→ class Simple {

```
    public static void main(String args[]){
        System.out.println("Hello Java");
    }
}
```

→ 4 types :-

1) Standalone :- Desktop Apps ⇒ Media Player, Antivirus.

2) Web Appln :- runs on server side ⇒ Servlets, JSPs.

3) Enterprise Appln :- Banking Apps ⇒

4) Mobile Appln :-

→ Platforms :-

1) JAVA SE :- java.lang, java.io, java.net, java.util, java.sql, java.rmi

↳ OOPs, String, Regex, Exception, InnerClass, Multithreading, ...

2) Java EE :- develop web & enterprise applns.

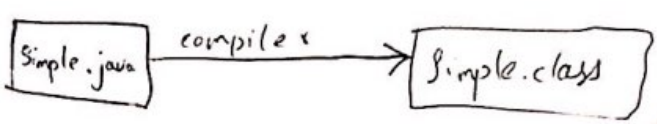
↳ Servlets, JSPs

3) Java Micro Edition :- dev. mobile appln's.

4) JavaFX :- dev. rich internet applns.

→ Features :-

* Simple
* Obj Oriented ⇒ Object, Class, Inheritance, Polymorphism, Abstraction Encapsulation
* platform inde, portable
* Secured ⇒ no pointers,
* Robust ⇒ no ptrs, GC, exception handling.
* Dishibuted, Multi-threaded.



JVM :- Java byte code can be executed
↳ loads code → verifies code → executes code → provides runtime env.

JRE :- set of all tools used for developing

JDK :- dev java applns and applets. physically exists, JRE + dev. tools

Java Variables :- www.javatpoint.com/java-variables.

- local → inside the method
- instance → inside the class.
- static → declare variable as 'static', mmry allocation happens only once when the class is loaded

Primitive Datatypes :-

byte, short, int, long, double, float, boolean, char

Non-Primitive :- String, Array

- abstract :- to declare abstract class. Abstract class can provide the implementation of interface. It can have abstract & non abstract methods.
- boolean :- holds True & False
- break :- breaks loop or switch stmt.
- byte :- declare a variable.
- case :- used in switch stmnts
- catch :- used after 'try', catch the exceptions generated by try stmts.

↓ www.javatpoint.com/java-keywords.

Java Programs :-

Fibonacci :-

```
int n=10; n1=0; n2=1;
System.out.println("Fibonacci Series");
for(int i=1; i<n-1; i++)
{
    if(n1==0 && n2==1)
    {
        S.O.P("  "+n1);
        S.O.P("  "+n2);
    }
    int fib = n1+n2;
    n1 = n2;
    n2 = fib;
    S.O.P("  "+fib);
}
```

Fibonacci Series :-

0
1
1
2
3
5
8
13
21
34

Check Prime Number :-

My code

```
public class CheckPrime{
    P S V main(String args[])
    {
        S.O.P(checkPrime(7));
    }
    public static String checkPrime(int a){
        int counter=0;
        for(int i=1; i<a; i++)
        {
            if(a%i==0){
                counter++;
            }
        }
        return (counter>0)? "NOT Prime" : "PRIME";
    }
}
```

online

```
for(int i=1; i<a/2; ++i)
{
    if(a%i==0){
        counter++; break;
    }
}
```

Palindrome :-

```
public static String checkPalin(int a){
  int temp=a, sum=0, remainder;
  while (a>0) {
     remainder = a % 10;
     sum = remainder+(sum*10);
     a = a/10;
  }
  return (temp==sum) ? "PALINDROME" : "NOT PALIN";
}
```

Factorial :-

→ call this from main fn

ex:- Factorial(4);

```
public static int FactorialNum(int a){
    if (a<=2) { return a; }
    else { fib=1;
      for(int i=1; i<=a; i++){
          fib = fib*i;
      }
      return fib;
    }
}
```

Armstrong :-

→ ArmstrongNum(370);
ArmstrongNum(153);
called from main() fn.

```
public static String ArmstrongNum(int a){
  int temp=a, r, arm=0;
  while(a>0) {
     r=a%10;
     a=a/10;
     arm = arm+(r*r*r);
  }
  return (arm==temp) ? "ARMSTRONG" : "NOT ARMSTRONG";
```

More Programs :-

www.javatpoint.com/java-programs.

OOPs Concepts :-

Object, Class, Inheritance, Polymorphism, Abstraction, Encapsulation

Object :-

→ instance of a class
→ contains an address and takes up some space in memory.
→ It is physical and logical. → "new" to create obj.

Class :- data mems + mem + fns

→ collection of objects
→ logical entity.
→ class do not consume any space.
→ It is a blueprint from which you can create an indi. obj.

Inheritance :-

→ mechanism in wch 1 class acquires the property of other class
→ we can reuse the fields & methods of the existing class.
→ "IS A" relationship.

```
class Doctor{
   void Doctor_details() {
      S.O.P("Doctor...");
   }
}
class Surgeon extends Doctor{
    void Surgeon_details(){
       S.O.P("surgeon....");
    }
}
public class Hospital{
   P S V main(String args[])
    Surgeon S = new Surgeon();
    S.Surgeon_details();
    S.Doctor_details();
  }
```

Polymorphism :-

→ A task performed in different ways is known as Polymorphism.
→ Ex :- A cat speaks Meow, Dog barks Bow, etc;
→ we use method overloading, method overriding.

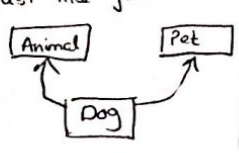
Static Polymorphism	Dynamic Polymorphism
→ method overloading	→ method overiding
→ same method name	→ same method name & signature.
void sum(int a, int b); void sum(float a, float b);	//reference of Parent pointing to child object. Doctor obj = new Surgeon();

Abstraction :-

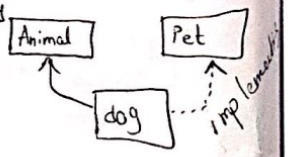
→ Hiding internal details and showing functionality.
→ we used abstract class & interface to achieve Abstraction.
→ An 'Abstract' class may or may not contain abstract methods
→ Reference of an 'abstract' class can point to object of its sub-classes.
→ A class must be 'abstract' if it has ≥1 abstract methods
→ "final" class may not be inherited.
→ "final" variable is a constant & can't be changed.
→ "final" method may not be overridden.

Interface :-

→ just like java class, has static constants & abstract method.



we can't extend 2 classes in java



→ An interface is an 100% abstract class & has only abstract methods.
→ class can implement any no. of methods.
→ www.guru99.com/interface-vs-abstract-class-java.html.

Encapsulation :-

→ Binding code and data together into a single unit.
→ declare variables of a class as Private.
→ provide public setter and getter methods to modify and view the variables values.

Constructor :-

→ constructor name must be same as its class name.
→ must have no explicit return type.
→ constructor can't be abstract, static, final, synchronized.
→ Parameterized constructor
→ Default " (No-arg)
→ Copy constructor.

Static :-

→ "static" is used for memory management in java.
→ Static Variable :- Company name of employees, college name of students...
↳ static variable gets memory only once in the class area at the time of class loading.
→ Static Methods :- The static method can not use non static data member or call non-static method directly.

→ static Blocks :

→ static { S.O.P("static ... "); }

→ It is executed before the main method

→ till JDK 1.6, it is possible to execute java class without the main method.

-> I have 1 year of hands-on experience in MySql, html, javascrit, jquery and PHP.

-> Extensive knowledge of core Java.

→ ~~designed~~ developed & handled stored procedures, cursors and queries.

→ Provided technical support in QA and resolved customer service problems ~~bug~~ and fixed technical bugs.

→ understing & experience of using version control systems like Github and Gitlab.

→ we are a small team of 3 developers. so our roles are really blur

-> I also assist my managers with website developme

→ we also do pair programming //.

Java QA

→ "final" keyword :-

- is used to restrict the user.
- final variable : It will be constant.

HTML5

→ HTML + XHTML, drawing, video, audio

→ <!doctype html> is recognized by all modern browsers

→ embed video on the web pages without using s/w like Flash.

→ <svg> scalable vector graphics.(img's)

→ <canvas> is a rectangle.

→ new form elmnts :-

Color, Date, Datetime-local, Email, Time, Url, Telephone, Number, Search

→ learn API.
→ " javascript
→ " html.
→ " css.
→ web dev. questions
→ read Akshay "2'Atchuts"
→ Search questions

→ 3 types of video frmt :- mp4, WebM, Ogg.

<video controls>
<source ....>
</video>

Rest, Soap API.
Resume

→ 3 types of audio frmt :- mp3, wav, ogg.

<audio controls>
<source src= .... >
</audio>

→ <progress> tag used in conjunction with JavaScript to display the progress of a task.

→ <figcaption> for <figure> tag we get caption for Pic

→ <datalist> provides autocomplete feature.

→ html4 to html5

`<div id="header">`	`<header>`
`</div>`	`</header>`
:	:
`<div id="footer">`	`<footer>`
`</div>`	`</footer>`

→ Java script : pgming lang of html & web

- create interactive websites.
- client-side validation, displaying clocks, date-time sets
- no support for multi-threading.
- Java script is provided by Netscape.
- JScript " " " Microsoft.
- case-sensitive, Var is nt allowed, var is allowed.

BOM → Browser Object Model → Window.
DOM → Document " " → access & change content of html.

- history back(), history.forward(), history.go(number).

→ Primitive :- String, Number, Boolean, Undefined, Null
→ Non " :- Object, Array, RegExp.

- object :-

```
emp = { id:102, name:"sneha" }
   (or)
var emp = new Object();
emp.id = 101;
emp.name = "sneha";
   (or)
function emp(id, name) {
    this.id = id;
    this.name = name;
}
e = new emp(103, "sneha");
```

→ pop-up boxes :-
• Alert, • Confirm, • Prompt.

→ Map(); → map keys to values.

Ex:-
```
var map = new Map();
map.set(1, "jQuery");
document.writeln(map.get(1));
```

→ WeakMap(); → keys are objects

```
var wm = new WeakMap();
var objs = { };
wm.set(objs, "jQuery");
document.writeln(wm.has(objs));
```

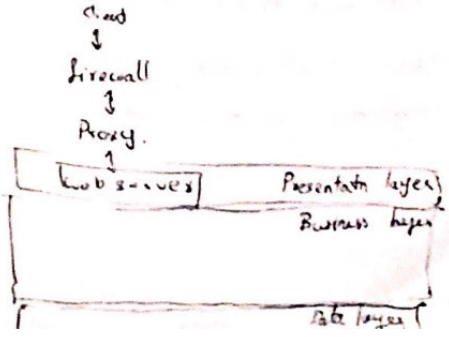
jQuery :- client-side JavaScript library.
↳ well written JS code for event handling, ajax, animatn.

→ $() is alias of jQuery() fn (or) jQuery( )

```
$(document).ready(function() {
    $("p").css("background-color", "pink");
});
```

→ $("p").html(" .... ");
↳ change entire content

→ 3 layered architecture for Web Applications :-



JSwing

```
$(function() {
    var name = $("#name");
    var greeting = $("#greeting");
    name.keyup(function() {
        greeting.text("Hello " + name.val());
    });
});
```

```
<input id="name" type="text">
<h2 id="greeting"></h2>
```

Change ...

// just BootStrap.

```
<script
    src="js/angular-1.0.0rc5.js">
</script>
```

```
<input type="text" ng-model="name">
<h2>Hello {{name}}</h2>
```

Angularjs → open src, JS framework.

- MVC

→ single page application.

Bootstrap :- JS framework

class = ".container", "container-fluid".

↳ BS classes can be added to HTML styling.

Laravel :- PHP web framework, M-V-C architecture

PHP :- server-scripting lang

make dynamic web pages

Angular JS

→ dev by Google

```
<head>
<script src="...js"></script>
</head>
<body ng-app="myapp"> → becomes the root object of Angular JS appln
  <div ng-controller="HelloContainer">
    <h2>Hello {{helloTo.title}}!</h2>
  </div>
<script>
  angular.module("myapp", [])
    .controller("HelloContainer", function($scope) {
        $scope.helloTo = {}
        $scope.helloTo.title = "World, Sneha!!!"
    });
</script>
</body>
```

View Part

Controller Part

Data Binding :- acts as a bridge b/w view and business logic of appln.

1-way :- value is taken from model & inserted into HTML elmnt.

2-way :- model is changed → view is changed & viceversa ✓

ng-app :- defines an AngularJS appln

ng-model :- binds value of html controls (input, select) to appln data.

ng-bind :- binds appln data to html

```
<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
```

ng-init :- to initialize AngularJS appln variables.

```
ng-init="name='John'"
```

Expressions :- {{name}} or {{5+5}}

```
$(function () {
    var name = $("#name");
    var greeting = $("#greeting");
    name.keyup(function () {
        greeting.text("Hello " + name.val());
    });
});
```

```
<input id="name" type="text">
<h2 id="greeting"></h2>
```

Angular JS

```
// just Bootstrap
<script
  src="js/angular-1.0.0rc6.js">
</script>
```

```
<input type="text" ng-model="name">
  <h2>Hello {{name}}</h2>
```

Angular JS → open src, JS framework.

- MVC
→ single page application.

Bootstrap :- JS framework

class = ".container", "container-fluid".

↳ BS classes can be added to HTML styling.

Laravel :- PHP web framework, M-V-C architecture.

PHP :- server-scripting lang.
make dynamic web pages

Angular JS

→ dev by Google

```
<head>
<script src="...angular.js"></script>
</head>
<body ng-app="myapp"> → becomes the root obj of AngularJS appln
  <div ng-controller="HelloController">
    <h2>Hello {{helloTo.title}}!</h2>
  </div>

  <script>
    angular.module("myapp", [])
      .controller("HelloController", function($scope) {
          $scope.helloTo = {};
          $scope.helloTo.title = "World, Sasha !!!";
      });
  </script>
</body>
```

View Part

Controller Part

Data Binding :- acts as a bridge btw view and business logic of appln.

1-way :- value is taken from model & inserted into HTML elmnt.

2-way :- model is changed ⟹ view is changed & viceversa ✓

ng-app :- defines an AngularJS appln

ng-model :- binds value of html controls (input, select) to appln data.

ng-bind :- binds appln data to html

```
<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
```

ng-init :- to initialize AngularJS appln variables.

```
ng-init="name='John'"
```

Expressions :- {{name}} or {{5+5}} //10

ng-repeat: clones html elmnts once for each item in a collectn.

```
<div ng-init="names=['Jones','Kim','Ellen']">
    <li ng-repeat="x in names">
        {{x}}
    </li>
</div>
```

directive: new directives are created.

Eg: `<w3-test-dir></w3-test-dir>`

```
<script>
    var app = angular.module("myApp", []);
    app.directive("W3TestDir", function() {
        return { template: "<h1>Hello!!!</h1>" };
    });
</script>
```

- **ng-model:-** adds/removes the fol. classes acco. to the status of the form field:-

→ ng-empty → ng-valid → ng-invalid ....

ng-click:-

```
<h1 ng-click="changeName()">{{firstName}}</h1>

<script>
    ...
    app.controller('myCtrl', function($scope) {
        $scope.firstName = "John";
        $scope.changeName = function() {
            $scope.firstName = "Harry";
        }
    });
```

→ Add filters by using the pipe character |, followed by filter.

{{lastname | uppercase}}

→ filters to transform data :-

currency, lowercase, uppercase, limitTo, orderBy, data...

-> Angular JS has builtin services like $location. ↓ gives the url of the page.

* **$http:-** most commonly used services in Angular JS.

↳ makes a request to server & lets ur appln handle the resp

→ $timeout service similar to window.setTimeout.

→ $interval " " " window.setInterval.

ng-DOM :- ng-hide, ng-show,

Events :- ng-click, ng-copy, ng-focus, ng-mousemove, ng-mouseover

w3schools.com/angular/angular_events.asp.

$event object contains the browser's event obj.

→ angular_validation.asp in w3schools.org.

API :- Appln Pgm Interface.

↳ set of global JavaScript fns for performing common tasks.

Eg:- angular.lowercase()

angular.isString()

→ we can use w3.css together with Angular JS.