Lecture 4

Loops

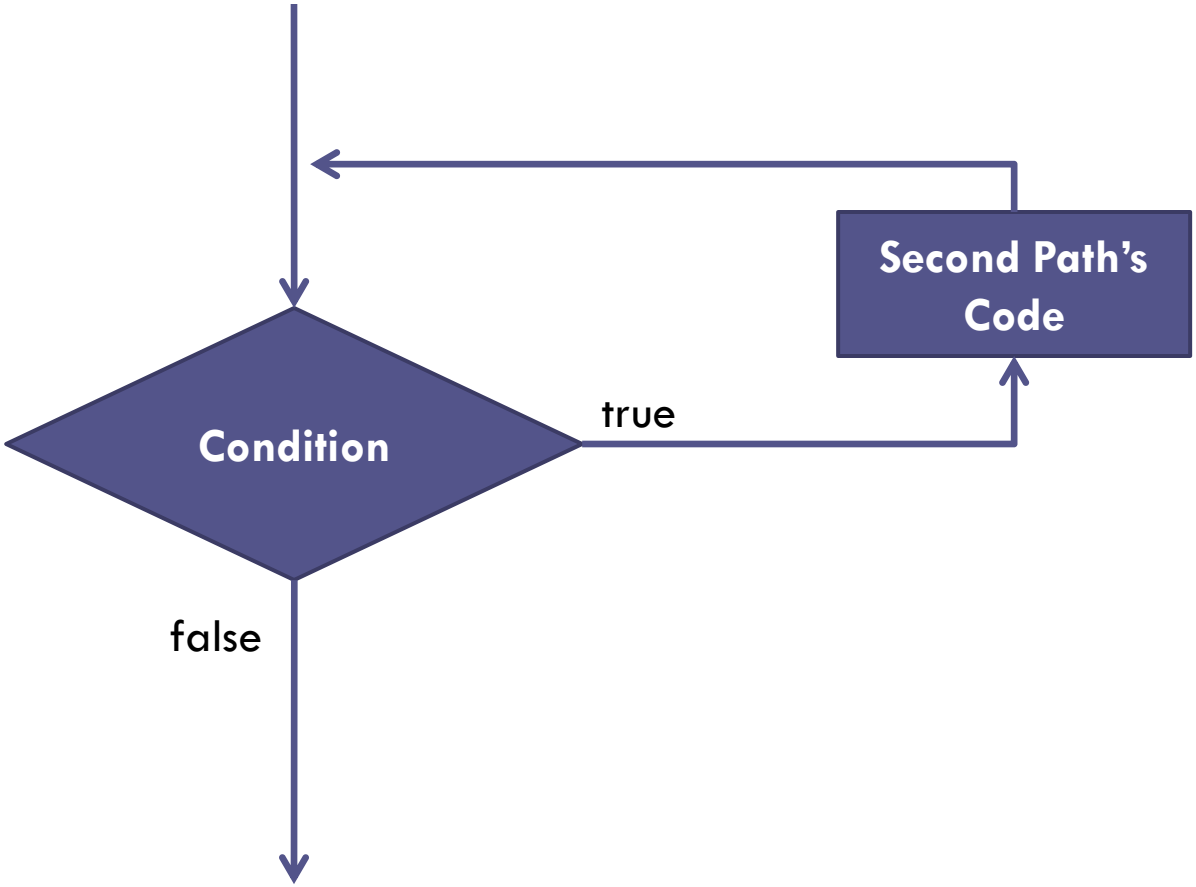
Repeating Code

- A **loop statement** is a control structure that repeatedly executes a statement or a series of statements while a specific condition is true or until a specific condition becomes true

while loop

Repeats a statement or a series of statements as long as a given conditional expression evaluates to true

while loop flow



while loop syntax

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

As long as the condition is true we will keep executing the statements repeatedly

while statement terms

- iteration: each time the loop is entered
- A while statement keeps repeating until its conditional expression evaluates to false
- counter: the variable that increments or decrements with each iteration of a loop statement

while example

- Looping from 1 to 10:

```
int counter = 1;

while(counter <= 10)
{
    println(counter);
    counter++;
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Looping Backwards

- Looping from 10 to 1:

```
int counter = 10;

while(counter >= 1)
{
     println(counter);
     counter--;
}
```

Output:

```
10
9
8
7
6
5
4
3
2
1
```

Going up by 2's

- Looping from 2 to 20 by 2's:

```
int counter = 2;

while(counter <= 20)
{
    println(counter);
    counter +=2;
}
```

Output:

```
2
4
6
8
10
12
14
16
18
20
```

do while loop

A do...while executes a statement or statements once, then repeats the execution as long as a given conditional expression evaluates to true

do while loop

- Note that some while statements will never execute

```
int x = 9;
while (x > 10) { statements; }
```

- But in a do while loop they'll still run one iteration of the loop even if the condition isn't met

do while syntax

```
do
{
  //statements
}
while(condition)
```

do while (…)

- **do...while** statements always execute once, before a conditional expression is evaluated

```
int counter = 11;

do
{
    println(counter);
    counter++;
}
while(counter <= 10);
```

Output:

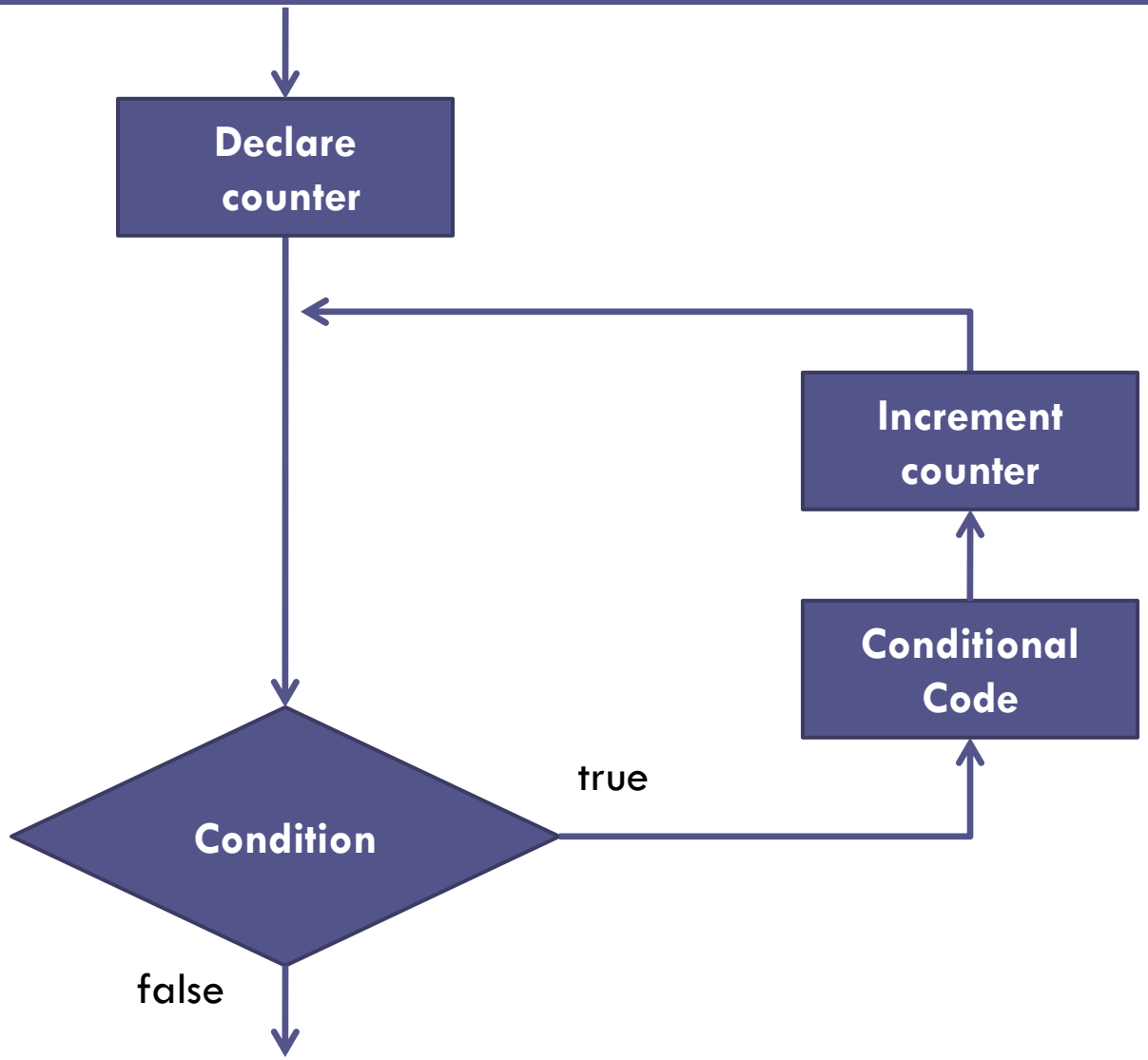
```
11
```

for loop

Used for repeating a statement or a series of statements as long as a given conditional expression evaluates to true

Particularly useful when you know how many times you want the loop to be run

for loop flow diagram



for syntax

```
for(initialization; condition; update statement)
{
    //statements
}
```

for example

- Looping from 0 to 10

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

Output:

```
0
1
2
3
4
5
6
7
8
9
10
```

for explained

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

Part	Explanation
int i = 0	Sets the loops counter to start at 0. Only executed once at beginning of loop
i <= 10	Loops condition… every time we go through the loop. If this is true we continue otherwise we skip to the code after the loop
i ++	Increments the counter each time after we've completed the loop
;	Semicolons separate the parts. Very necessary, don't forget

infinite loop

In an infinite loop, a loop statement never ends because its conditional expression is never false

E.g. You forgot to change the value of the counter inside of a loop.

infinite loop

- Here' s an example of an infinite loop.
- It's impossible for the condition to be false so it just loops forever

```
int counter = 10;

while(counter >= 1)
{
    println(counter);
    counter ++;
}
```

Common Infinite loop

```
while(true)
{
   //execute this forever
}
```

Be aware of this when debugging as it can cause you annoying errors.