

### Ternary Operator

A shortcut to assigning one of two values to a variable depending on a given condition

Like an **if-then-else** statement

Question mark comes after the condition

After the question mark, two values that can return are separated by a colon (:)

Takes 3 operands: `condition ? operand1 : operand2`  
**Condition** we're testing against  
**First value** to assign if first condition was true

Example:

```
int age = 20
boolean isOver18 = (age == 20) ? true : false
// is age equal to 20? if it is, isOver18= true if false, isOver18= false
```

### Operator Abbreviation

Original	Abbreviated
<code>result = result + 1;</code>	<code>result++;</code>
<code>result = result - 1;</code>	<code>result--;</code>
<code>result = result + 2;</code>	<code>result += 2;</code>
<code>result = result * 10;</code>	<code>result *= 10;</code>
<code>result = result / 3;</code>	<code>result /= 3;</code>
<code>result = result - 2;</code>	<code>result -= 2;</code>

### Access Modifiers

These are java keywords

Allows defining the scope, how other parts of the code can access this code

Access Modifiers	Access Levels
<code>public</code>	Same Class, same package, other subclass, other package
<code>protected</code>	Same Class, same package, other subclass
<code>no access modifier</code>	Same Class, same package
<code>private</code>	Same Class

### Code comments

These are used to describe methods for quick reference with an IDE

```
Start      /**      End comment */
comment                                block:
block:
Describe   Computes sum of two arguments
method:
Describe   @param a   an int operand
parameters
           @param b   an int operand
Describe   @ return   the sum of a and b
what
method
returns:
Method     public     static int   sum(int a, int b)
described:
Full example:
/**
 * Computes the sum of two arguments.
 *
 * @param a an int operand to be added
 * @param b another int operand
 * @return the sum of a and b
 */
public static int sum(int a, int b)
```

### For loop

For loops are used when you know exactly how many times you want to loop through a block of code

```
for (statement 1; statement 2; statement 3) { }
```

```
for (int i = 0; i < 5; i++) { }
```

sets a variable before the loop starts	Defines the condition for the loop to run. If true, the loop will start over again, if false, the loop will end.	increases a value (i++) each time the code block in the loop has been executed.
--	--	---

### For-Each Loop

Used exclusively to loop through elements in an array:

```
for (type variableName : arrayName) {}
```

e.g.

```
String[] cars = {"Volvo ", " BMW ", " Ford", " Mazda"};
```

```
for (String i : cars)
```

```
System.out.println(i);
```

### While Loops

A while loop will execute the enclosed statement as long as a boolean condition remains true.

Syntax: `while (boolean_condition) statement`

The condition must be boolean.

If the condition never becomes false, the loop never exits, and the program never stops.

Example:

```
n = 1;
while (n < 4) {
    System.out.println(n + " squared is " + (n * n));
    n = n + 1;
}
```

Result:

```
1 squared is 1
2 squared is 4
3 squared is 9
```

### When do you use each loop

for loop if you know ahead of time how many times you want to go through the loop.

while loop in almost all other cases.

do-while loop if you must go through the loop at least once before it makes sense to do the test.

### ArrayList

Create arraylist:

```
ArrayList <Type> varName = new ArrayList <
```

e.g.

```
ArrayList <String> names = new ArrayList <
```

Add object to arraylist `varName.add(object)`

e.g. `names.add("Alice")`

Get size of arraylist: `varName.size();`

e.g. `names.size();`

Change object with index `varName.set(index, object);`

e.g. `names.set(0, "Anna");`

Remove object with index `varName.remove(index)`

e.g. `names.remove(0)`

Passing as a parameter example: `public static double average(Arr`  
`ArrayList<Double> arr)`

### Arrays

Create array:	<code>type[] varName = new type[size];</code>
e.g.	<code>double[] arr= new double[5]; //5 objects</code>
Get length of array:	<code>varName.length</code>
e.g.	<code>arr.length</code>
Access object with index	<code>varName[index]</code>
	<code>arr[1]</code>

### Reading input

#### Structure

Import `import java.util.Scanner;`  
Scanner  
class:

Create a scanner, assign it to a variable:  
`Scanner scanner = new Scanner(System.in);`

`new Scanner(..)` creates a new one

System.in says scanner is to take input from the keyboard

Request user to input number  
`System.out.println("Please input data:");`

Read in number:  
`myNumber = scanner.nextInt();`

Read in String  
`String myString = scanner.nextLine();`

Read in double  
`String myDouble = scanner.nextDouble();`

Read in char:  
`char myChar = scanner.next().charAt(0);`

### break statement

Inside any loop, the break statement will immediately get you out of the loop.

If you are in nested loops, break gets you out of the innermost loop

It doesn't make any sense to break out of a loop unconditionally; you should do it only as the result of an if test

break should not be the normal way to leave a loop

Use it when necessary, but don't overuse it.

Example:

```
for (int i = 1; i <= 12; i++) {  
    if (badEgg(i))  
        break;  
}
```

### Inheritance

subclass (child) the class that inherits from another class

superclass (parent) the class being inherited from

To inherit from a class we use `extends`

class A extends B means class A (subclass) inherits attributes and methods from class B(superclass)

### Polymorphism

# C

By Bayan (Bayan.A)  
[cheatography.com/bayan-a/](https://cheatography.com/bayan-a/)

Not published yet.  
Last updated 19th July, 2023.  
Page 3 of 3.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>