.*

- Step 1: Identify the statements that need to be repeated.

- Step 2: Wrap these statements in a loop as follows:

while (**true**) {

   Statements;

}

- Step 3: Code the **loop-continuation-condition** and add appropriate statements for controlling the loop.

while (loop-continuation-condition) {
   Statements;
   Additional statements for controlling the loop;

}

Controlling a loop with Flag (Sentinel Value)

```java
class HelloWorld {
    public static void main(String[] args) {
        boolean flag = true;
        while (flag) {
            System.out.println("continue");
            if (CONDITION)
                flag = false;
        }
    }
}
```

```java
import java.util.Scanner;

public class GuessNumber {
    Run | Debug
    public static void main(String[] args) {
        int number = (int)(Math.random() * 101);
        System.out.println(x:"Guess a magic number between 0 and 100");

        boolean continueLoop = true;
        while (continueLoop) {
            System.out.print(s:"\nEnter your guess: ");
            Scanner input = new Scanner(System.in);
            int guess = input.nextInt();

            if (guess == number) {
                System.out.println("Yes, the number is " + number);
                continueLoop = false;
            }
            else if (guess > number)
                System.out.println(x:"Your guess is too high");
            else
                System.out.println(x:"Your guess is too low");
        }
    }
}
```

Caution

- **Don't use floating-point values for equality checking in a loop control.** Because floating-point values are approximations for some values, using them could result in **imprecise counter values** and inaccurate results.

do-while loop

- do-while loop executes the loop body first then checks the loop continuation condition.

```
while (loop-continuation-condition) {
    // Loop body
    Statement(s);
}



do {
    // Loop body;
    Statement(s);
} while (loop-continuation-condition);
```
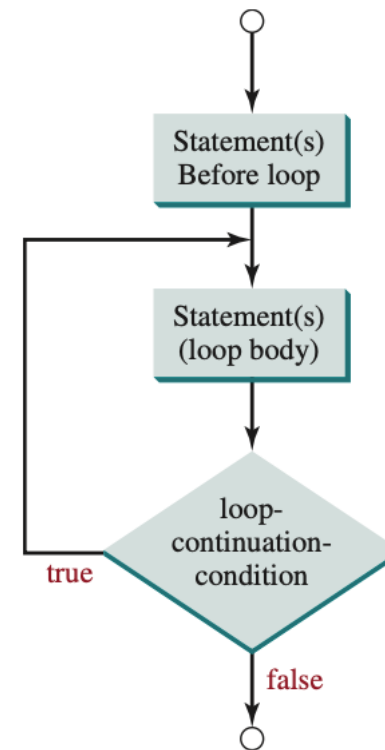
A Simple Vote

- What is sum after the following loop terminates?

```
int sum = 0;
int item = 0;
do {
  ++item;
  if (sum >= 4)
    continue;
  sum += item;
} while (item < 5);
```

a.   6

b.   7

c.   8

d.   9

e.   10

slido.com
slido #9592659

Problem: Compute Future Tuition

- Suppose the tuition for Algoma is $10,000 this year and increases 5% every year. In one year, the tuition will be $10,500. Write a program that displays the tuition in 10 years, and the total cost of four years' worth of tuition starting after the tenth year.

Problem: Compute Future Tuition

- Suppose the tuition for Algoma is $10,000 this year and increases 5% every year. In one year, the tuition will be $10,500.

> no input

> 10000*(1+0.05)*1.05*…

- Write a program that displays the tuition in 10 years,

> n = 10 in while loop

- and the total cost of four years' worth of tuition starting after the tenth year.

> cost10+cost11+cost12+cost13

Problem: Compute Future Tuition

```
double tuition = 10000;
int count = 1;
while (count <= 10) {
    tuition = tuition * 1.05;
    count++;
}
```

```
double sum = tuition;
int i = 1;
while (i <= 3) {
    tuition = tuition * 1.05;
    sum += tuition;
}
```

```
Tuition in ten years is 16288.9462677744418
The four-year total tuition in ten years is 70207.39453239122
```

for loop

for loop uses a variable to control how many times the loop body is executed.
The **initial-action** initialize the control variable and **action-after-iteration** will increase/decrease the control variable.

```
for (initial-action; loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
  Statement(s);
}
```
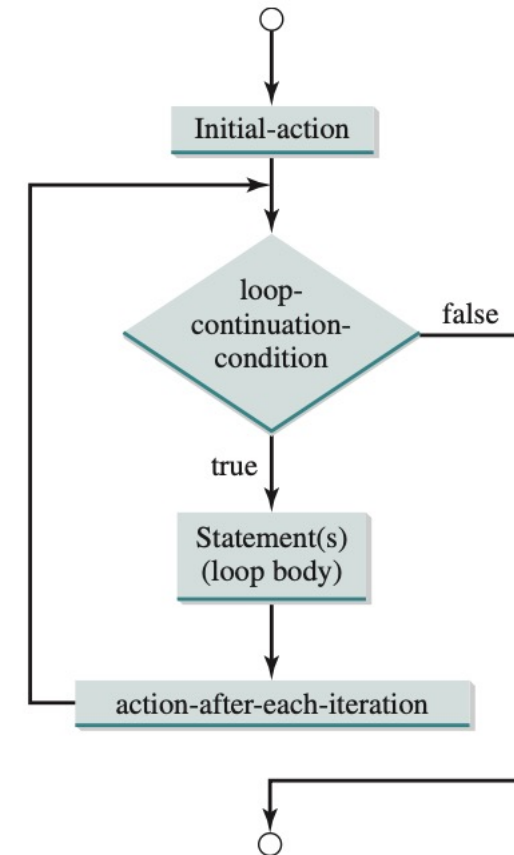
for loop

```java
for (int i = 0; i < 100; i++) {
  System.out.println("Welcome to Java!");
}
```

The **initial-action** in a **for** loop can be a list of zero or more comma-separated variable declaration statements or assignment expressions. For example:

```java
for (int i = 0, j = 0; i + j < 10; i++, j++) {
  // Do something
}
```

Caution

No compiling error:


```
for ( ; ; ) {
    // Do something
}
```

(a)

Equivalent

```
while (true) {
    // Do something
}
```

(b)
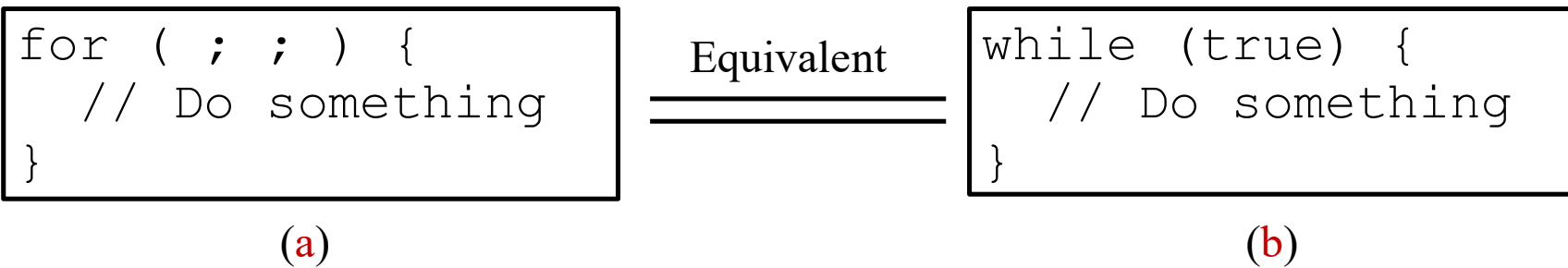
Example

Equivalent???

```
for (int i = 0; i < 10; ++i) {
    sum += i;
}
```

(a)

```
for (int i = 0; i < 10; i++) {
    sum += i;
}
```

(b)

Which Loop to Use?

- The three forms of loop statements—while, do-while, and for—are expressively equivalent; that is, you can write a loop in any of these three forms.

- A for loop places control <u>variable initialization, loop continuation condition, and adjustment after each iteration</u> all together. It is more **<u>concise</u>** and enables you to write the code with less errors than the other two loops.

Which Loop to Use?

- In general, a for loop may be used if the **number of repetitions** is known in advance.

- A while loop may be used if the **number** of repetitions is **not fixed**,

- A do-while loop can be used to replace a while loop if the loop body **has to** be executed before the continuation condition is tested.

- Use the loop statement that is most intuitive and **comfortable for you**.

Caution

Error

```java
for (int i = 0; i < 10; i++);
{
    System.out.println("i is " + i);
}
```

(a)

Empty body

```java
for (int i = 0; i < 10; i++) { };
{
    System.out.println("i is " + i);
}
```

(b)

Error

```java
int i = 0;
while (i < 10);
{
    System.out.println("i is " + i);
    i++;
}
```

(c)

Empty body

```java
int i = 0;
while (i < 10) { };
{
    System.out.println("i is " + i);
    i++;
}
```

(d)

Caution

```
int i = 0;
do {
    System.out.println("i is " + i);
    i++;
} while (i < 10);
```
← This is correct.

Errors?

```
1   public class Test {
2     public void main(String[] args) {
3       for (int i = 0; i < 10; i++);
4         sum += i;
5
6       if (i < j);
7         System.out.println(i)
8       else
9         System.out.println(j);
10
11      while (j < 10);
12      {
13          j++;
14      }
15
16      do {
17          j++;
18      } while (j < 10)
19    }
20  }
```

Nested Loops

```
for (int i = 0; i < 10000; i++)
  for (int j = 0; j < 10000; j++)
    for (int k = 0; k < 10000; k++)
      Perform an action
```

The action will be performed for 10000*10000*10000 times

MultiplicationTable

```java
public class MultiTable {
    Run | Debug
    public static void main(String[] args) {
        // Display the table heading
        System.out.println(x:"          Multiplication Table");

        // Display the number title
        System.out.print(s:"    ");
        for (int j = 1; j <= 9; j++)
            System.out.print("   " + j);

        System.out.println(x:"\n————————————————————————————————————————");

        // Print table body
        for (int i = 1; i <= 9; i++) {
            System.out.print(i + " | ");
            for (int j = 1; j <= 9; j++) {
                // Display the product and align properly
                System.out.printf(format:"%4d", i * j);
            }
            System.out.println();
        }
    }
}
```

```
              Multiplication Table
        1   2   3   4   5   6   7   8   9
————————————————————————————————————————
1 |     1   2   3   4   5   6   7   8   9
2 |     2   4   6   8  10  12  14  16  18
3 |     3   6   9  12  15  18  21  24  27
4 |     4   8  12  16  20  24  28  32  36
5 |     5  10  15  20  25  30  35  40  45
6 |     6  12  18  24  30  36  42  48  54
7 |     7  14  21  28  35  42  49  56  63
8 |     8  16  24  32  40  48  56  64  72
9 |     9  18  27  36  45  54  63  72  81
```

Nested Loops

$$
\begin{array}{c}
\begin{array}{cccc}
\textcolor{red}{1} & \textcolor{red}{2} & \textcolor{red}{\ldots} & \textcolor{red}{n}
\end{array} \\
\begin{array}{c}
\textcolor{green}{1} \\
\textcolor{green}{2} \\
\textcolor{green}{3} \\
\vdots \\
\textcolor{green}{m}
\end{array}
\begin{bmatrix}
a_{1\textcolor{green}{1}} & a_{1\textcolor{green}{2}} & \ldots & a_{1\textcolor{red}{n}} \\
a_{2\textcolor{green}{1}} & a_{2\textcolor{green}{2}} & \ldots & a_{2\textcolor{red}{n}} \\
a_{3\textcolor{green}{1}} & a_{3\textcolor{green}{2}} & \ldots & a_{3\textcolor{red}{n}} \\
\vdots & \vdots & \vdots & \vdots \\
a_{\textcolor{green}{m}\textcolor{red}{1}} & a_{\textcolor{green}{m}\textcolor{red}{2}} & \ldots & a_{\textcolor{green}{m}\textcolor{red}{n}}
\end{bmatrix}
\end{array}
$$

```java
class NestedLoops {
    public static void main(String[] args) {
        for (int i = 0; i < m; i++)
            for (int j =0; j < n; j++)
                a_ij = ...
    }
}
```

Mar 12 (Week 10): Single-Dimensional Arrays Array operations.

Mar 19 (Week 11): Multidimensional Arrays

Avoid Floating-point Numbers

- Using floating-point numbers in the **loop continuation condition** may cause numeric errors.

```java
class NestedLoops {
    public static void main(String[] args) {
        float sum = 0;
        for (float i = 0.01f; i <= 1.0f; i = i+0.01f)
            sum += i;
        System.out.println("sum is " + sum);


        double sumd = 0;
        for (double i = 0.01; i <= 1.0; i = i+0.01)
            sumd += i;
        System.out.println("sum is "+ sumd);
    }
}
```

Avoid Floating-point Numbers

- If you display **i** for each iteration in the loop, you will see that the **last i** is slightly **larger than 1** (**not exactly 1**) when i is in double format.

- This causes the last i not to be added into sum. The fundamental problem is the floating-point numbers are represented by approximation.
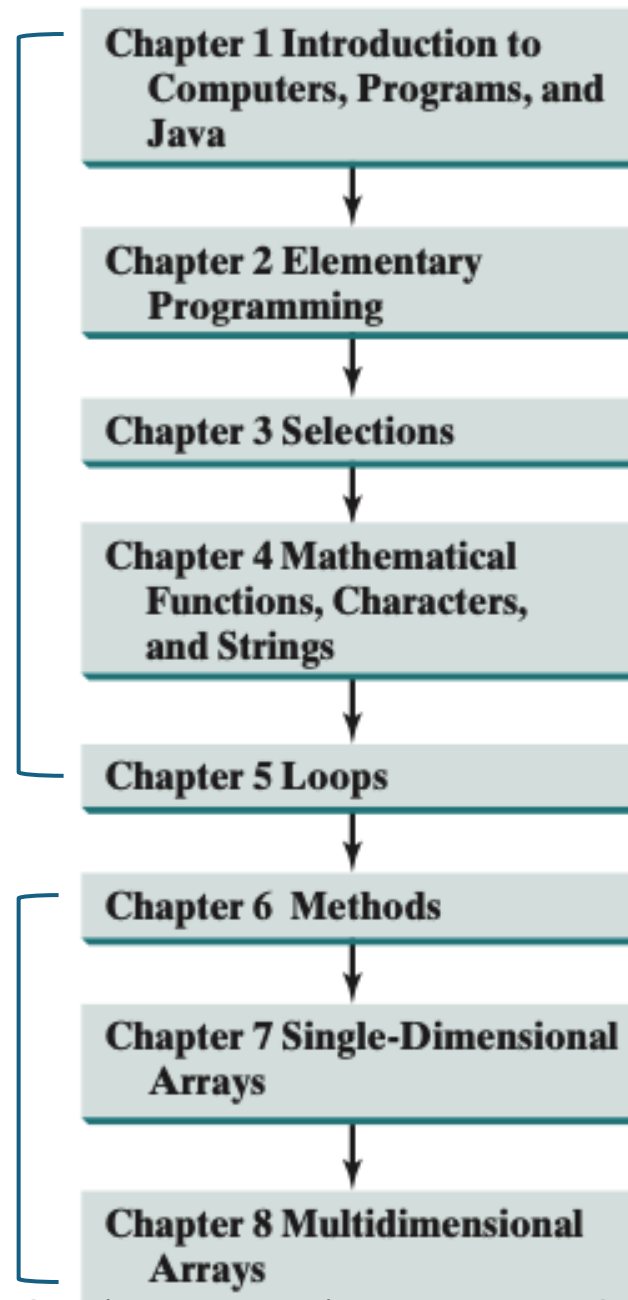
Bonus Points

- Start from this week:
  - Each problem is worth 2 points.
  - Each student can get maximum 8 bonus points.
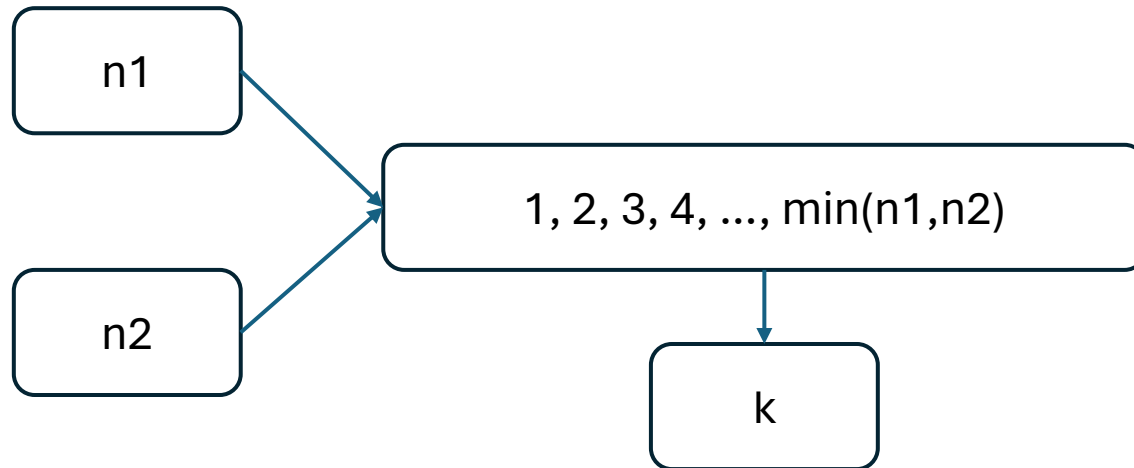
10 Min Break

Case Study: Finding the Greatest Common Divisor

- In mathematics, the greatest common divisor (GCD) of two or more integers, which are not all zero, is the **largest positive integer that divides** each of the integers.

- The GCD of the two integers 4 and 2 is 2. The greatest common divisor of the two integers 16 and 24 is 8.

GCD

n1

n2

1, 2, 3, 4, …, min(n1,n2)

k

GCD

n1

n2

1, 2, 3, 4, ..., min(n1,n2)

k
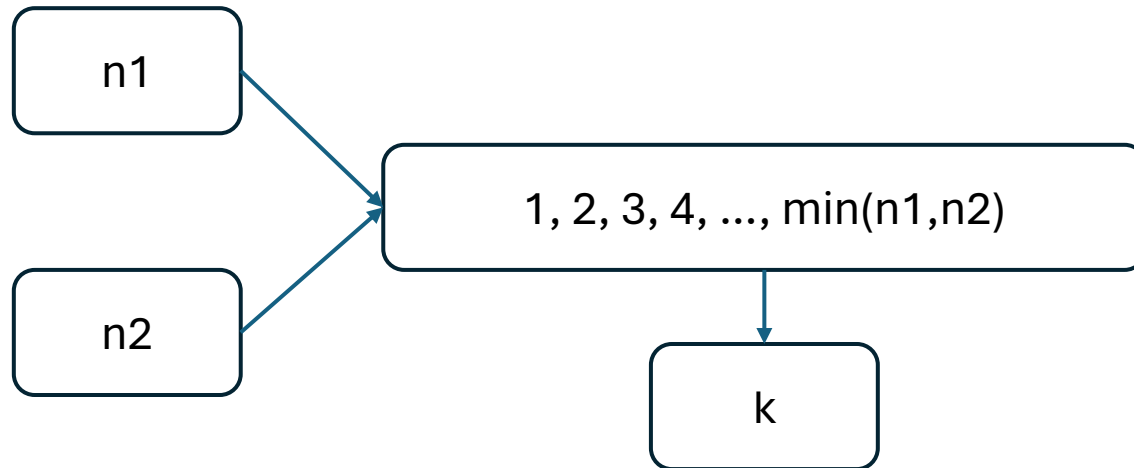
what do we need:
input:

calculation:

output:

```java
import java.util.Scanner;

public class GreatestCommonDivisor {
    Run | Debug
    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter two integers
        System.out.print(s:"Enter first integer: ");
        int n1 = input.nextInt();
        System.out.print(s:"Enter second integer: ");
        int n2 = input.nextInt();

        int gcd = 1;
        int k = 2;
        while (k <= n1 && k <= n2) {
            if (n1 % k == 0 && n2 % k == 0)
                gcd = k;
            k++;
        }

        System.out.println("The greatest common divisor for " + n1 +
            " and " + n2 + " is " + gcd);
    }
}
```

Case Study: Predicting the Future Tuition

- Suppose the tuition for Algoma is $10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

```
input:
no input, use $10,000
calculation:

output:
number of years
```

FutureTuition

```java
public class FutureTuition {
    Run | Debug
    public static void main(String[] args) {
        double tuition = 10000;    // Year 0
        int year = 0;
        while (tuition < 20000) {
            tuition = tuition * 1.07;
            year++;
        }

        System.out.println("Tuition will be doubled in "
            + year + " years");
        System.out.printf(format:"Tuition will be $%.2f in %1d years",
            tuition, year);
    }
}
```

break and **continue**

- break and continue, can be used in loop statements to provide additional controls

- Using break and continue can simplify programming in some cases

break

```java
public class TestBreak {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      sum += number;
      if (sum >= 100)
        break;
    }

    System.out.println("The number is " + number);
    System.out.println("The sum is " + sum);
  }
}
```

```
The number is 14
The sum is 105
```

continue

Adds integers from 1 to 20 except 10 and 11 to sum

```java
public class TestContinue {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      if (number == 10 || number == 11)
        continue;
      sum += number;
    }

    System.out.println("The sum is " + sum);
  }
}
```

The sum is 189

Note

- You can **always** write a program without using break or continue in a loop.

- Using break and continue is appropriate if it **simplifies** coding and makes programs easier to read.

Case Study: Checking Palindromes

- A string is a palindrome if it reads the same forward and backward. The words "mom," "dad," and "noon," for instance, are all palindromes.

- Write a program that prompts the user to enter a string and reports whether the string is a palindrome.

Case Study: Checking Palindromes

- A string is a palindrome if it reads the same forward and backward. The words "mom," "dad," and "noon," for instance, are all palindromes.

- Write a program that prompts the user to enter a string and reports whether the string is a palindrome.

- Solution: Check whether the first character in the string is the same as the last character. If so, check whether the second character is the same as the second-to-last character. Repeat until a mismatch is found or all the characters are checked.

Checking Palindromes

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
W	e	l	c	o	m	e		t	o		J	a	v	a

low   high =s.length()-1
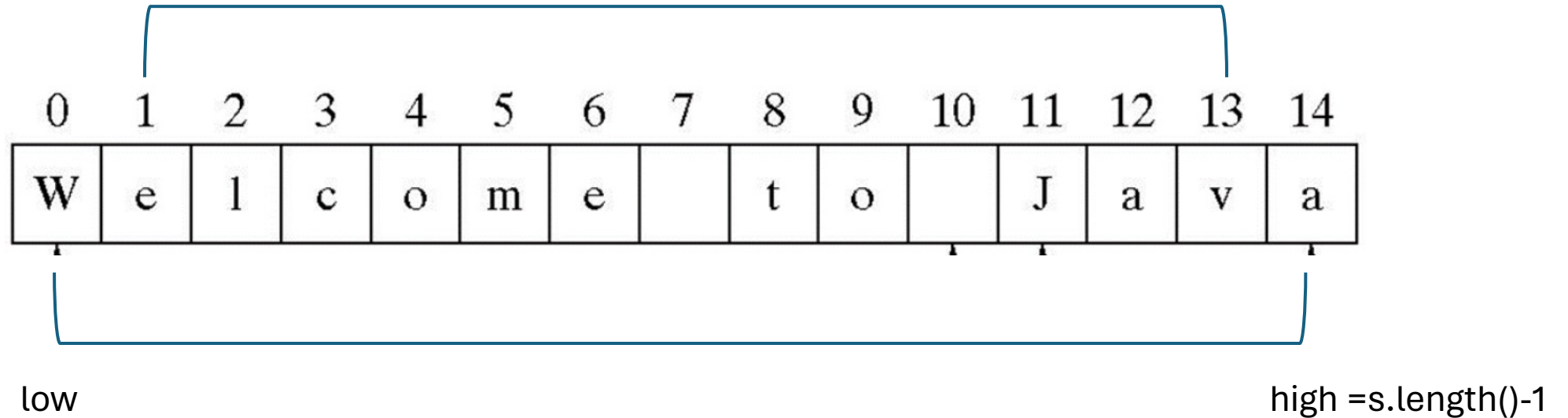
```
if (s.charAt(low) != s.charAt(high))
    not a palindrome;
else
    low++;
    high--;
```

```java
import java.util.Scanner;

public class Palindrome {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s:"Enter a string: ");
        String s = input.nextLine();


        int low = 0;
        int high = s.length() - 1;

        boolean isPalindrome = true;
        while (low < high) {
            if (s.charAt(low) != s.charAt(high)) {
                isPalindrome = false;
                break;
            }

            low++;
            high--;
        }

        if (isPalindrome)
            System.out.println(s + " is a palindrome");
        else
            System.out.println(s + " is not a palindrome");

    }
}
```

Leetcode

- https://leetcode.com/

- LeetCode is an online platform for **coding interview** preparation. The service provides coding and algorithmic problems intended for users to practice coding.

- LeetCode has gained popularity among job seekers and coding enthusiasts as a resource for technical interviews and coding competitions.

Leetcode - Palindrome Number

- https://leetcode.com/problems/palindrome-number/description/

Assignment

- Our course has a bottom line: copying others' assignments by simply **replacing the file name is not acceptable**.

- Assignment:
  - Java is the only accepted language for the assignment and exams.
  - Please convert word doc to pdf format and submit a pdf file.
  - Please copy-paste your script into the document

Q&A