Control Structures and Methods
“For Newcomers in Silicon Valley, the Dream of Entrepreneurship Still Lives”

This is called the initialization statement and is performed before the loop starts.

This is called the step or increment and is performed at the end of each loop iteration.

```java
for (int i = 0; i < 3; i++) {
    ...
}
```

This is called the loop condition or termination condition. The loop will check whether this statement is true before each execution.
Nyan nyan nyan nyan, nyan nyan nyan nyan nyan nyan, nyan, nyan nyan nyan nyan nyan ...
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```java
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

```java
int i 0
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 0
for (int i = 0; i < 4; i++) {
  println("Nyan!");
}

int i 0
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i      0

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  0

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 1

Nyan!

Console Program
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 1

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 1

Console Program

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i

1

Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 1

Console Program

Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 2

Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  2

Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

```java
int i 2
```

Console Program

Nyan!
Nyan!
Nyan!
Nyan!
```java
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i = 3

Console Program

Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Console Program

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

Console Program

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!

int i 4
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i  4

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  4

Console Program

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
Accessing the Loop Counter

```java
for (int i = 0; i < 4; i++) {
    println("Value is "+ i);
}
```

Console Program

Value is 0
Value is 1
Value is 2
Value is 3
```java
for (int i = 5; i > 0; i--) {
    println(i + "...");
}
println("Lift-off!");
```
T-15 Seconds: Guidance is Internal
T-9 Seconds: Ignition Sequence Start
T-0 Seconds: All Engines Running

```java
for (int i = 30; i > 0; i--) {
    println("T-" + i + "...");
}
println("Lift-off!");
```
if statement

- General form: if (condition) {
  statements
}
  
  if (first > second) // can omit braces
  println("First is bigger"); // if one statement

  if (first > second) {
    println("Brace yourself...");
    println("...for an embrace!");
  }

- Use braces with if with more than one statement
- Good idea to use braces (block) even if there is only one statement in the if

Based on slides by Mehran Sahami
private static final int COUNTDOWN_START = 30;
private static final int GUIDANCE_START = 15;
private static final int IGNITION_START = 9;

public void run() {
    /* Do the launch countdown! */
    for (int i = COUNTDOWN_START; i > 0; i--) {
        println("T-" + i + " seconds.");
    }

    /* Specific mission commands. */
    if (i == GUIDANCE_START) {
        println("Guidance is internal.");
    }
    if (i == IGNITION_START) {
        println("Ignition sequence start.");
    }
}

println("All engines running. Lift-off!");
private static final int COUNTDOWN_START = 30;
private static final int GUIDANCE_START = 15;
private static final int IGNITION_START = 9;

public void run() {
    /* Do the launch countdown! */
    for (int i = COUNTDOWN_START; i > 0; i--) {
        println("T-" + i + " seconds.");

        /* Specific mission commands. */
        if (i == GUIDANCE_START) {
            println("Guidance is internal.");
        }
        if (i == IGNITION_START) {
            println("Ignition sequence start.");
        }
    }
    println("All engines running. Lift-off!");
}
if-else statement

- General form: if (condition) {
  statements
} else {
  statements
}

```java
if (((num % 2) == 0) {
  println("num is even");
} else {
  println("num is odd");
  println("and so are you");
}
```

Based on slides by Mehran Sahami
Cascading if

```java
if (score >= 90) {
    println(" AWWWW YEAHHHHH ");
} else if (score >= 80) {
    println(" <(^_^)> ");
} else if (score >= 70) {
    println(" : - | ");
} else if (score >= 60) {
    println(" ง_ง ");
} else {
    println(" (°□° )'(_,_) ");
}
```

Based on slides by Mehran Sahami
Control Structures in Karel

for
if
while
Control Structures in Karel

for

if

while
The **while** Loop

```java
while (condition) {
    ... statements ...
}
```

- Checks *condition* before each iteration and executes *statements* if true.
- Does **not** check *condition* in the middle of the loop.
while loop

Example:

```java
int x = 15;
while (x > 1) {
    x /= 2;
    println(x);
}
```

Based on slides by Mehran Sahami
break-ing out of a Loop

• The **break** statement immediately exits a loop.

```java
int x = 120;
while (x > 1) {
    x /= 2;
    if (x % 2 == 1)
        break;
    println(x);
}
```

```
60
30
```
Looping Forever

- Recall: **while** loops iterate as long as their condition evaluates to **true**.

- A loop of the form **while (true)** will loop forever (or until a **break** is executed).

```
while (true) {
    ...
}
```
The “Loop-and-a-Half” Idiom

- Often you will need to
  - read a value from the user,
  - decide whether to continue, and if so
  - process the value.

- The **loop-and-a-half idiom** can be used:

```java
while (true) {
    /* … get a value from the user … */
    if (condition)
        break;
    /* … process the value … */
}
```
for versus while

for (init; test; step) {
    statements
}

init
while (test) {
    statements
    step
}

• **for** loop used for *definite* iteration.
• Generally, we know how many times we want to iterate.

• **while** loop used for *indefinite* iteration.
• Generally, don’t know how many times to iterate beforehand.
Each point $k$ is connected to point $k + 2$, after wrapping around.
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$.
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$

$(x, y)$

$(x + r \cos \theta, y - r \sin \theta)$

$1 \ (51.43^\circ)$
Passing Parameters

- A method can accept **parameters** when it is called.

- Syntax:

  ```java
  private void name(parameters) {
      /* ... method body ... */
  }
  ```

- The values of the parameters inside the method are set when the method is called.

- The values of the parameters can vary between calls.