Priority

- What is Java? • James Gosling "Oak"
  • 1995 - Java

- Java is a Popular programing language, Created in 1965.
- It owned by oracle, and more then 3 billion devices run Java.

It is used for :-

- Mobile application (Specially Android apps)
- Desktop applications
- Web application
- Web Servers and application servers
- Games
- Database Connection
-

Scanned with CamScanner

Priority

- Why Use Java?

- Java work on different platforms (windows, mac, linux, Pi, Raspberry etc.)
- It is one of the Most Popular Programming language in the world.
- It is easy to learn and Simple to use.
- It is open Source and free.
- It is Secure, Fast and Powerful
- It has a huge Community Support (tens of millions of developers)
- Java is an object oriented language which give a clear Structure to programs and allows code to be reused, lowering development Cost.
- As Java is close to C++ and C#, it makes it easy For programmers to switch to Java or vice versa.

Wk	22	23	24	25	26
Mon		4	11	18	25
Tue		5	12	19	26
Wed		6	13	20	27
Thu		7	14	21	28
Fri	1	8	15	22	29
Sat	2	9	16	23	30
Sun	3	10	17	24	

Scanned with CamScanner

Priority

★ Setup For Windows

To install Java on Windows:

1. Go to "System Properties" (can be found on Control Panel > System and Security > System > Advanced System Settings)

2. Click on the "Environment variable" button under the "Advanced" tab

3. Then, Select the "Path" variable in System variables and click on the "edit" button

4. Click on the "New" button and add the path where Java is ~~in~~ is installed. Followed by \bin.

Then click on "OK" and save the setting.

Scanned with CamScanner

Priority

5. At last, Open Command Promipt (cmd.exe) and type java-version to see if Java is running on your machine.

★ Java Quick Start

- In Java, every application begins with a class name, and that class must match the Filename.

- Let's Create Our First Java File, Called MyClass.Java, which Can be done in any text editor (like Notepad).

- The file Should Contain a "Hello World" Message, which is written with the Following Code:

Myclass.Java

```
Public class Myclass {
    Public static void main(String[] args)
    {
      System.Out.Println("Hello world");
    }
}
```


Scanned with CamScanner

Priority

Example explained

Every line of Code that run in Java must be inside a class. In our example, we named the class MyClass. A class should always start with an uppercase First letter.

Note :- Java is case-sensitive : "MyClass" and "myclass" has different Meaning.

The name of the Java file must match the class name. When saving the file, save it using the class name and add ".Java" to the end of the file name.

The Output should be :-

```
Hello World
```


Scanned with CamScanner

Priority

The Main Method

The Main() method is required and you will see it in every Java Program:

```
Public static void main(String[] args)
```

Any Code inside the ~~main~~ Main() method will be executed. you don't have to understand the keywords before and after main.

System.Out.Println()

Inside the Main() method, we can use the ~~Println~~ (~~)~~ Println() method to Print a line of text to the screen:

```
Public Static void Main (String [] args) {
System.Out.Println ("Hello World");
}
```

Note:

- The curly braces {} ~~Make~~ marks the beginning and the end of a block of Code.
- Each Code statement Must end with a semicolon.

Scanned with CamScanner

Priority

Java Comments

Comments can be used to explain to Java Code, and to make it more readable. It (can also) be used to prevent execution when testing alternative code.

Single-line Comments start with two Forward slashes (//).

Any text between // and the end of the line is ignored by Java (will not be executed).

Example

```
// This is a Comment
System.out.println("Hello World");
```


Scanned with CamScanner

Java multi-line Comments:-

Multi-line Comments Start with /* and ends with */.

Any text between /* and */ will be ignored by Java.

"This example uses a multi-line Comment (a comment block) to explain the Code:

Example:

```
/* The Code below will print the words
   Hello World to the screen, and it is amazing */

System.out.Println("Hello world");
```

Note

It is up to you which you want to use. Normally, we use // for short comment, and /* */ for longer.

Scanned with CamScanner

Priority

Java Variables

Variables are containers for storing data values.

In Java, there are different type of variables, for example:

- String - Stores text, such as "Hello". String value are surrounded by double quotes.
- int - Stores integers (whole number), without decimal, such as 123 or -123
- Float - Stores Floating Point numbers, with decimal, such as 19.99 or -19.99
- Char - Stores single character such as 'a' or 'B' char value are surrounded by single quotes.
- boolen - Stores value with two states: True or false.

May 2018
Wk 18 19 20 21 22
Mon 7 14 21 28
Tue 1 8 15 22 29
Wed 2 9 16 23 30
Thu 3 10 17 24 31
Fri 4 11 18 25
Sat 5 12 19 26
Sun 6 13 20 27

Scanned with CamScanner

Declaring (creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax:-

Type Variable = Value;

where type is one of Java's type (such as int or String) and Variable is the name of the Variable (such as x or name). The equal sign is used to assign values to the variable.

- To create Variable that should store text.

Example:-

Create a variable called name of type String and assign it the value "John":

```
String name = "John";
System.out.Println (name);
```

Priority

To create a variable that should store a number:

Example :-

Create a variable that should store a number,
Create a variable called MyNum of type int
and assign it the value 15:

```
int myNum = 15;
System.out.println(myNum);
```

~~You can also declare~~

Other Types:-

A demonstration of how to declare variable of other types:

Example:-

```
int myNum = 5;
Float MsFloat Num = 5.99F;
Char myletter = 'D';
boolen MyBool = true;
String mytext = "Hello";
```


Scanned with CamScanner

★ Display Variables

The Println() Method is often used to display Variables.

To Combine both text and a variable, use the + Character:

Example :-

```
String name = "John";
System.out.Println("Hello" + name);
```

You Can also use the + Character to add a Variable to another Variable:

Example

```
String FirstName = "John";
String lastName = "Doe";
String FullName = FirstName + lastNamo;
```

~~String~~

```
System.out.Println(FullName);
```


Scanned with CamScanner

Priority

#

For numeric value, the + character works as a mathematical operator (Notice that we use int (integer) variable here)

Example:-

```
int x = 5;
int y = 6;
System.out.Println(x+y);
// x Stores the value 5
// y Stores the value 6
// Then we use the Println() method to display the value of x+y, which is 11
```


Scanned with CamScanner

Priority

Declare Many Variables:-

To declare more than one variable of the same type, use a comma-separated list:-

Example:-

```
int x = 5 , y = 6 , z = 10;
System.out.Println(x+y+z);
```

Priority

Java Identifiers

- All Java Variables must be identified with unique names.
- These unique names are called identifiers.
- Identifiers can be short name (like x and y) or more descriptive name (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

Example:-

```
// Good
int minutesPerHour = 60;
// ok, but not so easy to understand what
   m actually is
int m = 60;
```


Scanned with CamScanner

The general rules for Constructing name for Variables (unique identifiers) are :-

- Names Can Contain letters, digits, underscores and dollar Signs.
- Names Must begin with a letter.
- Names Should Start with a lowercase letter and it Cannot Contain whitespace
- Names Can also begin with $ and _
- Names are case sensitive. (like Java keyword, such as int or boolean) Cannot be used as name.

Priority

Java Data Types

As explained in the a variable in Java must be a ~~sole~~ specified data type:

Example

```
int myNum = 5;                    // Integer (whole number)
Float myFloatNum = 5.99F;         // Floating Point number
Char myLetter = 'D';              // Character
bollean myBool = true;            // Boolean
String myText = "Hello";          // String
```

Datatype are Divided into two group.

- Primitive Data type :-
  Includes byte, Short, int, long, Float, double, boolean and Char

- Non Primitive Data type :-
  Such as String, Array, Classes

Scanned with CamScanner

Primitive Data Type

A primitive data type specifies the size and type of variable values, and it has no additional methods.

There are eight primitive data types in Java:

Datatype	Size	Description
byte	1 byte	Stores whole number -128 to 127
Short	2 byte	Store whole number -32 768 to 32767
int	4 byte	Store whole number - 2147 483 648 to 2147483647
long	8 byte	" " " - 9223 3372 036854 - → +
double	8 byte	Store fractional number, id store 15 decim
boolean	1 bit	Store true or false value
Char	2 byte	Stores a single character / letter or ASCII values.

Priority

Non-Primitive Data Types

Non-Primitive data types are called reference types because they refer to objects.

The main difference between Primitive and Non-primitive data types are

- Primitive types are predefined (already defined) in Java. Non-Primitive types are created by the Programmer and is not defined by Java (except For String).
- Non-Primitive type can be used to call Methods to perform certain operations, while Primitive types cannot.

Scanned with CamScanner

- A primitive type has always a value, while non-primitive types ~~starts with an uppercase letter~~. can be null.
- A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
- The size of a primitive type depends on the data type, while non-primitive type have all the same size.

Examples of non-primitive type are String, Arrays, Classes, Interface, etc.

You will learn more about these in a later chapter.

Scanned with CamScanner

Priority

Java type Casting

Type Casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of Casting:

- Widening Casting (automatically) → Converting a smaller type to a larger type size.

byte → Short → char → int → long → Float → Double

- Narrowing casting (manually) → Converting a larger type to a smaller size type

double → Float → long → int → char → Short → byte

Scanned with CamScanner

Priority

Widening Casting

Widening Casting is done automatically when passing a smaller size type to a larger size type:-

Example

```
Public Class MyClass {
    Public static void main(String[] args)
    {
        int myInt = 9;
    double myDouble = myInt;
System.Out.Println(myInt); // output 9
System.Out.Println(myDouble); //output 9.0
    }
}
```


Scanned with CamScanner

Narrowing Casting

Narrowing Casting must be done manually by placing the type in parenthesses in front of the value:

Example :-

```
Public Class Myclass {
    Public Static Void Main(String[] args {
    double myDouble = 9.78 ;
    int MyInt = (int ) MyDouble ;
    System . Out. Println (myDouble);
    System . out. Println (MyInt) ;
    }
}
```


Scanned with CamScanner

Java Operators

- Operators are used to perform operations on variables and values.
- In the example below, we use the + operator to add together two value:

Example

```
int x = 100 + 50;
```

Although the + operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable:-

```
int sum1 = 100 + 50;
int sum2 = sum1 + 250;
int sum3 = sum1 + sum2;
```


Scanned with CamScanner

Priority

Java divides the operators into the following groups:-

- Arithmetic Operators
- Assignment operators
- Comparison operators
- Logical Operators
- Bitwise Operators

★ Arithmetic Operators:- (common math oper

Operator	Name	Description	example
+	Addition	Add together two values	x+y
-	Subtraction	Subtract one value from other	x-y
*	multiply		x*y
/	Division		x/y
%	Modulus		x%y
++	Increment		++x
--	Decrement		--x

May 2018
Wk 18 19 20 21 22
Mon 7 14 21 28
Tue 1 8 15 22 29
Wed 2 9 16 23 30
Thu 3 10 17 24 31
Fri 4 11 18 25
Sat 5 12 19 26
Sun 6 13 20 27

Scanned with CamScanner

Java Assignment Operators

- Assignment Operators are used to assign values to variables.

- In the example below, we use the assignment operator (=) to assign the value 10 to a variable called x:

Example :-

```
int x = 10;
```

The addition assignment operator (+=) add a value to a variable:

Example :-

```
int x = 10;
x += 5;
```


Scanned with CamScanner

A list of all assignment Operator :-

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x+3
-=	x -= 3	x = x-3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x/3
%=	x %= 3	x = x%3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Scanned with CamScanner

Java Comparison Operators / Relational

Comparison Operators used to compare two value.

operator	Name	example
==	Equal to	x == y
! =	Not Equal to	x != y
>	Greater than	x > y
<	less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Scanned with CamScanner

Priority

Java Logical Operators

Logical operators are used to determine the logic between variables or values.

Operator	Name	Description	example
&&	logical and	Return true if both statement are true.	x<5 && x<10
||	logical or	Return true if one of the statement is true.	x<5 || x<4
!	logical Not	Reverse the result, return false if the result is true.	!(x<5 && x<10)

Scanned with CamScanner

Java Strings

Strings are used for storing text.

A String variable contains a collection of characters surrounded by double quotes:

Example:-

Create a variable of type String and asign it a value.

```
String greeting = "Hello";
```

Priority

String:- Length

A String in Java is actually an object, which contain methods that can perform certain operations on string. For example, the length of a string can be found with the length() method:

Example:-

```
String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println("The length of the text string is:" + txt.~~leng~~ length());
```

Output

The length of the txt String is: 26

Scanned with CamScanner

More String Methods

There are many string methods available, For example toUpperCase() and toLowerCase();

Example:-

```
String txt = "Hello World";
System.out.Println(txt.touppercase());
system.Out.Println(txt.toLowerCase());
```

Output

```
HELLO WORLD
hello world
```


Scanned with CamScanner

Priority

Finding a Character in a String

The indexOf() method return the index (the position) of the first occurence of a specified text in a String (including whitespace):

Example:-

```
String txt = "Please locate where 'locate' occurs!";
System.out.Println(txt.indexOf("locate"));
```

Output

7

Scanned with CamScanner

Priority

String Concatenation :

The + operator can be used between string to combine them. This is called concatenation :

Example :-

```
String FirstName = "John";
String lastName = "Doe";
System.out.Println(FirstName + " " + lastname);
```

Output

```
John Doe
```

Note

You can also use the concat() method to concatenate two string:

Scanned with CamScanner

Priority

Special Characters

Because String must be written within quotes, Java will misunderstand this string, and generate an error:

```
String txt = " we are the so-called "Vikings" from the north.";
```

- The solution to avoid this problem, is to use the backslash escape character.

- The backslash (\) escape charater ~~ter~~ turns special characters into string characters.

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Scanned with CamScanner

Example :-

```
String txt = "We are the so-called \"Vikings\" from the north." ;
String txt = "It\'s alright." ;
String txt = " The character \\ is called backslash." ;
```

Six other escape sequence are valid in Java.

Code	Result
\n	New line
\r	Carriage Return
\t	tab
\b	BackSpace
\f	FormFeed

Scanned with CamScanner

Priority

Adding Number and String

Warning :-

- Java uses the (+) operator for both addition and Concatenation.
- Number are added. String are Concatenated.

If you add two number, the result will be number :-

Example :-

```
int x = 10;
int y = 20;
int z = x + y;
```

Output

30

Scanned with CamScanner

- If you add two string, the result will be a string Concatenation:

Example:-

```
String x = "10";
String y = "20";
String z = x + y;
```

Output
10 20

- If you add a number and a string, the result will be a string Concatenation:

```
String x = "10";
int y = 20;
String z = x + y;
```

Output
10 20

Scanned with CamScanner

Priority

Java math

The Java Math Class has many methods that allows you to perform mathematical tasks on numbers.

- Math.max (x,y)

The Math.max (x,y) method can be used to find the highest value of x and y;

- Math.min (x,y)

The Math.min (x,y) method can be used to find the lowest value of x and y:

- Math.sqrt (x)

The Math.sqrt (x) method returns the square root of x:

- Math.abs (x)

The Math.abs (x) method returns the absolute (positive) value of x;

Scanned with CamScanner

Priority

Example

```
Public Class My class {
    Public Static Void main (String [] args){
    System. out. Println (Math. min (5, 10));
    System. out. Println (Math. max (5,10));
    System. out. Println (Math. sqrt (64));
    System. out. Println (Math. abs (-4.7));
    }
}
```

Output

```
5
10
8.0
4.7
```


Scanned with CamScanner

Priority

Java Booleans

Very often in Programming you will need a data type that can only have one of two values, like:

- Yes / No
- ON / OFF
- True / False

For this, Java has a boolean data type, which can take the value true or False.

Scanned with CamScanner

Priority

Boolean Value

A boolean type is declared with the boolean keyword an can only take the values true or false :

Example

```
boolean isJavaFun = true;
boolean isFishTasty = False;
System.out.Println(isJavaFun);
System.out.~~Printl~~ Println(isFishTasty);
```

Output

```
true
False
```

However, it is more common to return boolean values ~~For~~ from boolean expressions, for conditional testing.

Scanned with CamScanner

Boolean Expression

- A Boolean expression is a Java expression that returns a Boolean value: true or False
- you Can use a Comparison operator, Such as the greater than (>) operator to Find out if an expression (or a variable) is true:

Example

```
int x = 10;
int y = 9;
System.out.Println(x > y);//return true:-
```

- Use the equal to (==) operator to evaluate an expression.

Example

```
int x = 10;
System.out.Println(x == 10);
```

13 June Wednesday

Priority

Java If ... Else

Java Conditions and If Statements:-

Java supports the usual logical conditions from mathematics,

- Less than: a<b
- Less than or equal to: a<=b
- Greater than: a>b
- Greater than or equal to: a>=b
- Equal to: a==b
- Not Equal to: a!=b

You can use these conditions to perform different actions for different decisions.

Scanned with CamScanner

Priority

8. Java has the following Conditional Statement:

9

- use if to specify a block of block of code

10 to be executed, if a specified Condition is

11 true.

12

- use else to specify a block of code to be

13 executed, if the same Condition is False.

14

- use else if to specify a new Condition to

15 test if the first Condition False

16

17. use switch to specify many alternative

18 blocks of code to be executed.

Scanned with CamScanner

Priority

The if statement

Use the if statement to specify a block of Java Code to be executed if a Condition is true.

Syntax

```
if (condition) {
    //block of Code to be executed if the Condition is true.
}
```

Note:- That if is in lowercase letters. Uppercase letter (If or IF) will generate an error.

Scanned with CamScanner

In the example below, we test two values to find out if 20 is greater than 18. if the Condition is true, print some text:

Example

```
if (20 > 18) {
  System.out.Println("20 is greater than 18");
}
```

We can also test variables:-

Example

```
int x = 20;
int y = 18;
if (x>y) {
  System.out.Println("x is greater than y");
}
```


Scanned with CamScanner

Priority

The else statement

Use the else statement to specify a block of code to be executed if the condition is False.

Syntax:-

```
if (condition) {
    // block of code to executed if the condition is true.
} else {
    // block of code to be executed if the condition is false.
}
```

Example

```
int time = 20;
if (time < 180){
    System.out.Println("Good day.");
} else {
    System.out.Println("Good evening.");
}
// Output "Good Evening."
```


Scanned with CamScanner

The else if statement

Use the else if statement to specify a new condition if the first condition if false.

Syntax

```
if (condition1) {
    }
else if (condition2) {
    }
else {
    }
```

Example

```
int time = 22;
if (time < 10){
  System.out.Println("Good morning.");
}
else if (time < 20) {
  System.out.Println(" Good day.");
} else {
  System.out.Println("Good evening.");
}
```

// output - Good evening

Scanned with CamScanner

Priority

Short Hand If ---- Else

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statement.

Syntax :-

Variable = (condition) ? Expression True :
Expression False ;

Instead of writing :-

Scanned with CamScanner

Priority

Instead of writing

Example

```
int time = 20;
if (time < 18) {
  System.out.Println("Good day.");
}
  else {
System.out.Println("Good evening.");
```

You Can simply write:

~~int~~ Example

```
int time = 20;
String result = (time<18) ? "Goodday." :"Goodevening";
System.out.Println(result);
```


Scanned with CamScanner

Priority

Java Switch Statement

Use the Switch statement to select one of many code blocks to be executed.

Syntax

```
Switch (expression) {
    case x:
        // code block
    break ;
    Case y:
        // code block
    break ;
    default :
        // code block
}
```


Scanned with CamScanner

This is How it work

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- The break and default keywords are optional, and will be described later in the chapter.

Scanned with CamScanner

The example below use the weekday number to calculate the weekday name:

```
int day = 4;
switch (day) {
  case 1:
  System.out.Println("Monday");
  break;
  case 2:
  System.out.Println("Tuesday");
  break;
  case 3:
  System.out.Println("Wednesday");
  break;
  Case 4:
  System.out.Println("Thursday");
  break;
  case 5:
  System.out.Println("Friday");
  break;
```


Scanned with CamScanner

```
Case 6:
System.out.Println("Saturday");
break;
case 7:
System.out.Println("Sunday");
break;
}
// Output
Thursday
```


Scanned with CamScanner

The Default Keyword

The Defaut Keyword specifies Some Code to run if there is no case match:

Example :-

```
int day = 4;
Switch (day) {
    Case 6:
    System.Out.Println("Today is Saturday");
break;
    case 7:
    System.Out.Println("Today is Sunday");
break;
    defaut:
    System.out.Println("Looking Forward to
                        weekend");
}
```

// Outputs

"Looking Forward to weekend"

Scanned with CamScanner

LOOPS

- Loops Can execute a ~~blo~~ ~~block~~ block of code as long as a Specified Condition is reached

- Loops are handy because they Save time, reduce errors, and they Make Code more readable.

Java while loop

The while loop loops through a block of code as long as a specified Condition is true:

Syntad

```
while (condition) {
    // code block to be executed
}
```


Scanned with CamScanner

In the example below, the Code in the loop will run, over and over again, as long as a ~~Variable~~ Variable (i) is less then 5:

Example:

```
int i = 0;
while (i<5) {
    System.Out.Println(i);
    i++;
}
```

NOTE:= Do not Forget to increase the variable used in the Condition, Otherwise the loop will never end!

The Do/while Loop

The do/while loop is a variant of the while loop. This loop will execute the Code block once, before checking if the Condition is true, then it will repeat the loop as long as the Condition is true.

Syntax

```
do {
    // Code block to be executed
}
while (Condition);
```


Scanned with CamScanner

Priority

Example

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
int i = 0;
do {
  System.out.Println(i);
  i++;
}
while (i<5);
```


Scanned with CamScanner

For Loop

when you know exactly how many times you want to loop through a block of code, use the For loop instead of a while loop.

Syntax :-

```
For (Statement 1 ; Statement 2 ; Statement 3) {
    // code block to executed
}
```

Statement 1 :- is executed (one time) before the execution of the code block.

Statement 2 :- define the condition for executing the code block.

Statement 3 :- is executed (every time) after the code block has been executed.

Scanned with CamScanner

Priority

① the Example below will print the number 0 to 4 :-

Example

```
For (int i=0; i<5; i++) {
    System.out.Println (i);
}
```

② This example will only print the even value between 0 and 10 :

Example

```
For (int i=0; i<=10; i=i+2) {
    System.out.Println (i);
}
```


Scanned with CamScanner

Priority

For-Each Loop

There is also a "For-each" loop, which is used exclusively to loop through element in an array.

Syntax :-

```
For ( type VariableName : arrayName ) {
    // Code block to be executed
}
```

The following example outputs all elements in the Cars array, using a "for-each" loop :

Scanned with CamScanner

Priority

Example

```
String[] cars = { "Volvo", "BMW", "Ford", "Mazda" };
for (String i : cars) {
    System.out.Println(i);
}
```


Scanned with CamScanner

Priority

Java Break and Continue

Java Break

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

* This example jumps out of the loop when i is equal to 4:

Example

```
Public class myclass {
Public static void main(String[] args) {
For (int i = 0; i<10; i++) {
    if (i == 4) {
        break;
    }
    System.out.Println(i);
}
}
}
```

Output

```
0
1
2
3
4
```


Scanned with CamScanner

Priority

Java Continue

The Continue Statement breaks one iteration (in the loop), if a specified Condition occurs, and Continue with the next iteration in the loop.

* This example skips the value of 4 :

Example

```
Public class My class {
Public static Void main (String[] args) {
For (int i = 0; i < 10 ; i ++) {
  if (i == 4) {
    Continue ;
  }
  System.out. Println (i);
}
```


Scanned with CamScanner

Priority

Break and Continue in while loop

You can also use break and continue in while loops:-

Break Example:-

```
int i = 0;
while (i < 10) {
    System.out.Println(i);
    i++;
    if (i == 4) {
        break;
    }
}
```


Scanned with CamScanner

Priority

Continue Example

```
int i = 0;
while (i < 10) {
  if (i == 4). {
    i++ ;
    Continue ;
  }
System.out. println (i) ;
  i++ ;
}
```


Scanned with CamScanner

Priority

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets:-

```
String[] cars;
```

We have now declared a variable that holds an array of string. To insert value to it, we can use an array literal - place the value in a comma - separated list, inside curly bracs:

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

To create an array of integers, you write:

```
int[] myNum = {10, 20, 30, 40};
```


Scanned with CamScanner

Priority

Access the Elements of an Array:-

You access an array element by referring to the index number.

★ This statement accesses the value of the first element in cars:

Example

```
String[] cars = {"Volvo", "Bmw", "Ford"};
System.out.Println(cars[0]);
```

// Output

Volvo

Scanned with CamScanner

Change an Array Element:

To change the value of specific element, refer to the index number:

Example

```
Public class MyClass {
    Public Static Void main(String[] args) {
    String[] cars = {"volvo", "BMW", "Ford"};
    Cars[0] = "Opel";
    System.Out.Println(cars[0]);
    }
}
```

Output

```
Opel
```


Scanned with CamScanner

Priority

Array Length

To find out how many elements are array has, use the length Property:

Example

```
Public Class Myclass {
    Public Static Void Main (String [] args) {
    String [] cars = {"volvo", "BMW", "Ford"};
    System.out.Println(cars.length);
    }
}
```

Output

```
4
```


Scanned with CamScanner

Loop Through an Array

you Can loop through the array element with the For loop, and use the length Property to specify how many times the loop should run.

The following example output all element in the Cars array:-

Example:-

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

For (int i = 0; i < Cars.length; i++) {
    System.Out.Println(cars[i]);
}
```

Output

```
Volvo
BMW
Ford
Mazda
```


Scanned with CamScanner

Priority

Loop Through an Array with For-Each

There is also a "For-each" loop, which is used exclusively to loop through elements in arrays:

Syntax

```
For ( type variable : arrayname) {
    ...
}
```

The following example output all element in the Cars array, using a "for-each" loop:

Example

```
String [] cars = {"Volvo","BMW","Ford","Mazda"};
For (String i : cars) {
    System.Out.Println(i);
```

Output

```
Volvo
BMW
Ford
Mazda
```

August 2018

Wk	31	32	33	34	35
Mon		6	13	20	27
Tue		7	14	21	28
Wed	1	8	15	22	29
Thu	2	9	16	23	30
Fri	3	10	17	24	31
Sat	4	11	18	25	
Sun	5	12	19	26	

AUG SEP OCT NOV DEC

Scanned with CamScanner

The example above can be like this:

For each String element (called i - as in index) in cars, print out the value of i.

If you compare the ~~for~~ For loop and for-each loop, you will see that the for-each method is easier to write, it does not require a counter (using the length property), and it is more readable.

Scanned with CamScanner

Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

To create a two-dimensional array, add each array within its own set of curly braces.

Example :-

```
int[][] myNumbers = { { 1, 2, 3, 4 }, { 5, 6, 7 } };
```

myNumbers is now an array with two arrays as its elements.

To access the elements of the myNumber array, specify two indexes: one for the array, and one for the element inside that array. This example accesses the third element (2) in the second array (1) of myNumber:

Scanned with CamScanner

Priority

Example

```
int [] [] myNumber = { { 1, 2, 3, 4 }, { 5, 6, 7 } };
int x = myNumber [1] [2];
System.out.Println(x);
```

Output

7

We Can use a for loop inside another For loop to get the elements of a two-dimensional array (we still have to Point to the two indexes):

Scanned with CamScanner

Priority

Example

```
Public class MyClass {
  Public Static Void main (String [] args) {
    int [] [] myNumbers = {{ 1,2,3,4}, {5, 6,7}} ;
  For (int i = 0 ; i < MyNumber.length ; ++i) {
  For (int j = 0 ; j < myNumbers [i] . length ; ++j) {
  System.out. Println (myNumbers [i] [j] ) ;
    }
  }
}
}
```

Output

```
1
2
3
4
5
6
7
```


Scanned with CamScanner

Priority

JAVA METHODS

- A method is a block of Code which only runs when it is Called.
- you Can pass data, known as Parameters, into a method.
- Methods are used to Perform certain action, and they are also known as Functions.
- why use methods? To reduse code: define the Code once, and use it many times.

Scanned with CamScanner

Priority

Create a Method

A method must be declared within a class. It is defined with the name of the method, followed by Parentheses (). Java Provides some Pre-defined methods, Such as System.out.Println(), but you can also create your own methods to Perform Certain actions:

Example:-

Create a method inside MyClass:

```
Public Class MyClass {
    Static void MyMethod () {
        // Code to be executed
    }
}
```


Scanned with CamScanner

Example Examplained :-

- MyMethod () is the name of the method
- Static means that the method belongs to the MyClass class and not an object of the MyClass class. you will learn more about objects and how to access methods through object later in this tutorial.

- Void mean that this method does not have a return value. you will learn more about return value later in this Chapter.

Scanned with CamScanner

Call a Method

To Call a Method in Java, write the method's name. Followed by two Parentheses () and a Semicolon;

In the following example, MyMethod () method: is used to Print a text (the action), when it is Called:

Example:-

Inside Main, Call the myMethod () method:

```
Public Class MyClass {
Static Void myMethod () {
System. Out. Println ("I Just got executed!");
}
Public Static Void Main (String [] args) {
My Method ();
}
}
```

Output

I Just got executed!

August 2018

Wk	31	32	33	34	35
Mon		6	13	20	27
Tue		7	14	21	28
Wed	1	8	15	22	29
Thu	2	9	16	23	30
Fri	3	10	17	24	31
Sat	4	11	18	25	
Sun	5	12	19	26	

Scanned with CamScanner

A method Can also be called Multiple times:

Example :-

```
Public class My Class {
Static Void myMethod () {
System . out . Println ("I Just executed!");
}
Public Static Void Main (String [] args {
myMethod ();
myMethod ();
myMethod ();
}
}
```

Output

```
I Just executed!
I Just executed!
I Just executed!
```


Scanned with CamScanner

Java Method Parameter

Parameters and argument :-

Information Can be Passed to methods as ~~Para~~ Parameter. ~~are~~ act as variables inside the Method.

Parameter are specified after the method name, inside the Parentheses. You Can add as ~~Many~~ Many Parameters as you want, Just Separate them with a comma.

The Following example has a method that takes a ~~Str~~ String called fname as Parameter. When the Method ~~that~~ is ~~takes~~ a Called, we pass along a first name, which is used inside the Method to print the full name.

Scanned with CamScanner

Example

```
Public Class MyClass {
    Static Void myMethod (String Fname){
    System.out.Println(Fname + "Refsnes");
    }
Public Static Void main (String [] args) {
    myMethod("Liam");
    myMethod("Jenny")
    myMethod("Anja");
    }
}
```

Output

```
Liam    Refsness
Jenny   Refsness
Anja    Refsness
```

Note:- When a Parameter is passed to the method, it is called an argument. So from the example above: Fname is a parameter, while liam, Jenny and Anja are Argument.

Scanned with CamScanner

Priority

Multiple Parameters

Example :—

```
Public Class Myclass {
  Static Void myMethod(String Fname, int age) {
    System.Out.Println(Fname + " is" + age);
  }
Public Static void Main(String[] args) {
    myMethod ("Liam", 5);
    myMethod ("Jenny", 8);
    myMethod ("Anja", 31);
  }
}
```

Output

```
Liam is5
Jenny is 8
Anja is31
```


Scanned with CamScanner

Priority

Return Values

The Void keyword, used in the examples above, indicates that the Method Should not return a Value. if you want the method to return a Value, you Con the ~~r~~ return keyword the Method:

Example :-

```
Public Class MyClass {
Static int myMethod (int x) {
return 5 + x ;
} }
public Static Void main (String [] args) {
System. Out. Println (myMethod (3)) ;
}
}
```

Output

8

Note: This example return the Sum of a method's Two Parameters :-

Scanned with CamScanner

Priority

This example return the sum of a method's Two Parameters:—

```
Public Class MyClass {
Static int myMethod (int x, int y) {
  return x + y ;
}
Public Static void main ( String [] args) {
System.out. Println (myMethod (5, 3)) ;
}
}
```

Output

8

Scanned with CamScanner

Priority You Can also store the result in a variable (recommended, as it is easier to read and maintain):-

Example

```
Public Class MyClass {
Static int myMethod ( int x , int y ) {
    return x + y ;
}

Public Static void main ( String[] args ) {
int z = myMethod ( 5 , 3 ) ;
System . out . Println (z) ;
}
}
```

OutPut

8

Scanned with CamScanner

Priority

A Method with if .... Else

It is Common to use if ...else statements inside methods:

Example:-

```
Public class MyClass {
// Create a checkAge() method with an integer variable called age
Static Void checkAge ( int age ) {
// If age is less than 18, Print " Access denied"
if ( age <18 ) {
System.Out.Println (" Access denied") ;
}
else {
System.out.Println (" Access granted") ;
}
}
Public static Void main ( String [] args )
{
checkAge (20) ;
}
}
```

Output

Access granted

Scanned with CamScanner

Priority

Method Overloading

With method overloading, Multiple Methods Can have the Same name with different Parameters:-

Example

```
int myMethod ( int x )
Float myMethod (Float x)
double myMethod ( double x , double y )
```


Scanned with CamScanner

Priority Consider the following example, which have two Methods that add numbers of different type:

Example

```
static int PlusMethodInt (int x, int y) {
  return x+y;
}
static double PlusMethodDouble (double x, double y)
  {
    return x + y;
  }
Public static void Main (String[] args) {
int myNum1 = PlusMethodInt (8 , 5);
double myNum2 = PlusMethodDouble (4.3 , 6.26);
System.out.Println ("int :" + myNum1);
System.out.Println ("double :" + myNum2);
}
```

Insted of defining two methods that should do the same thing, it is better to overload one.

Scanned with CamScanner

Priority In the example below, we overload the PlusMethod method to work for both int and double.

Example

```
Public class myClass {
Static int plusMethod (int x, int y) {
  return x + y;
}
Static double plusMethod (double x, double y) {
  return x + y;
}
Public static void main (String[] args) {
int myNum1 = plusMethod (8,5);
double myNum2 = plusMethod (4.3, 6.26);

System.out.Println("int:" + myNum1);
System.out.Println("double:" + myNum2);
}
}
```

NOTE:- Multiple methods can have the same name as long as the number and/or type of parameter are different.

Scanned with CamScanner

Priority

JAVA CLASSES

WHAT IS OOP ?

OOP stand for Object-Oriented programming.

procedural Programming is about writing procedures or methods that Perform operations on the data, while object-oriented programming is about Creating object that Contain both data and methods.

Advantage of OOP

- OOP is Faster and easier to execute
- OOP provides a clear structure for the program.
- OOP helps to keep Java Code DRY "Don't Repeat yourself", and makes the Code easier to maintain, modify and Debug.
- OOP make it Possible to create full reusable application with less code and shorter development time

Scanned with CamScanner

Priority

JAVA:- What is Classes and Objects?

- Classes and objects are the two main aspects of object-oriented programming.

- Look at the following illustration to see the difference between class and objects:-

Class	Object
Fruit	Apple, Banana, Mango

- So, a class is a template for object, and an object is an instance of a class.
- when the individual object are created, they inherit all the variable and methods from the class.
- You will learn much more about

Scanned with CamScanner

Priority

Java Classes and Object

★ Create a Class

To create a class, use the keyword Class:

MyClass.Java

Create a class named "MyClass" with a variable x:

```
Public Class "MyClass" {
int x = 5;
}
```

Remember from the Java Syntax chapter that a Class should always start with an uppercase first letter, and that the name of the Java File should match the class name.

Scanned with CamScanner

Priority Create an object

TO Create an object of MyClass, specify the class name, followed by the Object name and use the keyword new :

Example :-

```
Public class MyClass {
  int X = 5;
Public static void main (String[] args) {
MyClass myObj = new MyClass();
System.out.Println (myobj.X);
}
}
```

Output

5

Scanned with CamScanner

Multiple Object

Example:- Create two object of one class

```
Public class Myclass {
    int x = 5;
Public static void main (String [] args) {
  MyClass myObj1 = new myclass ();
  myClass myObj2 = new myclass ();
  System.Out.println (myObj1.x);
  System.Out.println (myObj2.x);
  }
}
```

Output

```
5
5
```


Scanned with CamScanner

Priority

Using Multiple Classes:-

You can also create an object of a class and access it in another class. This is often used for better organization of classes (one class has all the attributes and methods, while the other class holds the ~~main()~~ main() method (code to be executed)).

Remember that the name of the Java file should match the class name. In this example, we have created two files in the same directory / folder:

- MyClass.Java
- OtherClass.Java

Scanned with CamScanner

Example

MyClass.Java :-

```
Public class MyClass {
    int x = 5;
}

Class OtherClass {
Public static void main(String [] args) {
  MyClass myObj = new Myclass ();
  System.Out.Println (myObj.x);
}
}
```

Output

5

Priority

JAVA CLASS ATTRIBUTES

In the Previous Chapter, we used the term "Variable" for x in the example (as shown below). It is actually an attribute of the class. or you could say that class attributes are variables within a class:-

Example:-

Create a class called "MyClass" with two attribute: x and y:

```
Public class MyClass {
int x = 5;
int y = 3;
```

Another term for class attribute is Fields.

Scanned with CamScanner

Priority

Java Class ~~Accessing~~ Attributes

In the previous chapter, we used the term "variable" for x in the example (as shown below). It is actually an attribute of the class. Or you could say that class attributes are variable within a class:

Example :-

Create a class called "MyClass" with two attribute x and y :

```
Public Class MyClass {
int x = 5 ;
int y = 3 ;
}
```


Scanned with CamScanner

Priority

Accessing Attributes

You Can access attributes by creating an object of the class, and by using the dot Syntax (.):

The Following example will create an object of the MyClass Class, with the name myObj. We Use the x attribute on the the Object to print its Value:

Example:- Create an object Called "myobj" and Print the Value of x:

```
Public class MyClass {
  int x = 5;
Public static void main (String[] args) {
  MyClass myObj = new MyClass ();
  System.out.Println (myObj.x);
  }
}
```


Scanned with CamScanner

Priority

Modify Attributes

Example

Set the value of x to 40:

```
Public class ~~My Class~~ MyClass {
    int x;
    Public Static Void Main(String [] args) {
    MyClass ~~MyObj~~ myObj = new MyClass ();
    myObj.x = 40;
    System.out.Println(myObj.x);
    }
}
```

Overriding existing values:-

Example

```
Public class MyClass {
    int x = 10;
    Public Static Void Main (String [] args) {
    MyClass myObj = new MyClass ();
    myObj.x = 25;
    System.out.Println(myObj.x);
    }
}
```


Scanned with CamScanner

If you don't want the ability to overriding existing values, declare the attribute as Final:

Example

```
Public Class MyClass {
Final int x = 10;
Public Static Void main (String[] args) {
MyClass myObj = new MyClass ();
myObj.x = 25; // will generate an error: Cannot assign a value to a Final Variable.
System.out.Println (myObj.x);
}
}
```

☆ The Final keyword is useful when you want you want a variable to always store the same value, like PI (3.14159...).

☆ The Final keyword is called a "Modifier".

Scanned with CamScanner

Multiple Object

If you Create Multiple Objects of one class, you Can change the attribute values in one object, without affecting the attribute values in the other:

Example:- Chang the Value of x to 25 in myObj2, and leave x in ~~myObj~~ MyObj1 unchanged:-

```
Public Class MyClass {
    int X = 5;
Public Static void Main (String [] args) {
MyClass myObj1 = new MyClass ();
Myclass myObj2 = new MyClass ();
    MyObj2.X = 25;
System.Out.Println (MyObj1.X);
System.out.Println (myObj2.X);
}
}
```

Output
5
25

Multiple Attribute

You Can specify as many attributes as you want:

Example:-

```
Public class Person {
String Fname = " John";
String Lname = "Doe";
int age = 24;
Public Static Void main(String [] args) {
    Person MyObj = new Person ();
    System.Out. Println("Name:" +
            MyObj. Fname + "  " + MyObj. Lname);
    System.Out. Println ("Age:" + MyObj.age);
  }
}
```

Output

Name: John Doe

Age : 24

Scanned with CamScanner

Priority

Java Class Methods

You learned from the ~~Java~~ Java Methods chapter that methods are declared within a class, and that they are used to perform certain actions:-

Example:-

Create a method named myMethod() in MyClass:

```
public class MyClass {
  Static void myMethod () {
  System.out.Println ("Hello World");
  }
}
```


Scanned with CamScanner

myMethod() Prints a text (the action), when it is Called. To call a method, write the Method's name Followed by two parenthess () and a Semicolon ;

Example (Inside main, Call myMethod();

```
Public class MyMethod {
  Static void myMethod (){
    System.out. ~~Println~~ Println ("Hello World!");
  }
Public static void main (String [] args) {
    myMethod ();
  }
}
```

Output

Hello World!

Scanned with CamScanner

Priority

Static Vs. Non-Static

- You will often see Java Program that have either Static or Public attributes and Methods.
- In the ~~example~~ example above, we created a Static Method, which means that it can be accessed without creating an object of the Class, unlike Public, which can only be accessed by object.

Example

An example to demonstrate the differences between Static and Public method:-

```
Public Class MyClass {
    // Static method
    Static void myStaticMethod () {
        System.out.Println ("Static methods can be
        Called without creating object")
    }
```


Scanned with CamScanner

```
// Public Method
Public void myPublicMethod () {
System.out.Println(" Public methods must be called by creating object");
}
// Main method
Public Static Void Main (String[] args ) {
My StaticMethod (); // Call the Static Method
// myPublicMethod (); This would compile an error.
MyClass myObj = new MyClass (); // create an object of myClass.
myObj. myPublicMethod (); // Call the Public method on the object
}
}
```

Output

- Static Methods can be called without creating objects
- Public methods must be called by creating objects.

Scanned with CamScanner

Priority

Access Method With an Object

Create a Car object named myCar. Call the FullThrottle () and Speed () Method on the ~~Car~~ myCar object, and run the Program:-

```
// Create a Car class
Public Class Car {
// Create a FullThrottle () method
Public Void FullThrottle () {
  System.Out.Println("The Car is going as fast as it Can!");
}
// Create a Speed() Method and add a parameter
Public Void Speed (int MaxSpeed) {
System.out.Println("Max Speed is: " + MaxSpeed);
}
```


Scanned with CamScanner

```
// Inside main, Call the methods on the MyCar object.
Public Static void Main (String [] args) {
Car MyCar = new Car (); //Create a MyCar object.
MyCar.FullThrottle (); // Call the FullThrottle () Method
MyCar.Speed (200); // Call the speed () method
}
}
```

Output

```
The Car is going as fast as it Can!
Max Speed is: 200
```


Scanned with CamScanner

Priority

Example Explained

1) We created a custom Car class with the class keyword.

2) We created the FullThrottle and Speed() Method in the Car class.

3) The FullThrottle() method and the Speed() method will print out some text, when they are called.

4) The Speed() method accepts an int parameter called maxSpeed - we will use this in 8)

5) In order to use the Car class and its methods, we need to create an object of the Car class.

6) Then, go to the Main() Method, which you know by now is a built-in Java method that run your program (any code inside main is executed).

7) By using the new keyword we created a Car object with the name myCar.

Scanned with CamScanner

87 Then, we call the FullThrottle() and Speed() Method on the myCar object, and run the program using the name of the object (myCar), followed by a dot (.), followed by the name of the method (FullThrottle(); and Speed(200);). Notice that we add an int parameter of 200 inside the Speed() method.

Remember that...

- The dot (.) is used to access the object's attributes and method.
- To call a method in Java, write the method name, followed by a set of parentheses (), followed by a semicolon (;).

A class must have a matching filename (Car and Car.Java).

Scanned with CamScanner

Priority

USING MULTIPLE CLASSES

Like we specified in the Classes Chapter, it is a good practice to create an object of a class and access it in another class.

Remember that name of the Java file should match the class name. In this example, we have created two file in the same directory:

- Car.Java
- OtherCar.Java

Car.Java

```
Public Class Car {
  Public Void FullThrottle () {
  System.Out.Println("The Car is going as fast as it Can!");
  }
  Public Void Speed (Int MaxSpeed) {
  System.Out.Println("Max Speed is:" + maxSpeed);
  }
}
```


Scanned with CamScanner

Other Class. Java

```
Class OtherClass {
Public static void main(String[] args) {
Car myCar = new Car (); // create a myCar object
myCar. FullThrottle (); // call the FullThrottle() method
myCar. Speed (200); // call the Speed() method
}
}
```

☆ When both Files have been Compiled :-

C:\users\ your Name > Javac Car. Java

C:\users\your Name > Javac OtherClass. Java

☆ Run the Other Class. Java File :

C:\users\your Name > Java OtherClass

Output

The Car is going as fast as it can!
Max Speed is : 200

September 2018
Wk 35 36 37 38 39
Mon 3 10 17 24
Tue 4 11 18 25
Wed 5 12 19 26
Thu 6 13 20 27
Fri 7 14 21 28
Sat 1 8 15 22 29
Sun 2 9 16 23 30

Scanned with CamScanner

JAVA CONSTRUCTORS

A Constructor in Java is special Method that is used to initialize object. The constructor is called when an object of the class is created. It can be used to set initial values for object attributes:

Example

Create a Constructor:-

```
// Create a MyClass class
Public Class MyClass {
    int X; // Create a class attribute.

// Create a class Constructor for the MyClass Class
Public MyClass () {
    X = 5; // Set the initial value for the class attribute X.
}

Public Static Void Main (String [] args) {
    MyClass myObj = new MyClass ();
    // create an object of class Myclass (This will Call the Constructor)
    System.out.Println(myObj.X);
}
}
```


Scanned with CamScanner

Constructor Parameters

Constructors Can also take Parameters, which is used to initialize attributes.

The following example adds an int Y Parameters to the Constructor. Inside the Constructor we set X to Y (X = Y). when we ~~can~~ Call the Constructor. we Pass a Parameter to the Constructor (5), which will set the Value of X to 5:

Example

```
Public class MyClass {
int x;
Public MyClass(int y) {
    x = y;
}
Public Static void main(String[] args) {
MyClass myObj = new MyClass(5);
System.out.Println(myObj.x);
}
}
```

Output

5

September 2018
Wk 35 36 37 38 39
Mon 3 10 17 24
Tue 4 11 18 25
Wed 5 12 19 26
Thu 6 13 20 27
Fri 7 14 21 28
Sat 1 8 15 22 29
Sun 2 9 16 23 30
SEP OCT NOV DEC
Scanned with CamScanner

Priority

You Can have as Many Parameters as you want.

Example

```
Public class Car {
int modelYear;
String ModelName;
Public Car (int year, String name) {
    modelYear = year;
    modelYear = name;
}
Public Static Void main(String[] args) {
Car myCar = new Car (1969, "mustang");
System.out.Println (myCar.ModelYear +
                    " " + myCar.modelName);
}
}
```

Output

1969 Mustang

Scanned with CamScanner

Java Modifiers

```
Public Class MyGlass
```

The Public keyword is an access modifier, meaning that it is used to set the access level for classes, attribute, Methods and Constructors.

We divide modifiers into two groups:

- Access Modifiers - Controls the Access level
- Non-Access Modifiers - Do not Control access level, but Provides other functionality.

Access Modifiers:-

For classes, you can either Public or default:-

Modifier	Description
Public	The class is accessible by any other class.
Default	The class is only accessible by class in the same Package. This is used when you don't specify a modifier. you will learn more about packages in the Package chapter.

September 2018
Wk 35 36 37 38 39
Mon 3 10 17 24
Tue 4 11 18 25
Wed 5 12 19 26
Thu 6 13 20 27
Fri 7 14 21 28
Sat 1 8 15 22 29
Sun 2 9 16 23 30

Scanned with CamScanner

Priority

For attributes, methods and Constructors, you can use the one of the following.

Modifier	Description
Public	The code is accessible for all classes
Private	The code is only accessible within the declared class.
Protected	The code is accessible in the same package and subclasses. you will learn more about subclasses and superclass in the Inheritance chapter.
default.	The code is only accessible in the same package. This is used when you don't specify a modifier. you will learn more about package in the packages chapter.

Scanned with CamScanner

Non-Access Modifiers

Modifier	Description
final	The class cannot be inherited by other class (You will learn more about inheritance in the Inheritance chapter)
abstract	The class cannot be used to create object (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters)

Scanned with CamScanner

For attributes and methods, you can use the one of the following:

Modifier	Description
Final	Attribute and methods cannot be overridden / modified.
Static	Attribute and methods belongs to the class, rather than an object.
abstract	Can only be used in an abstract class, and can only be used on methods; The method does not have a body, for example abstract void run(); The body is provided by the subclass (inherited from). ~~you will learn more about it~~
transient	Attributes and methods are skipped when serializing the object containing them
Synchronized	Methods can only be accessed by one thread at a time.
Volatile	The value of an attribute is not cached thread-locally, and is always read from the "main memory"

August 2018
Wk 31 32 33 34 35
Mon 6 13 20 27
Tue 7 14 21 28
Wed 1 8 15 22 29
Thu 2 9 16 23 30
Fri 3 10 17 24 31
Sat 4 11 18 25
Sun 5 12 19 26

Scanned with CamScanner

Final

If you don't want the ability to override existing attribute values, declare attributes as Final :-

Example

```
public class MyClass {
final int x = 10 ;
final double PI = 3.14 ;
Public Static Void Main(String [] args) {
MyClass myObj = new MyClass();
myObj. x =50; // will generate an error :
        Cannot assign a value to a Final
        variable.
myObj. PI =25 ; // will generate an error :
        Cannot assign a value to a final
        variable.
System.out. println (myObj.x);
}
}
```


Scanned with CamScanner

Static

A Static Method means that it can be accessed without Creating an object of the class, unlike Public :

Example

An example to demostrate the difference between Static and Public method:

```
Public class MyClass {
    // Static method
Static void myStaticMethod() {
System.out.Println("Static Method can be called
        without creating object");
  }
  // Public Method
Public void myPublicMethod() {
System.out.Println("Public Method Must be called
        by creating objects");
  }
  // Main Method
Public static void Main(String[] args) {
  myStatic Method(); // call the static method
  MyClass myObj = new MyClass();
  myobj.MyPublicMethod();
  }
}
```


Scanned with CamScanner

Abstract

An abstract method belongs to an abstract class, And it does not have a body. The body is provided by the sub-class:

Example

```
// Code From Filename : Person.Java
// abstract Class
~~abstract Class Person {~~
abstract ClassPerson {
  Public String Fname = "John";
  Public int age = 24;
  Public abstract Void Study();// abstract method
}
// SubClass (inherit from Person)
Class Student extends Person {
  Public int graduationYear = 2018;
  Public Void Study() { // the body of the abstract method is provided here
    System.out.PrintLn("Studing all day long");
  }
}
```

```
// End Code From File name : Person.Java
// Code From filename : MyClass.Java
  Class MyClass {
  Public Static Void Main(String[] args) {
// Create an object of the Student class (which inherits attributes and method from Person)
  Student MyObj = new Student ();
  System.out.Println (" Name: " + MyObj.fname);
  System.out.Println ( "Age: " + MyObj.age);
  System.out.Println ( " Graduation Year:" + MyObj.graduationYear);
  MyObj.Study(); // Call abstract method
  }
}
```


Scanned with CamScanner

JAVA Encapsulation

Encapsulation:-

The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- Declare class variables/attribute as Private
- Provide public ~~ge~~ get and set methods to access and update the value of a private variable.

Get and Set

- You learned from the previous chapter that Private variables can only be accessed within the same class (an ~~output~~ outside class has no access to it). However, it is possible to access them ~~is~~ if we provide public get and set methods.

- The get method returns the ~~a~~ variable value, and the Set method sets the value.

- Syntax for both is that they start with either ~~get~~ get or ~~set~~ Set, followed by the name of the variable, with the first letter in uppercase:

Scanned with CamScanner

Example

```
Public class Person {
  Private String name ; // private = restricted access.
  // Getter
  Public String getName () {
    return name;
  }
  // Setter
  Public void setName (String newName) {
    this.name = newName;
  }
}
```

Example Explained

* The get Method return the value of the variable name.
* The set Method takes a Parameter (newName) and assigns it to the name variable.

* The This keyword is used to refer to the current object.

* However, as the name variable is declared as Private, we cannot access it from outside the class:

October 2018
Wk 40 41 42 43 44
Mon 1 8 15 22 29
Tue 2 9 16 23 30
Wed 3 10 17 24 31
Thu 4 11 18 25
Fri 5 12 19 26
Sat 6 13 20 27
Sun 7 14 21 28
OCT NOV DEC
Scanned with CamScanner

Example :-

```
Public class MyClass {
Public Static Void main(String[] args) {
Person myObj = new Person();
myObj.name = "John"; // error
System.out.Println(myObj.name); //error
}
}
```

If the variable was declared as Public, we would expect the following output:-

John

However, as we try to access a Private variable, we get an error.

MyClass.Java:4: error: name has private access in Person
myObj.name = "John";
^

~~Same~~ Same in like 5:

2 error

Scanned with CamScanner

Insted, ~~my~~ we use the getName() and Setname() methods to access and update the Variable.

Example:-

```
Public Class MyClass {
Public Static void main(String [] args) {
  Person MyObj = new Person();
  MyObj. SetName("John");
  // Set the value of the name variable to the "John"
  System.out. Println(MyObj. getName());
}
}
```

Output

John

Scanned with CamScanner

Priority

★ Why Encapsulation ?

- Better Control of Classes attribute and methods.

- Class attribute Can be made read-only (if you only use the get method) or write-only (if you only use the ~~set~~ Set method)

- Flexible: The programmer Can change one Part of the code without affecting other Parts

- Increased security of data

Scanned with CamScanner

Priority

☆ JAVA PACKAGES

Java Packages and API

A Package in Java is used to group related Classes. Think of it as a Folder in a file directory. We use Packages to avoid name Conflicts, and to write a better maintainable Code. Packages are divided into two categories.

- Built-in-Packages (Packages From the Java API)
- User-defined Packages (Create your own Packages)

Built-in-Packages

The Java API is a library prewritten Classes, that are free to use, included in the Java Development Environment.

The library Contains Components for managing input, database Programming, and much much more.

The library is divided into **Packages** and Classes. Meaning you Can either import a Single Class (along with its method and attribute), or a whole Package that Contain all the classes that belong to the specified Package.

To use a Class or a Package from the library, you need to use the import keyword:

Scanned with CamScanner

Syntax

```
import Package.name.class; // import a single class
import Package.name.*;     // Import the whole package.
```

Import a Class

If you find a class you want to use, for example, the Scanner class, which is used to get user input, write the following code:

Example

```
Import Java.util.Scanner;
```

In the example above, Java.util is a package, while Scanner is a class of the Java.util Package.

To use the Scanner class, create an Object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read a Complete line:

Scanned with CamScanner

EXAMPLE

using the Scanner class to get user input:

```
Import Java.util.Scanner;
Class MyClass {
Public Static void main (String[] args){
Scanner myObj = new Scanner (System.in);
System.out.Println("Enter username");
String UserName = myObj.nextline();
System.out.Println("Username is:"+ UserName);
}
}
```


Scanned with CamScanner

Priority

Import a Package

There are many Packages to choose from.

In the previous example, we used the Scanner Class from the Java.util Package. This Package also Contains date and time Facilities, random-number generater and other utility Classes.

To Import a whole Package, end the Sentence with an asterisk Sign (*). The Following example will import ALL the classes in the Java.util Package.

~~Example~~

Scanned with CamScanner

Example:

```
import java.util.*; //import the java.util Package.
Class MyClass{
Public Static void main(String[] args){
  Scanner myObj = new Scanner(System.in);
  String UserName;
  // Enter Username and Press enter
  System.out.Println("Enter username");
  UserName = MyObj.nextLine();
System.out.Println("username is: " + UserName);
  }
}
```

Output

```
Enter Username
Deepak
User name is Deepak
```


Scanned with CamScanner

User-defined Packages

To Create your own Package, you need to ~~tel~~ that Java uses a file system directory to store them. Just like folders on your Computer:

<u>Example</u>

```
└ root
  └ MyPack
    └ MyPackageClass.java
```

To Create a Package, use the Package keyword.

Scanned with CamScanner

MyPackageClass.Java

```
Package MyPack;
Class MyPackageClass {
  Public static Void Main(String[] args) {
  System.out.Println(" This is my Package!");
  }
}
```

Save the File as MyPackageClass.Java, and Compile it.

```
C:\users\yourName > javac MyPackageClass.Java
```

~~Then~~ Then Compile the Package:

```
C:\users\Your Name > Javac -d . MyPackageClass.Java
```

When we Compiled the Package in the example above, a new folder was created, called "MyPack".

To run the MyPackageClass.Java File, write the Following.

```
C:\user\yourName > Java MyPack.
                   MyPackageClass
```

OutPut

```
This is My Package!
```


Scanned with CamScanner

JAVA INHERITANCE

Java Inheritance (SubClass and SuperClass)

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- Subclass (child) - the class that inherits from another class
- Superclass (Parent) - the class being inherited from

To inherit from a class, use the extends keyword.

In the example below, the Car class (SubClass) inherits the attributes are method from the vehicle class (superclass):

Scanned with CamScanner

Priority

Example

```
Class Vehicle {
  Protected String brand = "Ford";
  Public void honk() {
    System.out.Println("Tuut, tuut!");
  }
}

Class Car extends Vehicle {
  Private String modelName = "Mustang";
  Public static void main(String [] args) {
    Car myfastCar = new Car();
    myfastCar.honk();
    System.out.Println(myfastCar.brand + " " +
                       myfastCar.modelName);
  }
}
```

Output

```
Tuut, tuut!
Ford Mustang
```


Scanned with CamScanner

The Final Keyword

If you don't want other classes to inherit from a class, use the final keyword:

If you try to access a ~~final~~ final class, Java will generate an error.

```
final class vehicle {
    Protected String brand = "Ford";
    Public void honk () {
        System.out. ~~Println~~ Println ("Toot, toot!");
    }
}
Class Car extends Vehicle {
    Private String ModelName = "Mustang";
    Public Static void main (String [] args) {
        Car myfastCar = new Car ();
        myfastCar. honk ();
        System.out. Println (myfastCar. brand + " " +
                myfastCar. modelName);
    }
}
```

output

error

Scanned with CamScanner

Priority

Java Polymorphism

- Polymorphism means "many forms", and it occurs when we have many classes that related to each other by inheritance.

- Like we specified in the previous chapter: Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

- For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the Pig oinks, and the Cat meows, etc):

Scanned with CamScanner

Example

```
Class Animal {
  Public Void animalSound () {
  System. out. Println (" The animal Makes a sound");
  }
}
Class Pig extends Animal {
Public Void animalSound () {
    System.out. Println ("The Pig Says: wee wee");
}
}
Class Dog extends Animal {
  Public Void animalSound () {
  System.out. Println (" The dog Says: bow bow") ;
  }
}
```

Note

- Remember from the Inheritance chapter that we use the ~~extends~~ extends keyword to inherit from a class.

Scanned with CamScanner

Priority

Now we can create ~~Pig~~ Pig and Dog Object and call the animalSound() method on both of them:

Examples

```
Class Animal {
Public Void animalSound () {
System.out.Println("The animal makes a Sound");
}
}
Class Pig extends Animal {
Public Void animalSound () {
System.out.Println("The Pig says: wee wee");
}
}
Class Dog extends Animal {
Public Void animalSound () {
System.out.Println("The dog Says: bow wow");
}
}
```


Scanned with CamScanner

```
Class MyMainClass {
Public ~~void~~ ~~Animal~~ Static void main(String[] args){
Animal myAnimal = new Animal();
Animal myPig = new Pig();
Animal myDog = new Dog();

myAnimal.animalSound();
my Pig.animalSound();
my Dog.animalSound();
}
}
```

Output

The animal makes a sound
The pig says: wee wee
The dog says: bow wow

Why And When To use "Inheritance" and "Polymorphism"?
It is useful for code reusability: reuse attributes and method of an existing class when you create a new class.

JAVA INNER CLASS

In Java, it is also possible to nest classes (a class within a class). The purpose of nested class is to group classes that belong together, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class.

Example

```
Class OuterClass {
  int X = 10;
Class InnerClass {
  int Y = 5;
}
}
Public Class MyMainClass {
Public Static void main(String[] args) {
OuterClass myOuter = new OuterClass();
Outer Class . InnerClass myInner = myOuter.
                  new InnerClass();
System.out.Println(myInner.Y + myOuter.X);
}
}
```

Output

15 (5+10)

October 2018
Wk 40 41 42 43 44
Mon 1 8 15 22 29
Tue 2 9 16 23 30
Wed 3 10 17 24 31
Thu 4 11 18 25
Fri 5 12 19 26
Sat 6 13 20 27
Sun 7 14 21 28
OCT NOV DEC

Private Inner Class

Unlike a "regular" class, an inner class can be Private or Protected. If you don't want outside objects to access the inner class, declare the class as Private:

Example

```
Class OuterClass {
  int x = 10;
  Private class InnerClass {
    int y = 5;
  }
}
Public class MyMainClass {
Public Static Void main(String[] args) {
OuterClass myOuter = new OuterClass();
OuterClass.InnerClass myInner = myOuter.
                        new InnerClass();
System.out.Println(myInner.y + myouter.x);
}
}
```

Note → If you try to access a private inner class from an outside class and error occur.

Static Inner Class

An inner class also be static, which means that you can access it without creating an object of the outer class:

Example

```
Class OuterClass {
  int x = 10;
  Static class InnerClass {
    int y = 5;
  }
}
Public class MyMainClass {
Public Static Void main(String[] args) {
OuterClass.InnerClass myInner =
    new OuterClass.InnerClass();
System.out.Println(myInner.y);
  }
}
```

Outputs

5

Access Outer Class from Inner Class

One advantage of inner classes, is that they can access attributes and methods of the outer class:

Example

```
Class OuterClass {
  int x = 10;
Class InnerClass {
Public int myInnerMethod () {
  return x;
  }
  }
}
Public Class MyMainClass {
  Public Static void Main(String args[]) {
OuterClass myOuter = new OuterClass ();
OuterClass.InnerClass myInner =
        myOuter. new InnerClass ();
System.Out.Println(myInner.myInnerMethod ());
}
}
```

Output

10

Scanned with CamScanner

Priority

Java Abstract Classes And Methods

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achived with either abstract classes or interfaces (which you will learn more in the next chapter).

The abstract keyword is a non-access modifier, used for classes and methods:

- Abstract Class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

- Abstract Method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

Scanned with CamScanner

An abstract class can have both abstract and regular methods:

```
abstract class Animal {
Public abstract Void animalSound ();
Public Void Sleep () {
System.out.Println("Zzz");
}
}
```

From the example above, it is not possible to create an object of the Animal Class:

```
Animal MyObj = new Animal () ; // will generate an error.
```

To access the abstract Class, it must be inherited from another Class. lets convert the animal Class we used in the polymorphism Chapter to an abstract class:

Scanned with CamScanner

Remember from Inheritance chapter that we use the extends keyword to inherit from class.

Example

```
// Abstract Class

abstract class Animal{

// Abstract method (does not have a body)
public abstract void animalSound();

// Regular Method
public void sleep(){

System.out.Println("Zzz");
}
}

// Subclass (inherit from Animal)

Class Pig extends Animal{

public void animalSound(){

// The body of animalSound() is provided here

System.out.Println("The Pig Says: weewee");
}
}
```


Scanned with CamScanner

```
Class MyMainClass {
Public static Void main(String[] args) {
Pig myPig = new Pig(); // Create a Pig Object
myPig . animalSound ();
myPig . Sleep ();
  }
}
```

Output

```
The Pig Says: Wee Wee
Z22
```

JAVA INTERFACE

Another way to achive abstraction in Java, is with interfaces.

An interface is a Completely "abstract class" that is used to group related methods with empty bodies:

Example

```
// interface
interface Animal {
Public Void AnimalSound ();// interface method &
                              (does not have a body)

Public Void run ();// interface method
                              (does not have a body)
}
```

To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (insted of extends). The body of the interface method is provided by the "implement" class:

Scanned with CamScanner

Example

```
// Interface
interface Animal {

Public Void animalSound ();// interface method
                              (does not have a body)
Public Void Sleep ();// interface method
                              (does not have a body)
}
// Pig "implements" the Animal interface
Class Pig implements Animal {

Public Void animalSound {

// The body of animal Sound () is provided here.

System.out.Println("The Pig Says: wee wee");
}
Public Void Sleep () {

// The body of Sleep () is Provided here

System.out.println("222");
}
}
```


Scanned with CamScanner

```
Class MyMainClass {
Public static Void main (String[] args) {
Pig myPig = new Pig (); //Create a Pig Object.
myPig . animalSound ();
myPig . sleep ();
  }
}
```

Output

The Pig Says: wee wee
Zzz

Scanned with CamScanner

Multiple Interfaces

To implement multiple interfaces, separate them with a comma:

Example

```
interface FirstInterface {
 Public void myMethod ();// interface method
}
interface SecondInterface {
 Public void myOtherMethod ();// interface method
}
Class DemoClass implements FirstInterface, SecondInterface {
Public void myMethod () {
  System.out.Println("Some text..");
}
Public void myOtherMethod () {
  System.Out.Println("Some other text...");
 }
}
```

```
Class MyMainClass {
Public Static Void Main (String [] args) {
DemoClass myObj = new DemoClass ();
myObj. myMethod ();
myObj. myOtherMethod ();
}
}
```

Output

Some text....

Some other text...

Scanned with CamScanner