

Iteration and Loops



Objectives

- While loops
- Do While
- Basic For loops.

Code flow

- Application starts in main method
- Application terminates at end of main method
- Need to stop the application from reaching end of main method.

```
public static void main(String[] args) {  
    //Used for taking user input from console  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.println("How old are you?");  
    int age = scanner.nextInt();  
  
    if(age >= 18) {  
        System.out.println("You are an adult");  
    } else {  
        System.out.println("You are a child");  
    }  
}
```

What can we do to keep the
program from terminating?

Application starts

Instantiating a Scanner object

Prints out a String

Stores input from scanner in int age

Checks if age is 18 or above

If true print a message

If false print another message

Application terminates



Code Flow

- We want to:
 - Print a message
 - Take user input
 - Evaluate the input
 - Print out result of evaluation
 - Terminate the program only if user enter integer below 18

Introducing the While loop

Key word Parentheses

Curly bracers { }
wrapping the body

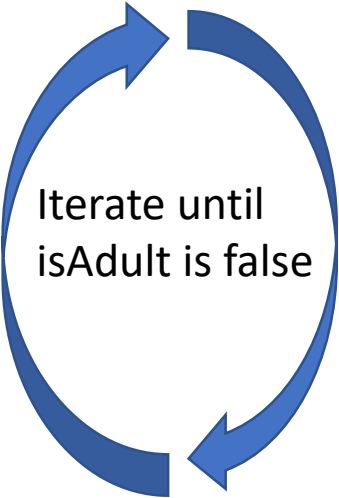
```
while(booleanExpression) {  
    // Body  
}
```

The diagram illustrates the syntax of a while loop. It shows the code `while(booleanExpression) { // Body }`. Annotations include: a vertical arrow pointing to `while` labeled 'Key word'; a horizontal double-headed arrow spanning `(booleanExpression)` labeled 'Parentheses'; and a vertical double-headed arrow spanning the curly braces `{ }` labeled 'Curly bracers { } wrapping the body'.

This could be a good solution
for the termination problem.

Boolean controlling the loop

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    boolean isAdult = true;  
  
    while(isAdult) {  
        System.out.println("How old are you?");  
        int age = scanner.nextInt();  
  
        if(age >= 18) {  
            System.out.println("You are an adult");  
        } else {  
            System.out.println("You are a child");  
            isAdult = false;  
        }  
    }  
}
```

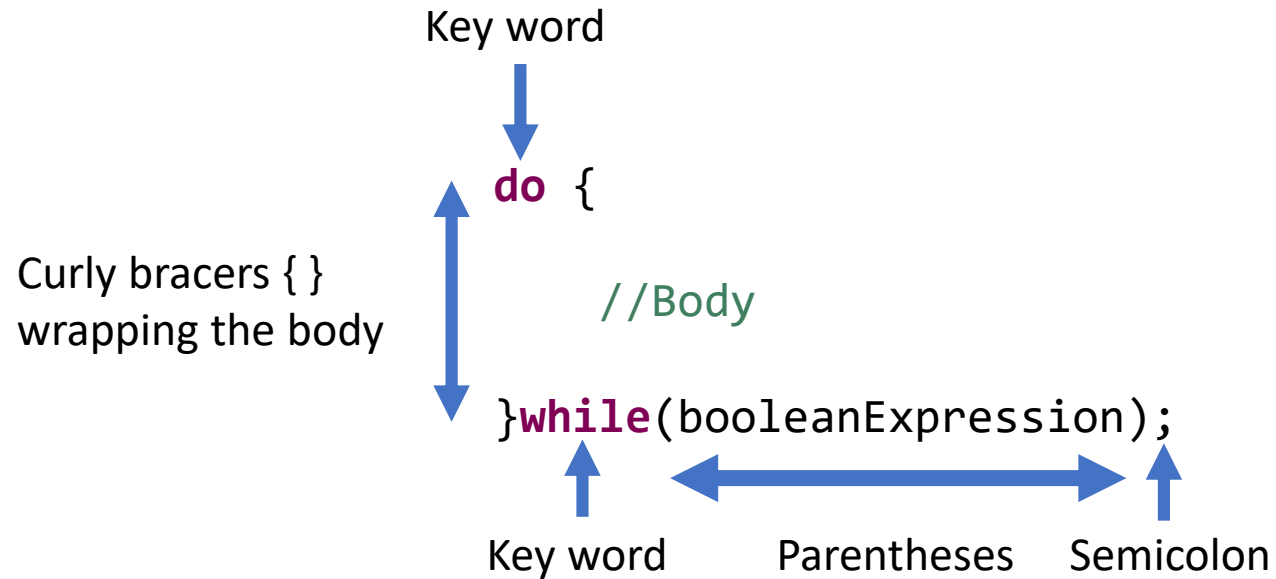


Iterate until
isAdult is false

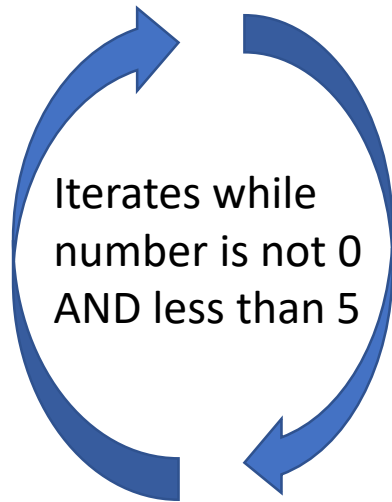
When to use a While loop?

- When we need to repeat a code section an **unknown** number of times.
- It will repeat as often as necessary until it accomplish its goal.

Introducing the Do While loop



This loop will always run at least once



```
int number = 5;  
  
do {  
    number++;  
}while(number != 0 && number < 5);  
  
System.out.println(number);
```

What do you think it will print?

For loops

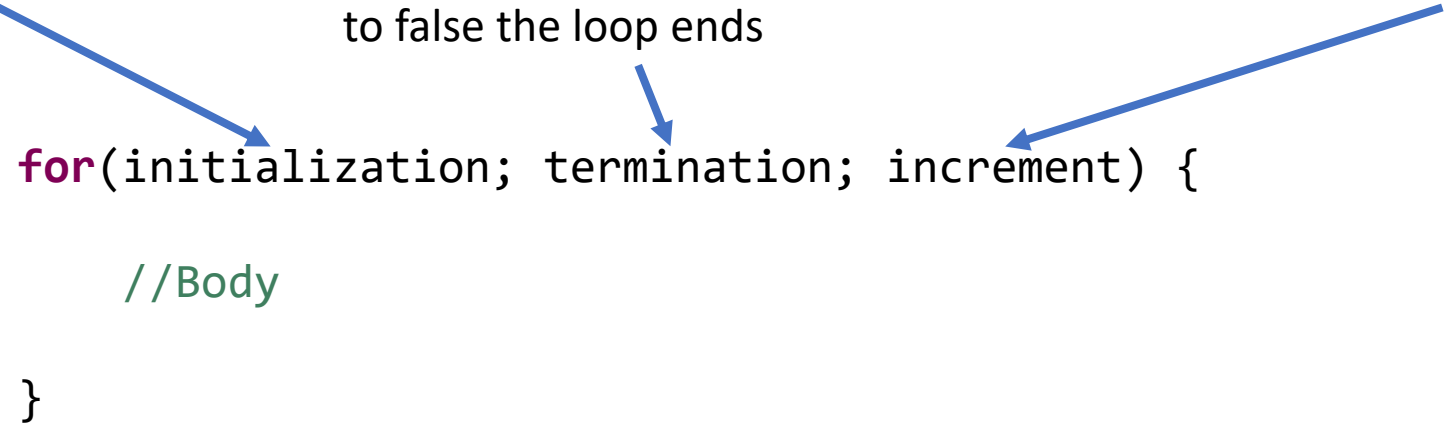
- Iterates a known amount of times
- Used when iterating through a range of values. (Arrays, Lists etc..)

General for construct

Initializes the loop and its
executed once, as the loop
begins.

Termination is a boolean
expression. When it evaluates
to false the loop ends

The increment expression is
executed after each iteration.

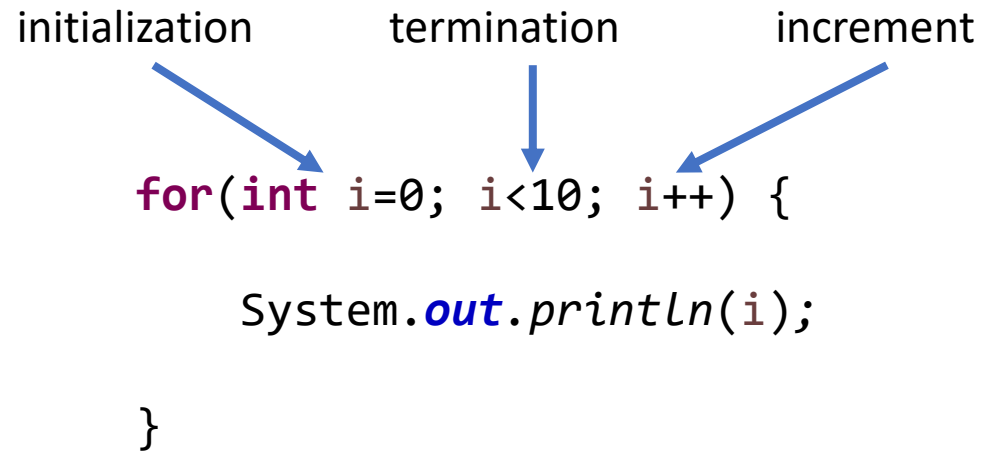


```
for(initialization; termination; increment) {  
    //Body  
}
```

The diagram illustrates the general for loop construct. Three blue arrows point from explanatory text to the corresponding parts of the code: one from 'Initializes the loop and its executed once, as the loop begins.' to the 'initialization' part, one from 'Termination is a boolean expression. When it evaluates to false the loop ends' to the 'termination' part, and one from 'The increment expression is executed after each iteration.' to the 'increment' part. The code is written in a monospaced font, with 'for' in purple and the body comment '//Body' in green.

Example

initialization termination increment



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

The increment statement
can both increment,
decrement, calculate etc..

For – each loop

- Used when iterating through an array or collection.
- Do not have a counter like the other for loop.
- Instead of the counter you declare a variable with the same type as the elements of the Array/Collection, followed by a colon and then the Array/Collection name.
- You use your declared variable to access each element in the loop.

For – each example


```
public static void main( String[] args ){  
    int [] numbers = {11,4,7,3,32,8,9,12,5,23,35,22};  
  
    int max = 0;  
    for(int number : numbers) {  
        if(number > max)  
            max = number;  
    }  
  
    System.out.println("Highest number is " + max);  
}
```

break statement

- break
 - A **break** statement inside a loop immediately terminates the loop and the program resumes at the next statement after the loop.
 - Is also used to terminate a case in the **switch** statement.

break example

```
public static void main( String[] args ){  
  
    Scanner scanner = new Scanner(System.in);  
    String input = "";  
  
    // Create an infinite loop  
    while(true) {  
        System.out.println("Hello! Enter some text");  
        System.out.print("> ");  
        input = scanner.nextLine();  
        System.out.println("You typed in " + input);  
        System.out.println("Continue? (y/n)");  
        input = scanner.nextLine();  
        if(input.equalsIgnoreCase("n"))  
            break;  
    }  
}
```



Break loop if input is "n" or "N"

continue statement

- Continue is sometimes used inside loops. If it is present inside a loop, control directly skips the remaining statements and jumps to the beginning of the next iteration.

```
for(int i=1; i<=100; i++) {  
    if(i % 7 != 0) {  
        continue; //Continue statement abort this iteration and continue on the next  
    }  
    System.out.println(i); //Should only print numbers divisible by 7  
}
```