Java

s of Java :

1) Secure (Rich in libraries)
2) Architechtural Neutral
3) Distributed
4) Object oriented
5) Data types
6) Single Programme
7) Command line argument
8) How to read value from Keyboard

JDK (Java Development kit)
JVM (Java virtual machine)
+
Libraries

Syntax :

not mandatory
package pkg.name;
import java packagename.classname;
import java.pkgname *;
class classname
{ datatype data_mem;
method1()
{
}

method 2
{

use } publically returnn nothing command line
public static void main (String args[])
{ objects not required function (special)
}
}

* Convert string to integer

```
class TestCond
{ public static void main (String args[])
{ int a = Integer.parseInt(args[0]);
  int b = Integer.parseInt(args[1]);
```

	args[]
0	20
1	hello
2	hi
3	30
4	60
5	70.6

```
    int add = a+b;
    System.out.println("Add = " + add);
    }
}
```

* Read Value

```
import java.util.Scanner;
class TestCmd
{ public static void main (String args[])
{ Scanner sc = new Scanner( System.in);
  int a = sc.nextInt();
  String str = sc.nextline(); }}
```

* WAP to read subject marks for particular student and calculate percentage

```
import java.util.Scanner;
class Perc
{ public static void main (String args[])
 { Scanner sc = new Scanner (System.in);
   int s1, s2, s3, s4, s5, s6, perc;
   string name;
   System.out.println ("Enter name:" + name);
   String name = sc.nextline();
   System.out.println ("Enter subject1 marks:" + s1);
   int s1 = sc.nextInt();
   System.out.println ("Enter Subject 2 marks:" + s2);
   int s2 = sc.nextInt();
   :
   System.out.println ("Percentage = " + perc);
   perc = (s1+s2+s3+s4+s5+s6/6);
 }
}
```

- Conditional Statements
  (if, else-if, switch)
- Looping statement / structure
  (for, while, do-while)
- Break, continue

* WAP to check whether given number is +ve, -ve or zero.

```
import java.util.scanner;
class check
{ public Static void main(String args[])
 { Scanner sc = new Scanner(System.in);
   int no1 = sc.nextInt();
   if (no1 >0)
     { system.out.println("Positive number");
   else if (no1<0)
     { System.out.println("Negative number");
   else
     System.out.println("Number is zero");
     }
}}
```

* WAP to print a day which is linked with integer no.:
0: Sunday 1: Mon ...

```
import java.util.Scanner;
class Day
{ public static void main (String args[])
 { Scanner sc = new Scanner (System.in);
   int day = sc.nextInt();
   System.out.println("Enter number of day");
 {Switch (day)
  { case 0:
      System.out.println("Sunday");
    break;
```

```
case 1:
    System.out.println("Monday");
    break;

default case:
    System.out.println("There exist no day for this no.");
    break;
}
```

* WAP to print a pattern.

```
*
* *
* * *
* * * *
```

```
import java.util.Scanner;
class Pattern
{ public static void main(String args[])
  { Scanner sc = new Scanner(System.in);
    int line = sc.nextInt();
    System.out.println("Enter no. of lines");
    for (i=1; i<5; i++)
    { for (j=1; j<=i; j++)
      { System.out.println("*");
      }
      System.out.println();
    }
```

Eg:

```
for (i=1; i<5; i++)
  { for (sp=1; sp<=4-i; sp++)
    { System.out.println(" "); }
    for (j=1; j<=i; j++)
    { System.out.println("*"); }
  System.out.println();
}
```

```
*
* *
```

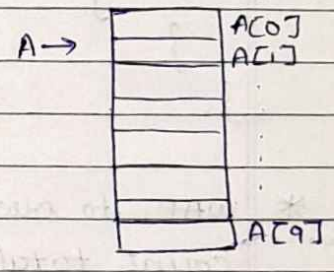
Page No.
Date :

Eg:
```
for (i=0; i<50; i++)
{ if (i==25)
    break;
}
```
This will print 26 times

3/8/23

Array: (1 dimensional)

int A[]={1,2,3,4,5};




```
datatype array_ref[];
datatype[] array_ref;
datatype []array_ref;
```

```
datatype arrayname[] = new datatype[size];
int A[] = new int[10];
```

* WAP to create an array with 10 elements and calculate addition, average, minimum & maximum value from it.

```
import java.util.Scanner;
class array
{ public static void main(String args[]);
  { Scanner sc = new Scanner(System.in);
    int A[] = new int[10];
    System.out.println("Enter inputs:");
    for(int i=0; i<10; i++)
      { A[i] = sc.nextInt(); }
    int sum = 0;
    for(int i=0; i<10; i++)
      { sum += A[i]; }
    float avg = sum/10.0;
    int max = A[0];
    int min = A[0];
```

```
for(int i=0; i<10; i++)
  { if (max < A[i])
        max = A[i];
    if (min > A[i])
        min = A[i];
    }
  }
}.
```

* WAP to create an array with size input by user and count total no. of even and odd nos. present in array.

```
import java.util.Scanner;
class Test
{ public static void main (String args[])
  { Scanner sc = new Scanner(System.in);
    int A[] = new int[
    int size = sc.nextInt();
    int A[] = new int[size];
    for (int i=0; i<10; i++)
     { System.out.println("Enter no.");
        A[i] = sc.nextInt(); }
     int e=0;
     for (int i=0; i<10; i++)
      { if (A[i] % 2 == 0)
            e++;
      }
     int o = size-e;
     System.out.println("Even are: "+e+"\n Odd are: "+o);
  }
}
```

Page No.
Date :

Array : (2 dimensional)

datatype array_ref[][]; → array_ref = new int[3][3];
datatype[][] array_ref;
datatype [][]array_ref;

int A[][] = new int [3 rows][3];
int A[][] = {{1,2,3},{4,5,6},{7,8,9}};

* Read values for 2-D array

```
for (int i=0; i<3; i++)
  { for (int j=0; j<3; j++)
      {
        A[i][j] = sc.nextInt();
      }
  }
```

4/8/23

* WAP to add ~~tb~~ 2 matrices

```
import java.util.Scanner;
class matrix
{ public static void main (String args[])
  { Scanner sc = new Scanner (System.in);
    int A[][] = new int[3][3];
    int B[][] = new int[3][3];
    int C[][] = new int[3][3];
   for (int i=0; i<3; i++)                    // Matrix A input
    { for (int j=0; j<3; j++)
       { A[i][j] = sc.nextInt(); }
    }
   for (int i=0; i<3; i++)                    // Matrix B input
    { for (int j=0; j<3; j++)
       { B[i][j] = sc.nextInt(); }
    }
```

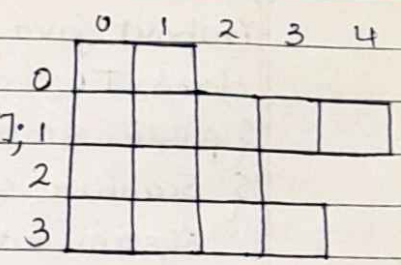
```
System.out.println("Sum is:");
for(int i=0; i<3; i++)
  {for(int j=0; j<3; j++)
    { C[i][j] = A[i][j] + B[i][j]; }
  }
}
}
```

* WAP to multiply two matrices

```
import java.util.Scanner;
class mul
{public static void main(String args[])
 { Scanner sc = new Scanner(System.in);
   int A[][] = new int[3][3];
   int B[][] = new int[3][3];
   int C[][] = new int[3][3];

   for(int i=0; i<3; i++)
     {for(int j=0; j<3; j++)
       { A[i][j] = sc.nextInt(); }
     }
   for(int i=0; i<3; i++)
     {for(int j=0; j<3; j++)
       { B[i][j] = sc.nextInt(); }
     }
   System.out.println("Multiplication is:");
   for(int i=0; i<3; i++)
    {for(int j=0; j<3; j++)
      {for(int k=0; k<3; k++)
        { C[i][j] += A[i][k] * B[k][j]; }
      }
    }
 }
}
```

Jagged Array :



Jagged array (diff columns)

```
datatype array-name[][] = new datatype[size][];

int A[][] = new int [4][];
    A[0] = new int [2];
    A[1] = new int [5];
    A[2] = new int [3];
    A[3] = new int [4];

for (int i=0; i<A.length; i++)
    {for (int j=0; j<A[i].length; j++)
        {
            A[i][j] = sc.nextInt();
        }
    }
```

String :

```
String s = new String("Hello");
String s1 = "Hi";
String s2 = "hello";
String s3 = "Hi";
String s4 = "hello";
```

s1 → | H | i |
s3 ↗

s2 → | h | e | l | l | o |
s4 ↗

no new memory allocation for same string....

String class 11/8/23

- constructor
- methods

- length
- split
- contains
- ends with
- startsWith
- toUpperCase
- toLowerCase
- trim
- charAt
- substring
- concat
- indexOf

```
import java.util.Scanner;
class T1
{public static void main(String args[])
{ Scanner sc = new Scanner(System.in);
  System.out.println("Enter string of length 10");
  String s1 = sc.nextline();
//length  System.out.println("String length is :"+ s1.length());
//charAt  System.out.println("char at position 4 is :"+ s1.charAt(4));
//toUpper System.out.println("String in Upppercase:"+s1.toUpperCase());
Case
//toLowerCase System.out.println("String in Lowercase:"+s1.toLowerCase());
//endsWith System.out.println(
  if (s1.endsWith("Hello"))
     System.out.println("String endswith Hello");
//substring String s2 = s1.substring(6);
  System.out.println("String s2:"+s2);
//indexOf int index = s1.indexOf(s2);
  System.out.println("String s1 contains s2 at:"+index);
//contains String s3="hello";
  if (s1.contain(" "))
     System.out.println("String s1 contains s3");
  else
     System.out.println("String s1 doesn't contain s3");
//concat String s4 = s3.concat(s1);
  System.out.println("string s4="+s3);
//split String s5 ="hello @  I am@ Robot @GISFCU ";
  String strA[] = s5.split(" ");
  for (String s:strA)
     System.out.println(s);
//trim System.out.println("Before trim ="+s5);
  System.out.println("After trim ="+s5.trim());
  }
}
```

Class & object

Page No.
Date: 16/8/23

```
import java.util.Scanner;
class Student
{
  int rollno;
  String name;
  float spi;

  Void displayData()
  { System.out.println("Rollno.: "+rollno);
    System.out.println("Name: "+name);
    System.out.println("Spi: "+spi);
  }
  Void getData()
  { Scanner sc = new Scanner(System.in);
    rollno = sc.nextInt();
    name = sc.nextline();
    spi = sc.nextdouble();
  }
}

class Main
{ public static void main(String args[])
  {
    Student s = new Student()
    s.getData();
    s.displayData();
```

Run:
Save by; Student.java
OR
Main.java
Run; javac Student.java
java Main

Constructor

Page No.
Date :

```
class Ball
{
  int size;
  Ball()
  { size=10; }
  Ball(int size)
  { this.size = size; }
      ↳ make this function's object size = number func size
  void display()
  { System.out.println("size=" + size); }
}

class Main
{ public static void main(String args[])
  { Ball b1 = new Ball();
    Ball b2 = new Ball(25);
    b1.display();
    b2.display();
  } System.gc(); // garbage collector
                    which diallocate memory
}
```

* Create a class Arithmetic which contains method addition Substraction, multiplication & division
1 Create appropriate constructor for same

```
import java.util.Scanner;
class Add
{ int a, b, c;
  Add()
  { c = a+b;
    System.out.println("Sum is:" + c);
  }
}
class Substra
import java.util.Scanner
class Arithmetic
{ void Add()
  {
```

→

```
import java.util.Scanner
class Arithmetic
{ int no1, no2;
  Arithmetic(int no1, int no2)
  { this.no1 = no1;
    this.no2 = no2;
  }
  void Addition()
  { System.out.println("Sum = "+(no1+no2)); }
  void Substraction()
  { - - - - - - - - - - - - - - - - - - - - }
  void multiplication()
  { - - - - - - - - - - - - - - - - - - - - }
  void division()
  { - - - - - - - - - - - - - - - - - - - - }
```

```
class Main
{ public static void main (String args[])
  { Arithmetic a1 = new Arithmetic (10,12);
    a1. addition ();
    a1. substraction ();
    a1. multiplication ();
    a1. division ();
  }
}
```

17/8/23

Method Overloading

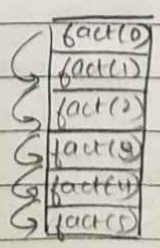
Function / method name is same
different → Arguments
→ datatypes

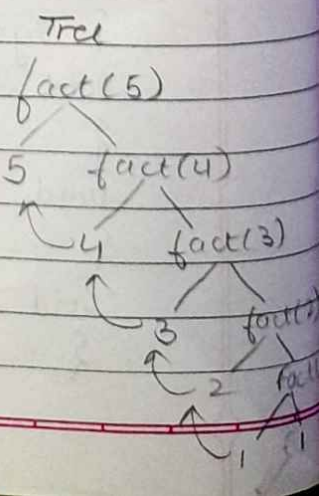
```
class Test
{ void add (int x)
  { x = x+10; }
  void add (int x, int y)
  { int ans = x+y; }
  void add (float x, float y)
  { int ans = x+y; }
  void add (String x, String y)
  { String ans = x+y; }
}
```

Recursion

factorial

```
int fact (int no)
{ if (no == 0)
    return 1;
  else
    return (no * fact (no-1));
}
```





fibonacci 1 1 2 3 5

Passing obj as method argument

* WAP to add two numbers of two different object.

```
Class Number
{ int x;
  Number (int x)
  { this.x = x; }
  Void add (Number n)
  { int ans = this.x = n.x;
    System.out.println ("Sum is:" + ans);
  }
}
class main
{ Number a1 = new Number (10);
  Number a2 = new Number (20);
  obj1.add (obj2);
}
```

* Create a class Employee having Empno, dipno, sal as datamumber. Create appropriate constructor & method for initialize and display data member.
Create a class Account having datamembers like total Expenditure, create a method calculate expense which calculate total salary drawn by each employe

Array of Objects

```
classname array name[] = new classname [size];
datatype vari[] = new datatype [size];
int A[] = new int [10];
class x                    X obj[] = new X[10]; → Array of objects of class x
{
}
```

Array of object

Page No.
Date : 18/8/23

* WAP to create class A having a1 as datamember. Create array of 5 objects instantiated and display the value

```
class A
{ int a1;
  A(int a1)
  {
    this.a1 = a1;
  }
  void display()
  {
    System.out.println("a1= "+a1);
  }

  public static void main(String args[])
  {
      A obj[]= new A[5];
      for(int i=0; i<5; i++)
        {
          obj[i]= new A(i+1);
          obj[i].display();
        }
  }
}
```

Returning obj from Method.

Page No.
Date :

```
class Number
{
  int no;
  Number (int no)
  { this.no = no; }

  Number temp incTen ()
  {
      Number temp = new Number (0);
      temp.no += 10;
      return (temp);
  }
  public static void main (String args[])
  {   Number n1 = new Number ();
      n1 = n1.Number (20); n1 = n1.incTen (20);
  }
}
```

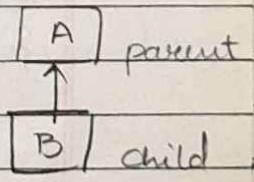
Inheritance: (single, multilevel, hierarchical)

- Single Inheritance

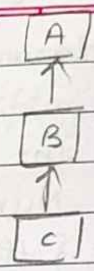
```
class A
{ int a;

}
class B extends A
{ int b;

}
```

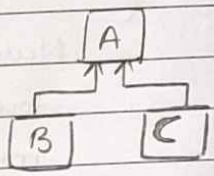


- Multilevel inheritance




```
class A
{ int a; }
class B extends A
{ int b; }
class c extends B
{int c;}
```

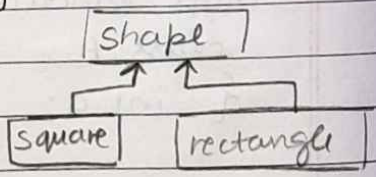
- Hierachical inheritance




```
class A
{int a;}
class B extends A
{int b;}
class C extends A
{int c;}
```

Method overriding

* WAP name Shape having method calcarea().
Create a class square and rectangle which inherit Shape class. Overwrite the method calcarea to find out area of square and rectangle.




```
class A
{ A()
  {S.o.p(" ");}
  void display()
  { . . . }
}
class B extends A
{ B()
  {S.o.p(" ");}
  void display()
  {diff definition}
}
```

```
main()
{ B obj = new B();
  obj.display();
}
```

Constructor chaining

```
class A
{ A()
  { s.o.p("I am A"); }
}
class B extends A
{ B()
  { s.o.p("I am B"); }
}
class C extends B
{ C()
  { s.o.p("I am C"); }
}

main()
{
  C obj = new C();
}
}
```

Output:
I am A
I am B
I am C

// C constructor calls B constructor and B constructor call A constructor on obj creation of Class C

23/8/23

Super keyword refer to datamember of immediate parent of child class.

Super can be used with variable, method & constructor

① (constructor)

```
class A
{ int a;
    A(int a)
    { this.a = a; }
}
class B extends A
{ int b;
  B(int a, int b)
  { super(a);
    this.b = b; }
```

```
class C extends B
{  int c;
     C(int a, int b, int c)
    { super (a,b);
      this.c=c;
    }
}
class Test
{ public static void main (String args[])
  { C obj = new C(1,2,3); }
}
```

② (Variable)

```
class A
{ int a;
  A (int a)
  { this.a=a; }
}
class B extends A
{ int b;
  super(a); B(int a, int b)
   { super(a);
     this.b=b; }

void add()
{ int addition = super.a + this.b;
  System.out.println("ans="+addition); }
}
```

// if class has abstract method then it needs to be abstract class; vice versa is not true...

③ (method)

```
class A
{ void disp()
  { System.out.println("I am A"); }
}
class B extends A
{ void disp()
  { super.disp();
    System.out.println("I am B");
  }
}
main()
{ B obj = new B();
  obj.disp();
}
```

Output:
I am A
I am B

If recursion then infinite loop.

Abstract class

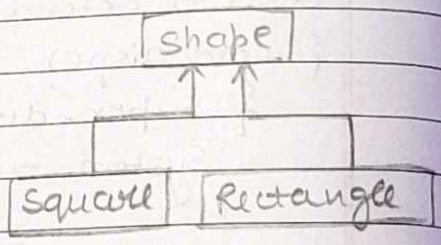
```
abstract class A                    // cannot create object of A class but can create refrence.
{ void sayHello() (fun" difined)
  {System.out.println("Hello");}
abstract void sayHi(); (fun" declared)
}
```

1)
```
class Test
{ public static void main(String args[])
  { // A obj = new A();   → Error
    A ref;
  }
}
class B extend A
{ // override
  Void sayHi()
  { System.out.println("Hi"); }
}
```

* Abstract class
* Abstract method
* Method overriding

Page No.
Date : 24/8/23

Q. Write a program name class shape having an abstract method calarea(). Create 2 subclasses of shape class name square() and rectangle() and overwrite calarea method.




```
abstract class shape
{ public static int float area;
abstract void calcarea();
}

class square extends shape
{ int a; float l;   Square(int l) { this.l=l; }
  void calcarea()
  { area = l*l;
    System.out.println("Area is = "+area); }
}

class rectangle extends shape
{ rectangle (i float l,b;
    Sqp rectangle (int l, int b)
        { this.l = l;
          this.b = b; }
  void calcarea()
  { area = l*b;
    System.out.println("Area is = "+area); }
}
class Test
{ ——main——(String args[])
  { square obj = new square(10);
    obj.calcarea();
    rectangle obj1 = new rectangle(10,20);
    obj1.calcarea(); }
}
```

Interface (contract)

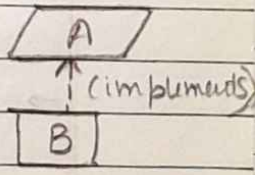
Page No.
Date :

* An interface in java contains static constant and abstract method.
* It is used to achieve abstraction.
* Since Java8 default and static methods are allowed in interface.
* Since Java9 private methods are allowed in interface.
* It is used to achieve loose coupling

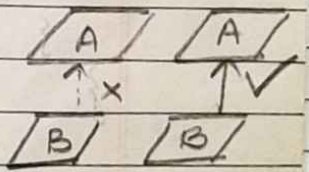
Syntax:
```
interface interfaceName
{ // static final variable
  // abstract method
}
```
// no variable, all methods are by default abstract, no method definition.
no constructor, only have variables with fix value, no final method, only have default methods

Ex:
```
interface Shape
{ void calArea(); }
```

Ex:
```
interface A
{ void getA();
  // void displayA();
  // int intA(int no);
  // default void sayHi()
  // { System.out.println("Hi"); }
}
```
```
interface C
{ }
```
// public, abstract by default

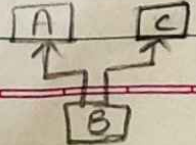



```
class B implements A , C
{ public void getA()
  { System.out.println("a is 10"); }
}
```



// interface can extend another interface but cannot implement

// interface allows multiple inheritance



Ex:
```
interface A
{ void disp(); }
interface B extends A
{  void draw(); }
```

25/08/23

Dynamic Method Dispatch

```
class TestA
{ void sayA()
   { System.out.println("A"); }
}
class TestB extends TestA
{ void sayA()
   { System.out.println("B"); }
}
class DynamicCD
{ public static void main(String args[])
  { TestA obj;
    obj = new TestA();
    obj.sayA();          // A Runtime polymorphism
    obj = new TestB();   // upcasting
    obj.sayA();          // B Runtime polymorphism.
    TestB obj1 = new TestB();
    obj1.sayA();
  }
}
```

Output:

```
A
B
B
```

→ 'final' keyword

1) variable → final datatype variableName = value;
→ you can't change its value

2) method → final returntype methodName (list of Parameters)
{
}
→ You can't override

3) class → final class className
{
}
→ You can't extend it.

Ex:
```
final float pi = 3.14;
final void disp()
{
}
```

Ex:
```
final class Testfinal
{ final int a=10;
  final void disp()
  { a++; // error
    System.out.println("a = " + a); }
}

class ChildTest extends Testfinal // Error
{ void disp()
  { System.out.println("hello"); } } // error
}
```

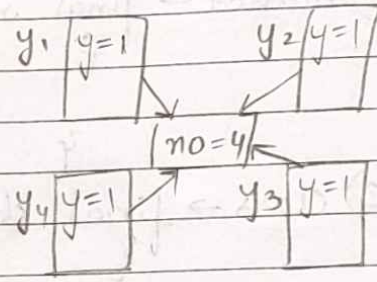
* One static method can call another static method directly in same class...

Page No.
Date :

→ 'static' keyword

Static variable is shared among all objects and have only one memory copy.

1) variable → static datatype varname;




```
class Y
{ int y;
  static int no;
  Y(int y)
  {this.y=y;
   no++;
  }
public static void main (String args[])
  { Y y1 = new Y(1);
    Y y2 = new Y(1);
    Y y3 = new Y(1);
    Y y4 = new Y(1);
  }
```

2) class →

```
class Test
{ void disp()
  { System.out.println("Hello"); }
  static void sayHi()
  { System.out.println("Hi"); }
public static void main (String args[])
  { sayHi();        //direct access within same class and no need of obj for it
    Test obj = new Test();
    obj.disp();
  }
}
```

* Static method cannot call non-static method...

3) Block →

2)
```
class Test
{ static ()
  { S.o.p("Static Block"); }
 public static void main(String args[])
  {S.o.p("Main Block"); }
}
```

Nested Class → Static nested
→ non-static nested class.

```
class outer                    // can never be static class
{
   class inner
   {
   }

}
```

```
class outer
  { void dispOut()
    { S.o.p ("Out"); }
  }
Static class inner
  { void dispIn()
    {S.o.p("In"); }
  }

class Test
  { public static void main (String args[])
    { outer o = new outer();
      o.dispOut();
      outer.inner i = new outer.inner();
      i.dispIn();
    }
```

static nested class

non-static nested class
outer.inner i = o.new inner();

final, finalize diff ...
variable method

Page No.
Date : 8/09/23

Inner class → normal
→ Anonymous

```
class A
 { void dispA()
   { S.o.p("A"); }
   void dispB()
   { class B
     { void dispB()
       {S.o.p ("B"); }
     }
     B obj1 = new B();
     obj1. dispB();
   }
 }
class Test
{ main ( )
   { A obj = new A();
     obj. dispA();
   }
}
```

// class inside method and its obj called within outer class

Output:
A
B

Anonymous

```
Interface X
{ void disp(); }
class Main
{ public static void main (String args[])
   { X obj = new X()
     { public void disp()
       { S.o.p ("I am A"); }
     };
     obj. disp();
   }
}
```

// abstract class x
// {abstract void disp();

// {void disp()

equals
finalize

→ wait(-)
→ clone()
→ finalize
→ hascode()
→ tostring()
→ notify()
→ notifyall()
→ wait()

java : object class java oracle

Page No.
Date :

Object class

- Object class is a parent (superclass) of every class

Finalize method

```
class Test
{   protected void finalize {s.o.p(" ");}
  public ——— (String — )
  { Test o = new Test();
    O=null;
    System.gc();
  }
}
```

```
package pkg_name;
import static java.lang.System;                    java.lang.Integer
class Test
{ — main —
    { out.println("Hello world"); }                int a = parseInt(" ");
}
```