07.12.2024

OBJECT ORIENTED PROGRAMMING

(code reusability)

- Encapsulation
- Inheritance
- Polymorphism
- Data Abstraction

JAVA

Grammar

CLASS

- class is a collection of similar types of objects.

OBJECT

- object is an instance of a class.

① CLASS

Data + Method

Data → variable

Method → operations / functions

Methods these are present inside the class can use data of the class.

② CLASS

Data

③ CLASS

Methods

④ CLASS

(empty class)

10.12.2024

Keyword → reserved word

H.W :- List all methods of Scanner Class.

Method used for string :- nextLine( )

Data types :-

1> Integers Data

byte → 1 byte byte a ;

short → 2 bytes short b;

int → 4 bytes

long → 8 bytes

* h.w → List the range of these data types.

2> float → 4 bytes

double → 8 bytes

1 byte = 8 bit

3> character

char → 2 bytes

4> boolean → 'true' , 'false' → 1 bit

Abstraction- hiding the details

Inheritance - code reusability

Character Input :-

```
import java.util.*;
class P3
{
 public static void main (String ar[])
 {
  Scanner sc = new Scanner (System.in);
  System.out.println ("Enter a character");
   char a = sc.next.charAt (0);
  System.out.println ("Entered character is "+a);
 }
}
```

Enter character : KIIT

Entered character is K

if we change char a = sc.next.charAt (1);

Entered character is I

Command line arguments :-

→ used to accept input in the program at command line.

WAP to display yr name using command line arguments.

```
class P4
{
 public static void main (String ar[])
 {
  System.out.println (" My name is " + ar[0] );
 }
}
```

javac P4.java

java P4 Rahul Satya Pritam

→ My name is Rahul (ar[0])

ar[1] → Satya

Static - assigned at compile time

dynamic - assigned at run time

WAP to add 2 nos , { take 2 inputs from command line }.

```
class P5
{
  public static void main ( String arr[] )
  {
    double a= Double. parseDouble ( arr[0] ) ;
    double b = Double. parseDouble ( arr[1] ) ;
    double c= a+b ;
    System. out. println (" Sum of two nos = " + c) ;
  }
}
```

javac P5.java

java P5 23.5 35.8

⟶ Sum of two nos = 59.3

Double is a wrappes class. (present in java.lang package)

↳ default

parseDouble() → converts a string to primitive data type

eg :- int a= Integer. parseInt (arr[0]) ;

instantiation → object creation

Array

```
// how to declare, instantiate, initialize & traverse array.
class A1
{
  public static void main (String args[])
  {
      int a[] = new int[5];    // declaration & instantiation
       a[0] = 10;              // initialization
       a[1] = 20;
       a[2] = 70;
       a[3] = 40;
       a[4] = 50;
          // traversing array
       for (int i=0 ; i<a.length ; i++)    // Length is the property of array
           System.out.println (a[i]);
  }
}
```

memory allocation ←

Output

10
20
70
40
50

System.out.println (a[3]);

Output

40

or

```
// a[3] = 40
System.out.println (a[3])
```

Output

0 → default value of java

```
int a[] = new int[5];
int []a = new int[5];
int[] a = new int[5];
```

} all are valid.

```
class A2
{
 public static void main (String args[])
   {
      int a[] = {33,3,4,5};  // declaration & instantiation
       // printing array
     for (int i=0 , i<a.length ; i++)
        System.out.println (a[i]);
   }
}
```

Output

33

3

4

5

```
// taking input from command line argument.
class A1
{
 public static void main (String args[])
   {
      int a[] = new int[5];
      for (int i=0;  i<a.length ; i++)
        {
          a[i]= Integer.parseInt (args[i]);
        }
      for (int i=0 , i<a.length ; i++)
         System.out.println (a[i]);
    }
 }
```

```
javac A1.java
java A1 4 7 8 3 2
4
7
8
3
2
```

```
java A1.java
java A1 Java k D L T
error
```

```
java A1 3.3 4.4 3.9 3.8
error
```

* if we input anything other than 'int', it will show error. {number format exception}

```
// input through Scanner
import java.util.*;
class A1
{
 public static void main (String args[])
  {
     int a[] = new int[5];
     Scanner sc = new Scanner (System.in);
    for (int i=0; i< a.length ; i++)
     {
        System.out.println (" Enter the no of "+ i +"th posit");
        a[i] = sc. nextInt( ) ;
      }
     for (int i=0 , i< a.length ; i++)
        System.out. println (a[i]).
    }
  }
```

17.12.24

2D Array

```
//2D Array
import java.util.*;
class A10
{
 public static void main (String args[])
  {
     int arr[][] = new int[3][5] ;
     Scanner sc = new Scanner (System.in) ;
     for ( int i=0 ; i<3 ; i++)
       {
          for( int j=0; j<5 ; j++)
            {
              arr[i][j] = sc.nextInt( ) ;
            }
       }
    for ( int i=0 ; i<3 ; i++)
      {
         for ( int j=0 ; j<5 ; j++)
           {
             System.out.print ( arr[i][j] + "   " ) ;
           }
          System.out.println ( ) ;          //newline
      }
  }
```

taking input

disp. output

filename class A3 ; A3.java ; jagged array

In 2D Array ; if we write arr[3][5]
arr.length gives no of rows.
arr[0].length gives no of columns in 0^{th} row.

```
class A3
{
   public static void main (String[] args)
     {
        int arr[][] = new int[3][];
        arr[0] = new int[3];
        arr[1] = new int[4];
        arr[2] = new int[2];
        // initialization of a jagged array
        int count = 0;
        for (int i=0 ; i<arr.length ; i++)
            for (int j=0 ; j < arr[i].length ; j++)
               arr[i][j] = count ++;
        // printing the data of a jagged array
          for (int i=0; i<arr.length ; i++)
           {
             for ( int j=0 ; j<arr[i].length ; j++)
              {
                System.out.print (arr[i][j] + "   ");
              }
               System.out.println ();
            }
     }
```

Jagged array is an array of array where each row of the array can have a diff. no of columns.

Output :-

```
0   1   2
3   4   5   6
7   8
```

For- each loop ; (only used in array) - (we can traverse)

A4.java

```
class A4
 {
    Public static void main (String args[])
     {
        int arr[] = {33,3,4,5};
        // printing array using for each loop
       for (int i : arr)
       System.out.println(i);
     }
  }
```

for each loop
for 2D Array

Output :-

33 3 4 5

Passing an array to a method.

A5.java

min method will display the min. value of the array.

```
class A5 {
 static void min (int arr[]) {
  int min = arr[0];
  for (int i = 1 ; i<arr.length ; i++)
   if (min > arr[i])
      min = arr[i];
      System.out.println(min);
   }
      public static void main (String args[]) {
      int a[] = {33,3,4,5};
      min(a);                    // passing array to method
     }
  }
```

Anonymous Array in Java :-

↳ no name (separate declarations of array variable)

2 static are present inside the same class; they can call each other.

If we declare a method as static that will be a class method.

without static ; it is instance method.

class method is the property of a class & only one copy of it can be created in the main memory & all object will share.

```
public class A6
{  // creating a method which receives array as parameter.
   static void printArray (int arr[])
     {
       for (int i=0 ; i<arr.length ; i++)
        System.out.println (arr[i]);
      }
       public static void main (String args[])
        {
           printArray (new int [] {10, 22, 44, 66} ; // passing
                                                      anonymous array
                                                       to method
        }
}
```

Output

```
10
22
44
66
```

Method Returns an Array

A7.java

method can also return arrays.

```
class A7
 {
     static int [] get ()
         {   // Creating methods which return an array
           return new int [] {10,30,50,90,60};
         }
           public static void main (String args[])
              {
                  // Calling method which returns an array
                   int arr [] = get ();
                     // printing the values of array
                   for (int i=0; i< arr.length; i++)
                     System.out.println (arr [i]);
                 }
              }
```

Output :-

```
10
30
50
90
60
```

Array Index Out of Bounds Exception :-

arr[5] → index 0 to 4

if we try to access 5th index then this error shows

A8.java

```
public class A8
  {
    public static void main (String args[])
      {
        int arr[] = { 50, 60, 70, 80 };
        for ( int i = 0; i <= arr.length ; i++)
          {
            System.out.println (arr[i]);
          }
      }
  }
```

Output :-

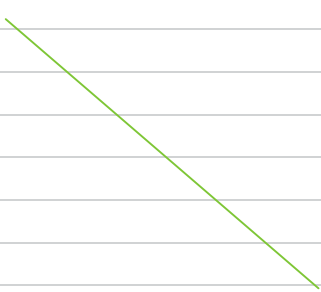
```
50
60
70
80
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException : 4
 at TestArrayException.main (TestArrayException.java:5)
```

A9.java

Initialization of 2D Array

Multidimensional array is an array of array where each element can be an array itself. It is useful for storing data in row & column itself.

```
class A9
{
    public static void main (String args[])
    {           // 3x3 matrix
        int arr[][] = {{1,2,3},{4,5,6},{7,8,9}};
                // printing the 2D array
        for (int i=0; i<3; i++)
        {
            for (int j=0, j<3; j++)
            {
                System.out.println(arr[i][j] + "  ");
            }
            System.out.println();
        }
    }
}
```



A10.java

```
import java.util.*;
class A10
 {
   public static void main (String [] args)
     {
        int arr[][] = new int [3][5] ;
        Scanner sc = new Scanner (System.in);
        for (int i=0 ; i<3 ; i++)
          {
            for ( int j=0; j<5; j++)
              {
                arr[i][j] = sc.nextInt() ;
              }
          }
           for (int i=0 ; i<3 ; i++)
            {
              for (int j=0 ; j<5 ; j++)
                {
                   System.out.println (arr[i][j] +" ");
                }
            }
           }
```

18.12.24

Constructor :- special method used to initialize objects (no return type)

- default constructor (no arguments)
- parameterised constructor

```
Class P8
{
  int a, b;          // instance variable
  string p;          // instance variable
  void display()     // instance method
  {                  // displaying the value of a, b, p.
    Sopln("Value of a : " +a);
    Sopln("Value of b : " +b);
    Sopln("Value of p : " +p);
  }
  public static void main(String arr[])
  {
    P8 ob = new P8();      // Creating object of the class
    ob.display();          // calling ob.display().
  }
}
```

/* Output :-

```
Value of a : 0
Value of b : 0
Value of p : null
```

} constructor allocates the number.

P8 ob = new P8() → a separate memory will allocate to object ob

ob.a
ob.b
ob.p
ob.display

```
class P8
{
  int a, b;          // instance variable
  string p;          // instance variable
  void display()     // instance method
  {                  // displaying the value of a, b, p.
    Sopln("Value of a : " +a);
    Sopln("Value of b : " +b);
    Sopln("Value of p : " +p);
  }
  public static void main(String arc[])
  {
    P8 ob = new P8();        // Creating object of the class
                             --> default constructor
    ob.a = 8;
    ob.b = 9;
    ob.p = "KIIT";
    ob.display();            // calling ob.display().
  }
}
```

*/ Output:-

```
Value of a : 8
Value of b : 9
Value of p : KIIT
```

Constructor will not initialize the local variable.
it can only initialize the instance variable.

```
Class P8
{
  int a, b;        // instance variable
  string p;        // instance variable
  P8()
  {
  }
  void display()   // instance method
  {                // displaying the value of a, b, p.
    Sopln("Value of a : " +a);
    Sopln("Value of b : " +b);
    Sopln("Value of p : " +p);
  }
  public static void main(String arc[])
  {
    P8 ob = new P8();
    ob.display();
  }
}
```

also possible

output :-

Value of a : 0

Value of b : 0

Value of p : null

```
P8()
{
  Sopln("Learn Java");
}
```

Output :-

Learn Java

Value of a : 0

Value of b : 0

Value of p : null

default constructor does its work first.

Static Variable, Static Method, static block

```
class P8
 {
    static int a,b;                    // class variable
    static String p;                   // class variable
    static void display()              // class method
     {
       Sopln("Value of a : " +a);
       Sopln("Value of b : " +b);
       Sopln("Value of p : " +p);
     }
      static                           // static block
      {
        Sopln("KIIT University");
      }
        public static void main(String args[])
          {
             a=9;
             b=7;
             p="KIIT";
             display();
          }
 }
```

Output :-

KIIT UNIVERSITY
Value of a : 9
Value of b : 7
Value of p : KIIT

* static will execute first then main method

Parameterized Constructor

```
class Static -> 3 progs
{
    int a,b;          // instance variable
    String p;         //      "          "
    P9 (int x, int y, String z)
    {
        a = x;
        b = y;
        p = z;
    }
    void display ( )      // instance method
    {
        Sopln ("Value of a : " +a);
        Sopln ("Value of b : " +b);
        Sopln ("Value of p : " + p);
    }
    public static void main ( String args[])
    {
        P9 ob = new P9 (7, 9, "KITT");
        ob.display( );
    }
}
    P9 (int x, String z)
    {
        Sopln (" learn java");
        a = b = x+z;
        p = z;
    }
```

- can display any message from parameterized constructor.

`a = b = x+x;` we can perform any computational part

`P9 (int x, String z) { a = x, p = z; }` } not necessary to pass values to all instance variable.

* If we define any parameterized constructor, we can't use only default constructor.

```
P9 ob = new P9();
```

↳ Compilation error

Compiler will not pass any default constructor value.

* Math is a class in java.lang package

Math class

```
class P10
{
  int a, b;
  P10(int x, int y)
  {
    a=x;
    b=y;
  }
  void display()
  {
    int c = Math.max(a,b);
    Sopln("Largest value is :" +c);
  }
  public static void main (String args[])
  {
    P10 ob = new P10(23,45);
    ob.display();
  }
}
```

// Method Overloading (Compile time polymorphism)

```
class P13
 {
void area (int a)
 {
    int b = a*a;
    Sopln (" Area of square = " + b);
 }
   void area (double r)        // circle
    {
      double b = 3.14*r*r;
      Sopln (" Area of circle = " + b);
    }
①     void area (int a)        // cube
      {
        int b = 6*a*a;
        Sopln (" Area of cube = " + b);
      }
      public static void main (String ar[])
       {
          P13 ob = new P13();
          ob.area (3.5);
          ob.area (5);
②         ob.area (100);
       }
}
```

error correction → refers to cond 1 → void area (long a)

① Output: error

② Output:
Area of circle = 38.465
Area of square = 25
Area of square = 10000

Output /*
Area of circle = 38.465
Area of square = 25
Area of cube = 60000

long → ob.area (100L);
we can also write l.

Cond 1: if no. of arguments are same in every method then their data type must be different.

Cond 2: if data types of all arguments in all methods are same, then these numbers must be different.

Return type doesn't matter; it will show errors

```
class P13
 {
void area (int a)
 {
    int b = a*a;
    Sopln (" Area of square = " + b);
 }
   void area ( double r)        // circle
    {
      double b = 3.14 * r * r;
      Sopln ( " Area of circle = " +b);
    }
       int area ( int a)       // cube
        {
          int b = 6*a*a;
          return b;
        }
          public static void main ( String as[])
           {
              P13 ob = new P13();
              ob. area ( 3.5);
              ob. area ( 5);
               int x = ob. area (100);
               Sopln (" Area of cube is " + x);
           }
        }
```

* In Method overloading return type doesn't matter in Java

Output /*

incompactible types.

Cond 2 :-

if data type same, the no must be diff.

```
Class P13
 {
void area (int a)
 {
    int b = a*a;
    Sopln (" Area of square = " + b);
 }
   void area ( int l, int b )
    {
     int b1 = l*b;
     Sopln ( " Area of rectangle = " +b);
    }
     void area ( int a, int b, int c )        // cuboid
      {
       int b2 = 2*(a*b + b*c + c*a)
       Sopln (" Area of cuboid = " + b);
      }
       public static void main ( String ar[])
        {
          P13 ob = new P13();
          ob.area ( 3 );
          ob.area ( 5, 7 );
          ob.area ( 2, 3, 6);
        }
      }
```

/* Output :-

Area of rectangle = 35

Area of square = 9

Area of cuboid = 52

Constructor Overloading :-

as method overloading is possible in java ; constructor is a special type of method

```
class P15
 {
    P15 ( )
     {
       Sopln ("Default");
     }
   P15 ( int a)
     {
      Sopln (" Value of  a = " + a) ;
     }
     P15 ( float a)
     {
       Sopln (" Value of float a = " + a);
     }
      public static void main ( String as [])
      {
         P15  ob1 = new P15 ( ) ;
         P15  ob2 = new P15 ( 3.6f);   → error
         P15  ob2 = new P15 ( 6.9 f);
         P15  ob3 = new P15  ( 5);
      }
 }
```

Output

```
// P15 ob2 = new P15 ( 3.6f);
Default
Value of float = 6.9
Value of a = 5
```

we can't declare ob2 twice.

corrected version

```
P15  ob1 = new P15 ( ) ;
P15  ob2 = new P15 ( 6.9 f);
P15  ob3 = new P15  ( 5);
```

Uses of this keyword :-

The this keyword in Java is used to refer to the current object or class instance. It's a reserved keyword that's used to access instance variables, invoke methods & constructors & more.

```
class 03
{
  int a, b;
  03 (int a, int b)
   {
     this.a = a;
     this.b = b;
   }
   void show ()
   {
     Sopln ("Value of a = "+a+" Value of b = "+b);
   }
}
 class Main
 {
    public static void main (String ar[])
    {
       03 ob = new 03 (5, 7);
       ob.show ();
    }
 }
```

*/ Output

Value of a=5 Value of b=7

Without this keyword.

Value of a=0

Value of b=0

Types of Nested Classes :-

SUMMARY

1) OOPs

i) Encapsulation - wrapping data & methods in a single unit.

ii) Inheritance - code reusability

iii) Polymorphism - enables object to take multiple forms

iv) Abstraction - hiding details

2) Scanner Class

* used for reading user input (nextLine() for strings & nextInt() for integers.)

3) Command Line Arguments

* inputs passed directly during program execution

4) Array :-

* array can be traversed using loops.
* supports 2D & jagged array
* use 'for each' loop for efficient array traversal.

5) Constructors :-

* used to initialize objects
* constructor overloading allows multiple constructor in a class.

6) Wrapper Class :-

- parseInt() - converts string to integer
- parseDouble() - converts string to double

7) constructor have no return type.

8) In method overloading return type doesn't matter. (if data types are same, the number must be different)