16/08/23

Unit - 2

⇒ Class

```
import java.util.scanner;
class student
{
    int rollno;
    String name;
    float spi;
    void displayData()
{
    System.out.println("rollno is" + rollno);
}
void getdata()
{
    Scanner sc = new Scanner(System.in
rollno=sc.nextInt();
name=sc.nextLine();
spi=sc.nextFloat();
}
class main
{
    public static void main(String args[])
```

```
{
    Student s = new Student();
    s.getdata();
    s.displayData();
}
}
```

We can save file using student or main class but we can run only using main class.

⇒ Constructor

```
class Ball
{
    int size;
    Ball()
    {
        size = 10;
    }
    Ball(int size)
    {
        this.size = size;
    }
    void display(){ System.out.println("size:" + size);
    }
}
class main
{ public static void main(String args[])
```

```
Ball b1 = new Ball();
Ball b2 = new Ball(25);
b1.display();
b2.display();
}
}
```

* There is no destructor but by default garbage collector by jvm works itself.

for explicitly calling garbage collector
:- System.gc();

⇒ WAP to create a class arithematic which contains a method addition, subtraction, multiplication and division.

⇒ Create appropriate constructor for the same.

```
import java.util.Scanner;

class Arithematic
{
    int a, b;
    Arithematic(int a, int b)
    {
this.a = a;     Scanner sc = new Scanner();
this.b = b;     a = sc.nextInt(); 20;
                b = sc.nextInt(); 10;
    }
    void addition()
    {
        System.out.println("addition = "+
                                (a+b));
    }
void substraction()
    {
        System.out.println("substraction="
                                + (a-b));
    }
void multiplication()
    {
        System.out.println("multiplication="
                                + (a*b));
    }
```

```
Void division()
{
System.out.println("division=" +
                   (a/b));
}
}
Class Main
{
public static void main(String args[])
{
  Arithematic op = new Arithematic(20,10);
op.addition();
op.substraction();
op.multiplication();
op.division();
}
}
```

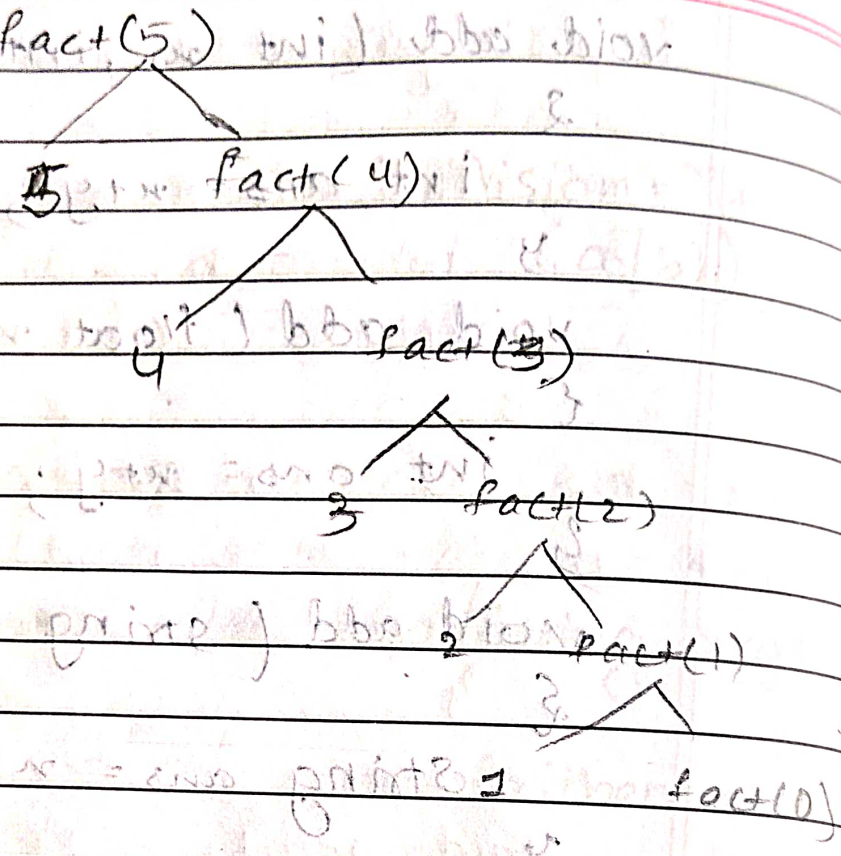
Method Overloading

```
Class Test
{
  void add(int x)
  {
    x = x+10;
  }
```

```
void add (int x, int y)
  {
     int ans = x+y;
  }
  void add (float x, float y)
  {
     int ans = x+y;
  }
  void add (string x, string y)
  {
     string ans = x+y
  }
}
```

Recursion

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



2) fibonacci Series

6. Create a class employee having employeeno., departmentno. and salary as data members, create appropriate constructor & method to initiliaze the value. Create a class account having data members like total expenditure. Create a method calculate expense which calculate total salary drawn by each employee.

Q. WAP to add two numbers of two different object.

```
class Number
{
int x;
Number (int x)
{
this.x = x;
}
void add (Number n)
{
int ans = this.x + n.x;
Sys..("answer=" + ans);
}
main()
{
Number obj1 = new Number(10);
Number ob2 = new Number(20);
obj1.add(ob2);

}
}
```

Array of Object

```
Class-Name  Arry_Name[] = new
                          Classname [size];
```

```
Class x
{
   int x1, x2;
x(int x1, int x2)
 {
   this.x1 = x1;
   this.x2 = x2; }

main ()
{
   X Obj[] = new X[10];
Obj[0] = new X (30,30);
Obj[1] = new X(40,40);
```

$x_1 = 30$	X[0]
$x_2 = 30$	
$x_1 = 40$	X[1]
$x_2 = 40$	
x_1	X[2]
x_2	

Q. WAP to create a class A having A1 as datamember, create array of 5 objects, instantiate it and display value.

⇒

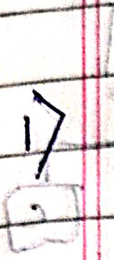
```
class A
{
    int a1;
    A(int a1)
    {
        this.a1 = a1;
    }
    void display()
    {
        System.out.println("a1 = " + a1);
    }
    public static void main(String args[])
    {
        A obj[] = new A[5];
        {
            obj[i] = new A[i+1];
            obj[i].display();
        }
    }
}
```

0	$a_1 = 1$	obj[0]
1	$a_1 = 2$	obj[1]
2	$a_1 = 3$	obj[2]
3	$a_1 = 4$	obj[3]
4	$a_1 = 5$	obj[4]

=> Returning obj from method

```
class Number
{
   int no;
Number ( int no)
   {
      this.no = no;
   }
Number intTen()
   {
      Number temp = new Number(10);
      temp.no += 10;
      return (temp);
   }
public static void main (String args[])
   {
      Number n1 = new Number()
      n1 = n1.IntTen Number(20);
   }
}
```

Inheritance



1) Single - level inheritance

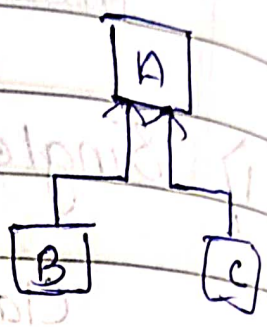
```
Class A
{
    int a;
}
Class B extends A
{
    int b;
}
```

2) Multi-level

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

3) Hierarchical

```
class A
{

}

class B extends A
{
}

class C extends A
{
}
```



Q. WAP name shape having method calculate area further create a class square and rectangle which inherit shape class overwrite the method calculate the area to find out area of Square & rectangle.

Constructor channing

```
1) Class A
   {
      A()
      {
        S.O.P("I am A");
      }
   class B extends A
   {
      B()
      {
        S.O.P("I am B");
      }
   class C extends B
   {  C()
      S.O.P ("I am C");
   }
     main()
   {
      C Obj = new C();
   }
}
```

2)

```
class A
{
    A()
    {
        System.out.println("I am A");
    }
}
class B extends A
{
    B()
    {
        System.out.println("I am B");
    }
}
class C extends B
{
    C()
    {
        System.out.println("I am C");
    }
}
class Test
{
    public static void main(String args[])
    {
        C obj = new C();
    }
}
```

Super Keyword

- Super keyword is used to refer to data member of immediate parent

1)

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

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

2>

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

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

Method overridding (Super)

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

```
public static void main (String args[])
{
    B obj = new B();
    obj.disp();
}
}
```

Abstract Class

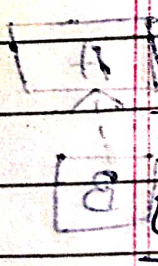
```
abstract class A
{
    void sayHello()
    {
        System.out.println ("Hello");
    }
abstract void sayHi();
}

class B extends A
{
    //override
    void sayHi()
    {
    System.out.println ("Hi");
    }
}
```

Q. WAP name shape having abstract method calculate area. Create two sub classes of shape name square and rectangle and overwrite calculate area method.

=>

```
abstract class shape
{
    float area;
    abstract void calculate_area();
}
class square extends shape
{
    int l;
    void calculate_area()
    {
        area = l*l;
        System.out.println("Area=" + area);
    }
}
Class Rectangle extends shape
{
    int l, b;
    Rectangle(int l, int b)
    {
        this.l = l;
```

```
        this.b = b;
    }
    void calculate_area()
    {
        area = l * b;
        System.out.println("Area = " + area);
    }
}
class Test
{
    public static void main(String args[])
    {
        Test
        Square obj = new Square(10);
        Rectangle object = new Rectangle(10,20);
        obj.calculate_area();
        object.calculate_area();
    }
}
```

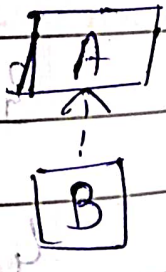


Interface

- An interface in java contains static constant and abstract method.
- It is used to achieve abstraction, since java 8 default and static method are allowed in interface.
- Since java 9 private methods are allowed in interface, it is used to achieve coupling.

Syntax:

```
interface interfaceName
{
  // Static final variable
  // abstract method
}
```



Example: interface shape

```
{
    public void calcArea();
}
```

```
interface A
{
    void getA();
    void dispA();
    int increA(int n0);
}
class B implements A
{
    public void getA()
    {
        System.out.print("a is 10");
    }
}
```

* We can achieve multiple inheritance using interface.

* One interface cannot implement another interface

* One interface can extend another interface.

```
interface A
{
    void disp();
```

25/08/22

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

Example:

interface a
{
int no=121;
void sayA();
default void sayHi()
{
System.out.println("Hi");
}
}
interface b extends a // b can extends
{
    void sayB();
}
abstract class Test implements a, b ->
{

public void sayA()
{
//no++; final variable can't
change the value.
```

```
    System.out.println("Hello A");
}
public static void main(String args[])
{
    Test obj = // = new Test(); // Abs-
                                tract class
                                can't be instantiate
    //obj.sayA();
}
}
```

a but can't implement a.

// must override, say "B" else abstract class

```
class TestA
{
    void sayA()
    {
        System.out.println("A");
    }
}
class TestB extends A
{
    void sayA() //method overridding
    {
        System.out.println("B");
    }
}
class DyanamicD
{
public static void main(String args[])
{
TestA objA;
 Obj = new TestA();
Obj.sayA(); //A Run time polymorphism
 obj = new TestB(); // upcasting
 Obj.sayA(); //B Run time polymorphism
 TestB obj1 = new TestB();
  Obj1.sayA();
}
}
```

:|09

'final' Keyword

final datatype VariableName = value;
final returntype methodName(first Of parameter)
{
}
className.

- You can't change the value
- you can't override
- you can't extend it.

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

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

class childTest extend Test final
↘ error

error { void disp()
{
S.O.P. ("Hello");
}
}

Keywords

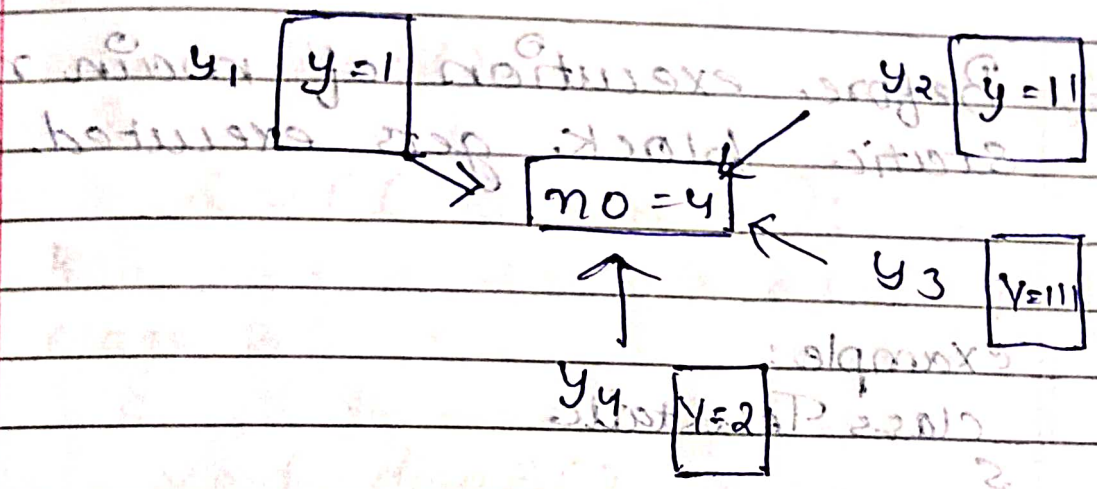
⇒ Static variable is shared among all object which has only one memory object.

⇒ Static variable

Example:

```
class y
{
int y;
static int no;
y(int y)
{
this.y = y;
no++;
}
main
{
y y1 = new y(1)
```

```
Y y1, y2 = new Y(1);
Y y3 = new Y(11);
Y y4 = new y(21);
```



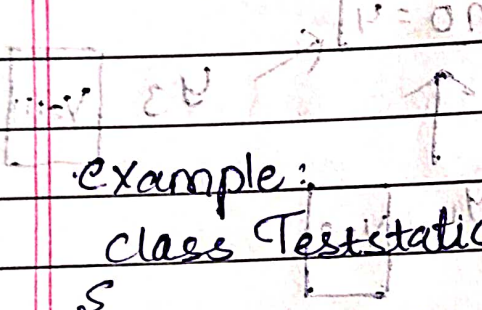
Static method example

```
class Test
{
  void disp()
  {
    S.O.P("Hello");
  }
  Static void sayHi()
  {
    S.O.P("Hi");
  }
}
main()
{
  sayHi();
  Test obj = new Test(); obj.disp();
}
```

In same class one static can directly call another static method.

Static block

=> Before execution of main method static block gets executed.



example:

```
class Teststatic
{
    static {
        s.o.p("static block");
    }
}
Main -> {
    s.o.p("main method");
}
```

o/p: static block
main method

8/03/23

Nested class

Nonstatic Static

Nonstatic example:

```
class A
{
  void dispA()
  {
    S.O.P("I am A");
  }
class B
  {
    void dispB()
  {
    S.O.P("I am B");
  }
}
class Test
{  P
{
  A obj = new A();
    obj.dispA();
Nonstatic: A.B obj1 = obj.new B();
// for static method in class B:
```

```
A.B obj1 = new A.B(); //
obj1.dispB();
}
}
```

Inner class

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

```
Public
{
A obj = new A();
   obj.dispA();
}
}
```

Anonymous Inner class

```
interface X
{
   void disp();
}
class Main
{
  Public
{
   X obj = new X(){
  public void disp()
  {
     S.O.P("I am A");
  }
  };
   obj.disp();
}
}
```

Using abstract class

```
abstract class X
{
abstract void disp();
}
class Main
{
public
{
    X obj = new X(){
    void disp()
    {
        s.o.p("I am A");
    }
};
    obj.disp();
}
}
```

⇒ Object class

It is parent of every class.

Finalize Method

```
class Test
{
  protected void finalize()
  {
    S.O.P("    ");
  }
  main
  {
    Test o = new Test();
    o = null;
    System.gc();
  }
}
```

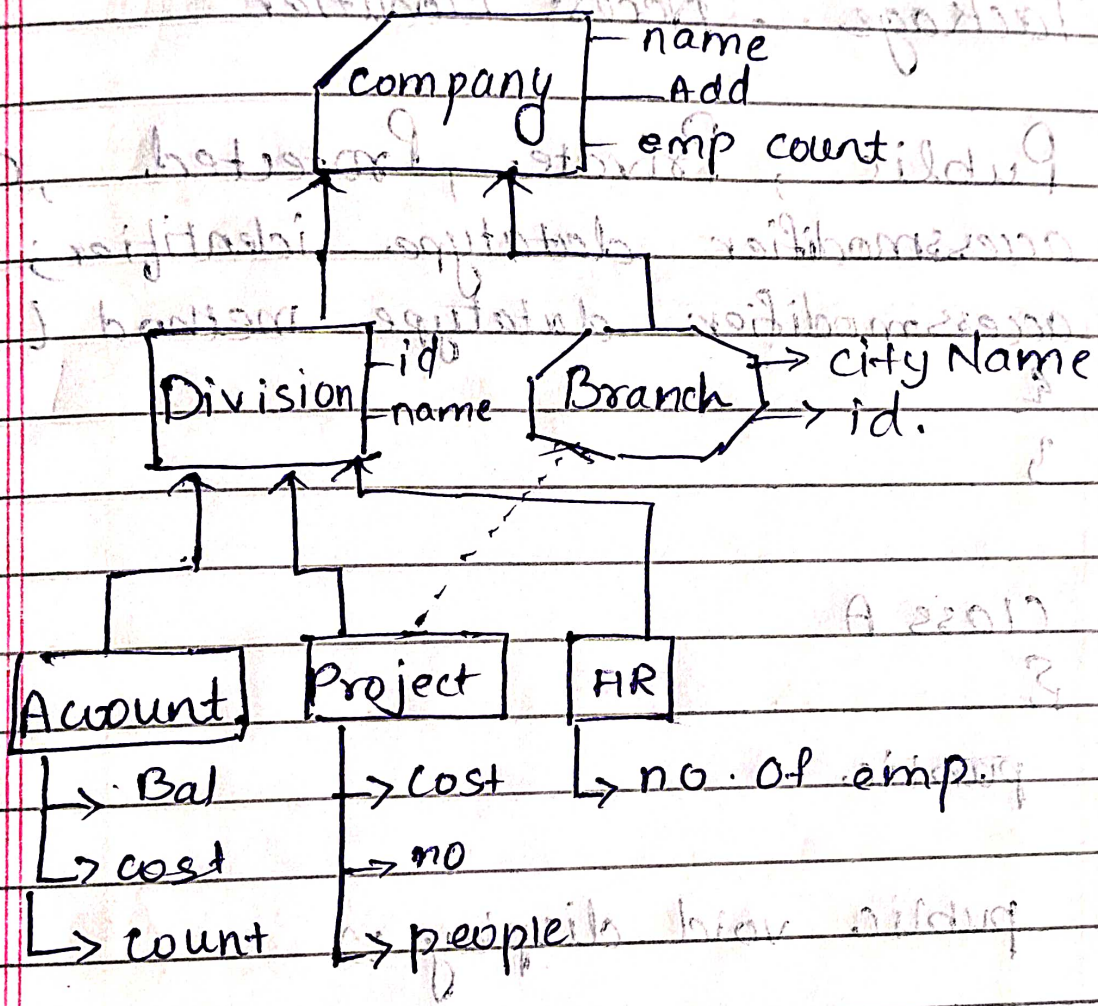
Object

i) tostring() - hashcode to string

ii) hashcode() - Internal representation of address.

⇒ Package

```
Pakage prgname;
import static java.lang.System;
class Test
{
    main
    {
    out.println("Hello world");
    }
}
```


Company
name
Add
emp count
Division
id
name
Branch
city Name
id.
Account
Bal
cost
count
Project
cost
no
people
HR
no. of emp.

Package - Access Modifier

Public, Private, Protected, default

accessmodifier datatype identifier;

accessmodifier datatype method ()

{

}

class A

{

public

public void display()

{

}

}

Visibility	Public	Protected	Private	default
from same class	✓	✓	✓	✓
from any class in same Pkg	✓	✓	✗	✓
from subclass in pkg	✓	✓	✗	✓
from subclass outside the same pkg	✓	✓	✗	✗
from any known Subclass outside the pkg	✓	✗	✗	✗

Example: Package Mypkg;

```
class A
{
public String str = "Hello";
       string msg = "GSFCU";
}
```

```
Package Mypkg;
Class B
{
  public static void main(String[] args
{
     A obj = new A();

     obj.str;
     obj.msg;

}
```

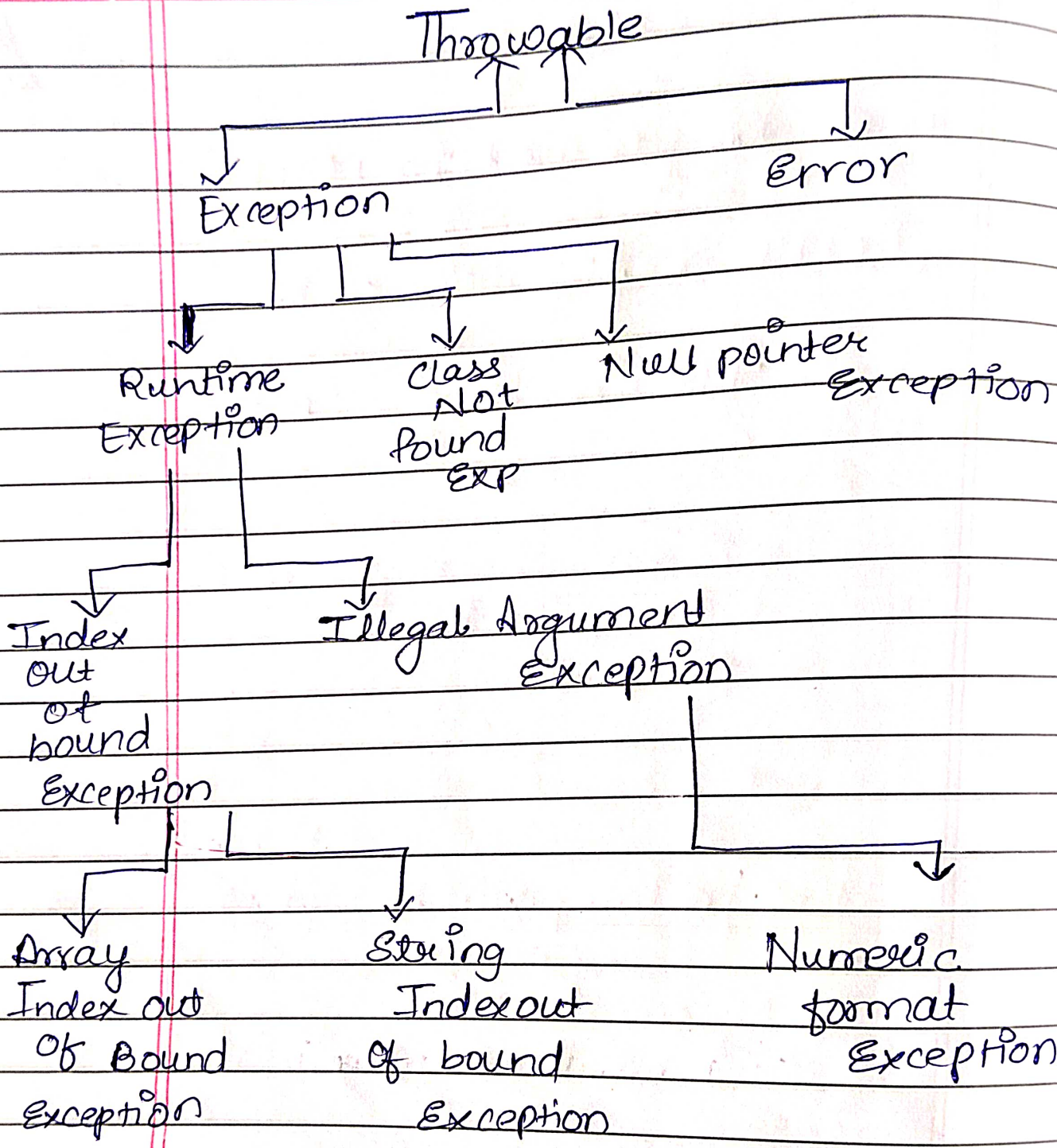
Unit - 3

Exception Handling

Types
-> cheked , uncheked
-> try , catch, finally

```
try
{

}
catch (--)
{

}
catch (--)
{

}
catch (--)
{

}
finally
{
}
```

After a catch block it goes to finally and then does not go back for remaining code lines.


Throwable
Exception
Error
Runtime Exception
Class Not found EXP
Null pointer Exception
Index out of bound Exception
Illegal Argument Exception
Array Index out of Bound exception
String Index out of bound Exception
Numeric format Exception

Java Interface

Throwable getCause()
String toString()
void printStackTrace()

Q. Write a program to create a method check which takes string as input argument. This method throws input exception if the string does not contains vowels.

```
class vowel extends Exception
{
 System.out.println("Enter a string");
 Scanner sc = new Scanner(System.in);
 String str = sc.nextLine();

    Vowel(String msg)
    {
      super(msg);
    }
}
class CheckString
 {
 Void check(String str) throws
                          vowel
```

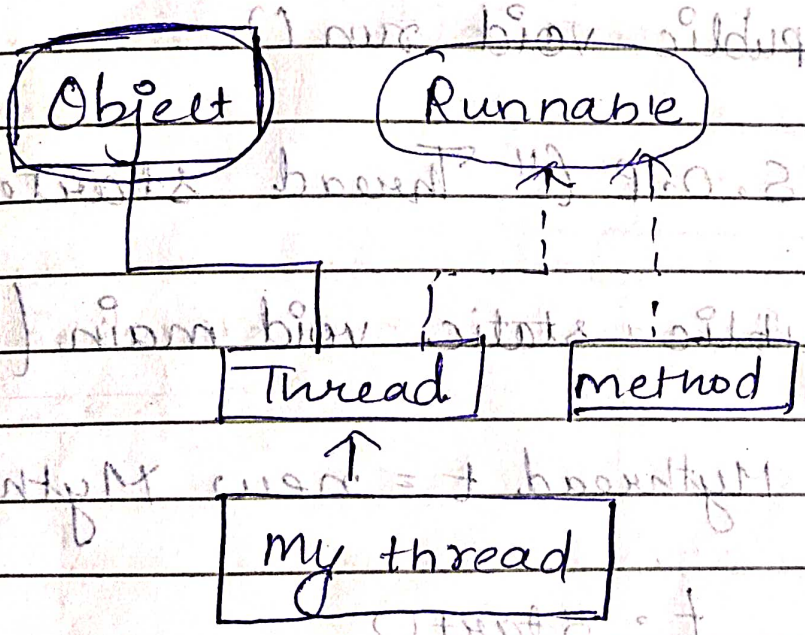
```
{
    bool f = false;
for(int i=0; i<str.length(); i++)
 {
  char ch = str.charAt(i);

if(ch=='a' || ch=='i' || ch=='o' ||
                    ch=='e' ||
                    ch=='u')
{
        f = true;
        break;
 }
}
 if(f)
  System.out.println("  ");
 else
  throw new Vowel("  ");
}
```

There are two ways to create a thread:

1) By extending thread class
2) By implementing Runnable interface

→ Thread class: provides constructor and methods to create and perform operation on a thread

→ Thread class extends object class and implement runnable interface



Constructor of Thread class

Thread()
Thread(String s)
Thread(Runnable r, String n)
Thread(Runnable r)

Methods

Start()

sum()

getName()

setName(String)

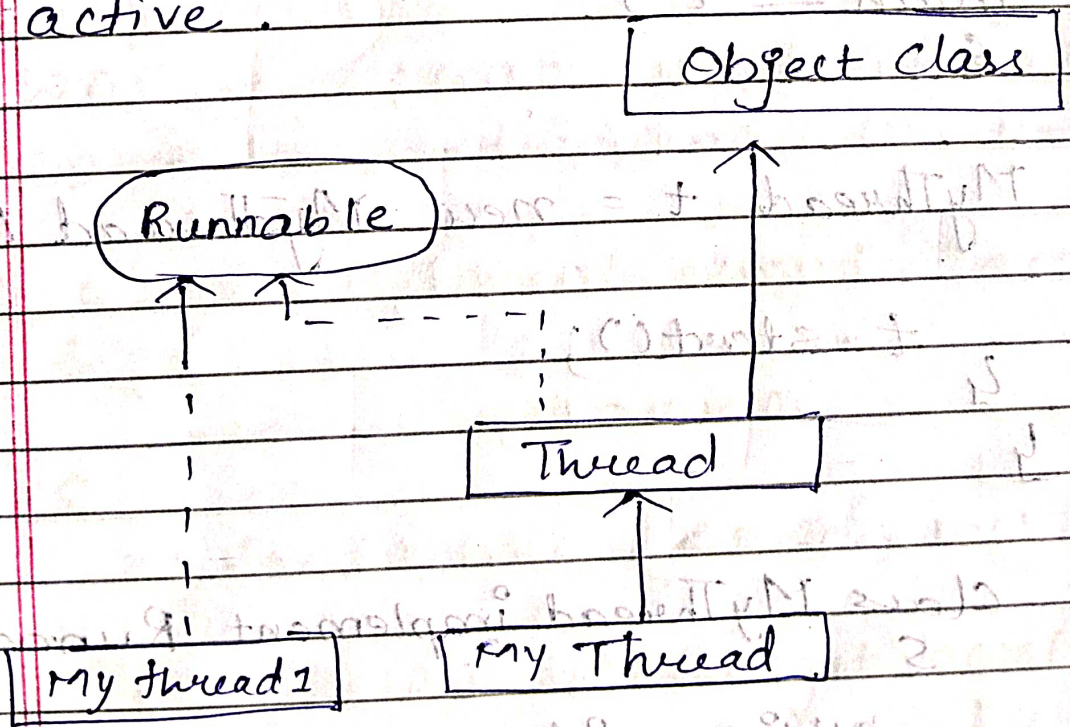
getPriority()

setpriority(int-priot)

```
class Mythread extends Thread
{
    public void run()
    {
        S.O.P("Thread started");
    }
    public static void main(String args
    {
        Mythread t = new Mythread();
        t.start();
        Mythread t1 = new Mythread();
        t1.start();
    }
}
```

4/10/23

- Runnable interface contains a method void run().
- The Runnable interface should be implemented by any class whose instances are intended to be executed by thread.
- The class must define a method run, this interface is designed to provide a common protocol for objects that wish to execute the code while they are active.



WAP to create a thread to print Hello world 1) by extending thread class 2) implementing runnable interface

1)
```
class MyThread extends Thread
{
public void run()
{
System.out.println("Hello world");
}
main ()
{

MyThread t = new MyThread();
t.start();
}
}
```

2)
```
class MyThread implement Runnable
{
public void run()
{
System.out.println("Hello world");
```

```
}
____ main( ____ )
{
  MyThread o = new Mythread();
  Thread t = new Thread(o);
    t.start();
  }
}
```

* We cannot use thread class when we need to extend another class

Q. WAP to create two thread which prints their name 10 times.

```
class MyThread extend Thread
{
    public void run()
{   int i;
    for(i=0; i<=10; i++)
  {
    S.O.P(this.getName()+" is running");
  }
main __ ( )
{
```

```
MyThread t1 = new MyThread();
t1.setName("t1");
t1.start();
MyThread t2 = new MyThread();
t2.setName("t2");
t2.start();
}
}
```

Q. WAP to print Hello world sum of numbers from 1 to 20.

```
class SumThread extends Thread
{
  int l, u, sum;
  SumThread(int l, int u, String name)
  {
    super(name);
    this.l = l;
    this.u = u;
  }
  public void run()
  {
    for(int i=l; i<=u; i++)
    {
      this.sum+ = i;
```

```
        this.sum += i;
        System.out.println (this.getName()
                            + " is running");
    }
    System.out.println("Sum by " +
            this.getName() + " \ + "
            + this.sum
            + is running");
    }
    main()
    {
        SumThread t1 = new SumThread
                                (1, 10, "+1");
        t1.start();
        SumThread t2 = new SumThread
                                (11, 20, "+2");
        t2.start();
    }
}
```

To run java sumthread.java

5/10/23

⇒ Join()

Signature: public final void join()
throws Interrupted exceptio

wait for this thread to die

⇒ Sleep()

Signature: Public static void sleep(long millisecond) throws interrupted exception

Sleep method cause the current executing thread to sleep for speitic no. of miliseco Subject to preision and accuracy of system, timer & scheduler

⇒ Wait()

Signature: public final void wait()
throws
~~that should~~ interrupted exception

causes the current thread to wait until another thread

Python

invoke the notify or notify all method for this object

Q. WAP to print sum of numbers from 1 to 20 using join

```
class SumThread extends Thread
{
    int l, u, sum;
    SumThread (int l, int u)
    {
        this.l = l;
        this.u = u;
    }
    public void sum ()
    {
        for(i = l; i <= u ; i++)
        {
        this.sum+ = i;
    }
    public — main
    {
        SumThread s1 = new SumThread
                                (1, 10);
    SumThread s2 = new SumThread
                                (11, 20);
```

```
s1.start();
s2.start();
try {
t1.join();
t2.join();
}
catch (InterruptedException e)
{
    S.o.p("Interrupted");
}
S.o.p("sum" + (t1.sum + t2.sum))
}
```

Now ⇒

Lab

Q 13

```
class sThread extends Thread
{
sThread (String name)
{
    super(name);
}
public void run()
{
for(int i=0; i<10; i++)
{
```

sleep is static so, it can be already called by thread class.

```
    Thread.sleep(5000);
    Thread.sleep(1500);
}
catch (InterruptedException e)
{
    e.printStackTrace
}
S.o.P(this.getName());
}
public main()
{
    SThread t1 = new SThread("Hello");
    SThread t2 = new SThread("world");
    t1.start();
    t2.start();
}
try {
    t1.sleep();
    t2.sleep();
}
catch (InterruptedException e)
{
    s.o.p("In
```

06/10

Sychronization

- In Java is the capability to control the access of multiple threads to any shared resources
- The sychronisation is mainly used
  i) To prevent thread interference
  ii) To prevent consistency problem.
- It can be achieved by using sychronised method or block.

⇒ WAP to perform bank withdrawal system using threading

```
Class BankThread extends Thread
{
    int Balance;
    int Amount;

BankThread
```

```
class Table
{
synchronized print (int n)
    {
        for (    )
        {
            S.O.P ——;
        }
    }
}
class my+hread extend Thread
{
    Table rest;
```

FileInputStream

Constructor

1) FileInputStream(File f)
2) FileInputStream (String name)

Method

1) int available - returns int
2) int read
3) ~~int byte array read()~~
3) int read(byte [] b)

Q. WAP to read file using FileinputStream.

```
import java.io.FileInputStream;
import java.
Class TestReadFile
{
    main
    {
        try
        {
```

```
FileInputStream fin = new FileInputStream
                        (in.txt);

  while (fin.available() > 0)
{
int c = fin.read();
  S.O.p ((char) c);
  }
}
catch (Exception e)
{
    e.printStackTrace();
  }
}
}
```

Q. WAP to create a copy of a file.

Constructor

1) FileOutputStream (File f)
2) FileOutputStream (String name)
3) FileOutputstream (File f, bool)
4)

Method

1) close()
2) write(int)
3) write(byte[])

```
import java.io.FileInputStream;
import java.io.FileOutputStream;

class TestFile
{
  main
  {
    try
    {
      FileInputStream fin = new
                  FileInputStream("in.txt");
      FileOutputStream fon = new
                  FileOutputStream("out.txt");
      while(fin.available()>0)
      {
        int c = fin.read();
        fon.write(c);
      }
```

```
catch (Exception e)
{
    e.printStackTrace();
}
}
}
```

① DataInputStream

- A dataInputStream lets an application read primitive java datatypes from an underline inputstream in a machine independent way.

- An application uses a data outputstream to write a data that can later be read by a dataInputStream.

Constructor

1) DataInputStream (InputStream ____)

Method

1) readInt()
2) readFloat()
3) readDouble()
4) readChar()

5) string readUTF()

6) readLong()

7) readbyte()

Q WAP to read 10 nos from a file and print the summation of it.

```
import java.io.FileInputStream;
import java.io.DataInputStream;

class sum
{
main
{ int sum=0;
try
{
FileInputStream F=new FileInputStream
                    ("sum.txt");
DataInputStream D = new DataInputStream
                    (F);
while(d.available >0)
{
int n = d.readInt();
```

```
            sum = sum + n;
        }
        s.o.p("sum=" + sum);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

WAP to create a class student having instance variable like, Id, name & CGPA. Create array list of 5 students

```
import java.util.ArrayList;
import java.util.Iterator;
class Student
{
    int Id;
    String name;
    double CGPA;
public static void main (String
                        args[])
{
    Student (int id, String name,
                double CGPA)
    {
        this.id = id;
        this.name = name;
        this.CGPA = CGPA;
    }
Vector void display ()
    {
        S.O.P ("id " + this.id");
        S.O.P ("name " + this.name);
```

```
        S.O.p("CGPA" + this.CGPA);
    }
}
class DemoArrayList
{
    public main (a)
    {
        Student S1 = new Student(1, "Riya", 8.9
        Student S2 = new Student(2, "Rima", 9.9f
        Student S3 = new Student(3, "Rita", 8.9f
        ArrayList<Student> = new ArrayList
                                <Student>();
        A.add(S1);
        A.add(S2);
        A.add(S3);
    Iterator<Student> i = A.iterator();
    while (i.hasNext())
    {
        Student obj = i.next();
        obj.display();
    }
```

Q. WAP to print a string using multithreaded programming make sure at one time only one string should be printed.

⇒ Class Multithread extends Thread
{
Multithread (String name)
{
super (name);
}
public void run ()
{ int i;
for (i=0; i<=2; i++)
{
s.o.p (this.getName () + "I am Thread ");
}

Sychronised void display ()
{
s.o.p (this.getName () + "I am thread ");
s.o.p (this.getName () + "I am thread 2");
s.o.p (this.getName () + "I am thread 3");
}

—main()—

{

Multithread t1 = new Multithread();
t1.setName("+1");
Multithread t2 = new Multithread();
t2.setName("+2");
Multithread t3 = new Multithread();
t3.setName("+3");

}

}

}

Q. WAP to create a simple web page server that can handle multipe client request simultaneously implement sychronization to ensure proper handling.

class web extends Thread

{

Q. WAP to calculate total expense on trip by Five friends and split the bill among five. Use multithreading concept to ask expense done each member and then calculating total expense & splitting the bill.

⇒

```
class Demothread extends thread
{
    double tot-exp;
    double exp;
```

```
DemoThread (double exp)
{
    this.exp = exp;
}
public void run()
{
    tot-exp += exp;
}
—main—
{
DemoThread t1 = new Demothread
                                (1000);

float avg = tot-exp/5;
f1.start();
f2.start();
f3.start();

f1.join();
f1.join.
```

1/11/2023

Javafx

Javafx is java library used to develop desktop application. The application built on Javafx can run on multiple platforms including mobile, desktop

It has three main component :-
1) scene builder
2) FXML
3) controller

Javafx. application package :

Application class :

~~Life cycle~~: init()
start()
stop()
} methods.

1) init() - initialization method type void.

2) Start() - Public abstract void
Start (Stage primary sta

9. stop() - public void stop() throws Exception.

This method is called when the application should stop.

Life cycle: The entry point of Javafx application is the Application class. The Javafx

1) constructs an instance of specified Application class
2) calls init() method
3) calls the start(javafx.stage.Stage) method
4) waits for the application to finish, which happen whens
   - platform.exit()
5) stop()

Methods of Javafx.

Set max height
set min height
set max width
set scene
set title
Show()
show and wait()
toBack()
tofront()

```
import javafx.application.Application;
import javafx.stage.Stage;
public class Hello_word extends Applicatio
```

Code ⇒

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;

public class MyFirstJavafx extends
  Application {

    Public void start(Stage primary
                              Stage)
{   Button btOK = new Button("OK");
```

```
Scene scene = new scene (bt OK, 200,
                                250);

primarystage.setscene(scene);
primarystage.setTitle("My JavaFX")
primarystage.show();

}

pub ——— main() ———
{
    launch(args);
}
}
```

Text Class

Constructors

Text()

creates an empty instance of Text.

Text(double x, double y,

Method

⇒ GetText()

Gets the value of property text

Setfont()

Gets the value of property text

⇒ Creating textbox

~~public class Text extends shape~~

import javafx.scene.text;

Text t = new Text(10, 50, "This is a test");

t.setFont(new Font(20));

import javafx.scene.text.*;

Text t = new Text();

text.setFont(new Font(20));

Line Class

Constructors:
- Line()
- Line(double startx, starty, endx, endy)

Example :-

```
import javafx.scene.shape.*;
Line line = new Line();
line.setStartX(0.0f);
line.setStartY(0.0f);
line.setEndX(100.0f);
line.setEndY(100.0f);
```

Rectangle class

Constructor
- Rectangle()
- Rectangle(double width, double height)
- Rectangle(double x, double y, double width, double height)
- Rectangle(double width, double height, Paint fill)

Example:

```
import javafx.scene.shape.*;
Rectangle r = new Rectangle();
```

```
r.setX(50);
r.setY(50);
r.setWidth(200);
r.setHeight(100);
r.setArcWidth(20);
r.setArcHeight(20);
```

⇒ Circle Class

Constructor

```
circle()
circle(double radius)
circle(double centerx, double
       centrey, double radius)
circle(double centerx, double
       centrey, double radius,
       Paint fill)
```

Example:

```
Circle circle = new circle();
circle.setCentreX(100.0f);
circle.setCentreY(100.0f);
circle.setRadius(50.0f);
```

Ellipse class
Arc class
Polygon
Polyline
* Pane
* Image class
constructor
Image(String url, double requestedWidth, double requestedHeight, boolean preserveRatio, boolean smooth).