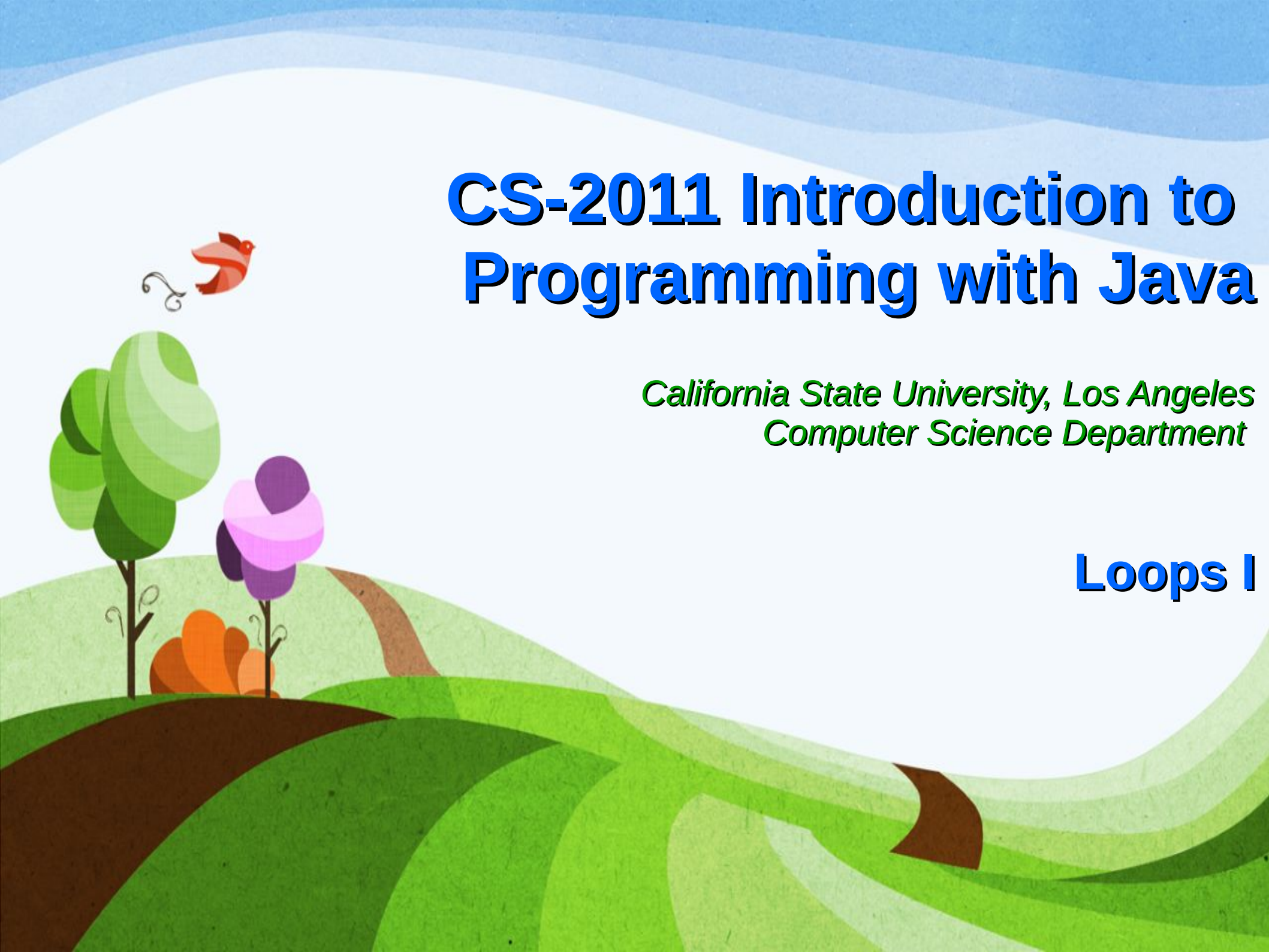


CS-2011 Introduction to Programming with Java

California State University, Los Angeles
Computer Science Department

Loops I

Introduction

- Suppose you wanted to print the string "Hello World!" 15 times.
- Naïve Approach – copy and paste 15 times:

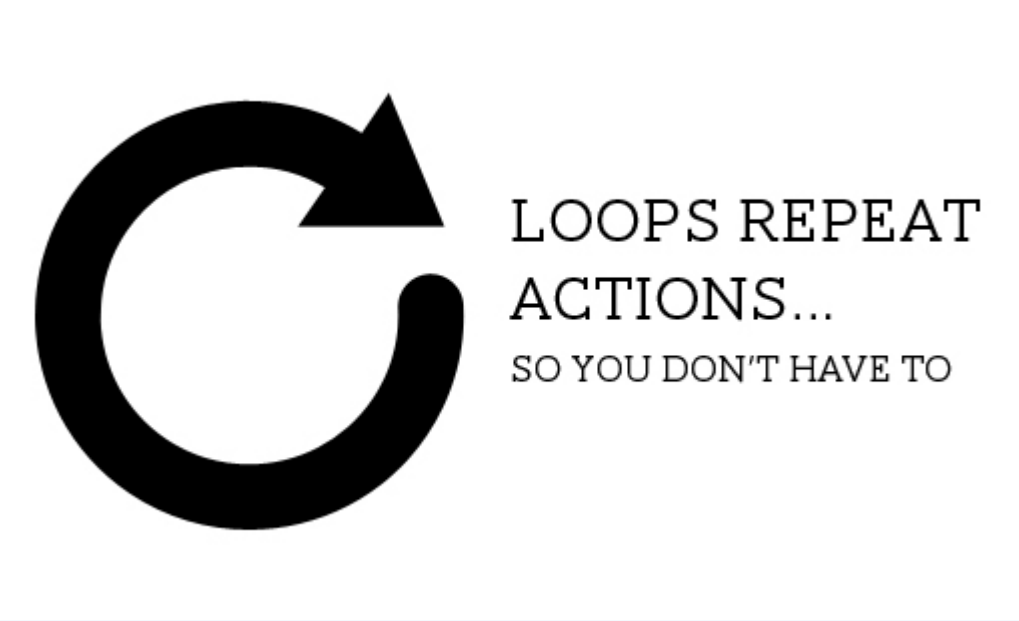
```
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
System.out.println("Hello, world");
```

- It works...but this is messy and very bad looking.
- Also, what happens if you wanted to repeat this 100, 200, 1000, etc. times.
- Or what about if you don't know how many times it should execute?

Introduction

- Java provides a construct called a loop that controls how many times an operation or sequence of operations is performed in succession.

- Four loop types:
  - **`while`**
  - **`do-while`**
  - **`for`**
  - **`for-each`**
    - we will see these when we talk about arrays



Introduction

- A better solution is to use a loop!

- Here we are replacing the previous code with a more elegant solution:

```
int count = 0;
while (count < 15) {
    System.out.println("Hello World!");
    count++;
}
```


The while Loop

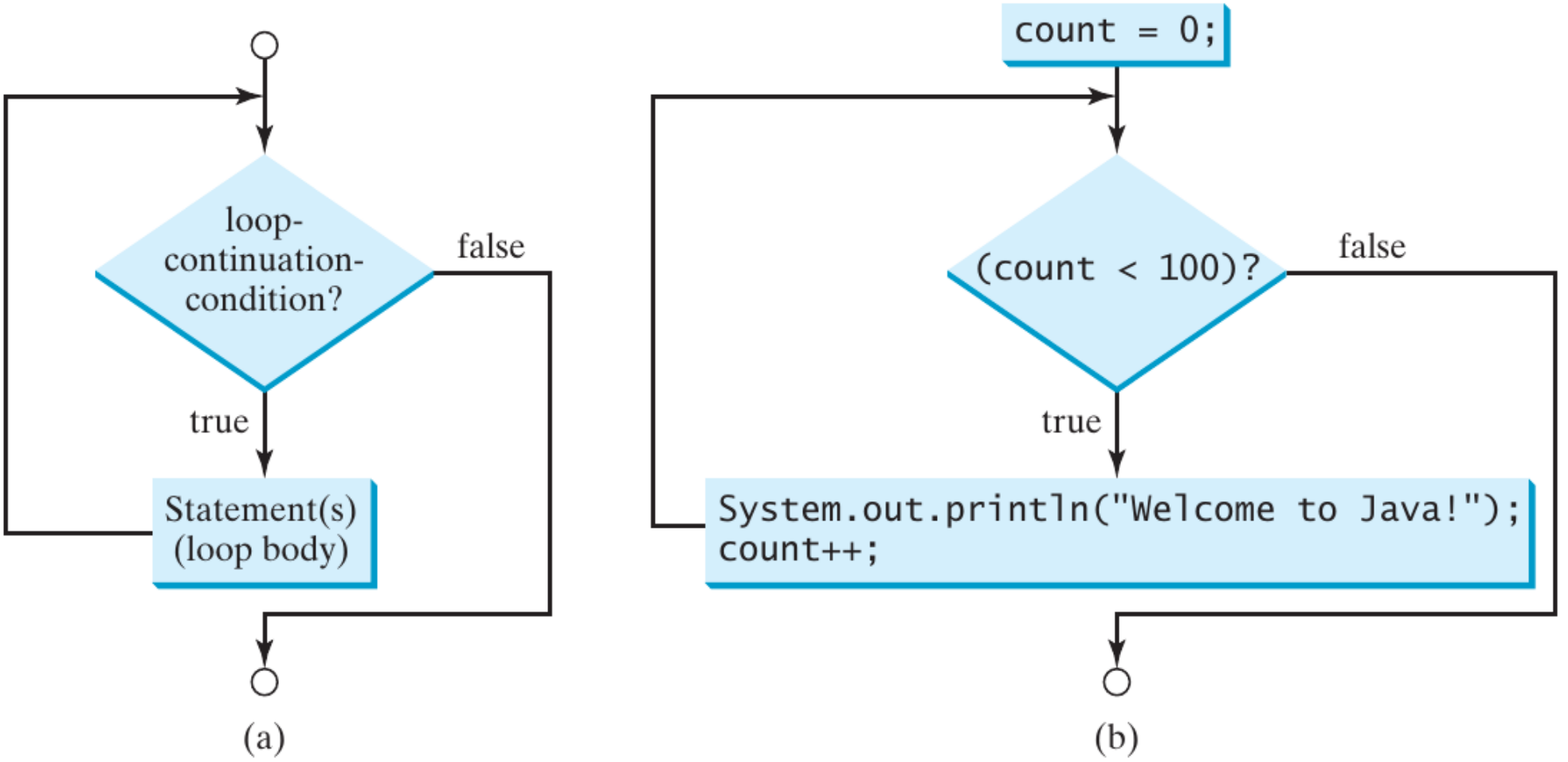
The `while` Loop

- Syntax:

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

- a **`while`** loop executes statements repeatedly while the condition is true.
- **loop body:** the part of the loop that contains the statements to be repeated. the statements can be anything we have seen so far (including if/else statements and other loops)
- **iteration** (or **repetition**) **of the loop:** a one-time execution of the loop body
  - 1 iteration = 1 execution of the loop body
- **loop-continuation-condition:**
  - is a Boolean expression that controls the execution of the body
  - is evaluated each time the loop repeats to determine if the loop body should be executed
  - if it evaluates to true the loop body is executed, false the entire loop terminates.

The `while` Loop



The `while` loop repeatedly executes the statements in the loop body when the `loop-continuation-condition` evaluates to `true`.

Counter-Controlled Loops

- loops which use a variable to count how many times a loop should iterate.

```
int count = 0;
while (count < n){
    statement;
    count = count + 1;
    //or count++;
}
```

- The loop body will execute **n** times.
- **NOTE:** The loop-continuation-condition must ALWAYS be inside the parentheses. The braces around the loop body can be omitted if the body contains one or no statements, but as we know, **it's better to always include them!.**

Counter-Controlled Loops

- Counter-controlled repetition requires:
  - Control variable (loop counter)
  - Initial value of the control variable
  - Increment/decrement of control variable through each loop
  - Loop-continuation condition that tests for the final value of the control variable
- The following code finds the sum of the integers 1 ~ 5 and prints it out:

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum); //sum is 15
```

- As long as **`(i <= 5)`** is **`true`**, then the current value of **`i`** is added to **`sum`** and then **`i`** is increased by **`1`**.

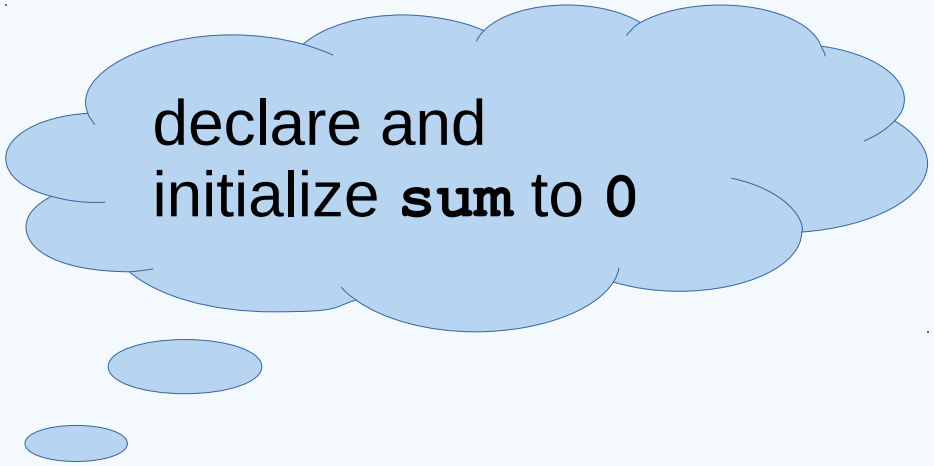

Debugging Loops

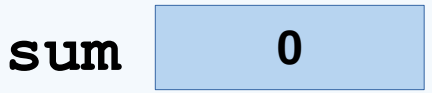
Debugging Loops

- Loops can be hard to debug, sometimes hand tracing is the best thing you can do.

- You can also trace the values of variables in your program.
  - Simply insert temporary output statements in your program to print the values of the variables of interest.

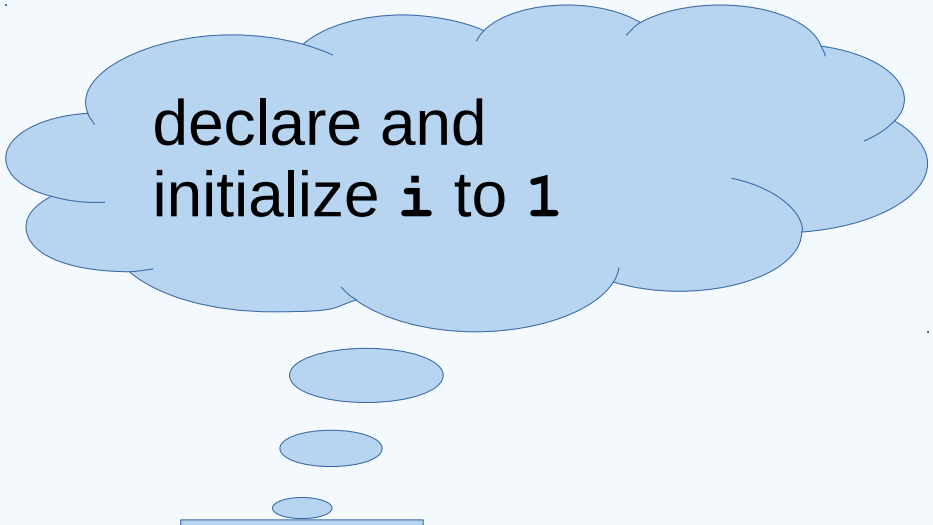
Tracing a `while` Loop






```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

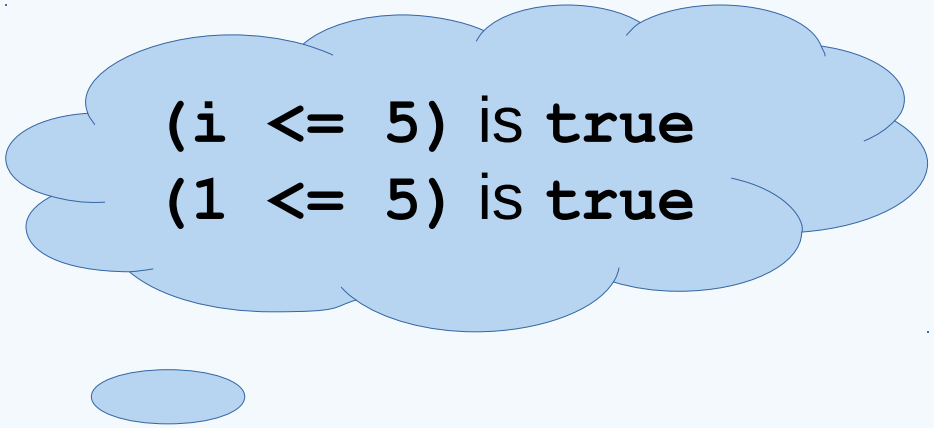
Tracing a `while` Loop



sum	0
i	1

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop



sum	0
i	1

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

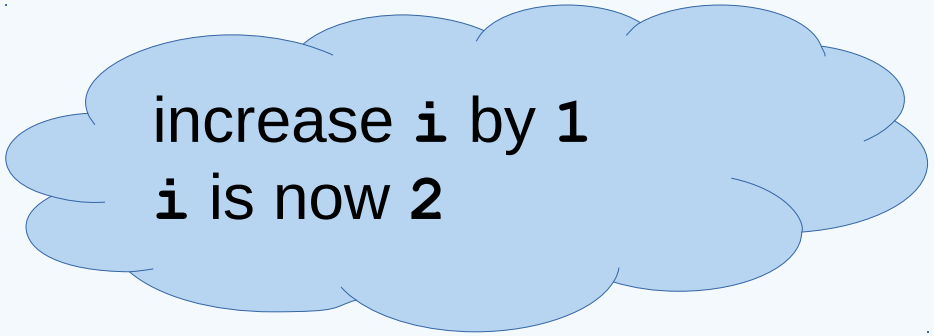
Tracing a `while` Loop

add `i` to `sum`
`sum = 0 + 1`
`sum` is now `1`

`sum`	`1`
`i`	`1`

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

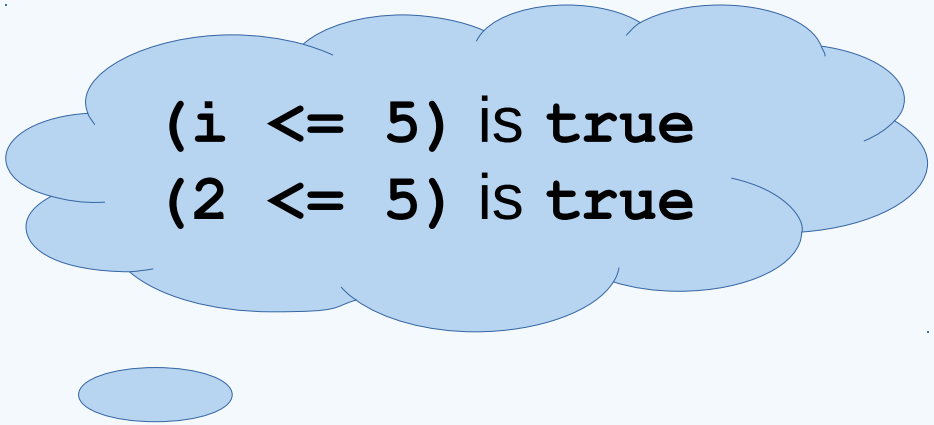
Tracing a `while` Loop



sum	1
i	2

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

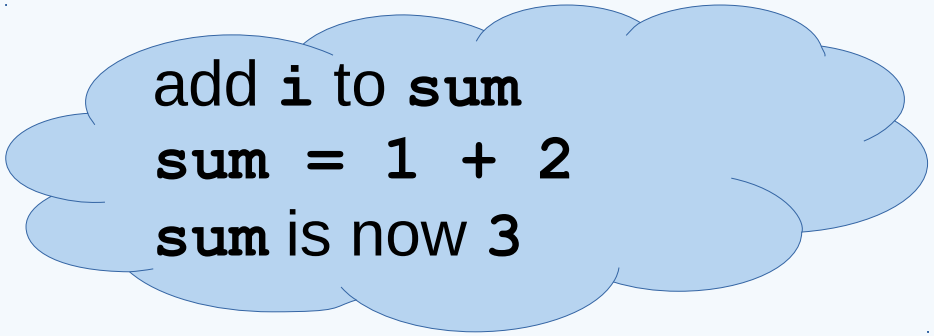
Tracing a `while` Loop



sum	1
i	2

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

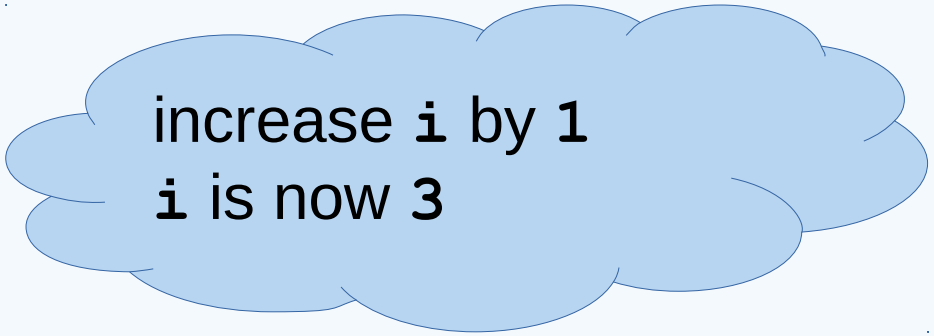
Tracing a `while` Loop



sum	3
i	2

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

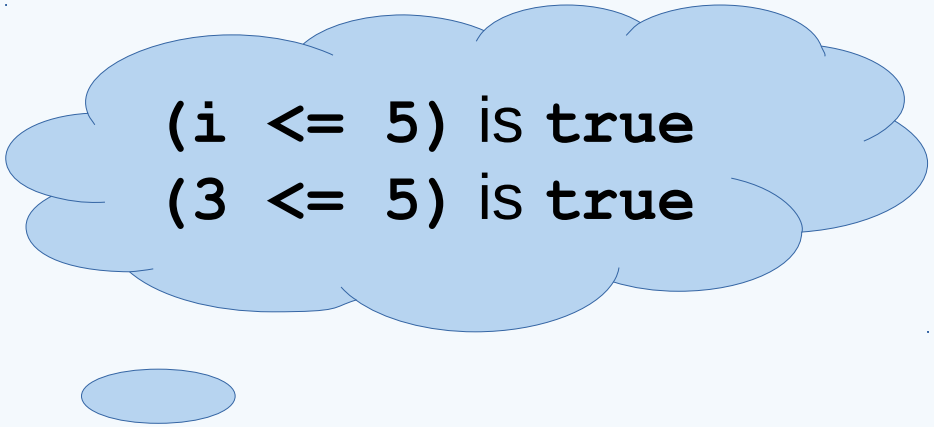
Tracing a `while` Loop



sum	3
i	3

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop



sum	3
i	3

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

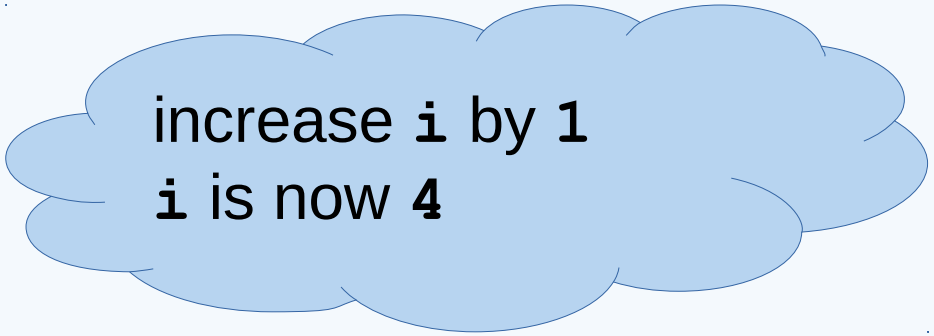
Tracing a `while` Loop

add `i` to `sum`
`sum = 3 + 3`
`sum` is now `6`

`sum`	6
`i`	3

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

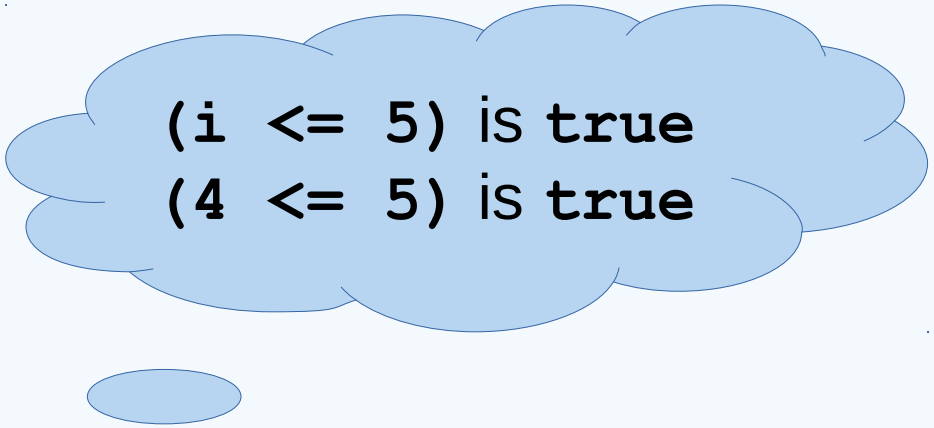
Tracing a `while` Loop



sum	6
i	4

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop



sum	6
i	4

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop

add `i` to `sum`
`sum = 6 + 4`
`sum` is now `10`

`sum`	10
`i`	4

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

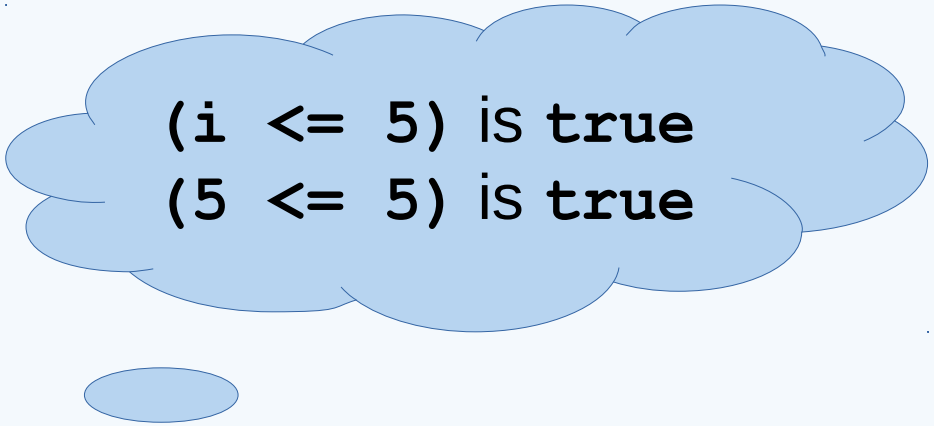
Tracing a `while` Loop

increase `i` by `1`
`i` is now `5`

`sum`	`10`
`i`	`5`

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop



sum	10
i	5

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop

add `i` to `sum`
`sum = 10 + 5`
`sum` is now `15`

`sum`	15
`i`	5

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop

increase `i` by `1`
`i` is now `6`

`sum`	`15`
`i`	`6`

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop

(i <= 5) is false
(6 <= 5) is false

sum	15
i	6

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Tracing a `while` Loop

Display the output

sum	15
i	6

```
int sum = 0, i = 1;
while (i <= 5) {
   sum = sum + i;
   i++;
}
System.out.println("sum is: " + sum);
```

Counter-Controlled Loops

- See Code:
  - **`RepeatAdditionQuiz.java`**
  - **`GuessNumberOneTime.java`**
  - **`GuessNumber.java`**
  - **`SubtractionQuizLoop.java`**

Controlling a Loop with a Sentinel Value

- ***sentinel value:*** a special input value used to control the loop execution.

- ***sentinel-controlled loop:*** a loop that uses a sentinel value to control its execution

- usually used to signify when the loop should terminate when you do not know beforehand how many times the loop will iterate.

- See Code: **`SentinelValue.java`**

Tracing Sentinel Example

- Prompt the user to enter a series of positive numbers and compute the average of all the numbers.
- The loop should end when the user enters a negative number.
- Numbers should be entered one per line.
- The negative number is the sentinel value.

```
System.out.println("Enter positive numbers 1 per line.\n"
  + "Indicate end of the list with a negative number.");

Scanner stdin = new Scanner(System.in);

int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	0

Tracing Sentinel Example

- Suppose the input is: **4.5 0.5 1.3 -1**

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	0
valueSum	0

Tracing Sentinel Example

- Suppose the input is: **4.5 0.5 1.3 -1**

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	0
valueSum	0
value	4.5

Tracing Sentinel Example

- Suppose the input is: **4.5 0.5 1.3 -1**

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	0
valueSum	0
value	4.5

Tracing Sentinel Example

- Suppose the input is: **4.5 0.5 1.3 -1**

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	0
valueSum	4.5
value	4.5

Tracing Sentinel Example

- Suppose the input is: **4.5 0.5 1.3 -1**

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	1
valueSum	4.5
value	4.5

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	1
valueSum	4.5
value	0.5

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	1
valueSum	4.5
value	0.5

Tracing Sentinel Example

- Suppose the input is: 4.5 0.5 1.3 -1

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	1
valueSum	5.0
value	0.5

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	2
valueSum	5.0
value	0.5

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	2
valueSum	5.0
value	1.3

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	2
valueSum	5.0
value	1.3

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	2
valueSum	6.3
value	1.3

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	1.3

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	-1

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	-1

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	-1

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	-1
average	2.1

Tracing Sentinel Example

- Suppose the input is: `4.5 0.5 1.3 -1`

```
int n = 0;
double valueSum = 0;

double value = stdin.nextDouble();

while (value >= 0) {
   valueSum = valueSum + value;
   n = n+1;
   value = stdin.nextDouble();
}

if (n > 0) {
   double average = valueSum / n;
   System.out.println("Average: " + average);
}
else {
   System.out.println("No list to average");
}
```

n	3
valueSum	6.3
value	-1
average	2.1

Caution with Floating-Point Numbers

- Don’t use floating-point values for equality checking in a loop control statement.
  - floating-point values are approximations
  - using them could result in imprecise counter values and inaccurate results.

- See **`SentinalWithDouble.java`**:
  - a floating-point type value is used for data
  - (data == 0) may be false even though data is mathematically 0.


The do-while Loop

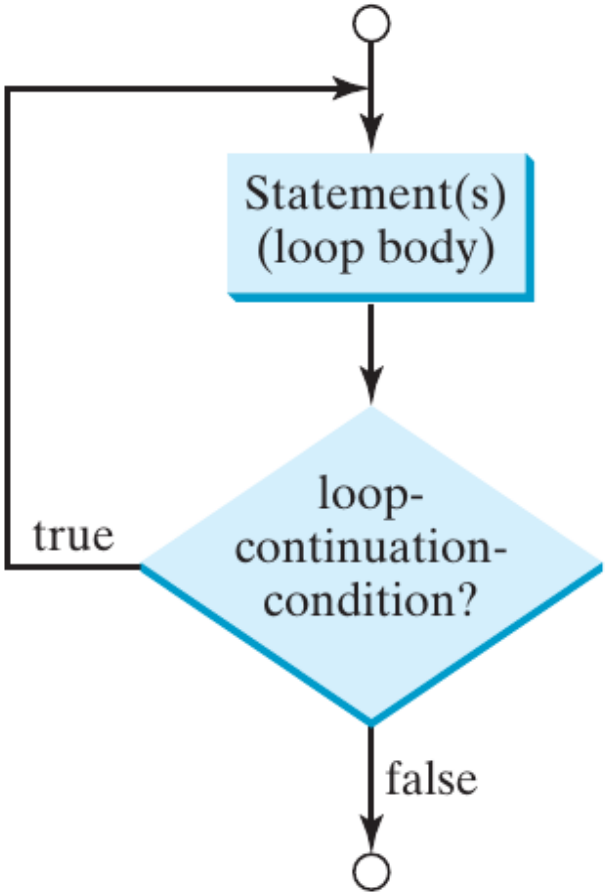
The do-while Loop

- **do-while** loops **execute the loop body first** and **then** checks the loop continuation condition
- Syntax:

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

- if the loop-continuation-condition is **true**, the loop will execute again, if it is **false** the loop will terminate.
- NOTE: You MUST include the semicolon (;) at the end of the loop-continuation-condition (otherwise its a syntax error)
- Code Example: **SentinalValueDoWhile.java**

The do-while Loop



The **do-while** loop executes the loop body first, then checks the **loop-continuation-condition** to determine whether to continue or terminate the loop.

Differences Between `while` and `do-while`

```
int count = 0;
while (count < n){
   statement;
   count = count + 1;
}
```

1. Check count < n?
2. Do the loop statement
3. The loop statement MAY NOT be executed!

```
int count = 0;
do {
   statement;
   count = count + 1;
} while (count < n);
```

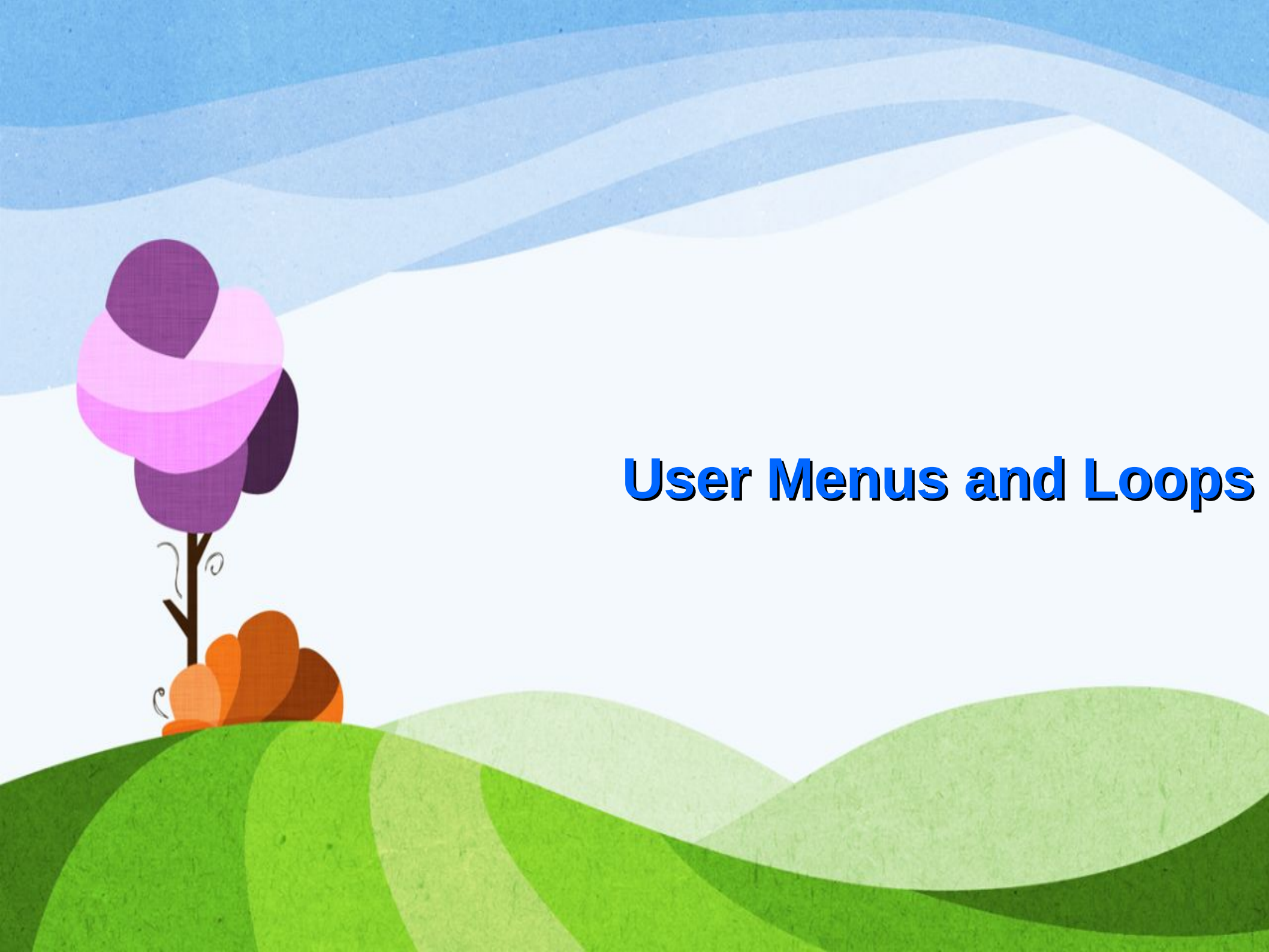
1. Do the loop statement
2. Check count < n?
3. The loop statement is executed AT LEAST once!

Differences Between `while` and `do-while`

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

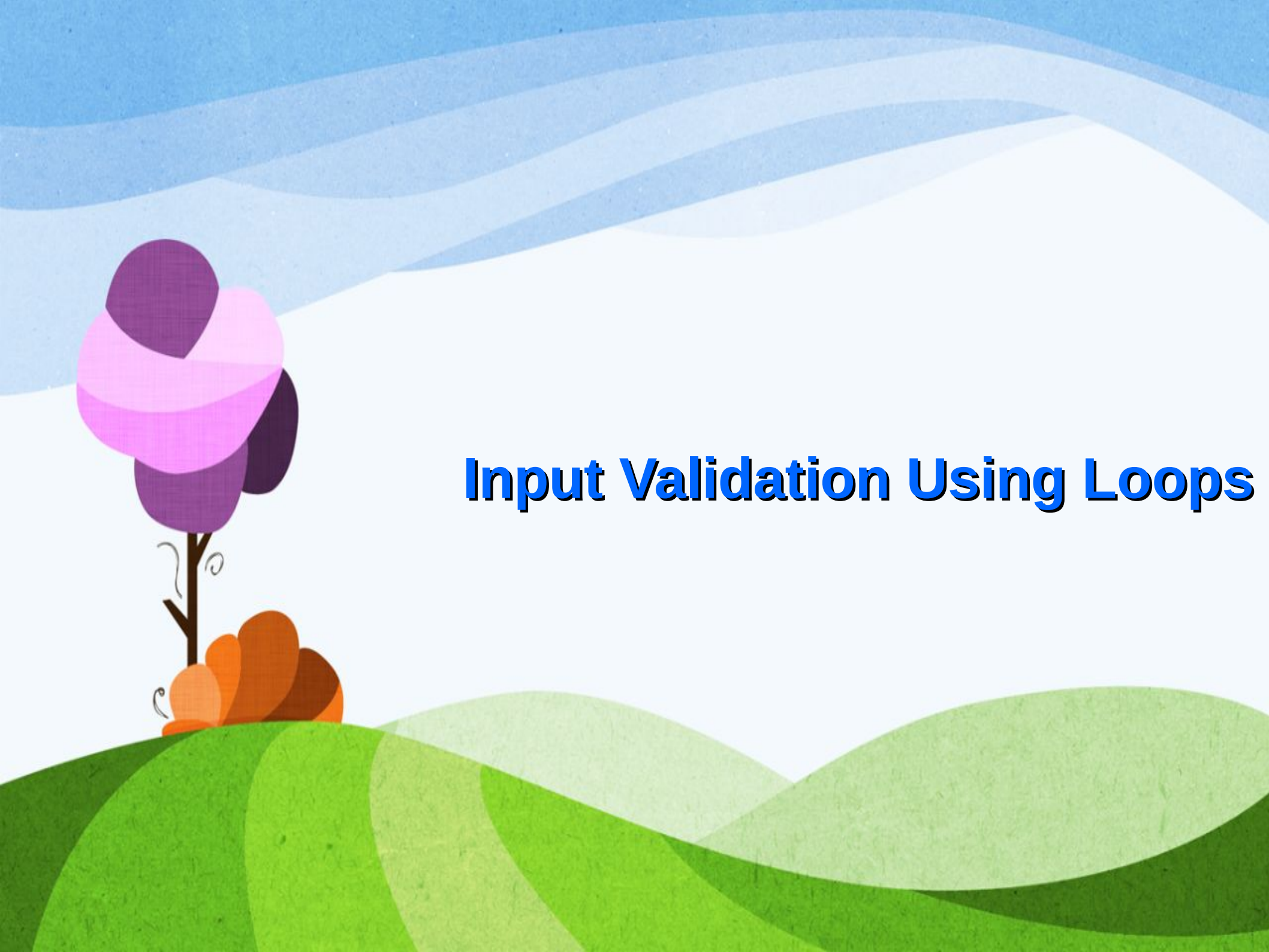
```
int i=0;
do {
   System.out.println(
      "i is " + i);
   i++;
} while (i < 10);
```

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


User Menus and Loops

Loops and User Menus

- Normally when creating a user menu, you want to keep the program running until the user chooses to exit.
- Loops allow you to redisplay the menu and allow the user to choose another option.
- See Code: UserMenuWithLoops.java


Input Validation Using Loops

Loops and Input Validation

- Previously when your user input fails a validation check, you simply exit the program.
  - This works up to a point, it is very tedious to restart a program everytime something fails.
  - Imagine if every time there was a tiny error in Windows or Mac OS, your computer shut down.
- Loops can be used to recover from simple user input mistakes and allow the user to try their input again.
  - All this can happen while keeping the program executing.
- The concept is simple, you trap the program in an infinite loop until the user gives you the correct input.
- See Code: InputValidationWithLoops.java