Page No.
Date

* Basics of Java Programming *

* Why we are here?

- Make Learning online.
- New technology in industry.
- know about new techie mind and what they thing about coding, programming & language.
- Make you easy about software company.

* A brief history of Java.

- "Java, whose original name was Oak, was developed as a part of the Green project at Sun. It was started in December '90 by Patrick Naughton, Mike Sheridan and James Gosling and was chartered to spend time trying to figure out what would be the "next wave" of computing and how we might catch it.

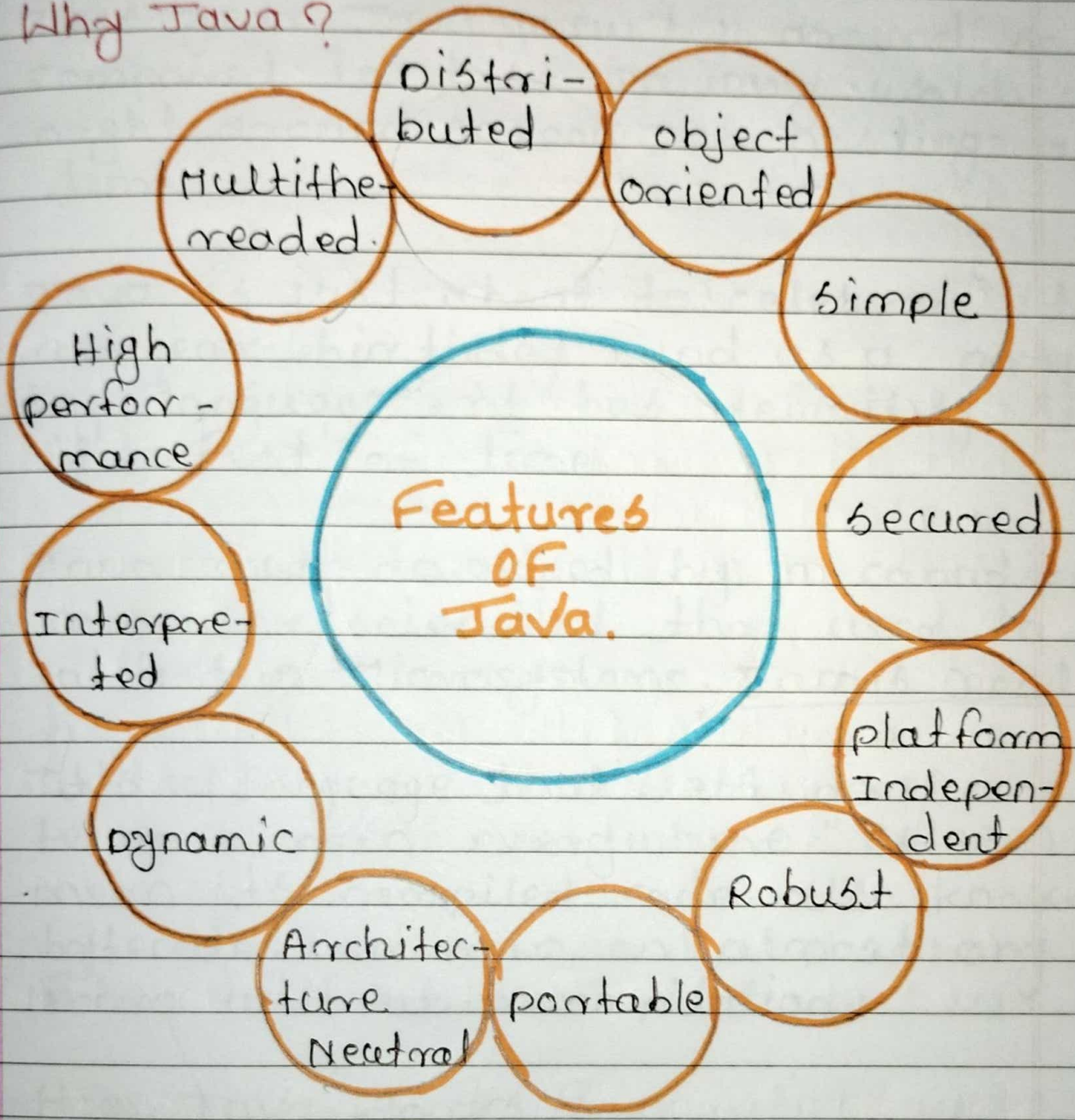
Page No.
Date

When Java ?

Version	Date.
JDK 1.0	January 23, 1996
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May, 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11 2006
Java SE 7	July 28, 2011
Java SE 8	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11	September 25, 2018
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020

Page No.
Date

Why Java ?



Page No.
Date

- This year Java grew by around 60% compared to last January, which was right around 62,000 Job postings at the time.

- Java is just about to celebrate its 24-year birthday, and as a programming language, it has definitely stood the test of time.

- Java was developed by a canadian computer scientist that used to work with Sun Microsystems, James Gosling

- It's a language that lets developers "write once, run everywhere;" (WORA), which mean its compiled code, also known as bytecode, can run on almost any platform without recompilation.

How Java Works ?

- Java's platform independence is achieved by the use of the Java virtual Machine (JVM)

- A java program consists of one or more files with a .java extension.

Page No.
Date

- When a java program is compiled the .java files are fed to a compiler which produces a .class file for each .java file
- The .class file contains java bytecode
- Bytecode is like machine language, but it is intended for the Java virtual Machine not a specific chip such as a Pentium or PowerPC chip
- To run a java program the bytecode in a .class file is fed to an interpreter which converts the byte code to machine code.
- Some people refer to the interpreter as "The Java virtual Machine" (JVM)
- The interpreter is platform specific because it takes the platform independent bytecode and produces machine language instructions for a particular chip.
- So a Java program could be run an any type of computer that has a JVM written for it.
  - PC, Mac, Unix, Linux, BeaOS, Spare

Page No.
Date

So What!

- The platform independence of Java may be a huge marketing tool, but is actually of little use to people learning object oriented Programming and Abstract Data Types.

- Java is a "pure" object oriented language

  - encapsulation, inheritance, and polymorphism.

  - all code must be contained in a class

Hello World.java Program

```
Public class HelloWorld
{
  Public static void main (String [] args)
  {
    System.out.println (" Hello World! ");
  }
}
```

Where Java?

- Notes:- year after year, Java is the #1 most searched programming language.

Page No.
Date

* Tools and Technology used in class:-
  - JDK (1.8 or above version)
  - Notepad
  - Eclipse Mars
  - Netbeans
* How to make Java as a hobby?
  - Start from Simple
  - co-Related it to day to day life.
  - once start, Finish any how!
  - Momentum slow but steady.

Basic features

- Data Types
  - primitives
  - classes / objects
- Expressions and operators
- control structures
- Arrays
- Methods
- Programming for correctness
  - Pre and post conditions
  - assertions

* Identifiers in Java :-

- Letters, digits, _, and $ (don't use $. can confuse the runtime system)
- Start with letter, _, or $
- By convention:

1) Start with a letter

2) variables and method names, lowercase with internal words capitalized e.g. honkingBigVariableName.

3) constants all caps with _ between internal words e.g. ANOTHER_HONKING_BIG_INDENTIFIER

4) classes start with capital letter, internal words capitalized, all other lowercase e.g. HonkingLongClassName.

* Data Types :-

- Primitive Data Types :-

  - byte short int long float double boolean char

  - stick with int for integers, double for real numbers.

- Classes and Objects :-

  - pre defined or user defined data types consisting of constructors, methods, and fields. (constants and fields (variables) which may be primitive or objects.)

Page No.
Date

Java Primitive Data Types:-

Data Type	characteristic	Range
byte	8 bit signed integer	-128 to 127
short	16 bit signed integer	-32768 to 32767
int	32 bit signed integer	-2,147,483,648 to 2,147,483,647
long	64 bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32 bit floating point number	±1.4E-45 to ±3.4028235E+38
double	64 bit floating point number	±4.9E-324 to ±1.79769313486231570E+308
boolean	true or false	NA, note Java booleans cannot be converted to or from other types
char	16 bit, unicode	unicode character, \u0000 to \uffff can mix with integer types.

Page No.
Date

What are classes and objects?

- class is synonymous with data type
- object is like a variable
  - The data type of the object is some class

    Referred to as an instance of a class
- classes contain :-
  - The implementation details of the data type
  - and the interface for programmers who just want to use the data type.
- objects are complex variables
  - usually multiple pieces of internal data
  - various behaviors carried out via methods.

Page No.
Date

Program :-

```
class calculation {// class
   int a = 10;
   int b = 20;
   void add() {
   int add = a+b;
    System.out.println("Addition of two nu-
                        mber" + add);
  }
public static void main(String args[]){
  calculation obj = new calculation(); //
   object creation -> instance of class

  System.out.println("value of 1st number:"
                                 +obj.a);
   obj.add();
  }
}
```

output :- value of 1st number : 10
Addition of two number: 30

Creating and Using Objects :-

- Declaration - Data Type identifier
  Rectangle r1;
- Creation - new operator and specified constructor
  r1 = new Rectangle ();
  Rectangle r2 = new Rectangle ();
- Behavior - via the dot operator
  r2.setSize (10, 20);
  String s2 = r2.toString ();

Built in Classes :-

- Java has a large built in library of classes with lots of classes with lots of useful methods
- ones you should become familiar with quickly.
- String
- Math
- Integer, character, Double.

- Systems
- File
- Arrays
- object
- Scanner
- Random
- Look at the Java API Page

Import :-

- Import is a reserved word
- packages and classes can be imported to another class
- Does not actually import the code (unlike the c++ include preprocessor command)
- Statement outside the class block

```
import java.util.ArrayList;
import java.awt.Rectangle;
public class Foo {
        // code for class Foo
}
```

- can include a whole package

```
import java.util.*;
```

- or list a given class
  - import java.util.Random;
- Instructs the compiler to look in the package for types it can't find defined locally
- The java.lang.* package is automatically imported to all other classes.
- Not required to import classes that are part of the same project in Eclipse.

The String class:-

- String is a standard Java class
  - a whole host of behaviors via methods
- also special (because it used so much)
  - String literals exist (no other class has literals)

```
String name = "Mike D.";
```

  - String concatenation through the + operator

```
String firstName = "Mike";
String lastName = "Scott";
String wholeName = firstName +
                   lastName
```

- Any primitive or object on other side of + operator from a string automatically converted to string.

★ Standard output :-

- To print to standard output the use

```
System.out.print(expression); // no newline
System.out.println(expression); // newline
System.out.println(); // just a newline
```

common idiom is to build up expression to be printed out

```
System.out.println("x is:" + x + " y is:" + y);
```

★ Constants :-

- Literal constants - " the way you specify values that are not computed and recomputed, but remain well, constant for the life of a program."

- true, false, null, 'c', "c++", 12, -12, 12.12345

- Named constants

- use the keyword final to specify a constant.

- Scope may be local to be a method or to a class

- By convention any numerical constant besides -1, 0, 1 or 2 requires a named constant

```
final int NUM_SECTIONS = 8;
```

* Program code:-

```
class DemoString{
    final static int age = 10;
public static void main (String args[])

    String firstName = "Tom";
    String lastName = "Jerry";

// age = 20; -> it will create error if we try this to change, due to constant

    String wholeName = firstName + lastName;
    System.out.print ("Whole Name:");
    System.out.println (wholeName);
}
}
```

Output:- Whole Name: TomJerry

Page No.
Date

* Operators :-

- Basic Assignment : =
- Arithmetic Operators : +, -, *, /, % (remainder)
  - integer, floating point, and mixed arithmetic and expressions.
- Assignment operators : +=, -=, *=, /=, %=
- increment and decrement operators : ++,

* Expressions :-

- Expressions are evaluated based on the precedence of operators
- Java will automatically convert numerical primitive data types but results are sometimes surprising.
  - take care when mixing integer and floating point numbers in expressions.
- The meaning of an operator is determined by its operands.

* Casting :-

- Casting is the temporary conversion of a variable from its original data type to some other data type.

- Like being cast for a part in a play or movie.

- With primitive data types if a cast is necessary from a ~~class~~ less inclusive data type to a more inclusive data type it is done automatically.

```
int x = 5;
double a = 3.5;
double b = a * x + a / x;
double c = x / 2;
```

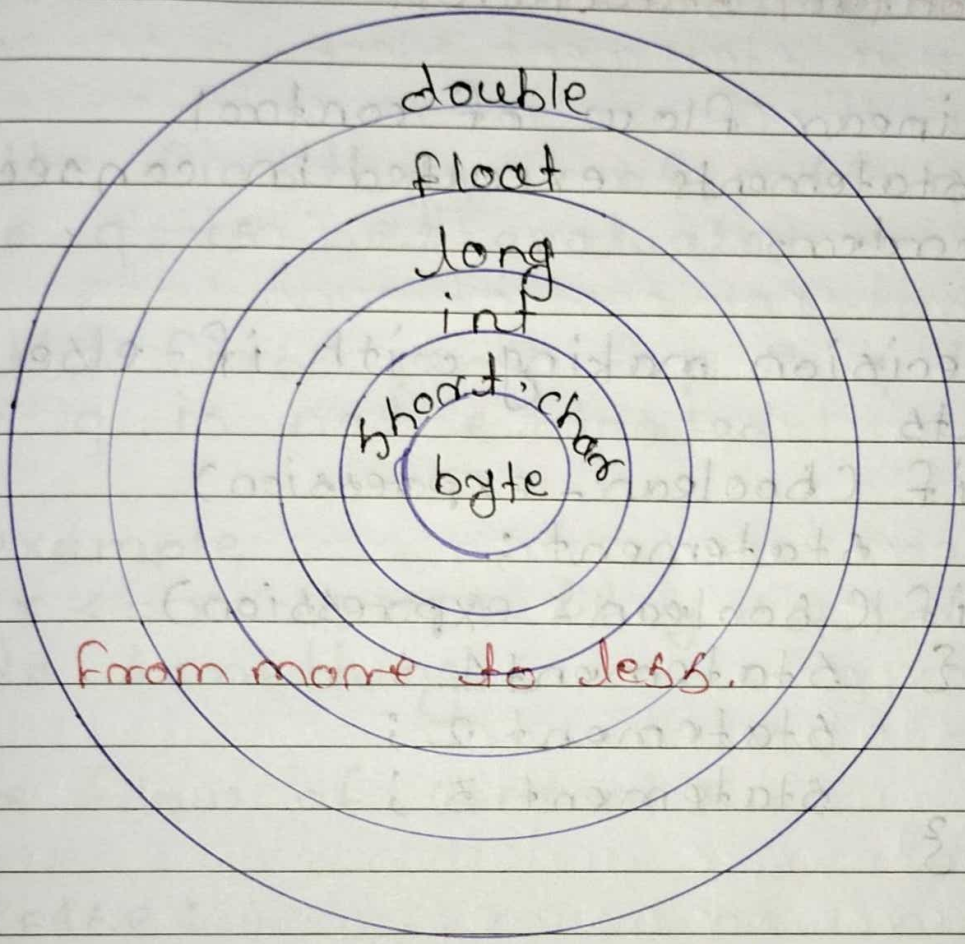
- if a cast is necessary from a more inclusive to a less inclusive data type the class must be done explicitly by the programmer.

- failure to do so results in a compile error.

```
double a = 3.5, b = 2.7;
int y = (int) a / (int) b;
y = (int) (a / b);
y = (int) a / b ; // Syntax error
```

Page No.
Date

Primitive casting.



- outer ring is most inclusive data type. Inner ring is least inclusive.
- In expressions variables and sub expressions of less inclusive data types are automatically cast to more inclusive.
- If trying to place expression that is more inclusive into variable into variable that is less inclusive, explicit cast must be performed.

Page No.
Date

Control structures:-

- linear flow of control
  - statements executed in consecutive order.

- Decision making with if-else statements

```
if (boolean-expression)
    statement;
if (boolean-expression)
{   statement1;
    statement2;
    statement3;
}
```

- A single statement could be replaced by a statement block, braces with 0 or more statements inside.

Boolean Expressions:-

- Boolean expressions evaluate to true or false.
- Relational operators : >, >=, <, <=, ==, !=
- Logical operators : &&, ||, !

- && and || cause short circuit evaluation
- if the first part of p && q is false then q is not evaluated.
- if the first part of p || q is true then q is not evaluated.

```
// example
if (x <= X_LIMIT && y <= Y_LIMIT)
// do something.
```

More flow of control :-

- if-else :

```
if ( boolean-expression)
    statement1;
else
    statement2;
```

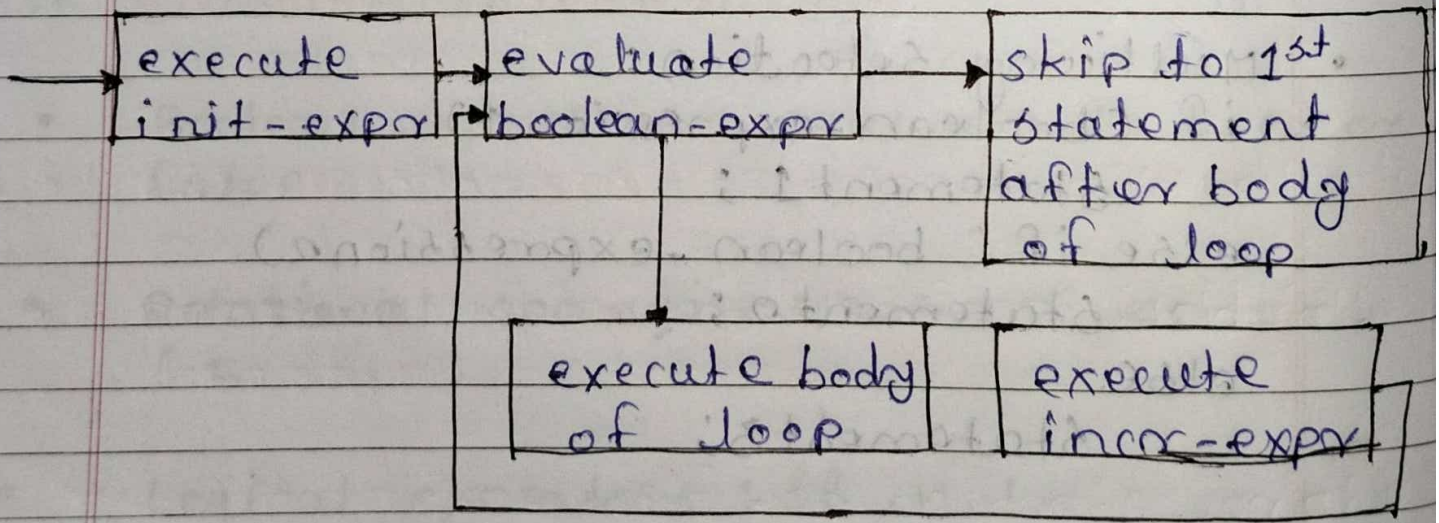
- Multiway selection :

```
if (boolean-expression 1)
    statement1;
else if ( boolean-expression2)
    statement2;
else
    statement3;
```

- individual statements could be replaced by a statement block, a set of braces with 0 or more statements.
- Java also has the switch statement, but not part of our subset.

For Loops:-

- For loops
  for (init-expr; boolean-expr; incr-expr)
  statement;
- init-expr and incr-expr can be more zero or more expressions or statements separated by commas
- statement could be replaced by a statement block.



Page No.
Date

While Loops :-

- While loops
  while (boolean-expression)
      statement ; // or statement block
- do-while loop part of language
  do
      statement;
  while (boolean-expression);
- Again, could use a statement block
- break, continue, and labeled breaks
- referred to in the Java tutorial as branching statements

keywords to override normal loop logic

use them judiciously (which means not much)

Attendance question 4 :-

What is output by the following Java code?

```
int x = 3;
double a = x / 2 + 3.5;
System.out.println(a);
```

1) a
2) 5
3) 4.5
4) 4
5) 5.0

Arrays :-

Arrays in Java :-

- Java has built in arrays. native arrays
- arrays hold elements of the same type
  - primitive data types or classes
  - space for array must be dynamically allocated with new operator.

```
Public void arrayExamples ()
{ int [] intList = new int [10];
  for (int i = 0 ; i < intList.length; i++)
  { assert 0 >= i && i < intList.length;
    intList [i] = i * i * i;
  }
  intList [9] = intList [4] * intList [3];
}
```

Array Details :-

- all arrays must be dynamically allocated
- arrays have a public, final field called length
  - built in size field, no separate variable needed.

- don't confuse length (capacity) with elements in use.

- elements start with an index of zero, last index is length - 1

trying to access a non existent element results in an ArrayIndexOutofBounds-Exception (AIOBE)

Array Initialization :-

- Array variables are object variables
- They hold the memory address of an array object.
- The array must be dynamically allocated
- All values in the array are initialized (0, 0.0, char 0, false, or null)
- Arrays may be initialized with an initializer list:

```
int[] intList = {2, 3, 5, 7, 11, 13};
double[] dList = {12.12, 0.12, 45.93};
String[] sList = {"Olivia", kelly",
                  "Isabelle"};
```

Page No.
Date

Arrays of objects :-

- A native array of objects is actually a native array of object variables.
  - All object variables in Java are pointers!

Array Utilities :-

- In the Arrays class, static methods
- binarySearch, equals, fill, and sort methods for arrays of all primitive types (except boolean) and arrays of objects.
  - overloaded versions of these methods for various data types.
- In the System class there is an arraycopy method to copy elements from a specified part of one array to another.
  - can be used for arrays of primitives or arrays of objects.

Page No.
Date

2D Arrays in Java:-

- Arrays with multiple dimensions may be declared and used

```
int [] [] mat = new [3] [4];
```

- the number of pairs of square brackets indicates the dimension of the array.
- by convention, in a 2D array the first number indicates the row and the second the column.
- Java multiple dimensional arrays are handles differently than in many other programming languages.

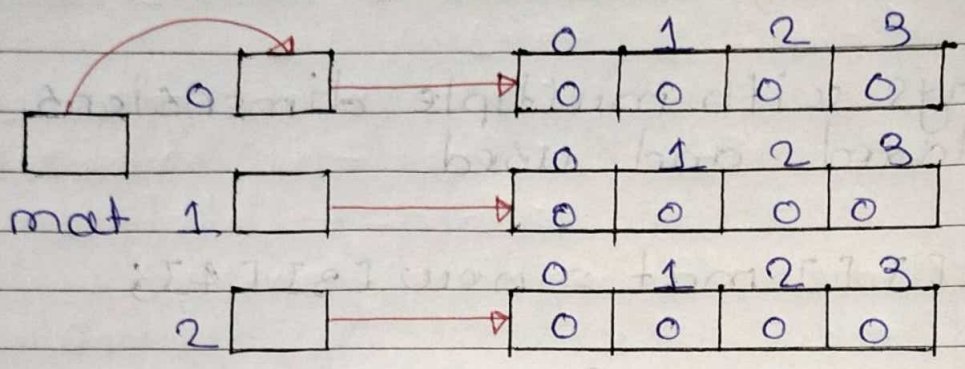
Two Dimensional Arrays :-

row \ column	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

- This is our abstract picture of the 2D array and treating it this way is fine.

```
mat [2] [1] = 12;
```

The Real Picture:-



- mat holds the memory address of an array with 3 elements. Each element holds the memory address of an array of 4 ints.

Arrays of Multiple Dimensions:-

- Because multiple dimensional arrays are treated as arrays of arrays of arrays.... multiple dimensional arrays can be ragged.

- each row does not have to have the same number of columns.

```
int [] [] raggedMat = new int [5] [];
for (int i = 0; i < raggedMat.length; i++)
    raggedMat [i] = new int [i + 1];
```

- each row array has its own length field.

Page No.
Date

Enhanced for Loop :-

- New in Java 5.0
- The for-each loop
- useful short hand for accessing all elements in an array (or other types of structures) if no need to alter values
- alternative for iterating through a set of values
- for (Type loop-variable : set-expression)
  statement
- logic error (not syntax error) if try to modify an element in array via enhanced for loop.

Examples :-

1)
```
Public static int sumListold (int [] list)
{  int total =0;
   for (int i =0; i< list.length; i++)
   { total += list [i];
     System.out.println (list [i]);
   }
   return total;
}
```

2) public static int sumListEnhanced
(int [] list)
{ int total = 0;
for (int val : list)
{ total += val;
System.out.println (val);
}
return total;
}

Attendance question 5 :-

What is output by the code to the right when method d1 is called?

A. 322 B. 323 C. 363
D. 366 E. 399 F.

```
public void d2 (int x) {
    x *= 2;
    System.out.println (x);
}

public void d1 () {
    int x = 3;
    System.out.print (x);
    d2 (x);
    System.out.print (x);
}
```

Ans :-

Page No.
Date

Attendance question 6 :-

What is output by the code to the right ?

A. output will vary from one run of program to next.

B. 00

C. 363

D. 7 then a runtime error

E. No output due to syntax error.

```
int [] list = {5, 1, 7, 3};
System.out.print( list[2]);
System.out.print (list[4]);
```

Ans:→ 7 then a runtime error.

Page No.
Date

Methods:-

- Methods are analogous to procedures and functions in other languages.
  - local variables, parameters, instance variables
  - must be comfortable with variable scope: where is a variable defined?
- Methods are the means by which objects are manipulated (objects state is changed) much more on this later.
- Method header consists of
  - access modifier (public, package, protected, private)
  - static keyword (optional, class method)
  - return type (void or any data type, primitive or class)
  - method name
  - parameter signature

* Static Methods :-

- The main method is where a stand alone Java program normally begins execution.

- common compile error, trying to call a non static method from a static one

```
public class StaticExample{
  public static void main(String[] args)
{ //starting point of execution
  System.out.println("In main method");
   method1();
   method2(); //compile error;
}
public static void main()
{ System.out.println("method 1"); }
public void method2()
{ System.out.println("method 2"); }
}
```

* Method Overloading and Return:-

- A class may have multiple methods with the same name as long as the parameter signature is unique

- may not overload on return type

- Methods in different classes may have same name and signature
  - this is a type of polymorphism, not method overloading
- If a method has a return value other than void it must have a return statement with a variable or expression of the proper type.

  Multiple return statements allowed, the first one encountered is executed and method ends

* Method Parameters :-

- A method may have any number of parameters
- Each parameter listed separately
- No VAR (pascal), &, or const & (C++)
- Final can be applied, but special meaning
- All parameters are pass by value

* Value Parameters vs. Reference Parameters

- A value parameter makes a copy of the argument it is sent.

  - changes to parameter do not affect the argument

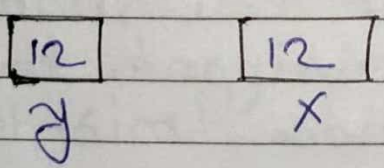
- A reference parameter is just another name for the argument it is sent.

  - changes to the parameter are really changes to the argument and thus are permanent.

* Value vs. Reference :-

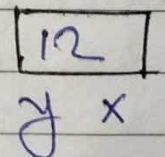
```
// value
void addTo(int x)
{ x += 10; }

void calls()
{ int y = 12;
  addTo(y);
  // y = ?
}
```




```
// c++, reference
void addTo(int& x)
{ x += 10; }

void calls()
{ int y = 12;
  addTo(y);
  // y = ?
}
```



Page No.
Date

* Assertions :-

- Assertions have the form
  assert boolean expression : what to output if assertion is false.

- Example

```
if ( (x<0) || (y<0) )
{ // we know either x or y is <0
  assert x<0 || y<0 : x + " " + y;
  x += y;
}
else
{ // we know both x and y are not less than zero
  assert x>=0 && y>=0 : x + " " + y;
  y += x;
```

- Use assertion liberally in your code
  - part of style guide

Page No.
Date

* Feature: object oriented Programming :-

* object-oriented Programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.
- Grady Booch.

Terms
- object
- class
- Abstraction
- Encapsulation
- Inheritance

* Object :-

- A thing in a real world that can be either physical or conceptual. An object in object oriented programming can be physical or conceptual.
- conceptual objects are entities that are not tangible in the way real-world physical objects are.

Page No.
Date

- Bulb is a physical object. While college is a conceptual object.
- conceptual objects may not have a real world equivalent. For instance, a stack object in a program.
- A bulb has state and behavior and that is switch on & switch off.

* Attributes and operations :-

- The object's state is determined by the value of its properties or attributes.
- Properties or attributes → member variables and /or data members
- The object's behaviour is determined by the operations that it provides
- operations → member functions or methods

* class :-

- A class is a construct created in object-oriented programming languages that enables creation of objects.

Page No.
Date

- Also sometimes called blueprint or template or prototype from which objects are created

- It defines members (variables and methods)

- A class is an abstraction.

* Abstraction :-

* Abstraction denotes essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perpective of the viewer.
- Grady Booch.

- Abstraction is the process of taking only a set of essential characteristics from something.

- Example
- For a Doctor → you are a patient
- Name, Age, old medical records
- For a Teacher → you are a student
- Name, Roll Number / Reg No, Education background.

Page No.
Date

* class and Encapsulation :-

Student	Rectangle	LinkedList
• -roll : long	• -length : int	• - Node
• -name : String	• -width : int	• + addFirst (Node n)
• + display() : void	• + area() : int	• + add (Node n, int pos)
• + read() : bool		• + remove (Node n)

- : private
+ : public.

* Reuse in object oriented Language :-

- Object oriented Languages also implements reuse in the same way that we do in real life.

- Using
  - has-a
  - is-a

- Has-a or composition relationship is implemented by having a class having another class as its member, or rather an object having another object as its member.

  - car has a stereo
  - college has Teachers and students.

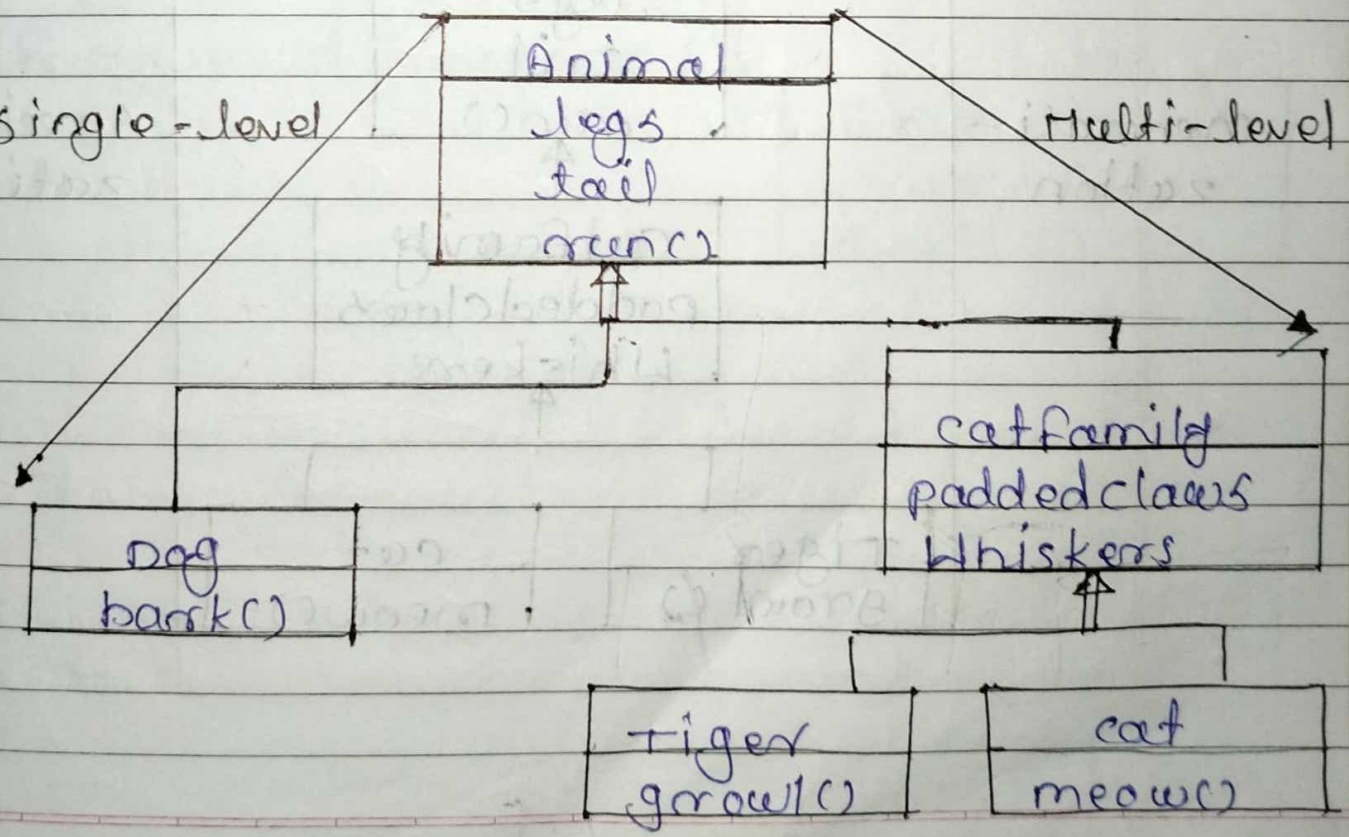
- Is-a is implemented through what we call inheritance relationship

* Inheritance :-

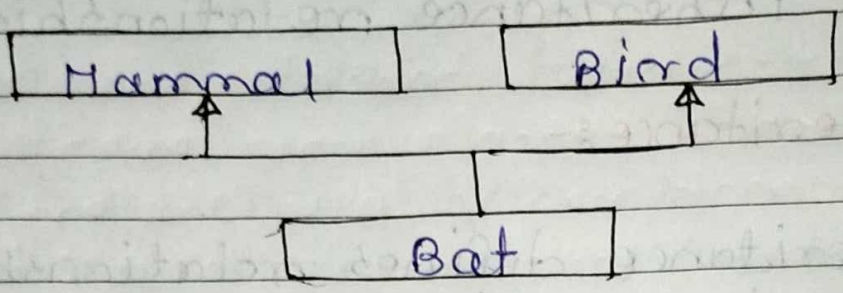
* Inheritance defines relationship among classed, wherein one class share structure or behaviour defined in one or more classes.
- Grady Booch

- Defines Is-A relationship between classes
- cat Is-A Animal
- car Is-A Vehicle
- Rose Is-A Flower

* Inheritance Hierarchy :-



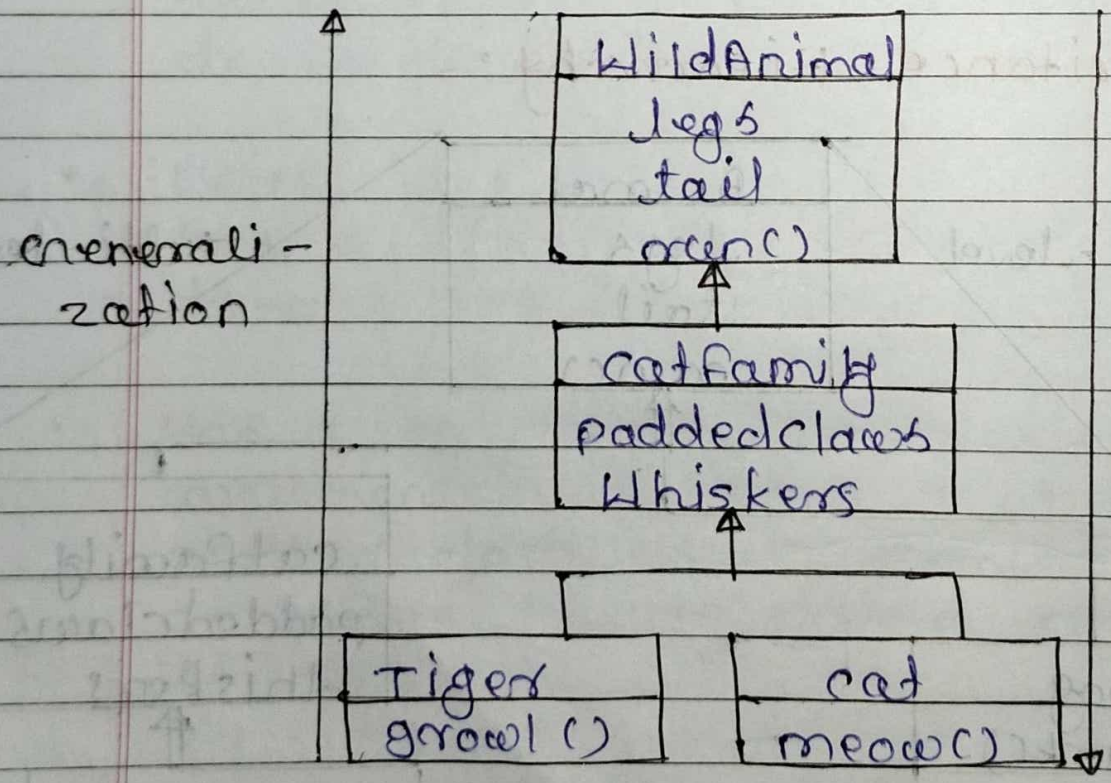
* Multiple Inheritance.



- Many object-oriented languages does not support this type of inheritance.

Java, c# are the examples of object-oriented language that does not support multiple inheritance through classes.

* Generalization and specialization:-

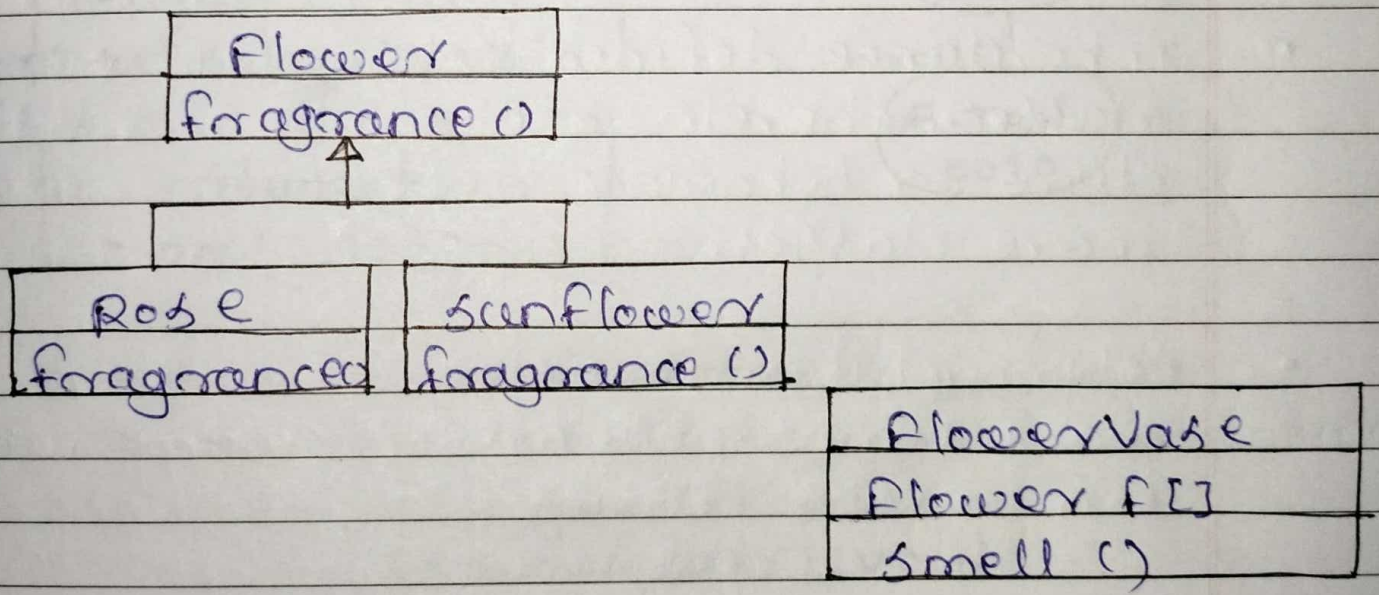


Page No.
Date

* Polymorphism :-

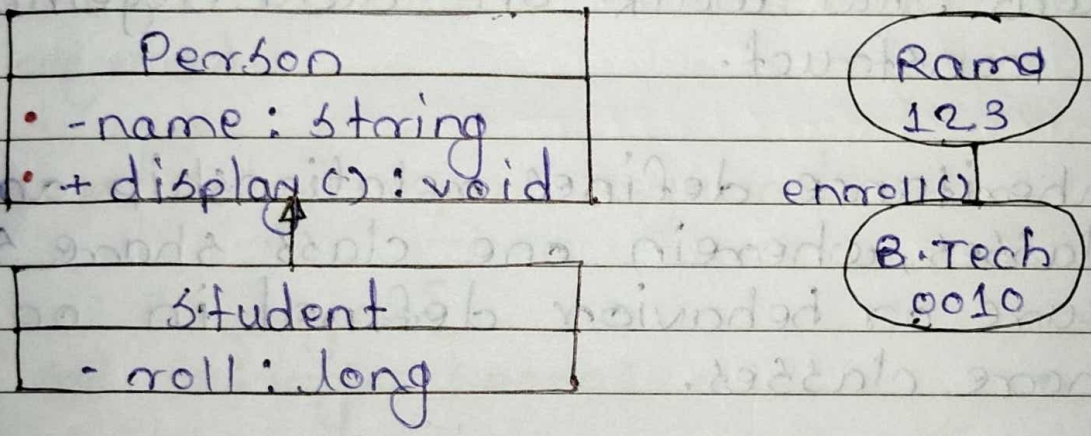
* A concept in type theory, according to which a name (such as a variable declaration) may denote objects of many different classes that are related by some common superclass; thus, any object denoted by this name is able to respond to some common set of operations in different ways.
- Grady Booch.

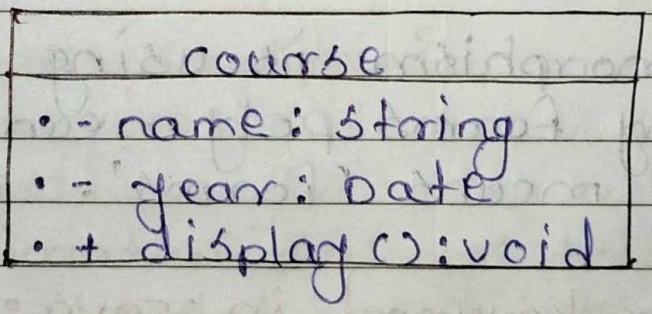
* polymorphism Example :-



* Revisiting definition :-

object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.





* Summary :-

- object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects.

- The object's state is determined by the value of its properties and its behavior is determined by the operations that it provides.
- Abstraction is the process of taking only a set of essential characteristics from something.
- Encapsulation is binding data and operations that work on data together in a construct.
- Inheritance defines relationship among classes, wherein one class share structure or behavior defined in one or more classes.
- Polymorphism is using a function in many forms. poly means 'many', morphism means 'forms'.

* Super keyword in Java :-

- Super keyword is a reference variable that is used for refer parent class object.
- Super keyword is mainly used at three level in java.

1) At variable level
2) At method level
3) At constructor level.

* Why use super keyword in java?

- Whenever inherit base class data into derived class it is chance to get ambiguity, because may be base class and derived class data are same so to difference these data need to use super keyword.

* Example of super keyword :-

```
class Employee{
float salary = 10000;
}
class HR extends Employee{
float salary = 20000;
void display(){
System.out.println("salary:" +super.salary);
                    // print base class salary
}
}
class supervariable{
public static void main (String[] args){
HR obj = new HR();
obj.display();"
}                          output :- salary : 10000.0
}
```

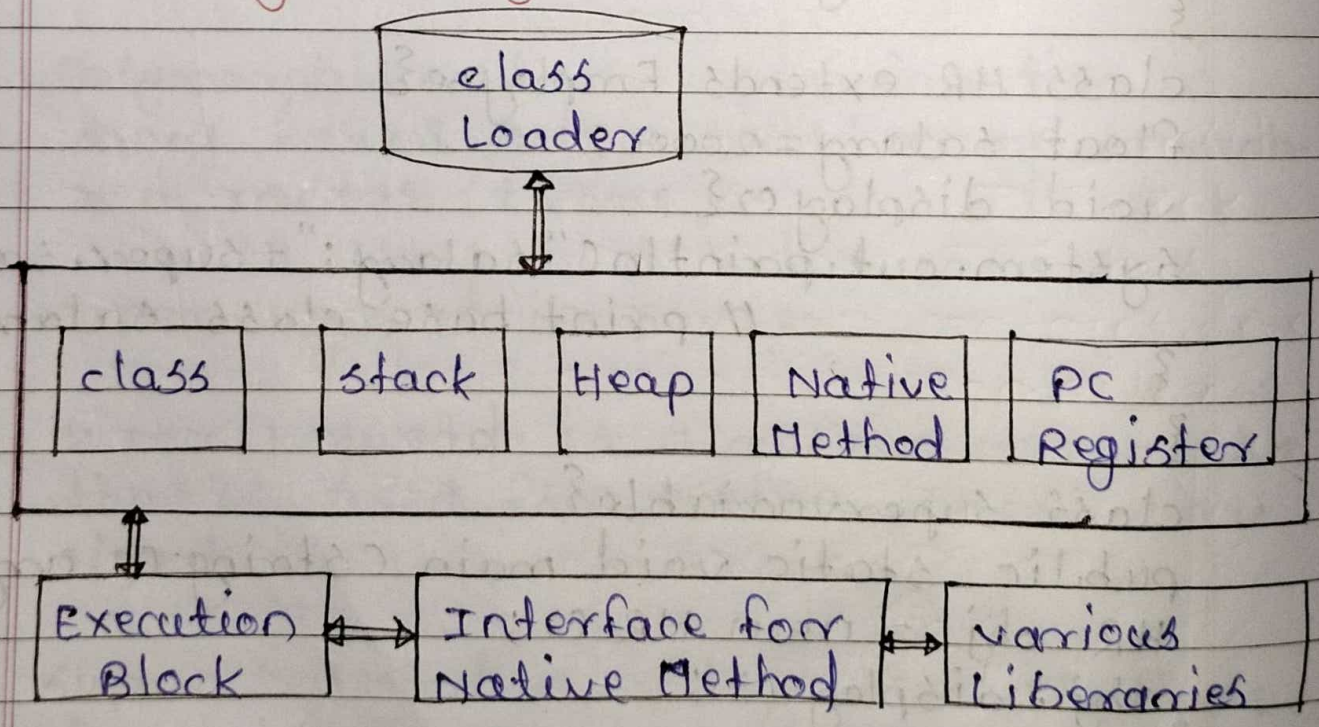
Page No.
Date

* This keyword in java:-

- This keyword is a reference variable that refers the current object in java.
- This keyword can be used for call current class constructor.
- The main uses of this keyword is to differentiate the formal parameter and data members of class.

- this.show() // method
- this.s = s // variable

* Memory Management:-



Page No.
Date

* Procedure:-

- After reading .class file, class loader save the corresponding byte code in the method area. Generally all JVMs have only one method area which is shared across classes which holds information related to each .class file.

- Heap is an integral part of JVM memory in which the objects actually rests. JVM produces the class object for each .class file.

- Unlike Heap, stack is used for storing temporary variables.

- PC - Registers used to keep exact information of all instructions. (which instruction is executing and which is going to be executed).

- A native method used to access the run-time data of the JVM (java virtual machine). Native Method interface enables java code to call by the native applications (programs that are specific to the hardware and OS).

* JRE (Java Runtime Environment)

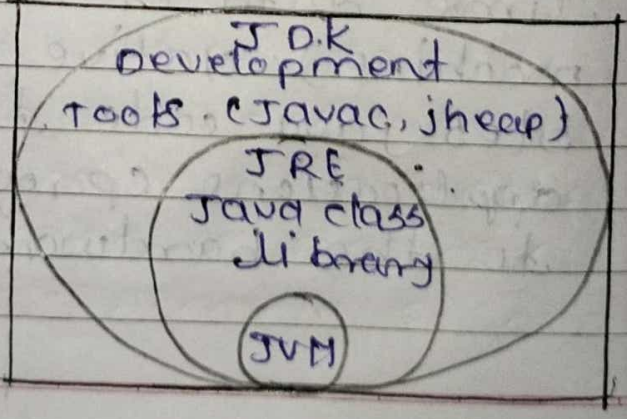
- Java Runtime Environment is within which the java virtual machine actually runs. JRE contains Java virtual machine and another files except development tools (debugger and compiler) so developer can run the source code in JRE but he/she cannot develop and compile the code.

* JVM (Java virtual Machine)

- JVM runs the program by using libraries and files provided by Java Runtime Environment.

* JDK (Java Development kit)

- Java Development kit can be considered as the super-set of JRE. JDK includes all features that JRE has and over and above it contains development tools such like compiler, debugger etc.



Page No.
Date

* Memory Management :-

- class (Method) Area

class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

- Heap

It is the runtime data area in which objects are allocated.

- Stack

- Java stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

- Each thread has a private JVM stack, created at the same time as thread.

- A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

- class Loader

Bookstrap classLoader, Extension class Loader, System/Application classLoader:

- Program counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

- Native Method Stack

It contains all the native methods used in the application.

- Execution Engine

It contains:
1. A virtual processor Interpreter
2. Just-In-Time (JIT) compiler

- Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc.

Page No.
Date

Java uses JNI framework to send output to the console or interact with OS libraries.

or

* packages :-

- Need for packages
- What are packages ; package declaration in Java.
- Import statement in Java
- How do packages resolve name clashes?
- Ex. package com.main.view
  package com.main.model
  package com.main.dto

* Miscellenous :-

- Double equals operator (==) ; comparision
- tostring() : to convert the value in string value.
- Reference variables, local variables, instance variables.

- A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword

2) Instance variable :-

- A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.
- It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable :-

A variable is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

Page No.
Date

* Example :-

```
class A {
  int data = 50; // instance variable
  static int m = 100; // static variable
  void method () {
    int n = 90; // local variable
  }
} // end of class.
```

Page No.
Date

* What is a static keyword? *

- The static keyword in Java is used for memory manggement mainly. We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class than an instance of the class.
- variable (also known as a class variable)
- Method (also known as a class method)
- Block
- Nested class.

* Static variable :-

- Fields that have the static modifier in their declaration are called static fields or class variables.
- They are associated with the class, rather than with any object.
- Every instance of the class shares a class variable, which is in one fixed location in memory.

- Any object can change the value of a class variable, but class variables can also be manipulated without creating an instance of the class.
- Syntax : <class-name>.<variable-name>

* Static Methods :-

- The Java programming language supports static methods as well as static variables.

main method is static, since it must be accessible for an application to run, before any instantiation takes place.

- Static methods, which have the static modifier in their declarations, should be invoked with the class name, without the need for creating an instance of the class, as in

ex. className.methodName(args)

- A common use for static methods is to access static fields.

- For example, we could add a static method to the Bicycle class to access the static field numberOfBicycles:

```
public static int getNumberOfBicycles()
{
    return numberOfBicycles;
}
```

* Not all combinations of instance and class variables and methods are allowed:

- Instance methods can access instance variables and instance methods directly.
- Instance methods can access class variables and ee class methods directly.
- class methods can access class variables and class methods directly.
- class methods cannot access instance variables or instance methods directly — they must use an object reference. Also, class methods cannot use the this or super keywords, as there is no instance to refer to.

Page No.
Date

Constants

- constant fields are often declared as static.
- For example NUM_OF_WHEELS which is a characteristic of any Bicycle, and not of a certain instance.
- If there's a sense to expose a constant field to the outside world, it is common to declare the field as public, rather than through a getter.

Singleton pattern

- Restricting the instantiation of a certain class to exactly one object.
- This is useful when exactly one object is needed to coordinate across the system.
- A singleton object is accessible globally using a static method.

Singleton

```
public class controller {
  private static controller instance = null;
  private controller() {----}
  public static controller getInstance(){
    if (instance == null) {
        instance = new controller();
    }
    return instance;
  }
  ...
}
```

* Not thread-safe

* A static Block *

- The static block, is a block of statement inside a java class that will be executed when a class is first loaded in to be the JVM.

A static block helps to initialize the static data members, just like constructors help to initialize instance members.

Page No.
Date

* Example of static *

ex. 1

```
class A2{
  static{
  System.out.println("static block is
  invoked");
  }
public static void main(String args[]){
  System.out.println("Hello main");
  }
}
```

output:- static block is invoked
Hello main

Page No.
Date

* Java static nested class *

- A static class i.e. created inside a class is called static nested class in java. It cannot access nonstatic data members and methods. It can be accessed by outer class name.

It can access static data members of outer class including private.

Static nested class cannot access non-static (instance) data member or method

* What is constructors?

- In java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

Page No.
Date

* Rules for creating constructor:-

- There are two rules defined for the constructor.
- constructor name must be the same as its class name.
- A constructor must have no explicit return type.
- A java constructor cannot be abstract, static, final, and synchronized.

```
class Bike1 {
Bike1 (){  // creating a default constru
System.out.println("Bike is created");
}
public static void main (String args[]){
Bike1 b = new Bike1 (); // calling a
                        default constructor
}
}
```

Page No.
Date

* Type of Constructors :-

- There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

If there is no constructor in a class, compiler automatically creates a default constructor.

Types of Java constructor:-

1. Default constructor.
2. parameterized constructor.

* Constructor chaining

- calling a constructor from the another constructor of same class is known as constructor chaining.

Page No.
Date

* This and Super constructors

- this() and super() are used to call constructors explicitly.

1. Using this() you can call the current class's constructor
2. Using super() you can call the constructor of the super class.

* Exception :-

- What? Why? How?

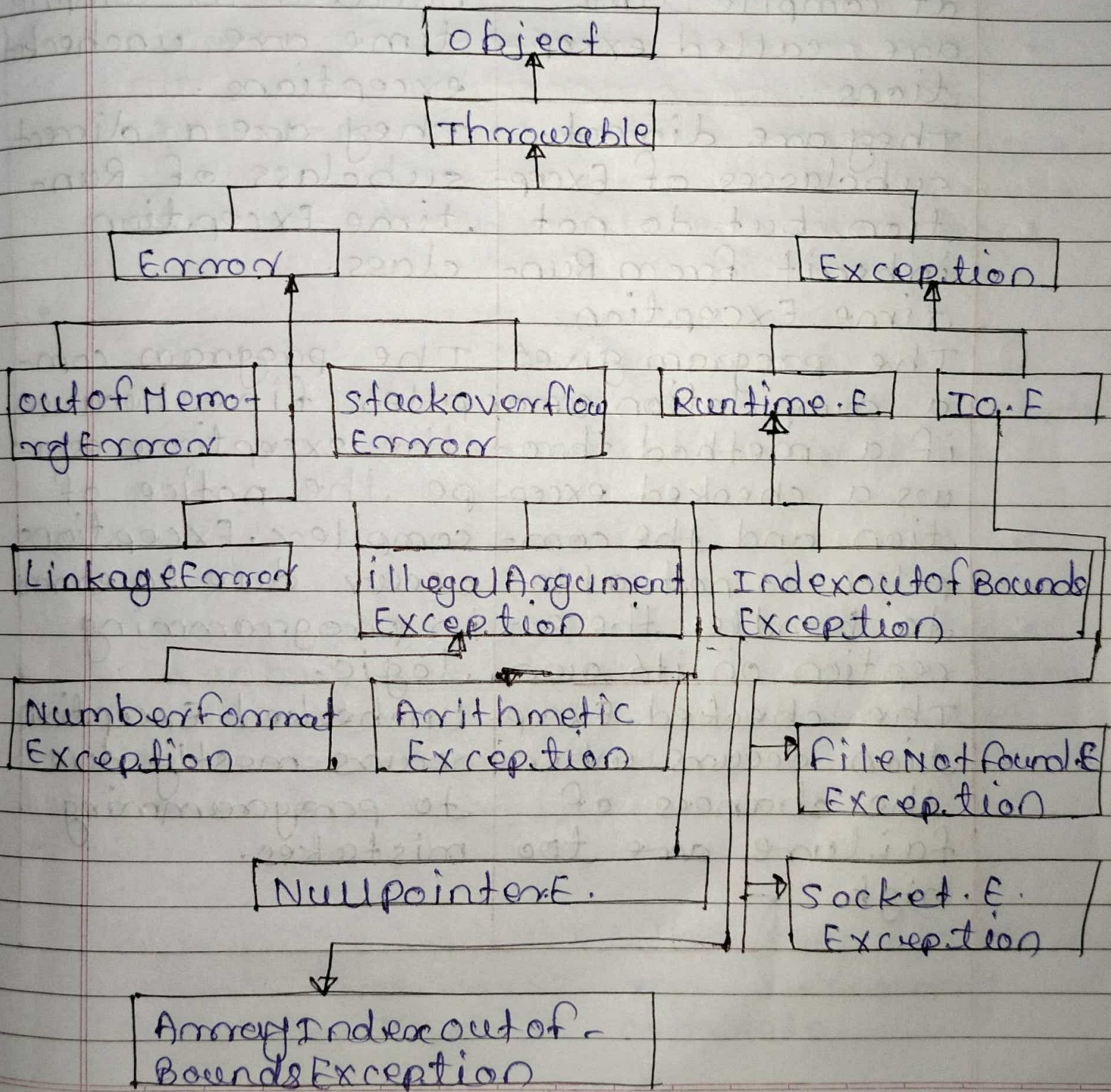
- In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Exception Handling is a mechanism to handle runtime errors such as classNotFoundException, IOException, SQLException, RemoteException, etc.

The core advantage of exception handling is to maintain the normal flow of the application.

An exception normally disrupts the normal flow of the application that is why we use exception handling.

* API hierarchy for Exceptions



Page No.
Date

* Checked Exception Vs Unchecked Exception *

Checked Exception	Unchecked Exception
Exception that are checked and handled at compile time are called exceptions.	Exceptions that are not checked and handled at compile time are unchecked exceptions.
They are direct subclasses of Exception but do not inherit from Runtime Exception.	They are a direct subclass of Runtime Exception class.
The program gives a compilation error if a method throws a checked exception and the compiler is not able to handle the exception on its own.	The program compiles fine because the exceptions escape the notice of compiler. Exceptions occur due to errors in programming logic.
The checked Exception occurs when the chances of failure are too high.	Unchecked Exception occurs mostly due to programming mistakes.

Page No.
Date

* Keywords for Java Exceptions

- Throws

  Describes the exceptions which can be raised by a method

- Throw

  Raises an exception to the first available handler in the call stack, unwinding the stack along the way.

- Try

  Marks the start of a block associated with a set of exception handlers.

- catch

  If the block enclosed by the try generates an exception of this type, control moves here; watch out for implicit subsumption.

- Finally

  Always called when the try block concludes, and after any necessary catch handler is complete.

Page No.
Date

* String in Java :-

* strings :-

- string is a sequence of characters placed in double quote (" "). // ("Java")
- strings are constant, their values cannot be changed in the same memory after they are created.
- There are two ways to create string: object :

1) By string literal.
2) By new keyword

* string creation :-

- By string literal :

For Example : string s1 = "welcome";
string s2 = "welcome";
// no new object will be created

- By new keyword:

For Example:

```
string s = new string ("sachin");
string s = newstring ("sachinTendulkar");
// create two objects and one reference variable.
```

* Immutability :-

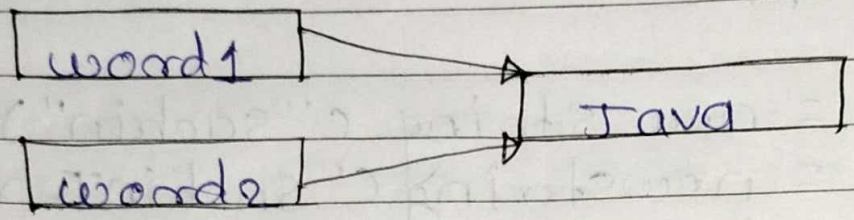
- In java, string objects are immutable. Immutable simply means unmodifiable or unchangeable.

once string object is created its data or state can't be changed but a new string object is created.

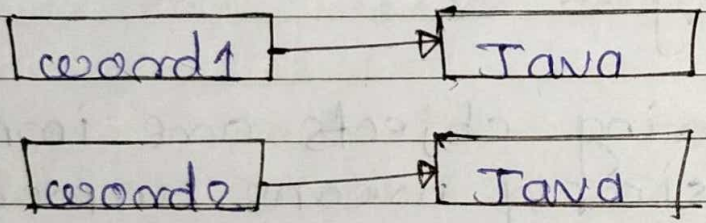
Advantage of immutability is use of less memory.

Disadvantages of immutability : Less efficient - you need to create a new string and throw away the old one even for small changes.

```
string word1 = "Java"
string word2 = word1;
```




```
string word1 = "Java";
string word2 = new string (word1);
```



Less efficient : wastes memory

* String Methods :

- substring (int begin) :-

Returns substring from begin index to end of the string.

Example :- string s = "nance";
system.out.println (s.substring(2)); //ce

- equals ():-

To perform content comparision where case is important.

Example :- string s = "Java";
system.out.println(s.equals("java"));
//true

- concat ():-

Adding two strings we use this method

Example :- string s = "nacre";
s = s.concat("software");
system.out.println(s); // nacresoftware

- length(), charAt()

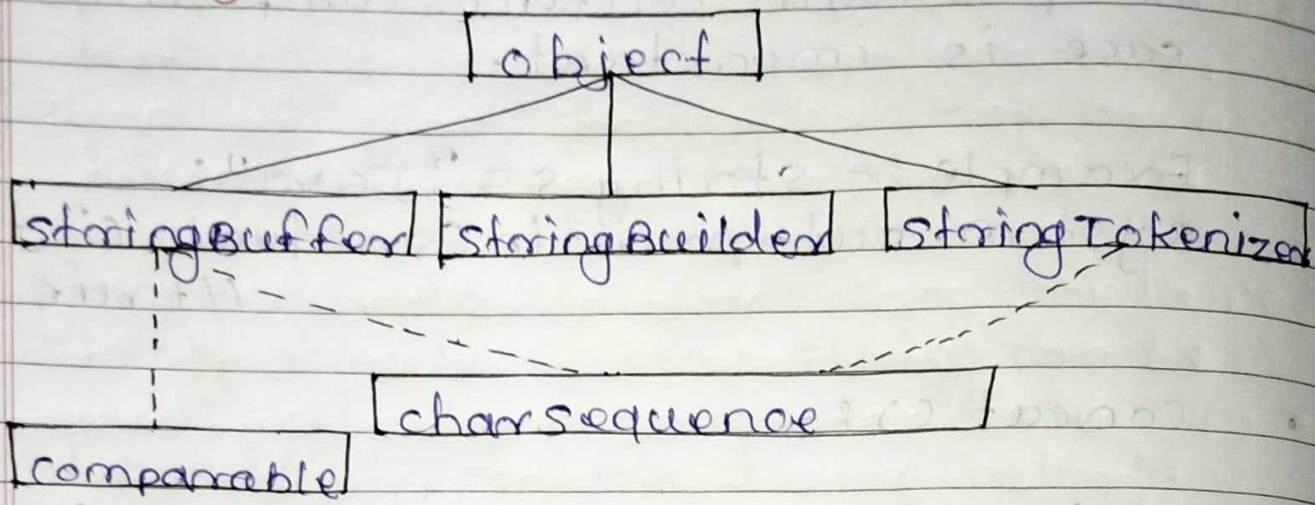
int length(); Returns the number of characters in the string

char charAt(i); Returns the char at position i.

character positions in strings are numbered starting from 0 - just like Arrays.

Returns:
"problem".length(); ⟶ 7
"Window".charAt(2); ⟶ 'n'

* StringBuffer, StringBuilder and StringTokenizer



All these classes are final and subclass of serializable.

* Limitation of string in Java?

What is mutable string?

- A string that can be modified or changed is known as mutable string. StringBuffer and StringBuilder classes are used for creating mutable string.

- Difference between string and StringBuffer in java?

Main difference between string and StringBuffer is string is immutable while StringBuffer is mutable.

Page No.
Date

* StringBuffer :-

- StringBuffer is a synchronized and allows us to mutate the string.
- StringBuffer has many utility methods to manipulate the string.
- This is more useful when using in multi-threaded environment.
- Always modified in same memory location.
- Methods :-

1) Append
2) Insert
3) Delete
4) Reverse
5) Replacing character at given index.

* StringBuilder :-

- StringBuilder is the same as the String-Buffer class.
- The StringBuilder class is not synchronized and hence in a single threaded environment, the overhead is less than using a StringBuffer.

Page No.
Date

* StringTokenizer

- A token is a portion of a string that is separated from another portion of that string by one or more chosen characters (called delimiters).
- The StringTokenizer class contained in the java.util package can be used to break a string into separate tokens. This is particularly useful in those situations in which we want to read and process one token at a time; the BufferReader class does not have a method to read one token at a time.

Index	string	stringBuffer	string Builder
storage Area	constant string pool	Heap	Heap
Modifiable	No (immutable)	yes (mutable)	yes (mutable)
Thread safe	yes	yes	No
Thread safe	fast	very slow	fast

Page No.
Date

* StringBuffer VS StringBuilder

1. Thread-safe	1. Not Thread-safe
2. synchronized	2. Not synchronized
3. since Java 1.0	3. since Java 1.5
4. slower	4. Faster.

THANK YOU

*