```
class Student
{
public static void main (String args[])   → Square bracket Parenthesis
int a=5, b=2;
int c=a+b;                                  } Variables can be declared wherever necessary and Java doesn't have header files, pointer concepts.
System.out.println(c);
System.out.println("c value is"+c);
}
```

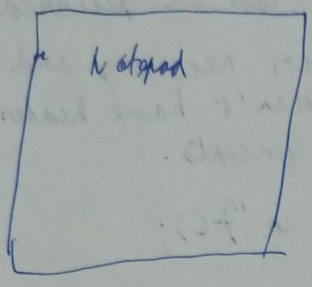
1) Print will be displayed in same line while println can be displayed in next line.

```
Public static void main (String args[])
  class Student
  {
  public static void main (String args[])
  {
     int a=2, b=3;
     int c;
     c=a+b;
     System.out.println(c);
     System.out.println("value of c"+c);
  }
}
```

Keywords

* class
* Public

Student.java



c:\Desktop\java\bin\> javac c Student.java

[create the class file]

Student.class

> java Student

Syntax

```
Class classname
```

Syntax

```
classname objectname;
```

Syntax

```
classname Object name = new classname();
```

Student s1 = new student();
Book b1 = new Book(); → Constructor
allocate memory space

C is a Procedure oriented Programming language
C++ is a partially OOP language
Java is a purely OOP language

1. Class
2. Object
3. Data abstraction
4. Data Encapsulation
5. Data Inheritance
6. Inheritance
7. Polymorphism
8. Dynamic Binding
9. Message Passing

Class is a collection of objects

Object is created by using a classname.

Data abstraction - Details about ~~data~~ Object / Data.
Properties of data / contents related to data
(Eg) Size, Price, Colour

Java is a platform independent. Java is a purely OOP language.

Java doesn't have any header files. It doesn't use pointers / pointer variable instead we use new operator.

Object - an entity in the world

4) Data Encapsulation (or) Data Hiding (or) Information Hiding

Wrapping of data (protecting) data

5) Inheritance

Deriving one class from another existing class

6) Polymorphism

Changing from one form to another form.

7) Dynamic binding

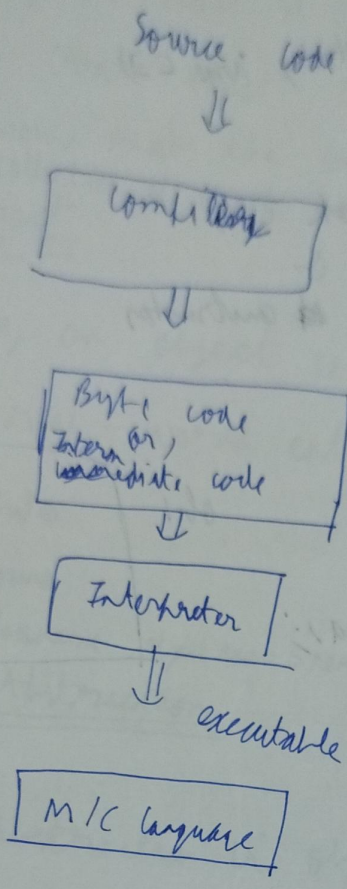
Runtime - At the time of execution,

8) Message Passing

Object name . method ()
or object

Data can be moved from one place to another

Java virtual machine (JVM)



Java is a platform independent language.

Java Program

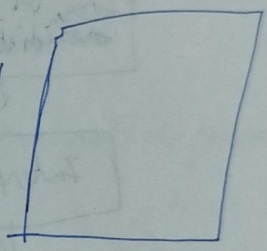
```
class book
{

int a=5, b=3, c;

return datatype ← void putdata()  ← user defined method name
{
    c = a+b;
    System.out.println(c);
    System.out.println("The result" + c);
}
}
```

```
Class Subject
{
public static void main (String args [])
{
```

book ob1 = new . book ();

- book → Class name
- new → memory reference pointer
- book () → constructor

ob1.putdata ();
↳ dot operator




```
System.out.println(ob1.a);
int K = ob1.a;
System.out.println(K);
}
}
```

Scanner

```
import java.util.Scanner.*;
class Student
{
 Public static void main (String args[])
 {
  Scanner ob = new Scanner (System.in)
```

memory reference

```
  String a;
  a = ob.nextline();
  System.out.println("A value" +a);
 }
}
```

```
import java.util.Scanner.*;
class Calk
{
   String a;
   ~~public~~ ~~int a=3, b=2, c=~~;
   void showdata()
   {
     ~~c=a+b;~~ Scanner ob = new Scanner (System.in)
     ~~System.out.println(c);~~ a = ob.nextline();
     System.out.println(a);
     ~~System.out.println("The result" +c);~~
```

↳ next Int();
next Double();

```
   }
}
```

```
class Board
{
Public static void main(String args[])
  {

    Scanner ob = New Scanner(System.in)
    chalk ob = new.chalk();
    ob1.data();

}
}
```

Primitive Data type

byte – 8 bits

Short – 16 bits

Int – 16 bits

long – 32 bits

Float – 32 bits

Double – 64 bits

Char – Single Bit (a group of character)

Boolean

```
byte b = 10;
short a = 588;
int c = 5000;
long l = 40000L
float c = 5.5f
double k = 5.1f
```

Type Casting

* Implicit
* Explicit

```
int a = 5, b = 6
double c = a+b;    // Implicit
System.out.println(c);

int e = (int) c; // Explicit
System.out.println(e);

class Student
{
public static void main (String args[])
{
String s1 = "Hai";
String s2 = "Hello";
String s3 = s1+s2;
System.out.println(s3);
}
}
```

String Builder

String Builder S1 = ~~"Hello"~~ new String Builder("welcome");

Program on String Builder

```
Public class window
{
Public static void main (String args[])
{
  String Builder Sb = new String Builder ("welcome");
  S.O.P (Sb);
  Sb.append (" to REC");
  Sb.Insert (3, "Java");
  Sb.replace (0, 5, "Hi");
  Sb.delete (8, 14);
  Sb.reverse ();
int capacity = Sb.capacity ();
char char At5 = Sb.charAt (5);
    S.O.P. (charAt 5);
String substring = Sb..substring (5, 10)
S.O.P (Subs);
```

Syntax and Program for Inheritance

Syntax

```
class Parent {
    ....
}
class Child extends Parent {
    ....
}
```

(Eg)

```
class Animal {
  void eat() {
    System.out.Println("This animal eats food.");
  }
}
class Dog extends Animal {
  void bark() {
    System.out.Println("The dog barks");
  }
}
Public class main {
  Public static void main(String[] args) {
    Dog myDog = new Dog();
    myDog.eat();
    myDog.bark();
  }
}
```

Syntax and Program for Polymorphism.

Syntax

```
class className{
void methodName(int a){
    ....
}
void methodName(int a, int b){
    ....
}
}
```

(Eg)

```
class Calculator{
int add(int a, int b){
    return a+b;
}
int add(int a, int b, int c){
    return a+b+c;
}}

Public class main{
Public static void main(String[] args){
Calculator calc = new Calculator();
System.out.Println("Sum of 2 numbers:"+calc.add(5,10));
System.out.Println("Sum of 3 numbers:"+calc.add(5,10,15));
}}
```

Syntax and Program for Dynamic Binding

```
class Superclass{
    void display(){
        System.out.println("Display from superclass");
    }
}
class Subclass extends Superclass{
    void display(){
        System.out.println("Display from subclass");
    }
}
Public class Main{
    Public Static void main(String[] args){
        Superclass obj = new Subclass();
        obj.display();
    }
}
```

Syntax and Program for Message Passing

Syntax

Object Name.MethodName(arguments);

(Eg)

```
class Sender{
    void Send Message(Receiver r, String msg)
    {
        r.receive Message(msg);
    }
}
```

```
class Receiver {
    void receiveMessage(String message) {
    System.out.println("Message received: " + message);
    }
}

Public class Main {
    Public static void main(String[] args) {
        Sender sender = new Sender();
        Receiver receiver = new Receiver();
        sender.sendMessage(receiver, "Hello from Sender!");
    }
}
```

Type Casting

Type casting in Java is converting a variable from one data type to another. It is of two types:

1. Widening (Implicit) Casting:
   Smaller → larger type (automatic)
2. Narrowing (Explicit) Casting:
   larger → smaller type (manual)

Syntax for Implicit Typecasting:

```
smallType variable = value;
largeType newvariable = variable;
```

Syntax for Explicit Type casting:

```
largeType variable = value;
smallType newvariable = (smallType) variable;
```

Program for Implicit and Explicit Type casting:

```
Public class Main{
    Public static void main (String [] args){
        int num =10;
        double wideResult = num;
        System.out.Println ("Widening: int to double =".
        + wideResult);

        double Price = 99.99;
        int narrow result = (int) price;
        System.out.Println (" Narrowing: double to int =" +
        narrowResult);
    }
}
```

Java development kit/tool (JDK)

let
Appletviewer

javac

java

javadoc

Jdb

Javah -

It is used to run Java Applets.

Java compiler:

It convert source code to byte code (Intermediate

Java:

Interpreter checks the code line by line.

Java doc

Creates Html Documentation from Java source code files.

Jdb (Java debugger)

It helps us to find errors in program.

Java h

Javah creates a header file

API

Application programmable Interface is a language support package, utility packages, Applet Packages, Abstract window Tool Kit - Awt

Comments in Java Program

1) Single line - //

2) Multi line - /* - - - */

3) Documentation - /** - - - - - - - */

Java Variables

variable name is in form of alphabets, digits....

first letter must alphabet

↳ upper case, lower case are allowed

↳ no special character is allowed.

↳ _ - underscore is allowed

↳ keywords are restricted strictly

Java operators

Arithmetic operators

+, -, /, *, %

Relational operators

>, <, ==, >=, <=, !=

logical operators

AND, OR, NOT

&& || !

Short hand assignment

a = a + b

(Eg) a += b

Increment and decrement operators

++, --

Special operators → conditional operator ? :
(Ternary operator)
→ new
→ . (dot operator)

Bitwise operator

& . | ^ ~ (Tilde)

```
class Java
{
public static void main (String [] args)
{

    int a = 5, b = 3;
    int c = a + b;
    int d = a - b;
    int e = a * b;
    int f = a / b;
    int g = a % b;

System.out.println ("Sum :" + c);
System.out.println ("Subtraction :" + d);
```

String

1) Immutable
   cannot change the value.
2) Contains the method under the package
   java.lang

import java.lang.*;

Class Example
{
  Public static void main (String [] args)
  {
    —
    —
    —
  }
}

```
Public class StringMethodsExample{
    Public static void main(String[] args){
    String originalString = "Hello Java!";
    System.out.println("length:" + originalString.length());
    System.out.println("character at index 1:" + originalString.
                                                    charAt(1));
    System.out.println("Contains 'Java':" + originalString.
    Contains("Java"));
    System.out.println("Substring (index 2):" + originalString.
                                                substring(2));
    System.out.println("Trimmed String:" + originalString.
                                            trim() + "!!!");
    System.out.println("Uppercase:" + originalString.toUpperCase());
    }
}
```

```
Public class StringBuilderExample{
    Public static void main (String [] args)
    {
        StringBuilder sb = new StringBuilder ();
        sb.append ("Hello");
        sb.append (" ");
        sb.append ("world");
        sb.append ("!");
        sb.insert (6, "Beautiful");
        sb.delete (6, 16);
        sb.replace (0, 5, "Greetings");
        sb.reverse ();
        int length = sb.length ();
        char ch = sb.charAt (0);
        int index = sb.indexOf ("world");
        String substring = sb.substring (0, 7);
```

Feature	String	StringBuilder	StringBuffer
mutability	Immutable	Mutable	Mutable
Thread safety	Immutable (therefore, thread safe)	Not thread safe	

Constructor V.V.I 

↳ Special method in a class.

↳ Gives the initial values to instance variables.

↳ It is automatically at the time of object creation.

Types ↳ with arguments
↳ without arguments

```
Constructor name (arguments)
{
  - -
  - -
}
```

Rule

Name of the constructor must be as the name of class name.

Constructor it's called explicitly.

Constructor has no return datatype.

→ without arguments called default constructor

→ with arguments called Parameterized constructor

(Eg)

```
class BoxOne
{
    BoxOne()
    {
    }
};
```

constructor . Example program

Default constructor

```
class examplecon
{
    int height;
    int weight;
    examplecon()
    {
        height = 5;
        weight = 6;
    }
int area()
```

```
{
 return height * weight;
}
}
```

used for initializing the value of variables - default constructor.

```
class Sample
{
 public static void main (String args[])
 {
  examplecon ec = new examplecon();
  int d = ec.area();
  System.out.println(d);
 }
}
```

Parameterized constructor

```
class examplepram
{
  int height;
  int weight;
examplecon(int a, int b)
  {
    height=a;
    weight=b;
  }
int area()
  {
  return height * weight;
  }
}
class sample
{
public static void main (String[]args)
  {
    examplecon param ep = new examplecon param (5,6);
    int d = ep.area();
    System.out.println(d);
  }
}
```

```
class Student
{
  int c;
  Student ()
  {
    int a = 0;
  }

  Student ( int b)
  {
     c = b;
  }
     void Printdata ()
class Example { System.out.Print(a);
{               System.out.Print(c);
              }
  Public static void main (String [] args)
     {
       Student s1 = new Student ();
       Student s2 = new Student ();
       s1. Printdata ();
       s2 .Printdata();
     }
}
```

```
Class details
{
  int rollno;
  String name;
details()
 {
  roll no = 5
  name = rec;
 }
details (int r, String n)
  {
    rollno = r;
    name = n;
  }
    void printdata()
    {
    System.out.print(rollno);
    System.out.print(name);
    }
}
```

```
class mdetails
{
  public static void main (String [] args)
  {
    details d[] = new details [3];

    d [0] = new details (1, "abc");
    d[1] = new details (2, "xyz");
    d [2] = new details ( 3, "klm");
    d [3] = new details (4, "qzr");
    d[0]. Printdata();
    d [1]. Printdata();
    d [2]. Printdata();
     d [3]. Printdata();
  }
}
```

Java Architecture

It reduces overhead during the execution of the program
Java uses Just In Time (JIT) compiler to run the compiler
compiles code on-demand basis when it only compiles those methods

Static methods and members

```
class Samplestatic
{
    static int k = 5; // Static member
static void putdata() // Static method
    {
      System.out.println(k);
    }

public static void main (String [] args)
      {
        Putdata();
      }
}
```