thod or

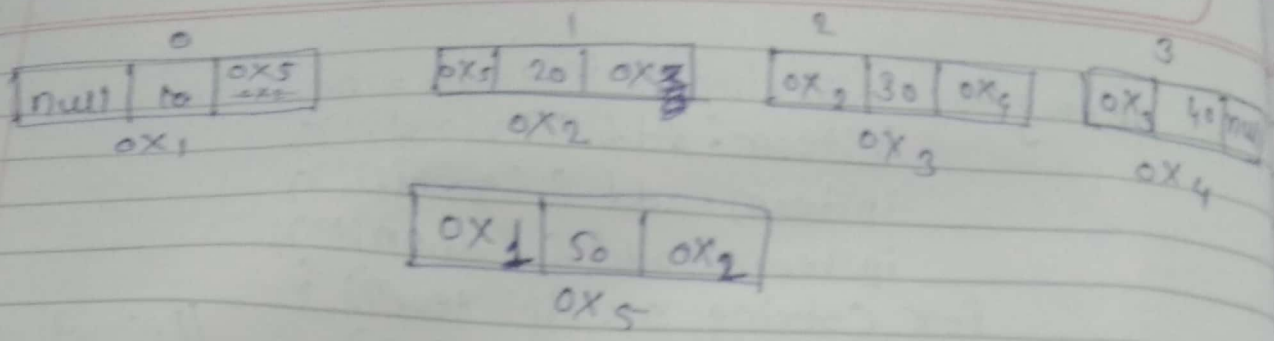
Page No.
Date :

```
ArrayList <String> iplteam = new ArrayList <> ();
        iplteam.add ("virat");
        iplteam.add ("sachin");
        iplteam.add ("dhoni");
    for (String s : ipl team) {
                sop (s + " ");      // virat sachin dhoni
        }
        sopln ( );
Arraylist < Integer > a = new Arraylist < > ();
                a.add (12);
                a.add (34);
                a.add (55);
            for (int i = a) {          // Auto unboxing
                sop (i + " ");         // 12 34 55
        }
    }
}
```

* Linked list

1) Linked list follows doubly linked list data structure.
2) There is no intial capacity for linked list
3) There is no incremental capacity :

```
e.g :- LinkedList ll = new LinkedList ();
        ll.add (10);
        ll.add (20);
        ll.add (30);
        ll.add (40);
        ll.add (1, 50);
```



- Points -

1) We usually don't Prefer using arraylist for insertion and deletion ~~insertion~~ operations beoz these kind of operations will lead to shifting of elements, hence Arraylist is bad choice for insertion and deletion operations. but Arraylist is highly Prefered for searching & Sorting operations.

2) LinkedList is Preferred for insertion and deletion operation & not for searching & sorting operations.

* Constructors in Arraylist

1) ArrayList ().
2) ArrayList (int initial capacity).
3) ArrayList (Collection c).

* Constructors in LinkedList -

1) LinkedList ().
2) LinkedList (Collection c).

```
* Public class A {
    psvm(s[] a) {
      LinkedList ll = new LinkedList();
  for(int i=0; i<=10; i++) {
          ll.add(i*10);
      }
      sopln(ll.size());          ll.
      ll.display();
    }
}
class LinkedList {
    Private int size;
    Private Node, head, temp;

    Public int size() {
        return size;
    }
  Public void add(int data) {
      Node n = new Node;
      n.val = data;
      if(head == null) {
          head = n;
          temp = n;
      }
      else {
        temp.next = n
          temp = n;
      }
      size++;
  }
  Public void display() {
      Node n = head;
```

```
while (n != null) {
    sop (n.val + " ");
    n = n.next;
}
sopln ();
}
```

O/P

10 20 30 40 50 60 70 80 90 100

* **Stack.**
  It follows first in last out datastructure

* **Methods in stack.**
  1) Push (object o) add करण्यासाठी
  2) Peek () top. बघण्यासाठी
  3) Pop () top चा देऊन नंतर तो remove करते
  4) isEmpty ().

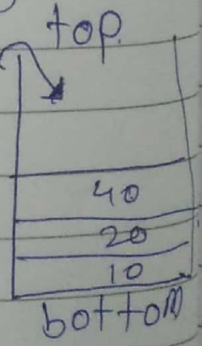
addition चालू होते Bottom to top

* Peek ()
  - It gives the topmost object present in stack

* Pop ()
  Gives the top most object and also removes it.

eg:

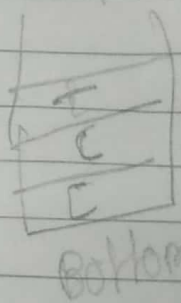
```
Stack<Integer> S = new Stack<>();
S.push (10);
S.push (20);
S.push (40);
int i = S.peek ();   // 40.
int j = S.pop ();    // 40
S.isEmpty            // false
```



Page No.
Date:

Q WAP to check whether brackets are balanced or not

```
class A {
psvm (S[]a) {
scanner sc = new scanner (system.in);
    sopln ("enter the string");
        string s = sc.nextline();
boolean b = A.isbalanced(s);
if (b == true) {
        sopln ("balanced");
    }
    else {
     sopln ("not balanced");
}
}
```




```
Public Static boolean isBalanced (string s) {
    Stack < character > st = new stack();
for (int i=0 ; i < s.length ; i++) {
        char c = s.charAt(i);
if [c == '{' || c == '[' || c == '('){
        st.push(c);
    }
else if (c == '}' || c == ']' || c == ')') {
    char x;
    switch (c) {
    case '}':
        x = st.pop();
    if( x == '[' || x == '(')
        return false;
    else
        break;
    case ']';
```

```
x = st.pop();
if (x == '{' || x == '(')
    return false;
else
    break;

case ']':
    if x = st.pop() && (
    if (x == '[' || x == '{')
        return false;
    else
        break;
    }
}
}
        st.isEmpty();
return isEmpty();
}
```

O/P

enter the string

{ ([ ]) }

is balanced.

H.W. Constructors in LinkedList

```
eg:- Public class A {
        Public static void main (String [] args){
    Arraylist < Integer > al = new ArrayList <>();
    for (int i=1; i<=10; i++) {
            al.add(i*2);
        }
        SopIn(al.size());          //10 :
    LinkedList < Integer > ll = new LinkedList(al);
        for(Integer i : ll) {
        Sop(i+" ");          //2 4 6 8 10 12 14 16 18 20
}}}
```

Page No.
Date :

Q.1 create list to store student object & Print every detail of all the object using to string method.

Ans →

```
Class student {
        int rollNum;
        string Name ;
        String Gender ;

student(int roll, String Name, string gender){
        rollNum = roll;
        Name = name;
        Gender = gender;
  }
  Public string tostring () {
    return (rollNum) + " " + Name + " " + Gender + " ");
}}

  Public class B {
    Public static void main (string [] args ) {
        student s1 = new student (901, "Pranali", "femele");
      student s2 = new student (902, "samiran", male);
      student s3 = new student (903, "yukta", "femele");

ArrayList < student > al = new ArrayList <> ();
        al. add (s1);
        al. add (s2);
        al. add (s3);

    for (student s: al) {
        sopln (s);
    }
  }
}
```

Vector :-

- 1) Vector is similar to Arraylist but it is not multithreaded (~~single threaded~~) (It is single threaded)
- 2) The initial capacity is 10, ~~incremental capacity~~
- 3) incremental capacity = current capacity * 2.

Note :- Stack, Vector, Hashtable are called legacy classes of Java.

* Difference bet" Arraylist & Vector & LinkedList

	Arraylist	Vector	LinkedList
initial Capacity	10	10	NA
incremental capacity	$\frac{\text{current capacity} * 3}{2} + 1$	current capacity * 2	NA
Data structure	Growable Array	Growable array	doubly linked list
Constructors	3	4	2
Thread Safe	No.	yes.	No

Page No.
Date :

* SET

SET is ~~parent~~ child interface of collection which allows no duplicate value.

Characteristics of set:

1] The Important characteristics of set is. It cannot store duplicate elements

2] Insertion order is not maintained.

3] SET is not Index based

4] At most any set can store maximum one null value.

e.g

```
Public class A {
    Public static void main (string [] args) {
Hashset hs = new Hashset <>();
        hs.add("10");
        hs.add (10);
        hs.add (10);
        hs.add (34);
        hs.add ("object");
        hs.add (null);
        hs.add (null);
    for (object o : hs) {
        Sopln (o + " ");
}  }  }
```

* Hashset

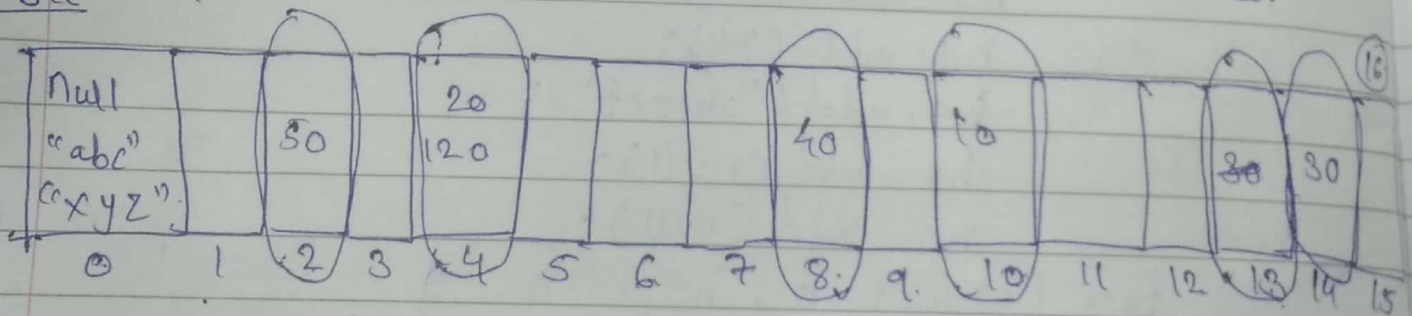
1) Hashset is implemented using hash table data structure
2) Hashtable Algorithm uses hashcode method internally
3) The intial capacity of hashSET is 16
4) The Load factor is 75%.

```
Public class A {
Public static void main (String [] args) {
Hashset hs = new Hashset <> ();
   int a [] = new int [5];
for (int i = 1 ; i <= 5 ; i++ ) {
   object o = i * 10;
a[i-1] = o.hashcode () % 16;
   hs.add (o);
}
for (int i : a) {
   ~~Sop~~ Sop (i + " ");
}
   Sopln ();
for (object o : hs) {
      Sop In (o + " ");
}
}
```

o/p
10 4 14 8 2
50 20 40 10 30

Always null element store in zeroth bucket bcoz it is not have address.

Hashtable




```
hs.add (120);
hs.add (20);
```

- **Loadfactor** : capacity to store how many elements to be store in each bucket i.e (75 % of bucket)

- **Constructors**
  1) ~~Public java.util.~~ Hashset ();
  2) ~~Public Java.util.~~ Hashset (java.util.Collection <? extends E>)
  3) Hashset (int, float);
  4) Hashset (int);

Stack → realtime e.g → Phone call log → FIFO/LIFO
Linked hash set → realtime → Play list (music) → FIFO
e.g.

Linked HashSET

1) Linked hashSET is implementing using hashtable and doubly linked datastructure
2) Linked hashSET maintain insertion order
3) since there are two datastructure are implemented the memory consumption is quite high.

```
Public class A {
Public static void main (S [] a). {
LinkedHashSET lhs = new linkedHashSET <> ();
for (int i=1; i<=6; i++) {
    lhs.add(i);
}
// trying to add duplicate elements
    lhs.add(2);
    lhs.add(4);
for (object o: hs) {
    system.out.print(o+" ");
}
}
}
```

o/p.

1 2 3 4 5 6

Tree set

1) Tree set implements balanced tree datastructure
2) Tree set maintains natural sorting order (default sorting)
3) you can only add Comparable type elements to tree SET
4) Tree set can only store homogeneous objects
5) If you add any heterogeneous element you end up with class cast Exception error

API → Application. Programing Interface → search on Google/youtube

natural Sorting = ascending order.

e.g:→

```
Public class A {
  PSVM (SI] a) {
Tree Set ts = new Treeset<>();
      ts.add (123);
      ts.add (34);
      ts. add (-99);
for (object o: ts) {
        S.O.P(o+" ");      o/p.
} } }
                           -99 34 123.
```

NOTE 1) Tree SET can not store even a single null bcoz null cannot compare with any type of elements.

2) If it is null. you end up with null point exception

e.g:→

```
Public class A {
  PSVM (S[] a) {
  Tree<String> ts = new TreeSET <> ();
  ts. add ("ac");
  ts. add ("abc");
  ts. add ("abd");
  ts. add ("xy");
  ts. add ("xyz");
// string will be compared lexicographically.
    for (string s: ts) {
      S.O.P(s+" ");
    }
  }
}
```

o/p

abc, abd, ac, xy, xyz

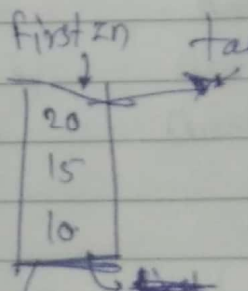
Stack → First In last out

Queue → Reservation Category [First in first out] added from 1 end & remove from another end.

Page No.
Date :

* **Queue** : Queue Implements first In first out Datastructure

```
Public class A {
    Public static void main (String [] args) {
Queue q = new priorityQueue <> ();
q.add (10);
q.add (15);
q.add (20);
Object P = q.Peek ()
    System.out.println (P)                    // 10
        P = q.Poll ();
    System.out.print (P);                     // 10
        P = q.Peek ();
    System.out.println (P)                    // 15
    System.out.println (q.isEmpty ());        // false
        q.clear ();
    System.out.println (q.isEmpty ());        // true
} }
```



Queue q = new Priority Queue ();
FIFO

q.add (10);
q.add (15);
q.add (20);

```
Object P = q.Peek ();
    SopIn (P)          ; // 10
```

show first element added

```
P = q.Poll ();
    SopIn (P)
```

// 10 → remove firstly added element & give you

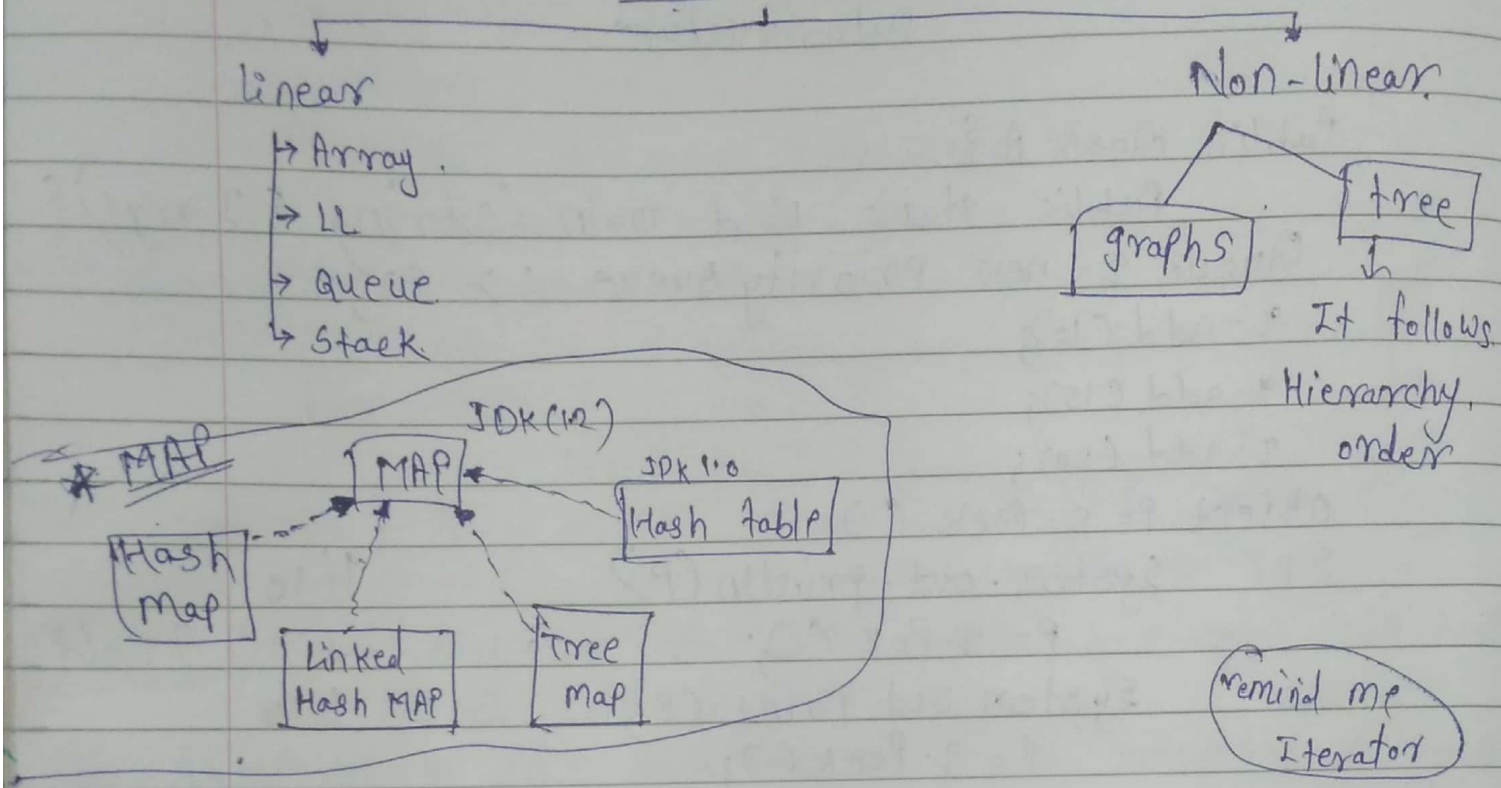
```
P = q.Peek ()
    SopIn (P);
```

// 15 → shows 2nd element

Legacy class → Hashtable, vector, stack

Page No.
Date :

Data structure.



Remind me Iterator

Methods of MAP Interface.

```
V          Put (K key, V val);
int        size ();
boolean    isEmpty ();
boolean    containsKey (Object key);
boolean    containsValue (Object value);
V          get (Object key);
V          remove (Object key);
void       clear ();
void       PutAll (Map<K, V> m);
Set<K>     keySet ();
Collection<V>  values ();
Set<Map.Entry<K, V>>  entrySet ();
```

Map → ludo, chess, snake & ladder

Page No.
Date :

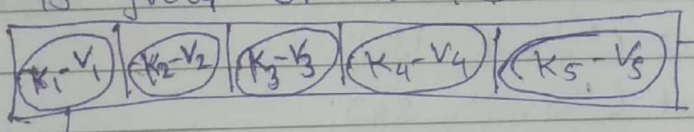
```
Interface Map {
    // Methods
    interface Entry {
        K   getKey();
        V.  getValue();
    }
}
```

→ nested interface

MAP

key value

1) Entry is Key & value Pair (K–V)
2) MAP is group of entries



1 entry

~~Key bas~~

Key must be unique, values can be duplicated in any MAP

e.g.

```
Public class A {
Public static void main (string [] args) {
    Map m = new HashMap <> ();
    m.put (1, 345);
    m.Put ("abc", 345);
    m.Put ("xyz", 345);
    m.Put ("abc", 345);
    m.Put (2, 345);
    m.Put (1, 345);
    sopln (m);
Map < Integer, String > m1 = new HashMap <> ();
    m1.Put (1, "abc");
    m1.Put (2, "abc");
    m1.Put (2, "xyz");
    m1.Put (3, "def");
```

```
m1.put(3, "abc");
m1.put(3, "hello");
Sop1n(m1);
}
}
```

O/P.

{1=345, 2=345, abc=345, xyz=345}

{1=abc, 2=xyz, 3=hello}

e.g

```
Public class A {
 Public static void main (string [] args) {
MAP<string, Integer> rcbteam = new HashMap<>();
 rcbteam.put ("virat", 18);
 rcbteam.Put ("siraj", 34);
 rcbteam.Put ("Padekar", 9);
 rcbteam.Put ("chahal", 4);
Map.< string, Integer> miteam = new HashMap<>();
   miteam.put ("sky", 63);
   miteam.put ("rohit", 45);
   miteam.Put ("bumrah, 27);
   miteam.put ("chahal", 14);
MAP< string, Integer> Indiateam = new HashMap<>();
   indiateam.putAll (rcbteam);
   indiateam.putAll (miteam);
 System.out.println (indiateam);
 System.out.println (indiateam.ContainsKey ("virat"));
System.out.println (indiateam.Contains value (45));
   Integer V = Indiateam.get ("chahal");
      System.out.println (V);
 V = Indiateam.remove ("Padekar");
      System.out.println (V);
Set< string> keys = indiateam.KeySet ();
   System.out.println (keys);
```

```
Collection < Integer > values = indiateam.values();
        System.out.println (values);
    }
}
```

O/P

{SKY=63, chahal=14, bumrah=27, virat=18, siraj=34, Padekar=3, rohit=45}
true
true
14
3
null
[SKY, chahal, bumrah, virat, siraj, rohit]
[63, 14, 27, 18, 34, 45]

23/02/23 * Iterator.

- It is an interface Present in java.util Package
- Technically iterator is a user present at the begining of the collection.
- Iterator is having 3 methods.

boolean	hasNext()
<E>	next()
(void)	remove()

Ex:-
```
Public class A {
    PSVM (S[] a)
    Hashset < string > hs = new Hashset <>();
for(int i=1; i<=6 ; i++) {
    hs.add (i + "abc");
    }
    SopIn(hs);
```

```
Iterator< string > o = hs.iterator();
    while(i.hasNext()) {
        Sop(i.next());
    }
}
```

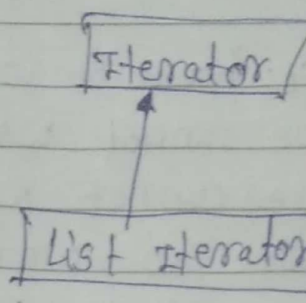
O/P = [5 abc, 1 abc, 6 abc, 3 abc, 4 abc, 2 abc]

Q. Create a arraylist and traverse the array using iterator.

→
```
Public class A {
    Psvm(S[] a) {
        Arraylist < Integer > = new Arraylist <>();
for(int i=1; i<=6; i++) {
            al.add(i);
        }
        Sop(al);
    }
Iterator < Integer> i = al.iterator();
        while(i.hasNext()) {
            Sop(i.next());
        }
    }
}
```

*
```
Public class A {
    Psvm(String[] args) {
linked list < Integer > ll = new LinkedList <>();
    (int i=1; i<=4; i++) {
        ll.add(i);
    }
list Iterator< Integer > i = ll.listIterator(ll.size());
```

```
while (i.hasPrevious()) {
    Sop (i.Previous.());
}
};
```



Exception is unexpected event happening at runtime.

* Exception Handling :-

• Exception :- It is a runtime interruption due to which the Program stops.

• what is exception handling?

→ Exception handling deals with resolving an exception.

* Risky Code :-

The code which might through you an exception:-

• Exception can be handled using i] try-catch block or
ii] try-catch-finally as well

i] try-catch block -

Syntax :-
```
try {
    //risky code
}
catch (Possible ExceptionType e)
{
    //statements
}
```

block.

- we have to write some risky code in try ~~book~~
- we cannot write catch block without try block
- we cannot write try block alone
- we can write try block with catch block. or try block with finally block or try block with both catch block and finally block.

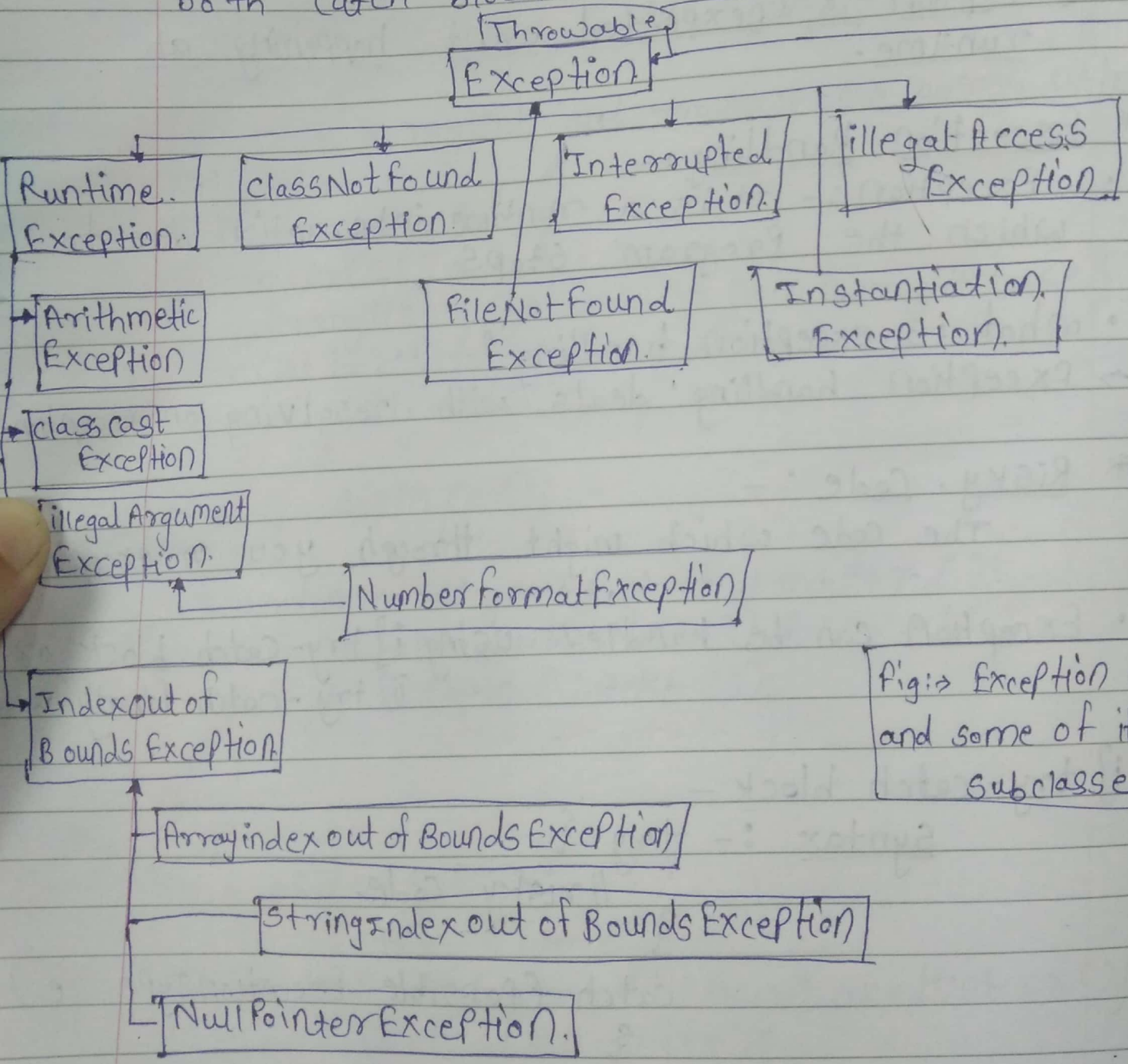


Fig:- Exception class and some of its subclasses

NOTE :⇒ How many exceptions you got in try block → only 1

JVM throws exception object

- Every Exception is object in java.

Scenario 1

when there is no exception occured in try block.

```
Public class A {
Ps v m (S[] a) {
int a = 10;
int b = 5;
try {
    int res = a/b
    sopln (res);                    //2
} catch (ArithmeticException e) {
    sopln ("catch block is executed");
}
    sopln ("Program ended");
}
}
```

O/P:

2

Program ended

Scenario 2 : when there is exception occured in try block the control goes ~~through~~ to catch block from the line where exception was occured

```
Public class A {
Ps v m (S[] a) {
int a = 10;
int b = 0;
try {
    Sopln ("try block");
int res = a/b;
sopln (res);
} catch (ArithmaticException e) {
    Sopln ("catch block is executed");
}
```

```
        SopIn("program ended");
    }
}
```

O/P

try block.
catch block is executed
Program ended.

Scenario : 3.

When there is exception occure in try block but if there is no matching catch block exception will not be handled.

```
Public class A {
    Psvm(S[]a) {
        int a=10;
        int b=0;
    try {
        SopIn("try block");
        int res = a/b;
        SopIn(res);
    } catch(ArrayIndexOutofBoundsException e){
        SopIn.("catch block is executed").
    }
    SopIn("program ended");
    }
}
```

O/P

Exception in thread "main" try block
java.lang.ArithmeticException : /
at gsf_jsp.A.main(A.java:11)

* Supermost class in Exception Hierarchy is throwable class

* Scenario: 4. : An ~~child~~ exception occure in child block can be called by a catch block with its Parent type.

```
Public class A {
Public static void main (String [] args) {
    int a = 10;
    int b = 0;
    try {
        SopIn ("try block");
    int res = a/b;
        ~~SopIn (RuntimeException e) {~~
        SopIn (res);
    } catch (RuntimeException e) {
        SopIn ("catch block is executed")
    }
        SopIn ("Program ended");
    }
}
```

O/P

try block

Catch block is executed

Program ended

NOTE :→ only 1 Exception can occure in try block bcoz (try block search for matching catch block exception & if there is no matching catch block u get exception & Program stops there but if u have matching Parent type in catch block only You did not get error Program Continues.)

NOTE

(If you write multiple catch block, catch block with that Particular exception that occured in try block will be executed.)

IMP NOTE ~~*~~ Supermost class in Exception Hierarchy is throwable class

Scenario: 4. : An ~~child~~ exception occure in child block can be called by a catch block with its Parent type.

```
Public class A {
Public static void main (string [] args) {
    int a = 10;
    int b = 0;
    try {
        SopIn ("try block");
    int res = a/b;
     ~~SopIn (RuntimeException e) {~~
        SopIn (res);
    } catch (RuntimeException e) {
        SopIn ("catch block is executed")
    }
        SopIn ("Program ended");
}
```

O/P

try block

Catch block is executed

Program ended

NOTE :⇒ only 1 Exception can occure in try block bcoz (try block search for matching catch block exception & if there is no matching catch block u get exception & Program stops there but if u have matching Parent type in catch block only You did not get error Program Continoues.)

NOTE

(If you write multiple catch block, catch block with that Particular exception that occured in try block will be executed.)

Scenario 5 :→ ~~They~~ You can write multiple catch block for single try block.

```
Public class A {
  public static void main(String []args){
    int a = 2;
    int b=5;
    String s = "abc";
    char c[] = s.toCharArray();
try {
    SopIn("try block");
    SopIn(s.charAt(a));
    SopIn(c[b]);
   int res = a/b;
    SopIn(res);
} catch (StringIndexOutOfBoundsException e){
    SopIN("String index exception");
} catch (ArithmeticException e){
    SopIn("arithmetic exception");
} catch (ArrayIndexOutOfBoundsException o ){
    SopIn("Array Index exception");
  }
    SopIn("program ended");
  }
```

O/P

try block

c

Array index exception

Program ended.

Sequence

1) try 1) try 1) try
2) catch 2) finally 2) catch.
3) finally

Page No.
Date:

Scenario 6.

When you write multiple catch blocks the sequence of catch block should be from child to parent else compilation error occures.

```
Public class A {
  Public static void main (String [] args) {
int a = 10;
int b = 0;
try {
  System.out.println("try block");
    int res = a/b;
  System.out.println(res);
} catch (ArithmeticException e) {
      SopIn ("arithmetic exception");
    } catch (RuntimeException e) {
      SopIn ("Runtime exception");
    }
    SopIn("end");
  }
}
```

o/p.
try block
arithmetic exception
end

* finally block :-

1) finally block can be written with try block or try catch block.
2) Usually finally block is used to close costly resources / or system resources.
3) finally block will be executed at any cost irrespective of the exception.
4) If we get or did not get exception but finally block always executed.

e.g

```
Public class A {
Public static void main (string [] args) {
Scanner sc = new Scanner (system.in);
SopIn ("enter the number");
try {
    int a = sc.nextInt();
    system.out.println("try block");
      int res. = 10/a
    system.out.println(res);
  } catch (RuntimeException e) {
      System.out.println("AE exception");
  } finally {
      SopIn ("finally block");
    }
      SopIn ("end");
  } }
```

O/P.

```
enter the number
  0
try block
Arithmatic exception.
 finally block
 end.
```

#

```
Public class # {
   Public static void main (string [] args) {
 Scanner sc = new scanner (system.in);
      SopIn ("enter the number");
   try {
        int a = sc.nextInt();
          SopIn ("try block");
```

* Exception object propagation: Exception travel from one method to another method

Page No.
Date:

```
int res = 10/a
    sopln(res)
} finally {
    sopln("finally block");
}
    sopln("end");
}
```

o/p.
enter the number
0
try block
finally block
Exception in thread "main"

* throws Keyword

1) throws is a keyword which is used to indicate an exception.
2) throws keyword does not resolve an exception. rather it just indicates there is a possibility of exception that can occure.

NOTE ⟹ throws keyword can be used with
1) method declaration.
2) Constructor declaration

- throw keyword used to create an exception (only 1 object at a time)
- throws -11- -11- -11- indicate an exception (indicate multiple exception)

* Checked Exception / Compile time Exception / unhandled Exception
- checked Exception is that kind of exception where compillar checks whether the exception is handled or not
- In other word handling a checked exception is mandatory

e.g:- ~~File Not~~ 1) fileNotFoundException
2) ClassNotfoundException
3) IOException

e.g

```
import java.io.File;
Public class A {
    Public static Void main(string[] args) throws
        FileNotfOundException {
```

throws keyword does not handled exception it only carry. exception to calling method.

```
        m1();
        sopln("main end");
    }
Public static void m1() throws FileNotFoundException{
file f = new file("c:\\users\\qspiders\\Desktop\\assignment.txt");
fileInputstream fis = new fileInputstream(f);
        sopln("method end");
    }
}
```

o/p

If Exception

Unchecked Exception / Run-Time Exception / handled Exception

unchecked Exception is that kind of Exception where compiller does not force you to handled that exception

Custom Exception

It is user defined exception, you can create your own checkedtype or uncheckedtype exception

2 Important methods in Throwable

1) Void printStackTrace()
```
{
}
```
2) String getMessage()
```
{
}
```

```
Public class InsufficientException extends RuntimeException{
    Public String getMessage() {
        return "less amount";
    }
}
Public ~~std~~ class A {
    Public static void main (string [] args) {
        transferlayot ();
        SopIn ("program donne");
    }
Public static void transferlayout () throws InsufficientFundsException {
    Scanner SC = new Scanner (System.in);
        SopIn ("enter the transfer amount");
        double available = 190.45;
        int amount = SC.nextInt ();
        if (amount > available) {
    InsufficientFundsException a = new InsufficientFundsException ();

        throw a;
    } else {
            SopIn ("amount transfered");
        }
}
```

* File Handling

1) File handling deals with reading data from a file and writing data into a file.

e.g

```
import java.io.file;
import java.io.IoException;
Public class A {
    Public static void main (string [] args) throws IoException {
File F= new file ("c:\\users \\@spiders\\ Desktop \\Aggile.txt")
   if (f.exist ()) {
       Sopln ("file is present in the system");
    } else {
       f.createNewfile ();
    }
     Sopln ("end");
} }
```

O/P

end.

Aggile created file in computer

* Blocks.

① syntax for nonstatic block

```
{
  // statement
}
```

② Syntax for static block

```
static {
   // statements
}
```

Blocks → static blocks
Blocks → non static blocks

Static block	Non static block
1) Static blocks will be executed only once at the time of class loading Process.	1) Non static blocks will be executed whenever u create an object
2) static block will be executed before main method	2) Non static blocks will be executed only if you create an object
3)	3) non static blocks will be called before a constructor

e.g

```
Public class A {
    static {
        sopln("static block-1");
    }
    static {
        sopln("static block-2");
    }
    A() {
        sopln("zero Param");
    }
    A(int n) {
        sopln("int Param");
    }
    Public static void main(string[] args){
        sopln("main method");
        A a1 = new A();
        A a2 = new A(10);
        sopln("main end");
    }
    {
        sopln("non static block-1");
    }
    {
        sopln("non static block-2");
    }
```

{
SoPln ("non static block-3");
}
static {
SoPln ("static block-3");
}
}

O/p.
static block 1
static block 2
static block 3.
non static block 1
non static block-2
non static block-3
Zero Parameter
non static block-1
non static block-2
non static block-3
int Parameter
main end.

* FileInputstream is used to read data/input from a file
: read() is used to give the ascii value of a character present in a file & cursor move to next character

filehandling is used to store output

# Public class A {
psvm (s[] a) throws Exception {
File f = new file ("c:\\users\\qspiders\\Desktop\\Agile.txt");
FileInputstream fis = new FileInputstream (f);
int i = fis.read();
StringBuffer sb = new StringBuffer();
while (i != -1) {
sb.append ((char) i);
i = fis.read();
}
sopln(sb);
} }

Off

4 too character.

Scanner class is used to read entire line from file

Page No.
Date

* FileOutputStream :=

```
Public class A {
Public static void main (string [] args) throws Exception {
file f1 = new file ("c:\\Users\\aspiders\\Desktop\\Agile.txt");
file f2 = new file ("c:\\Users\\aspiders\\Desktop\\Copy.txt");
FileInputStream fis = new FileInputStream (f1);
FileOutputStream fos = new FileOutputStream (f2, true);
    int i = fis.read();
    while (i!=-1){
        fos.write(i);
        i = fis.read();
    }
    sopln("new file created");
}
}
```

o/p
new file created

*

```
Public class A{
    psvm (s[] args) throws Exception {
    File f1 = new File ("c:\\Users\\aspiders\\Desktop\\Agile.txt");
Scanner sc = new Scanner (f1);
    String out = "";
    while (sc.hasNextLine ());
    out = out+s;
    out = out+(char) 10;
    }
        sopln (out);
    }
}
```

```
Public class A {
   PSVM(S[] a) throws Exception. {
file f1 = new file("c://users//@spiders//Desktop//Agile.txt");
file f2 = new file("c://users//@spiders//Desktop//demo.txt");
    scanner sc = new scanner (f1);
     string out = "";
   while(sc.hasNextLine()) {
     String s = sc.nextline();
       out = out + s;
       out = out + (char) 10;
     }
     byte b[] = out.getBytes();
   FileOutputstream fos = new fileoutputstream(f2);
       fos.write(b);
       Sopln("done");
   }
```

① [P.
one.

You can achieve multitasking using multithreading

* Multithreading.

=> Bookmy show & all real time application in mobie is example of multitasking.

```
Public class MyThread Implements
    Public void run()
```

```
Public class MyThread2 extends Thread {
        Public Void run()
```

Sleep() is asking for (milisecond) It doesnot accept for minits second time

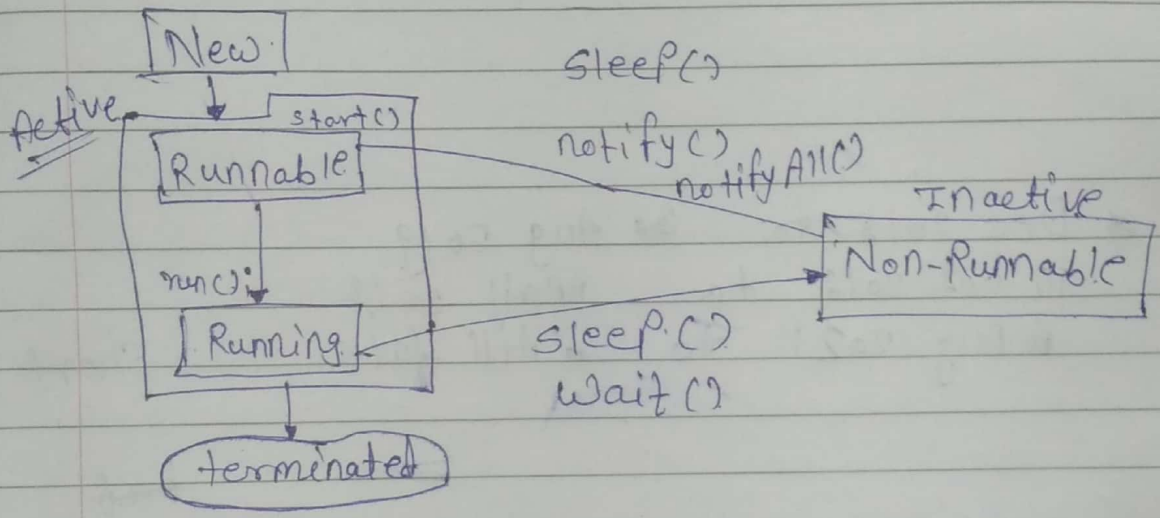
1 minit = 6000 milisecon nds

Page No.
Date:

Public class MyThread2 extends Thread {

O/P 0 1 2 500 499 498 3,4 ..... 0 500
My Thread 1 is done
My Thread 2 is done

Lifecycle of a thread :-



Both the thread are runnable.

$0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

integer literals

- Binary (0b / 0B) zero zero
- octal (0) zero
- decimal
- hexadecimal - (0x / 0X)

1) int i = 0b1010;
Binary sopln(i); // 10.

2) int i = 016 $1\times8^1 + 6\times8^0 = 8+6 = 14$
octal sopln(i); // 14

$10\times16^1 + 11\times16^0 = 160+11$

4) int i = 0X ab (10 11)
sopln(i); // 171
// a to f
10 15

3) int i = 098
octal
sopln(i); // CTE
in octal base is 8.
So u have on 0 to 7 nuber

10 % 9 = 1

Page No.
Date :

Difference betn -Private method & final meth

Private method

1) Private method cannot be inherited
2) Private method cannot be overridden.
3) it cannot be inherited

final method

final method can be inherited.
2) final method cannot be overridden
3) It can be inherited

Tessie → Dec 2018 to Aug 2019
Ref → 14 oct 2016 to April 2018
Jayraj → Aug 2021 to till date

Short
4.

~~bat~~

ball

I f boolean

128.

21.5
0.8

8 + 9 + 10