LEXICAL ISSUES :

1 WHITE SPACES : > normal space character
> new line character
> tab character

2 IDENTIFIERS > rules for naming variables

3 LITERALS > string consts, numeric constants

4 COMMENTS // single line /* multiline */ documentation comment (Html) /** */

5 SEPERATORS ( ) { } [ ] ; , .

6 KEYWORDS built in words, which are predefined 50 keywords
3 keywords for boolean variables (values)
→ True
→ False
→ Null

7 JAVA LIBRARIES similar to header files in C
They are a collection of predefined function

SCANNER CLASS IN JAVA

java.util package is used for obtaing inputs of primitive types

> To create an object of scanner class we pass the predefined object in System.in

> To read integers we use .nextInt()
.nextDouble()

```
// declares an array of integers
  int arr[];
// allocating memory of 5 integers
  arr = new int [5];
        ↳ keyword

  length = arr.length
```

else if statements are more efficient

$\underbrace{\text{import}}_{\text{importing keyword}}$ java.$\overbrace{\text{util}}^{\text{Package Name}}$.$\underbrace{\text{Scanner}}_{\text{class Name}}$

import java.util.$\underbrace{*}_{\text{wild card}}$

Indicates that we are gonna use all the classes of the utility library

$\underbrace{\text{Scanner}}_{\text{name of class}}\ \underbrace{\text{sc}}_{\text{obj name}} = \text{new}\ \underbrace{\text{Scanner(System in)}}_{\text{constructor (same as name of class)}}$

A class is a blueprint from which individual objects are created

> Specifying a class doesn't create any objects of that class
(In the same way specifying a structure in C doesnt create any variable

> To create objects we need keyword new

> The object variable type must be same as class

Creating an object is called instating it. INSTANTIATING

OBject is an instance of a class
The new operator instantiates a class by allocating memory for new object & returning a reference to that memory
also invokes object constr
Methods in JAVA is like functions in C

* The programme starts an runs the main method
* Anything outside the main method should be called first.

Jvm - is an additional layer between the language & OS which makes it platform independent.

static - With this keyword we need not instantiate main
Main is the only method without instantiating the object

void -> doesnt return any argument

string args[] -> for command line prompt

$$\begin{tikzcd}
\text{Program} \arrow[r, "\text{compiler}"] & \text{Bytecode} \arrow[r, "\text{intreprefor / executing}"] \arrow[d] & {} \\
& \text{platform indpen} &
\end{tikzcd}$$

no IDE and go through command line.

How to debug code using break points

Segmentation fault error

Java Constructor: A constructor in java is a method used to initialise objects

> constructor has same name as class

> invoked when obj of class is called
directly creates and initialises.

Try Catch In Java

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block

```
try {
    // statements that may cause a exception
}
catch (exception (type)
 {
 }
```

QUIZ 1 24/6/18

WHAT IS ENCAPSULATION

wrapping data & code to secure them from outside interface

HOW INHERITANCE IS USEFUL FOR PROGRAMMING

Allows only authorised users through class & obj concept

(reusing methods & data from some other class super class

POLYMORPHISM SUPPORTED IN C++ True

HOW MAIN FUNCTION OF JAVA CLASS is GETTING EXECUTED WITHOUT INSTANTIATION

only function which runs without instanting an obj
it's done automatically by th compl

OBJECTS AS PARAMETERS

```
class Box
{
double d, w, h;
Box (Box ob)
  { d = ob.d;
  w = ob.w;
  h = ob.h;
  }
Box(double a, double b, double c)
  {
   d = a;
  w = b;
  h = c;
 }
}
class new
{ public static void main(String args[])
 { Box mybox = new Box(1.1, 2.2, 3.3);
  Box myclone = new Box(mybox);
```

ACCESS SPECIFIERS

3 types

→ public - can be accessed by anybody

→ private - only by public methods inside the class

→ protected only in derived class

An acess specifier is a defining code element that can determine which elements of a program are allowed to acess a specific variable or other piece of data

Constructor Overloading

```
class Box
{ double w,d,h;
  Box(double a,double b, doublec)
   { w = a;
     d = b;
     h = c; }
  Box ( ) ·
  { w = -1 ; d = -1 ; h = -1; }
  Box (double len)
   { w = d = h = len; }
 double vol ( )
  { return w*d*h;
 } }
```

INTERFACES

> Interface looks like a class but it isn't

> It can have methods & variables just like class but methods declared are abstr

> used for full abstraction

> we can have more than one interface in class

> OBjects as arguments

> OBjects as parameters

Method overloading is a feature that allows a class to have more than one method having the same name

1) number of parameters add (int int) add (int int int)

2) data type of parameters add (int , int)
add (int float)

3) sequence of data type add (int , float)
add (float , int)

4) return types static or void

Constructor overloading

> Having more than one constructor with different parameter such that each constructor performs a different task

Constructors :

> A constructor is a method name of constructor is name of cls.
> when it is created constructor is automatically called
> no return value
> It is a block of code similar to other methods but it is called when an $\underset{\text{object}}{\underset{\downarrow}{\underline{\text{instance}}}}$ of that class is created

instantiation is the process of creating a class

default constructor

→ is included to the class which doesnot have a constructor when it is coming for compilation

```
class a
{ int b;
  a() { };
  void c()
  { SOP( a); }
```

11/8/15

Destructor

purpose :
The resources to be released when the object is no-long in use .

↓
Java System

* garbage collection ( ) (inbuilt method executed method)
  ↕
  periodically java collects
  ↓
  for objects not in use
  it will not allocate resource

We cannot call a destructor

Destructor will run at system level user will not know

```
class test
{ int a.
  public int b, // same as int b,
  private int c;
  void setc (int i)
  { c=i;}
  int getc ()
  { return c,
  }}
```

```
class testnow
{
public static void main
                (String args[])
{
test ob - new test(),
 ob.a = 10,
 ob.b = 20,
 ob.c = 30 , // error
 ob.setc(30),
a: ob.getc ()
S.O.P ("c"+a),
 }}
```

ARRAY OF OBJECTS :

```
class employee
 { int a,b,c;
   void Init ()
   { a=10,
     b = 20,
   }
 void add ()
   {
    c = a+b,
    SOP ("c = "+c).
  }}
class my
{
public static void main (String args[])
 { employee [] emp = new employee[3] ;
```

employee [] → away; emp → name; [3] ↳ square bracket mentioning the size

```
emp[0] = new employee();
emp [1] = new employee () ;
emp [2] = new employee () ;
```

17/7/18

METHOD OVERLOADING

Attributes of a method
→ name
→ parameters
→ return type

method name need not be unique
we can differenciate between them
with parameters and return type

```
int add()
int add(int a)
void add( double a, double)
```

polymorphism is acheivable by
method overloading

```
class over
{void test()
  {SOP ("No Param"); }
 void test (int a)
  {SOP ("a ="+a); }
 void test (int a, int b)
  {SOP (" a + b= " +(a+b));}
 double test (double a)
  {SOP (" double a =" +a);
   return a;
  }
class overloaddemo
{public static void main (String args [ ])
   over O = new over();
   double res;
   O.test();
   O.test (10);
```

```
O/P
No param
a =10
a+b= ?
double a= 12.25
```

Quiz 3

> LIST THE BITWISE LOGICAL OPERATORS :
|, &, ^, >>, << To perform binary operation on integer

> LIST THE BOOLEAN LOGICAL OPERATORS
&&, ||, != true, false condition

> GIVE AN EXAMPLE FOR TERNARY OPERATOR
(a>b)? c:d

> IF ELSE IF LADDER

> DIFFERENCIATE BREAK & C
> Jump states → in loop or switch
> when the control encounters break it comes out of the loop & stops the execution there in the loop
> when control encounters continue. it goes to the condition again

---

Object Oriented System Development OOSD

SOFTWARE ENGINEERING

> Methodology for developing computer based systems 2Q

$\rightarrow$ Analysis $\xrightarrow{\text{output}}$ algorithm,
$\rightarrow$ Design $\xrightarrow{\text{output}}$ flowchart
$\rightarrow$ Coding $\xrightarrow{\text{output}}$ implementation
$\rightarrow$ Testing $\xrightarrow{\text{output}}$ sample data samples are taken
$\rightarrow$ Maintainence

TRADITIONAL	OO 3Q 4Q
→ It uses the software as a collection of programs with isolated data	→ It views the software as the set of objects which combines the functionalities and the data, The fundionalities are achieved by the programs.
→ Programs focusses on the functionality of the system	An approach to system development that uses system objects to build new systems & rebuild ones