UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Dasar Dasar Pemrograman 2

Topik Pekan 02: Loop

Acuan: Introduction to Java Programming and Data Structure, Bab 5

Sumber Slide: Y Daniel Liang, Introduction to Java Programming and Data Structure, Pearson.

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Motivations

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement **a hundred times**:

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

So, how do you solve this problem?

Opening Problem

Problem:

100 times

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

...

...

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

Introducing while Loops

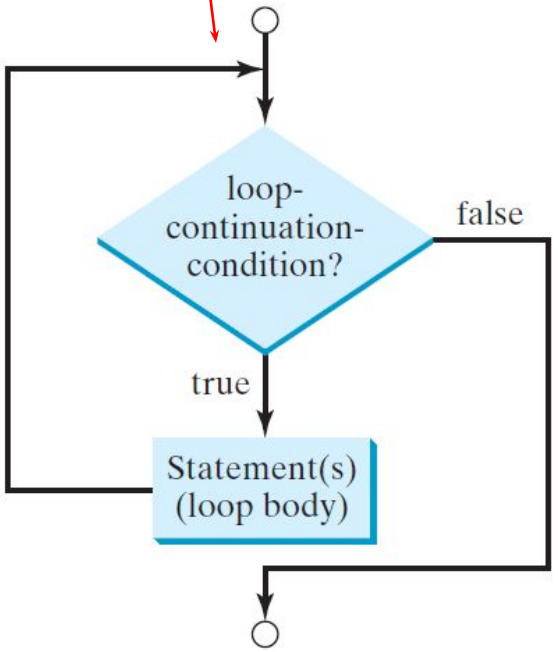
```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java");
  count++;
}
```

Objectives

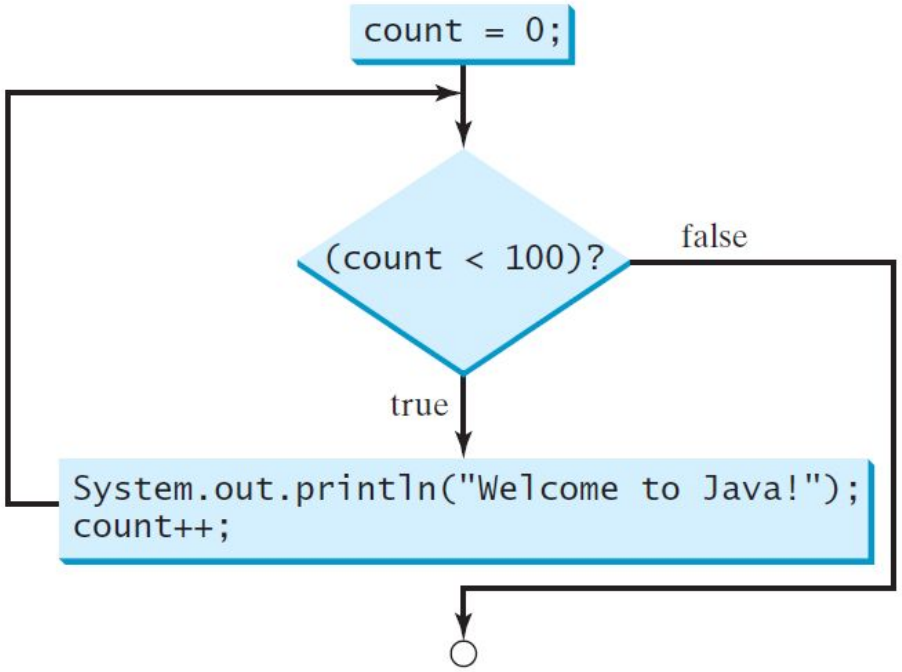
- To write programs for executing statements repeatedly using a **while** loop (§5.2).
- To follow the loop design strategy to develop loops (§§5.2.1–5.2.3).
- To control a loop with a sentinel value (§5.2.4).
- To obtain large input from a file using input redirection rather than typing from the keyboard (§5.2.5).
- To write loops using **do-while** statements (§5.3).
- To write loops using **for** statements (§5.4).
- To discover the similarities and differences of three types of loop statements (§5.5).
- To write nested loops (§5.6).
- To learn the techniques for minimizing numerical errors (§5.7).
- To learn loops from a variety of examples (**GCD**, **FutureTuition**, **Dec2Hex**) (§5.8).
- To implement program control with **break** and **continue** (§5.9).
- To write a program that displays prime numbers (§5.11).

`while` Loop Flow Chart

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




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



animation

Trace while Loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Initialize count

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is true

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Increase count by 1
count is 1 now

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is still true since count is 1

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Print Welcome to Java

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

Increase count by 1
count is 2 now

animation

Trace while Loop, cont.

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

(count < 2) is false since count is 2 now

animation

Trace while Loop

```
int count = 0;
while (count < 2) {
  System.out.println("Welcome to Java!");
  count++;
}
```

The loop exits. Execute the next statement after the loop.

Problem: Repeat Addition Until Correct

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#RepeatAdditionQuiz.java

Recall that Listing 3.1 AdditionQuiz.java gives a program that prompts the user to enter an answer for a question on addition of two single digits. Using a loop, you can now rewrite the program to let the user enter a new answer until it is correct.

Problem: Guessing Numbers

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#GuestNumber.java

Write a program that randomly generates an integer between 0 and 100, inclusive. 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 choose the next input intelligently. Here is a sample run:

Problem: An Advanced Math Learning Tool

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#SubtractionQuizLoop.java

The Math subtraction learning tool program generates just one question for each run. You can use a loop to generate questions repeatedly. This example gives a program that generates five questions and reports the number of the correct answers after a student answers all five questions.

Ending a Loop with a Sentinel Value

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#SentinelValue.java

Often the number of times a loop is executed is not predetermined. You may use an input value to signify the end of the loop. Such a value is known as a *sentinel value*.

Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

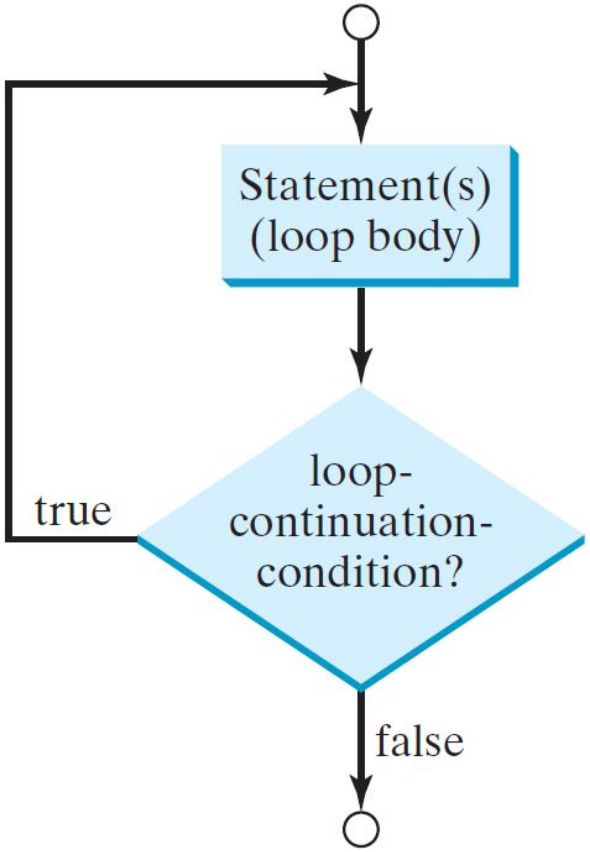
Caution

Don’t use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing 1 + 0.9 + 0.8 + ... + 0.1:

```
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
  sum += item;
  item -= 0.1;
}
System.out.println(sum);
```

do-while Loop

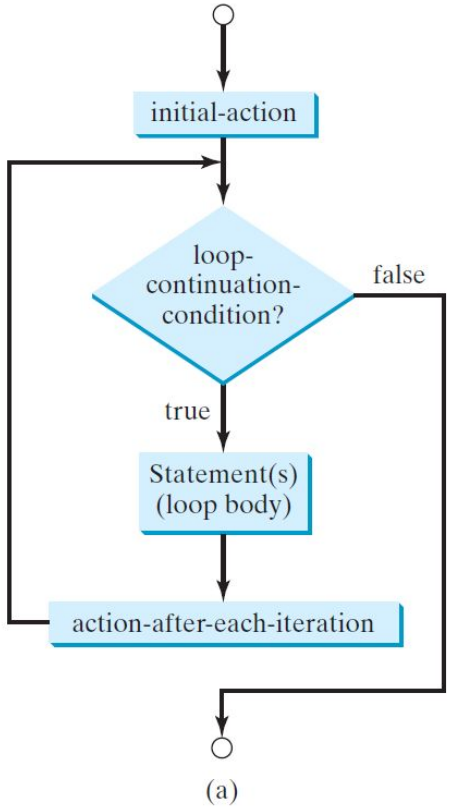
```
do {
  // Loop body;
  Statement(s);
} while (loop-continuation-condition)
```



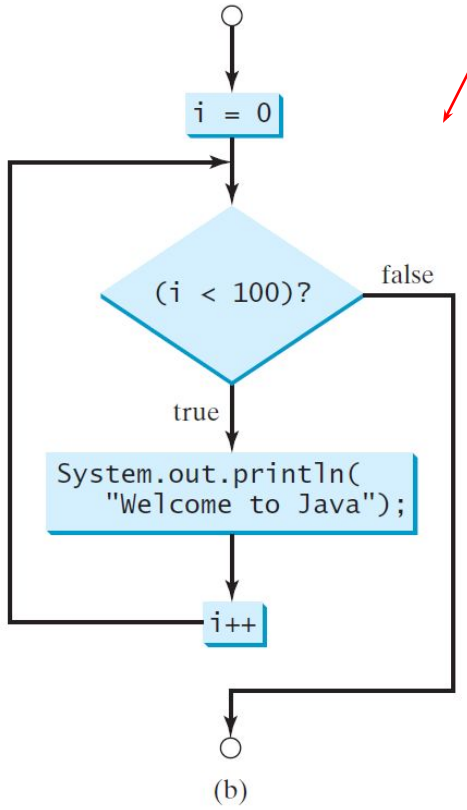
`for` Loops

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

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



(a)



(b)

Trace for Loop

Declare i

```
int i;
for (i = 0;   i < 2;   i++) {
  System.out.println("Welcome to Java!");
}
```

Trace for Loop, cont.

```
int i;
for (i = 0;   i < 2;   i++) {
  System.out.println("Welcome to Java!");
}
```

Execute initializer
i is now 0

Trace for Loop, cont.

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

(i < 2) is true
since i is 0

Trace for Loop, cont.

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java!");
}
```

Print Welcome to Java

Trace for Loop, cont.

```
int i;
for (i = 0;   i < 2;  i++) {
  System.out.println("Welcome to Java!");
}
```

Execute adjustment statement
i now is 1

Trace for Loop, cont.

```
int i;
for (i = 0;  i < 2;   i++) {
  System.out.println("Welcome to Java!");
}
```

(i < 2) is still true
since i is 1

Trace for Loop, cont.

```
int i;
for (i = 0;   i < 2;   i++) {
  System.out.println("Welcome to Java!");
}
```

Print Welcome to Java

Trace for Loop, cont.

```
int i;
for (i = 0;   i < 2;  i++) {
  System.out.println("Welcome to Java!");
}
```

Execute adjustment statement
i now is 2

Trace for Loop, cont.

```
int i;
for (i = 0;  i < 2;   i++) {
  System.out.println("Welcome to Java!");
}
```

(i < 2) is false
since i is 2

Trace for Loop, cont.

```
int i;
for (i = 0; i < 2; i++) {
  System.out.println("Welcome to Java");
}
```

Exit the loop. Execute the next statement after the loop

Note

The initial-action in a for loop can be a list of zero or more comma-separated expressions. The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements. Therefore, the following two for loops are correct. They are rarely used in practice, however.

```
for (int i = 1; i < 100; System.out.println(i++));
```

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

Note

If the loop-continuation-condition in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

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

(a)

Equivalent

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

(b)

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

Logic Error

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

Caution, cont.

Similarly, the following loop is also wrong:

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

Logic Error

In the case of the do loop,
the following semicolon is needed to end the loop.

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

Correct

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. For example, a while loop in (a) in the following figure can always be converted into the following for loop in (b):

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

(a)

Equivalent

```
for ( ; loop-continuation-condition; )
  // Loop body
}
```

(b)

A for loop in (a) in the following figure can generally be converted into the following while loop in (b) except in certain special cases (see Review Question 3.19 for one of them):

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

(a)

Equivalent

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

(b)

Which loop? -- Recommendations

Use the one that is most intuitive and comfortable for you.

In general, a ***for*** loop may be used if the **number of repetitions is known**, as, for example, when you need to print a message 100 times.

A ***while*** loop may be used if the **number of repetitions is not known**, as in the case of reading the numbers until the input is 0.

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

Problem:
Finding the Greatest Common Divisor

Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.

Solution: Suppose you enter two integers 4 and 2, their greatest common divisor is 2. Suppose you enter two integers 16 and 24, their greatest common divisor is 8. So, how do you find the greatest common divisor? Let the two input integers be n1 and n2. You know number 1 is a common divisor, but it may not be the greatest commons divisor. So you can check whether k (for k = 2, 3, 4, and so on) is a common divisor for n1 and n2, until k is greater than n1 or n2.

Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:

- For each number = 2, 3, 4, 5, 6, ...,
  - Determine whether a given number is prime.
  - Print each prime number, and print 10 numbers per line.
  - Count the prime numbers.
- Print total prime numbers

Selamat Berlatih!

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop

Perhatikan lagi List Objective yang perlu dikuasai pekan ini.
Baca buku acuan dan berlatih!
Bila masih belum yakin tanyakan ke dosen, tutor atau Kak Burhan.

Semangat !

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Dasar Dasar Pemrograman 2

Topik Pekan 03b: Loop, Debugger

Acuan: **Bab 5** , Y Daniel Liang, Introduction to Java Programming and Data Structure, Pearson

Sumber Slide: Y Daniel Liang, Introduction to Java Programming and Data Structure, Pearson.

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Minimizing Numerical Errors

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#TestSum.java

Numeric errors involving floating-point numbers are inevitable. This section discusses how to minimize such errors through an example.

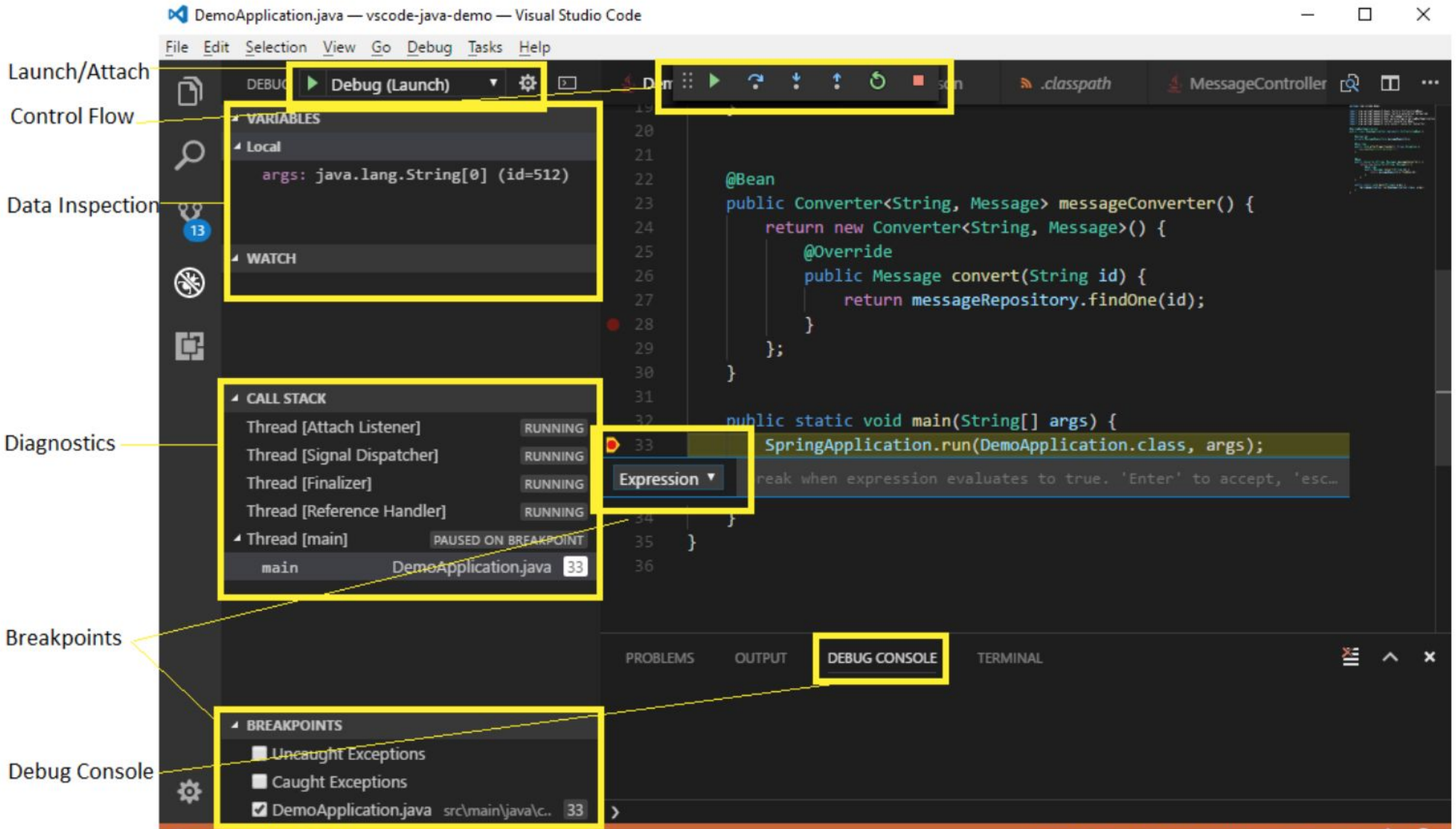
Here is an example that sums a series that starts with 0.01 and ends with 1.0. The numbers in the series will increment by 0.01, as follows: 0.01 + 0.02 + 0.03 and so on.

```
javac TestSum.java
java TestSum
Percobaan penjumlahan floating point.
Hasil seharusnya 50.50, tapi...?
Version 1: The sum is 50.499985
Version 2: The sum is 49.50000000000003
Version 3: The sum is 50.50000000000003
Version 4: The sum is 50.49999999999995
```

Debugging Loops in IDE Tools

https://code.visualstudio.com/docs/java/java-debugging

https://code.visualstudio.com/blogs/2017/09/28/java-debug



Problem: Predicting the Future Tuition

Problem: Suppose that the tuition for a university is $10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

Problem: Predicating the Future Tuition

```
double tuition = 10000;   int year = 0  // Year 0
tuition = tuition * 1.07; year++;       // Year 1
tuition = tuition * 1.07; year++;       // Year 2
tuition = tuition * 1.07; year++;       // Year 3
...
```

Case Study: *Converting Decimals to Hexadecimals*

Hexadecimals are often used in computer systems programming (see Appendix F for an introduction to number systems). How do you convert a decimal number to a hexadecimal number? To convert a decimal number *d* to a hexadecimal number is to find the hexadecimal digits h_n, h_{n-1}, h_{n-2}, ... , h_2, h_1, and h_0 such that

$$d = h_n \times 16^n + h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + \ldots + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0$$

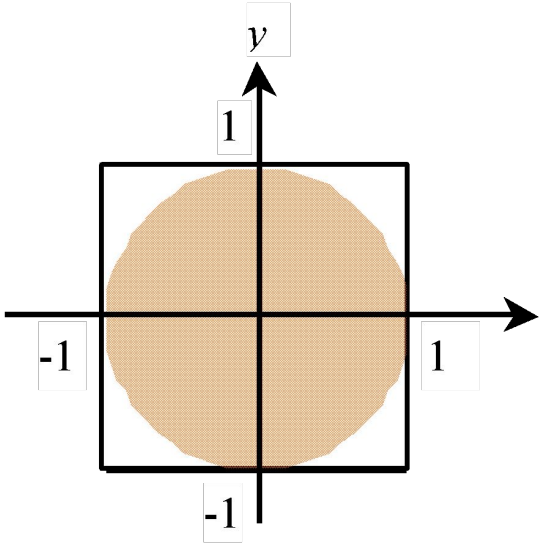
These hexadecimal digits can be found by successively dividing *d* by 16 until the quotient is 0. The remainders are h_0, h_1, h_2, ... , h_{n-2}, h_{n-1}, and h_n.

Monte Carlo Simulation

https://replit.com/@AdeAzurat/DDP2-Pekan03-Loop#MonteCarloSimulation.java

The Monte Carlo simulation refers to a technique that uses random numbers and probability to solve problems. This method has a wide range of applications in computational mathematics, physics, chemistry, and finance.

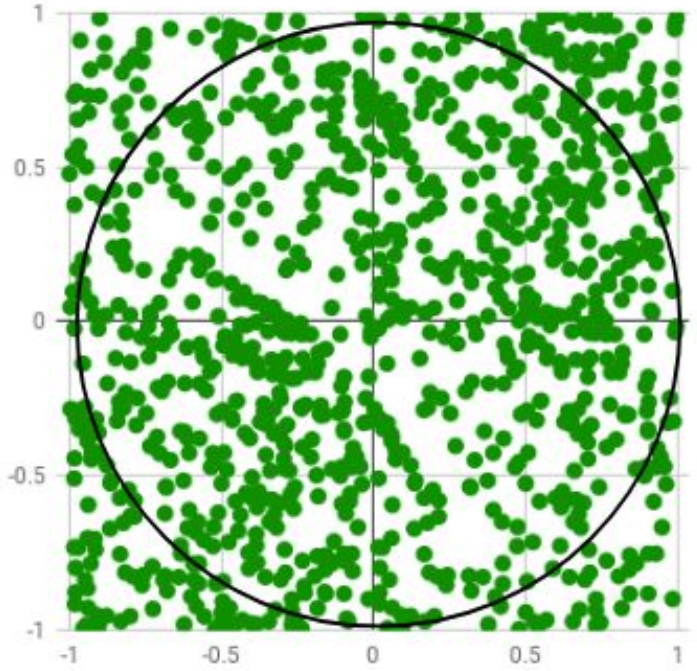
This section gives an example of using the Monte Carlo simulation for estimating π.



$$\frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

How to know that a point is inside the circle or not?

$$\pi = 4 * \frac{\text{no. of points generated inside the circle}}{\text{no. of points generated inside the square}}$$



Nested Loops

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop#MultiplicationTabel.java

Problem: Write a program that uses nested for loops to print a multiplication table.

```
java MultiplicationTable
          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
```

Using `break` and `continue`

Examples for using the `break` and `continue` keywords:

- TestBreak.java

- TestContinue.java

break

```
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);
  }
}
```

continue

```
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);
  }
}
```

Guessing Number Problem Revisited

Here is a program for guessing a number. You can rewrite it using a break statement.

Selamat Berlatih!

https://repl.it/@AdeAzurat/DDP2-Pekan03-Loop

Perhatikan lagi List Objective yang perlu dikuasai pekan ini.
Baca buku acuan dan berlatih!
Bila masih belum yakin tanyakan ke dosen, tutor atau Kak Burhan.

Semangat !

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER