Sanjay Adhikari

Quicksand Official

9848067387
bajeyz@gmail.com
quicksandofficial.com.np
lazimpat kathmandu

Java

console Application | web | standalone/desktop | Android

Java was released on 1991 AD → OAK (Green project)
1995 → Java
Main contribution → James Gosling, Mike Sherinium, Patrick Naughton)

Java is a pure object oriented.

Features of Java :
i) object oriented.
ii) platform independent
write once, run everywhere.

Java
- JDK — java development Kit
- JRE — Java Runtime Environment
  - → JVM (Java virtual Machine) (interpreter)

A.java
↓
A.class → bytecode)

iii) Architecture Neutral
(Memory & cpu resource changes → no effect)
iv) compiled and Interpreted
v) Multithreaded
vi) Robust.

comment // or /* */

```
//program to display "welcome to java"
import java.lang.*;  or import java.lang.System;
class Welcome        or import java.lang.String;
{
public static void main(String args[])
{
System.out.println("Welcome to Java");
}
}
```

double is a string double a; stack
Double is a class Double b; heap.

package is a collection of class.

(java.lang.String args[]) → fully qualified class name

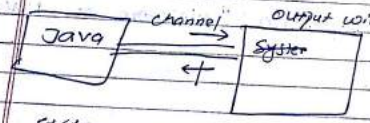
Naming convention

class name → Person
↳ PersonDetails

method → show()
↳ showInfo()

print → cursor in current line.

```
Class Welcome
{  public static void main(String args[])
   {  System.out.println("Welcome to java");
   }
}
```



String args is used to hold command line arguments.

①

cmd :

javac Welcome.java
java Welcome 10 20 ← command line argument.

// Addition of two numbers using command line argument.

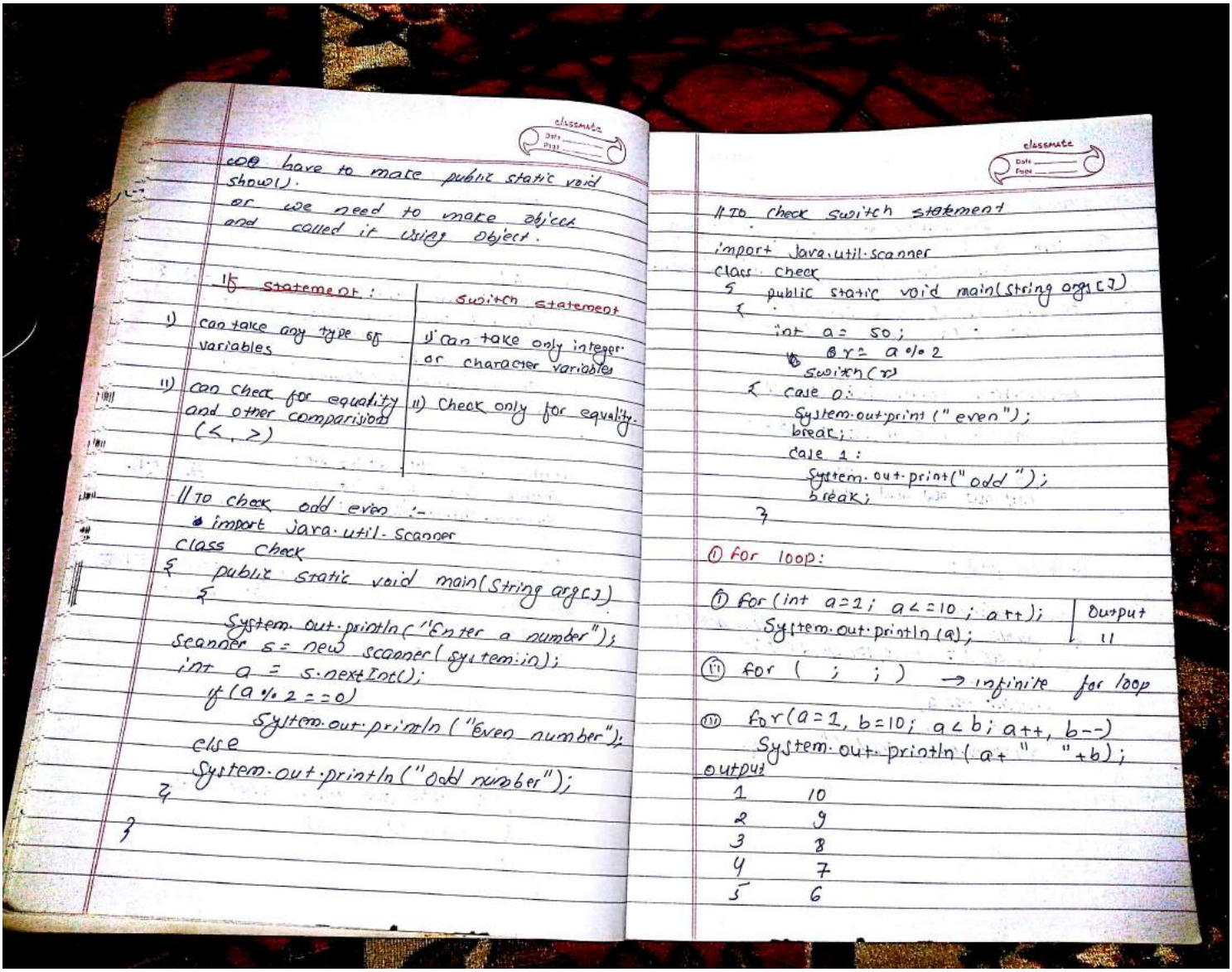
```
Class Add
{  public static void main(String args[])
   {  int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      int s = a+b;
      System.out.println("sum: " + s);
   }
}
```

```
double b = 4.0, h = 3.0;
double a = 1/2 * b * h;          1/2 = 0.5
System.out.println(a);           (answer=0)
```

%

10 % 3 = 1 जसलाई divide जातिल्ह
-10 % 3 = -1 दोघांचा sign
10 % -3 = 1
-10 % -3 = -1

```
Class X
{  public show()
   {
   }
   public static void main(String args[])
   {  show();  (X)
   }
}
```



code have to make public static void show().
or we need to make object and called it using object.

If statement :	switch statement
i) can take any type of variables	i) can take only integer or character variables
ii) can check for equality and other comparision (<, >)	ii) Check only for equality.

// To check odd even :-

```
import java.util.Scanner
class check
{ public static void main(String args[])
  {
    System.out.println("Enter a number");
    Scanner s = new Scanner(System.in);
    int a = s.nextInt();
    if(a%2==0)
      System.out.println("Even number");
    else
      System.out.println("Odd number");
  }
}
```

// To check switch statement

```
import java.util.Scanner
class check
{ public static void main(String args[])
  {
    int a = 50;
    int r = a%2
    switch(r)
  { case 0:
      System.out.print("even");
      break;
      case 1:
        System.out.print("odd");
        break;
  }
```

① for loop:

① `for(int a=1; a<=10; a++);` | Output
`System.out.println(a);` | 11

(ii) `for( ; ; )` → infinite for loop

(iii)
```
for(a=1, b=10; a<b; a++, b--)
  System.out.println(a+" "+b);
```

Output

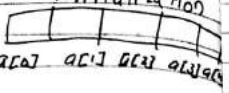
1	10
2	9
3	8
4	7
5	6

Array :

Declaration & without initialization

① int a[]

a = new int[5];



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

int []a = new int[5];

Declaration with initialization

int a[] = { 5, 10, 15, 20, 25 };

→

5	10	15	20	25
a[0]	a[1]	a[2]	a[3]	a[4]

Foreach loop :

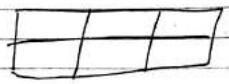
```
class SumArray
{  public static void main (String args[])
{  int a[] = {5, 10, 15, 20, 25 };
   int s = 0;
   for(int m: a)
       s += m;
   System.out.println("Sum of
```

2-D Array

without intialization.

int a[][] = new int [2][3];

or int [][]a = new int [2][3];



with Intialization

int a[][] = {{1, 2, 3}, {4, 5, 6}};

1	2	3
4	5	6

~~for~~ int s = 0

```
int a[][] = {{1, 2, 3}, {4, 5, 6}};
for(int m[]: a)
{   for (int n: m)
        s += n;
}
System.out.println("The sum is" + s);
```

Class and object

→ A class is a template for an object

→ A class is a logical entity ie- it does not have physical existence (memory is not allocated for it)

for eg:

```
Class Person
{
    private String name;
    private int age;
}
```

Memory is allocated only when an object is created.

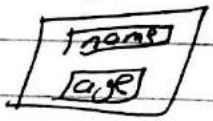
The object of above class can be created as:

a) Person p;
   p = new person();

b) Person p = new Person()

when

new person() means orphan object
no identity.
no reference object.



The Defect of this program is all the objects are intialized with the same value

to solve it, we can use parameterized method.

```
class Person
{   private String name;
    private int age;

public void setInfo(String n, int a)
{   name = n;
    age = a;
}
public void showInfo()
{
    System.out.printf("Name : "+ name);
    System.out.printf("Age: "+ age);

}
public static void main(String args[])
{   Person p = new Person();
        p.setInfo("Ram", 25);
        p.showInfo();
}
}
```

Method overloading:

→ Using the same methods several times in a class but with different return type or different no. of arguments.

→ Also called compile time polymorphism/ static binding/ early binding.

```
Class MethOverload
{   double dim1, dim2;
    public void setDim()
    {   dim1 = 2.5;
        dim2 = 3.8;
    }
    public void setDim(double d1, double d2)
    {   dim1 = d1;
        dim2 = d2;
    }
    public void area()
    {   double a = dim1 * dim2;
        System.out.println("Area" + a);
    }
    public static void main(String args[])
    {   MethOverload ob1 = new MethOverload();
        MethOverload ob2 = new MethOverload();
        ob1.setDim();
        ob1.area();
        ob2.setDim(5.5, 3.2);
        ob2.area();
    }
}
```

Constructor:

→ A special method that is used to initialize the instance variable.

→ It is special in the sense that it is exactly same as class name and it doesnot have any return type not even void.

→ It is called automatically when an object is created

```
//Program with constructor

class Person
{   String name;
    double salary;
    Person()
    {   name = "Ram";
        salary = 25000;
    }
    void showInfo()
    {   System.out.println("Name: " + name);
        System.out.println("salary: " + salary");
    }
    public static void main(String args[])
    {   Person p = new Person();
        p.showInfo();
    }
}

//Parameterized constructor

Class Person
{   String name;
```

```
double salary;
Person (string n, double s)
{   name = n;
    salary = s;
}
void showInfo()
{   System.out.println(" name: " + name);
    System.out.println(                    )
}

public static void main (string args[])
  {
    Person p = new Person("Ram", 25000);
    p.showInfo();
  }

}
```

16

Constructor Overloading :

```
person()
{  name = "sita";
   salary = 25000;
}
```

// Program using multiple inheritence:

```
class A
{   int length;
    void setLength(int l)
    {   length = l;
    }
}

class B extends A
{   int breadth;
    void setBreadth(int b)
    {   breadth = b;
    }
}

class C extends B
{   int height;
    void setHeight(int h)
    {   height = h;
    }
    void volume()
    {   int volume = length * breadth * height;
        System.out.println("volume : " + volume);
    }
}
class MultipleInh Demo
```

Use of 'super' keyword

→ It is used
  → To call base class constructor from derived class constructor.
  → To access the hidden member of base class from derived class.

#super use 1

```
Class A
{   int length;
    A(int l)
    {   length = l
    }
}
Class B extends A
{ int breadth;
    B(int l, int b)
    {   super(l);
        breadth = b;
```

sum = 80

#Super use 2:

```
class X
{ int m,n;
  void setData()
  { m=10
    n=20;
  }
}
class Y extends X
{ int n,o; // hides in X (n)
  void setData1()
  { n=50;
    o=20;
  }
  void sum()                    int j = m+super(n)+o;
  { int s= m+n+o;
    System.out.println("sum "+s);
  }
}

class Superuse2
{ public static void main(String args[])
  { Y ob= new Y();
    ob.setData();
    ob.setData1();
    ob.sum();
  }
}
```

Method overriding

using the same method name in the base class and derived class with same method signature.

— Also called runtime polymorphism, dynamic binding or late binding.

Example:

```
class A
{ public void show()
  { System.out.println("Inside A");
  }
}
class B extends A
{ public void show()
{ super.show(); System.out.println("Inside B");
  }
}
class MethOverRide
{ public static void main(String args[])
  { B ob = new B();
    ob.show();
  }
}
```

Useful of "abstract" keyword.

(i) To create abstract method.
→ A method which doesnot have body is called abstract method.
→ its body must be made in the derived class. ie it must be overriden.

(ii) To create abstract class.
→ A class which cannot be instantiated ie whose object cannot be created is called abstract class

(iii) Note: Reference variable of base class can point the object of derived class. However reference variable of derived class can not point the obj of base class

Use of 'final' keyword:

(i) To create constant
e.g final double PI = 3.14;

(ii) To create final method
→ A method that cannot be overridden in the derived class.
eg class A
{
final ~~sho~~ void show()
{
}
}

class B extends A
{ void show() X
{
}
}

(iii) To create final class
→ A class that cannot be inherited
final class A
{
{
class B extends A X
{
}

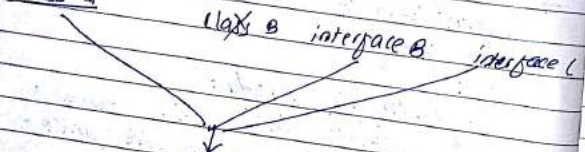
Interface:-
Interface is like class however it contains only static final fields and abstract methods.

→ keyword "interface" is used.
→ The methods of interface must be overridden in the class that "implements" the interface.
Eg:
interface product
{ static final String name = "FAN";
static final double price = 1500;
abstract void show();
}

Implementing Interface:-

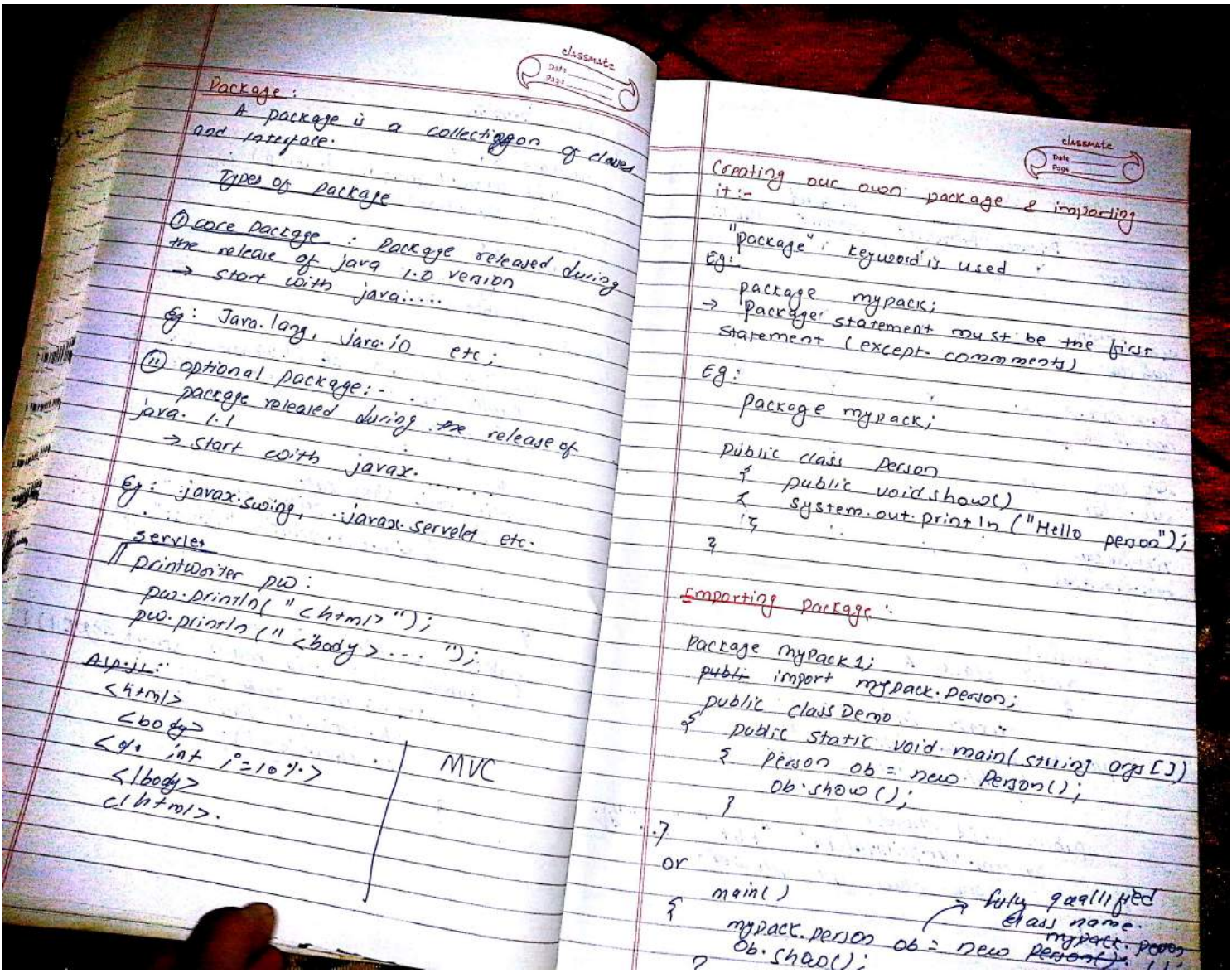
```
interface xY
{   String name= "Pen";
    double price = 1500;
    void showInfo ();
}
class Yz implement xY
{ public void showInfo()
  {  System.out.println("product Name: "+name)
                ("product Price : "+price);
  }
}

class ImplInterface
{ public static void main(String args[])
  {  Yz ob = new Yz();
     ob.showInfo();
  }
}
```

Multiple implements




```
class C extends A implement B,C
{ }
```

```
class ImplIn

interface xY {
    void setInfo(string n, double p) ;
    void showInfo();
}
class zA {
    string name;
    double price;
}

class Yz extends zA implement xY {
    public void setInfo(string n, double p) {
      { name = n;
        price = p;
      }
public void showInfo()
    {  System.out.println("Name: "+name);
       System.out.println("Price: "+price);
    }
}

public class ImplInterface {
    public static void main (String args[]) {
        Yz ob= new Yz();
        ob.setInfo("Pen", 1500);
        ob.showInfo();
    }
```



Package :
A package is a collection of class and interface.

Types of package

① core package : Package released during the release of java 1.0 version
→ start with java....
Eg: Java.lang, Java.io etc;

② optional package:-
package released during release of java. 1.1
→ start with javax.
Eg: javax.swing, javax.servelet etc.

servlet
PrintWriter pw;
pw.println("<html>");
pw.println("<body>...");

Alpha:
<html>
<body>
<% int i=10; %>
</body>
</html>

MVC

Creating our own package & importing it :-

"package" keyword is used.
Eg:
package mypack;
→ Package statement must be the first statement (except comments)
Eg:
package mypack;
public class Person
{ public void show()
{ System.out.println("Hello person");
}
}

Importing package :

package mypack1;
import mypack.Person;
public class Demo
{ public static void main(String args[])
{ Person ob = new Person();
ob.show();
}
}
or
main()
{ mypack.Person ob = new Person(); → fully qualified class name. mypack.Person
ob.show();
}

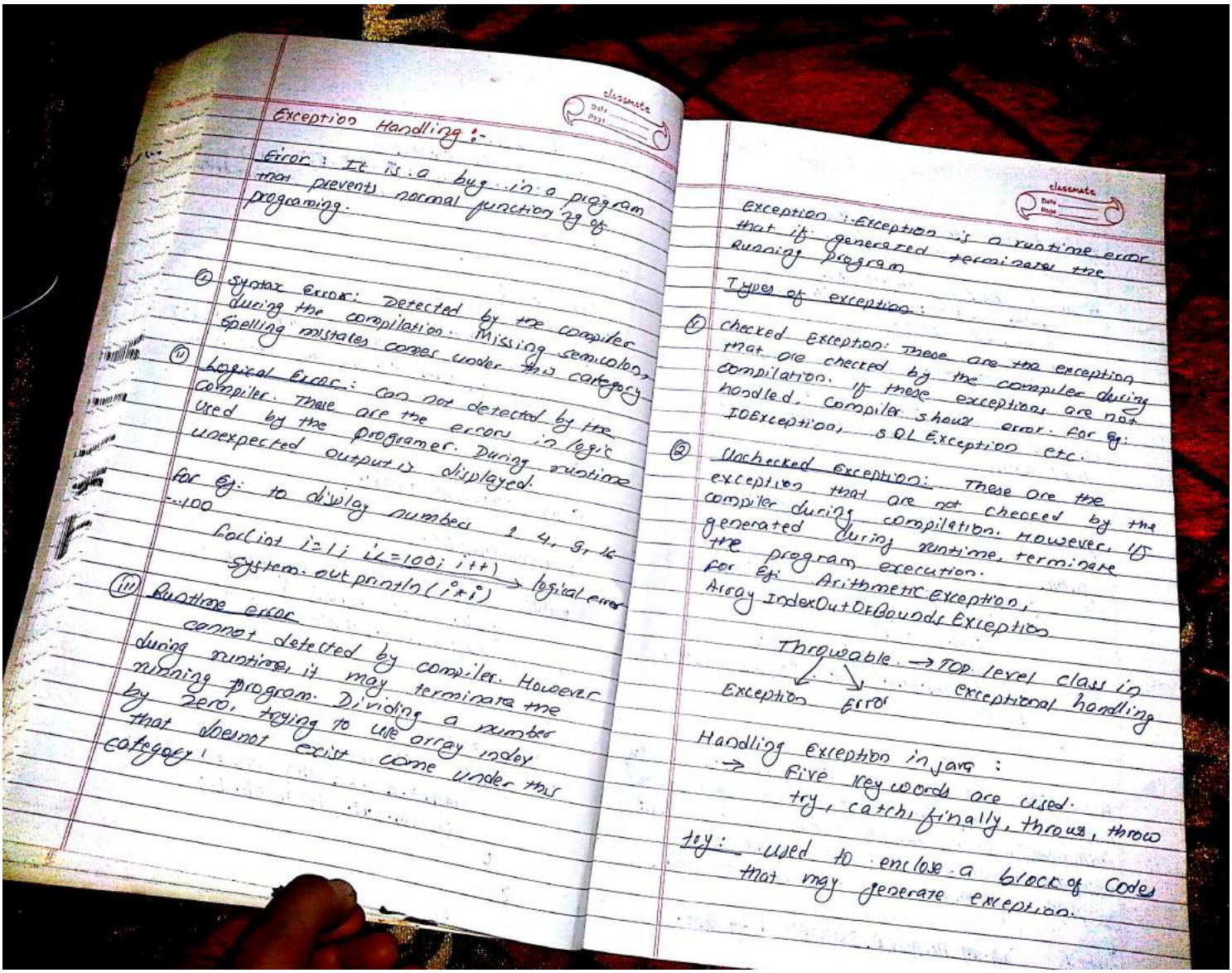
Access Protection :-

	Private	~~Default~~ Protected	Protected ~~Default~~	Public
Same class	Y	Y	Y	Y
Same package sub class	N	Y	Y	Y
Same package non-subclass	N	Y	Y	Y
Diff pack sub-class	N	N	Y	Y
Diff package non-subclass	N	N	N	Y

Example :

```
public class A
{   int a=10;
    private int b=20;
protected ~~private~~ int c=30;
    public int d=40;
public void show() {
    System.out.println(a+" "+b+" "+c+" "+d);
                    //same class all allowed.
}
}
```

```
class B extends A {
    public void show() {
        System.out.println(a+" "+b+" "+c+" "+d);
        //Same package subclass b not allowed.
    }
}

class C {
    public void show()
    {   A ob = new A();
        System.out.println(ob.a+" "+ob.c+" "+d);
        //same package non-sub class  Ob.b is not
        //allowed.
    }
}

// in different package

package mypack1;

public class D extends mypack.A {
    public void show() {
        System.out.println(c+" "+d);
    }
}

class E {
    public void show() {
        mypack.A ob = new mypack.A();
        System.out.println(ob.d);
    }
}
```



Exception Handling :-

Error : It is a bug in a program that prevents normal functioning of programing.

(i) Syntax Error: Detected by the compiler during the compilation. Missing semicolon, spelling mistakes comes under this category

(ii) Logical Error: can not detected by the compiler. These are the errors in logic used by the programer. During runtime unexpected output is displayed.

for Eg: to display numbers 1 4, 9, 16 ...100

for(int i=1; i*i<=100; i++) → logical error
System.out.println(i*i)

(iii) Runtime error

cannot detected by compiler. However during runtime, it may terminate the running program. Dividing a number by zero, trying to use array index that does not exist come under this category.

Exception : Exception is a runtime error that if generated terminates the running program

Types of exceptions :

(i) Checked Exception: These are the exception that are checked by the compiler during compilation. If these exceptions are not handled, compiler shows error. for Eg: IOException, SQLException etc.

(ii) Unchecked exception: These are the exception that are not checked by the compiler during compilation. However, if generated during runtime, terminate the program execution.

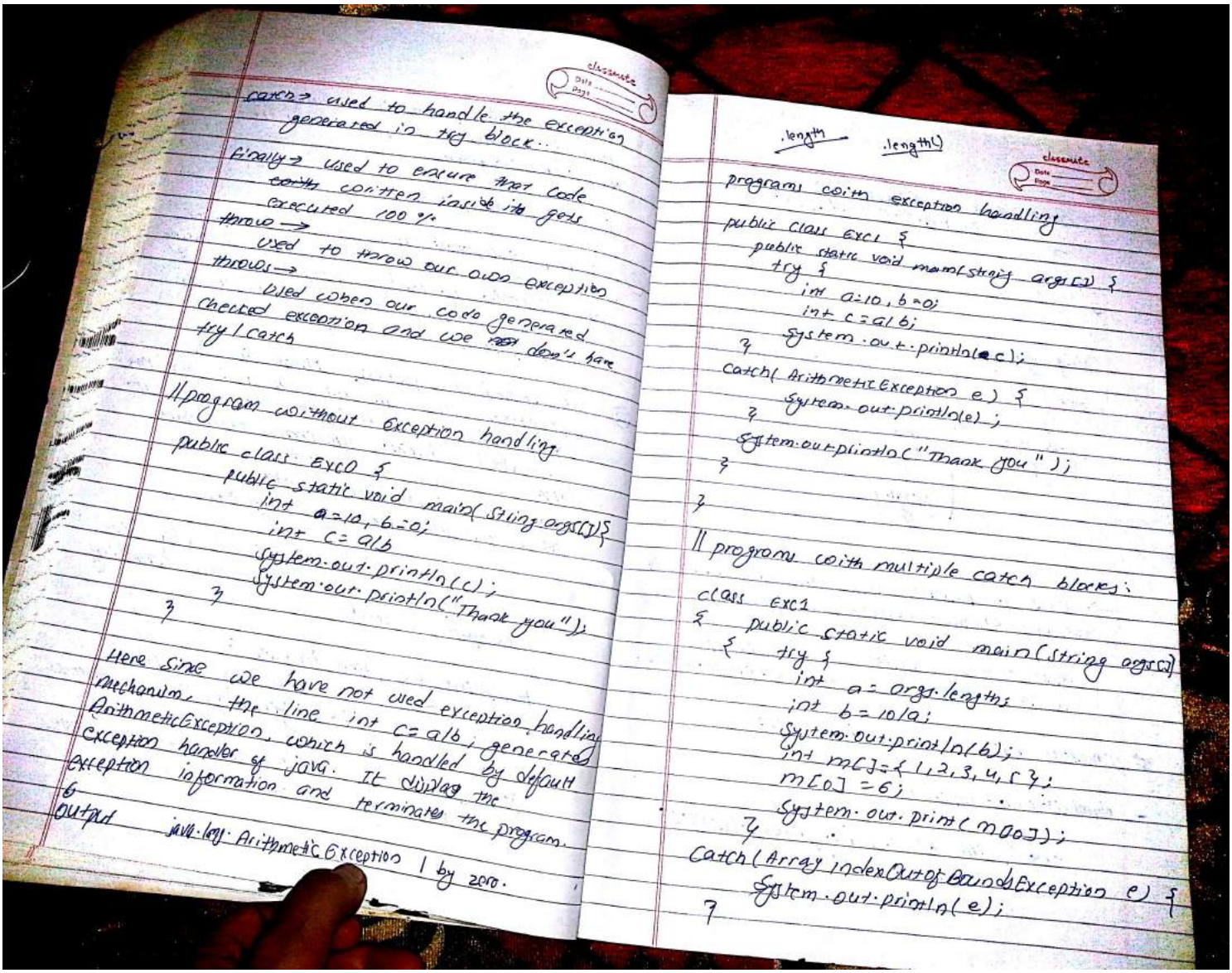
for Eg: Arithmetic exception,
Array IndexOutOfBounds Exception

Throwable → TOP level class in exceptional handling
Exception Error

Handling Exception in java :

→ Five key words are used.
try, catch, finally, throws, throw

try: used to enclose a block of codes that may generate exception.



catch → used to handle the exception generated in try block.

finally → used to ensure that code with written inside it gets executed 100 %.

throw →
used to throw our own exceptions

throws →
used when our code generated checked exception and we don't have try / catch

// program without exception handling

```
public class Exc0 {
    public static void main(String args[]){
        int a=10, b=0;
        int c=a/b
        System.out.println(c);
        System.out.println("Thank you");
    }
}
```

Here since we have not used exception handling mechanism, the line int c=a/b, generated ArithmeticException, which is handled by default exception handler of java. It displays the exception information and terminates the program.

output java.lang.ArithmeticException / by zero.

.length .length()

programs with exception handling

```
public class Exc1 {
    public static void main(String args[]) {
        try {
            int a=10, b=0;
            int c=a/b;
            System.out.println(c);
        }
        catch(ArithmeticException e) {
            System.out.println(e);
        }
        System.out.println("Thank you");
    }
}
```

// programs with multiple catch blocks:

```
class Exc1
{   public static void main(String args[])
    {   try {
            int a=args.length;
            int b=10/a;
            System.out.println(b);
            int m[]={1,2,3,4,5};
            m[0]=6;
            System.out.println(m[0]);
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.out.println(e);
        }
```

Some string functions:

(i) toUpperCase();

```
String name1 = new String("Sanjay Babu");
String name = name1.toUpperCase();
```

// toUpperCase() method is used to convert string to upper case.

(ii) toLowerCase()

```
String name = name1.toLowerCase();
```

// toLowerCase() method is used to convert string to lower case.

(iii) compareTo():

```
result = name1.compareTo(name2);
```

If (result == 0)
→ Two words are same
If (result < 0)
name1 comes first in dictionary order than name2
If (result > 0)
name2 comes first in dictionary order than name1

(iv) indexOf();

```
char check = '@';
String emailAddress = "Badrsanjay@gmail.com";
result = emailAddress.indexOf(check)
```

// it return the index of @ in that string in this string it returns 8. If not found it return 0 -1 If (result == -1)
→ It is invalid

length();

```
name1.length()
```

// Returns the length of string

trim();

```
name2.trim();
```

// Returns a copy of string, with leading & trailing // whitespace omitted

isEmpty();
check whether the string is empty or not. It returns 0 or 1. 1 for empty.

File handling

Stream → A stream is an object that either produces or consumes information
→ Generally a stream is linked to a physical device like keyboard, a network computer or even files in the computer.

Types of stream:
(i) Byte stream
(ii) Character stream

// Writing to file using byte stream

```
import java.io.*;
public class WriteBytes
{
    Public static void main(String args[]) throws Exception //or FileNotFoundException
    {
        FileOutputStream fout = new FileOutputStream("abc.txt");
        String str = "This is file handling using byte stream";
        byte b[] = str.getBytes();
        fout.write(b)
        System.out.println("Data written to file");
        fout.close();
    }
}
```

For path `FileOutputStream("d:/abc.txt");`

// For append:

```
FileOutputStream fout = new FileOutputStream("d:/abc.txt", true);
```

or

```
File f1 = new File("d:/abc.txt");
FileOutputStream fout = new FileOutputStream(f1, true);
```

// For professional

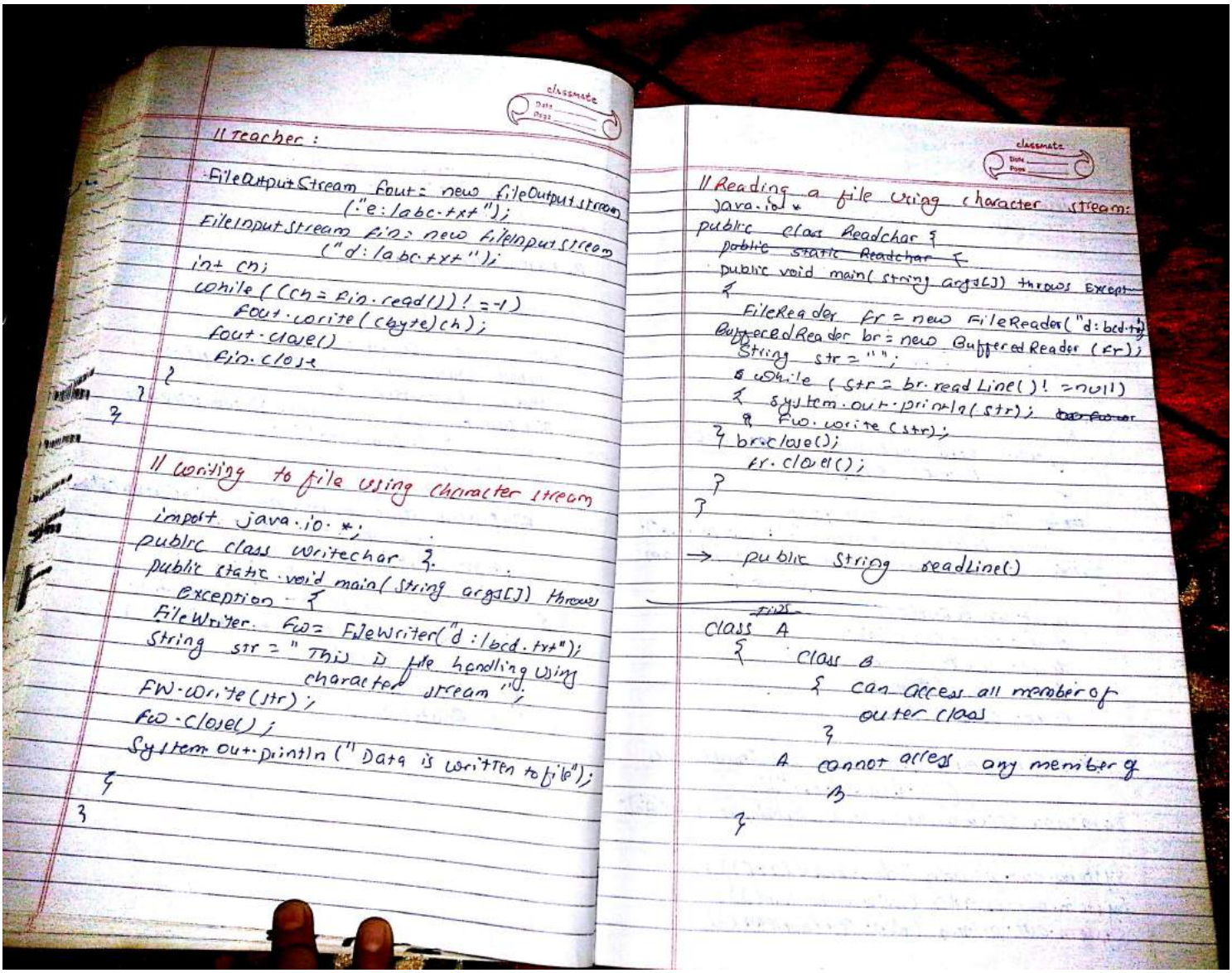
```
public class WriteBytes
{
    Public static void main(String args[]) {
        FileOutputStream fout = null;
        try {
            File f1 = new File("d:/abc.txt");
            fout = new FileOutputStream(f1, true);
            String str = "File handling";
            byte b[] = str.getBytes();
            fout.write(b);
            System.out.println("Data written to file");
        } catch (Exception e) {
            System.out.println(e)
        }
        finally {
            try {
                fout.close();
            }
            catch (Exception e) {
```

```
System.out.println(e);
        }
    }
}

// IO Read File.

import java.io.*;

public class ReadBytes {
    public static void main(String args[])
        throws Exception {
        FileInputStream fin = new FileInputStream
            ("d:/abc.txt");
        int ch;
        while((ch = fin.read()) != -1)
            System.out.print((char)ch);
        fin.close();
    }
}
```

Note:
-1 is EOF.
Public int read() it returns int

Q) create a program to copy the contents of d:/ab.txt into "e:/abc.txt" using byte stream.

```
Package filehandling;

import java.io.*;

public class CopyBytes {
    public static void main(String args[])
        throws Exception {
        FileInputStream fin = new FileInputStream
            ("d:/abc.txt");
        int ch;
        while (ch = fin.read()
        FileOutputStream fout = new FileOutputStream
            ("e:/abc.txt", true);
        String out;
        while (ch = fin.read()) != -1)
        {
            out = (char(ch); new String(ch);
            byte b[] = ob out.getBytes()
            fout.write(b);
        }
        fin.close();
        fout.close()
    }
}
```



// teacher :

```
FileOutputStream fout = new FileOutputStream
                        (":e:\abc.txt");
FileInputStream fin = new FileInputStream
                      ("d:\abc.txt");
int ch;
while ((ch = fin.read()) != -1)
    fout.write((byte)ch);
fout.close()
fin.close
}
}
```

// writing to file using character stream

```
import java.io.*;
public class WriteChar {
public static void main(String args[]) throws
    Exception {
FileWriter fw = FileWriter("d:\bcd.txt");
String str = "This is file handling using
             character stream";
FW.write(str);
fw.close();
System.out.println("Data is written to file");
}
}
```

// Reading a file using character stream:

```
java.io.*
public class ReadChar {
public static ReadChar {
public void main(String args[]) throws Except
{
  FileReader fr = new FileReader("d:\bcd.txt")
BufferedReader br = new BufferedReader(fr);
  String str = "";
while (str = br.readLine() != null)
{ System.out.println(str);
  fw.write(str);
} br.close();
  fr.close();
}
}
```

→ public String readLine()

```
class A
{   class B
    { can access all member of
      outer class
    }
  A cannot access any member of
    B
}
```

// write a program to copy from one file to another.

Writing and Reading Mixed Data.

```
import java.io.*;

public class ReadWriteMixed
{
    public static void main (String args[])
            throws Exception
    {
        FileOutputStream fout =
            new FileOutputStream ("d:/mixed.txt");
        DataOutputStream dout = new DataOutputStream
                                (fout);
        dout.writeInt(5);
        dout.writeChar('a');
        dout.writeBoolean(true);
        dout.close();
        fout.close();

        FileInputStream fin = new FileInputStream
                              ("d:\\mixed.txt");
        DataInputStream din = new DataInputStream(fin);

        System.out.println(din.readInt());
        System.out.println(din.readChar());
        System.out.println(din.readBoolean());
        din.close();
        fin.close();
    }
}
```

Note: जुन sequence मा write गरेको छ त्यही sequence मा read.

Serialization and De-Serialization :-

→ The process of writing (storing) the state of an object into a persistent store ie file is called serialization.

→ A class must implement serializable interface inorder to write its object to file.

→ ~~Object OutputStream~~. ObjectOutputStream class and its method writeObject() are used.

→ Process of restoring the state of an object from file is called de-serialization.

→ ObjectInputStream class and its method readObject() are used.

right click → Insert code → Getter

```
//
package filehandling

import java.io.*;

class Person implements Serializable{
    private int id
    private String name
    private double salary

    Person(int id, String name, double salary){
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    public void showInfo() {
        System.out.println("Id : "+id);
        System.out.println("Name: "+name);
        System.out.println("Salary: "+salary);
    }
}

public class ReadWriteObject {
    public static void main(String args[])
    {
        Person ob = new person(2,"Ram", 20000);
        FileOutputStream fout = new
            FileOutputStream("object.txt");
        ObjectOutputStream oout = new ObjectOutput
            Stream(fout);
        oout.close();
        fout.close();
```

```
FileInputStream fin = new FileInputStream("d:\object.tx
ObjectInputStream oin = new ObjectInputStream
                        (fin);
Person p = (Person) oin.readObject();
p.showInfo();
oin.close();
fin.close();
}
}

// This does not store that value into
file.

private transient int id;
```

→ If we dont want to use any of the Layout managers' we can pass null to setLayout() method.

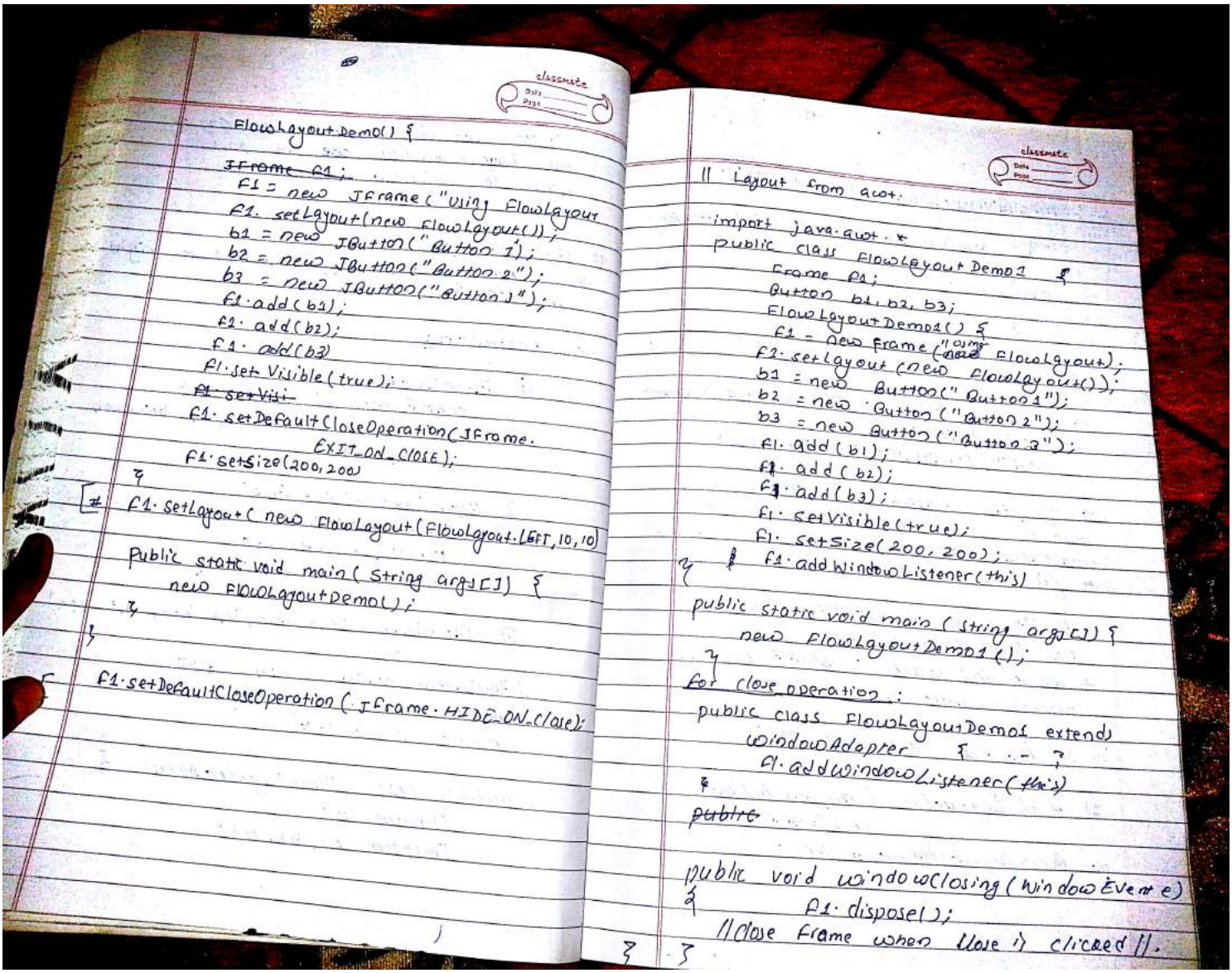
FlowLayout:

→ Arranges the component left to right and top to bottom

→ Constructors:

(i) FlowLayout()
center aligned by default
5px hor & 5px vert gap between components

(ii) FlowLayout(int how)
FlowLayout.LEFT



```
FlowLayoutDemo() {
    JFrame f1;
    f1 = new JFrame("Using FlowLayout
    f1.setLayout(new FlowLayout());
    b1 = new JButton("Button 1");
    b2 = new JButton("Button 2");
    b3 = new JButton("Button 3");
    f1.add(b1);
    f1.add(b2);
    f1.add(b3)
    f1.setVisible(true);
    f1.setVisi
    f1.setDefaultCloseOperation(JFrame.
                EXIT_ON_CLOSE);
    f1.setSize(200,200)
}
f1.setLayout(new FlowLayout(FlowLayout.LEFT,10,10)
public static void main(String args[]) {
    new FlowLayoutDemo();
}
}
f1.setDefaultCloseOperation(JFrame.HIDE_ON_close);
```

```
// Layout from awt.
import java.awt.*
public class FlowLayoutDemo1 {
    Frame f1;
    Button b1, b2, b3;
    FlowLayoutDemo1() {
    f1 = new Frame("using FlowLayout");
    f1.setLayout(new FlowLayout());
    b1 = new Button("Button 1");
    b2 = new Button("Button 2");
    b3 = new Button("Button 3");
    f1.add(b1);
    f1.add(b2);
    f1.add(b3);
    f1.setVisible(true);
    f1.setSize(200,200);
    f1.addWindowListener(this)
}
public static void main(String args[]) {
    new FlowLayoutDemo1();
}
for close operation:
public class FlowLayoutDemo1 extends
    WindowAdapter { ... }
    f1.addWindowListener(this)
}
public

public void windowClosing(WindowEvent e)
{
    f1.dispose();
// Close Frame when close is clicked //.
}
}
```

f1.addWindowListener(new windowClick());

class windowClick extends WindowAdapter
{
public vo -
}

BorderLayout :-

→ A Layout manager that arranges component in five regions:
NORTH, SOUTH, WEST, EAIT, CENTER.

→ If the components in north, south, east or west are missing then other component expand horizontally or vertically to fill the space.

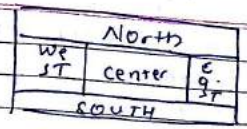
→ If the center component is missing, the space is left blank.

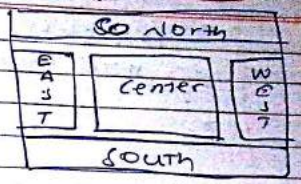
→ North and south components use the entire width of contain

→ It has two constructuctors:

1) BorderLayout() // no gap between the components.
2. BorderLayout(int hor, int ver).

①





//
import javax.swing.*;
import java.awt.*;

public class BorderLayoutDemo extends JFrame{
JButton b1, b2, b3, b4, b5;
BorderLayoutDemo() {
setLayout(new BorderLayout());
setVisible(true);
setSize(200,200);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
b1 = new JButton("North");
b2 = new JButton("South");
b3 = new JButton("west");
b4 = new JButton("centre");
b5 = new JButton("east");

add(b1, BorderLayout.NORTH);
add(b2, BorderLayout.SOUTH);
add(b3, BorderLayout.WEST);
add(b4, BorderLayout.CENTRE);
add(b5, BorderLayout.EAST);

}

```
JFrame f1;
JButton b1, b2;
JLabel l1, l2;
JTextField t1, t2;

GridLayoutDemo() {

f1 = new JFrame("using gridLayout");
f1.setLayout(new GridLayout(3,2));
f1.setDefaultCloseOperation(JFrame.EXIT_ON_
    CLOSE);
    l1 = new JLabel("Id");
    l2 = new JLabel("Name");
    t1 = new JTextField();
    t2 = new JTextField();
    b1 = new JButton("OK");
    b2 = new JButton("Cancel");
    f1.add(l1);
    f1.add(t1)
    f1.add(l2);
    f1.add(t2);
    f1-add(b1);
    f1.add(b2);
    f1.setVisible(true);
    f1.setSize(200,200);
}

public static void main(String args[]){
    new GridLayout();
}
}
```

#

```
import javax.swing;
import java.awt.*;

public class BorderDemo
{
    JFrame f1;
    JPanel p1;
    JButton b1, b2, b3, b4;
    BorderDemo() {
    f1 = new JFrame("using borderLayout");
    f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f1.setLayout(new BorderLayout());
    f1.setLayout p1 = new JPanel();
    b1 = new JButton("Button 1");
    b2 = new JButton("Button2");
    b3 = new JButton("Button 3");
    b4 = new JButton("Button 4");
    p1.setLayout(new FlowLayout());
    p1.add(b1);
    P1.add(b2);
    p1.add(b3);
    p1.add(b4);
    f1.add(p1, BorderLayout.SOUTH);
    f1.add(b4, BorderLayout.CENTER);
    f1.setVisible(true);
    f1.setSize(300,300);
    }
    public static void main(String args[]) {
        new BorderDemo()
    }
}
```

#

GridBagLayout :-

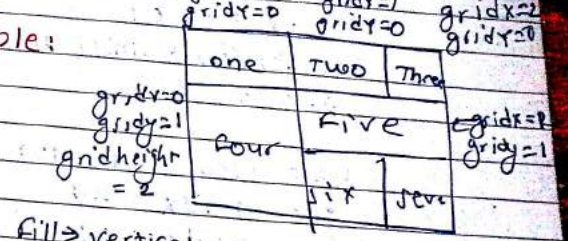
→ Similar to GridLayout however the size of each component can be different.

→ Size of component doesn't vary when the container is resized.

→ A helper class GridBagConstraints is needed

→ Constructor : GridBagLayout()

GUI Example:



#

```
import javax.swing.*
import java.awt.*;

public class GridBagDemo {
    JFrame f1;
    JButton b1, b2, b3, b4, b5, b6, b7;

    GridBagDemo() {
    f1 = new JFrame("using grid bag Layout");
    f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
GridBagLayout gl = new GridBagLayout();
fl.setLayout(gl);
GridBagConstraints gbc = new GridBagConstraints();
b1 = new JButton("one");
gbc.gridx = 0;
gbc.gridy = 0;
gl.setConstraints(b1, gbc);
fl.add(b1);

b2 = new JButton("two");
gbc.gridx = 1;
gbc.gridy = 0;
gl.setConstraints(b2, gbc);
fl.add(b2);

b3 = new JButton("three");
gbc.gridx = 2;
gbc.gridy = 0;
gl.setConstraints(b3, gbc);
fl.add(b3);

b4 = new JButton("four");
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridheight = 2;
gbc.setConstraints
gbc.fill = GridBagConstraints.VERTICAL;
gl.setConstraints(b4, gbc);
fl.add(b4);
```

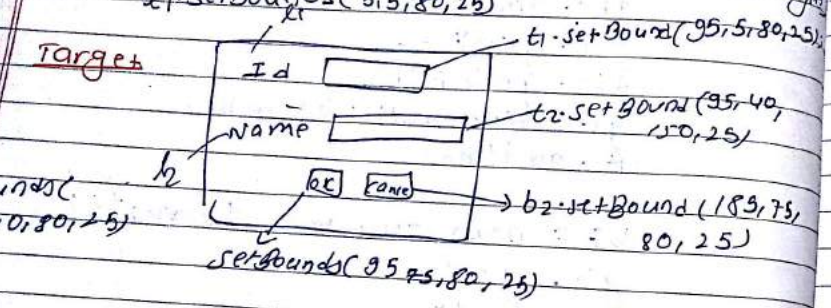
```
b5 = new JButton("five");
gbc.gridx = 1;
gbc.gridy = 1;
gbc.gridwidth = 2;
gbc.gridheight = 1;
gbc.fill = GridBagConstraints.HORIZONTAL;
gl.setConstraints(b5, gbc);
fl.add(b5);

b6 = new JButton("six");
gbc.gridx = 1;
gbc.gridy = 2;
gbc.gridwidth = 1
gl.setConstraints(b6, gbc);
fl.add(b6)

b7 = new JButton("seven");
gbc.gridx
gbc.gridy
gl.setConstraints(b7, gbc());
fl.add(b7);
fl.add(
fl.setSize(300, 300);
fl.setVisible(true);
{
public static void main(String args[])
{   new GridBagDemo();
}
}
```

Using NO Layout Manager:

→ If we don't want to use any of the Layout managers provided by java, we can pass "null" to setLayout();

→ However, we must use setBounds() method for each components.

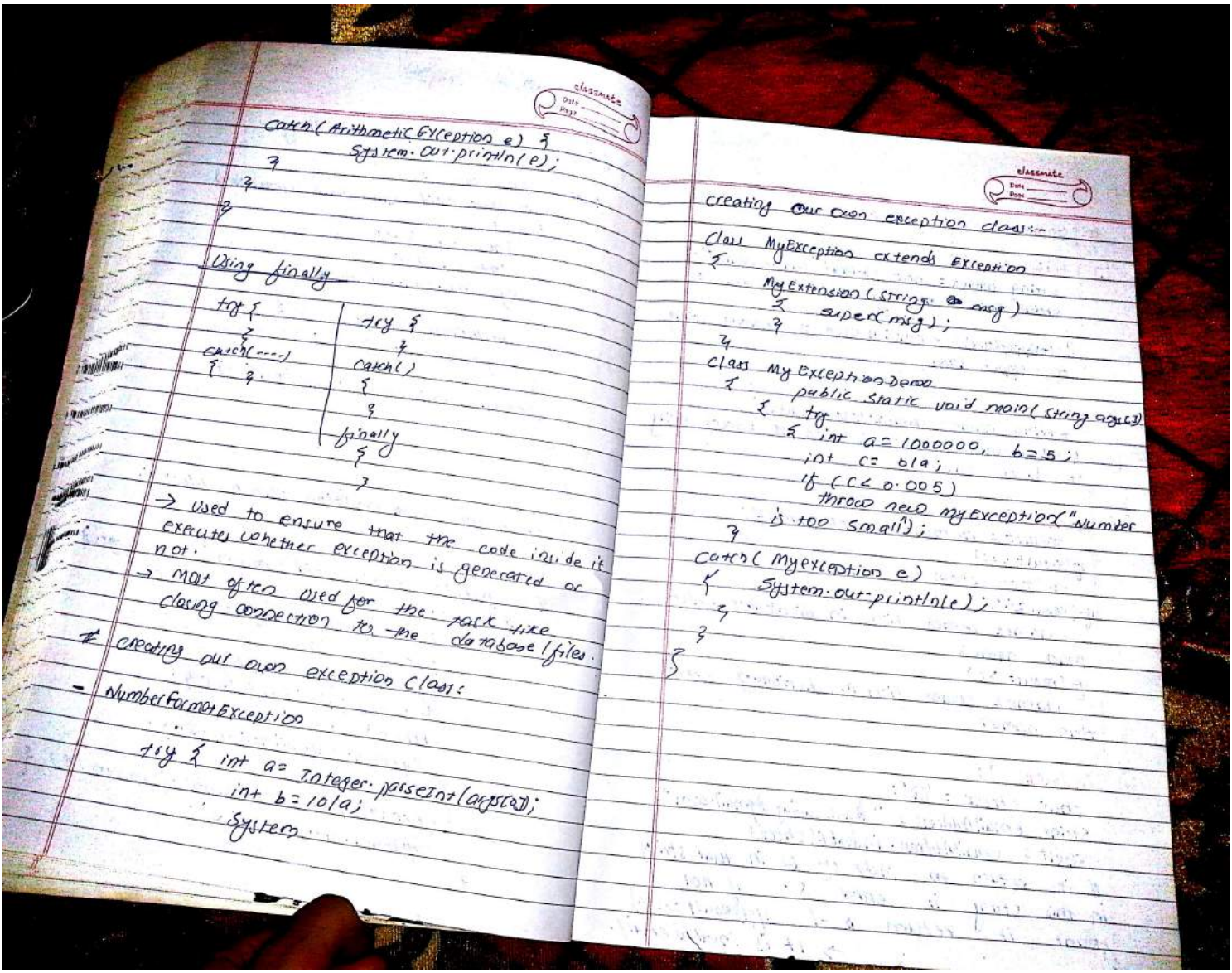
→ setBounds(int x00, int y00, int width, int height);



Target

setBounds(5,40,80,25)

```
import javax.swing.*;
import java.awt.*;

public class NoLayoutDemo {
    JFrame f1;
    JLabel l1, l2;
    JTextField t1, t2;
    JButton b1, b2;

    NoLayoutDemo() {
        f1 = new JFrame("Using no Layout");
        f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f1.setLayout(null);
        l1 = new JLabel("id");
        l1.setBounds(5,5,80,25);
        f1.add(l1);

        t1 = new JTextField();
        t1.setBounds(95,5,80,25);
        f1.add(t1);

        l2 = new JLabel();
        l2.setBounds(5,40,80,25);
        f1.add(l2);

        t2 = new JTextField();
        t2.setBounds(95,40,150,25);
        f1.add(t2);

        b1 = new JButton("OK");
        b1.setBounds(95,75,80,25);
        f1.add(b1);

        b2 = new JButton("Cancel");
        b2.setBounds(185,75,80,25);
        f1.add(b2);

        f1.setSize(300,300); f1.setResizable(false);
        f1.setVisible(true);
    }

    public static void main(String args[]) {
        new NoLayoutDemo();
    }
}
```



catch (ArithmeticException e) {
System.out.println(e);
}
}

Using finally

```
try {                    try {
}                        }
catch(---)               catch()
{ }                      {
                         }
                         finally
                         {
                         }
```

→ used to ensure that the code inside it executes whether exception is generated or not.
→ Most often used for the task like closing connection to the database / files.

* creating our own exception class:-

- NumberFormatException

try { int a = Integer.parseInt(args[0]);
int b = 10/a;
System

creating our own exception class:-

class MyException extends Exception
{
MyException(String msg)
{ super(msg);
}
}
class MyExceptionDemo
{ public static void main(String args[])
{ try
{ int a = 1000000, b = 5;
int c = b/a;
if (c < 0.005)
throw new MyException("Number is too small");
}
catch(MyException e)
{ System.out.println(e);
}
}
}

Event handling

Delegation Event Model

→ An approach in event handling that uses the three concepts:-

ⓐ Event ⓑ EventSource

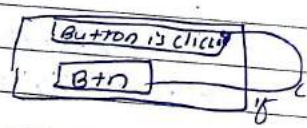
ⓒ EventListener

ⓐ Event :- An object that describes state change in an eventSource.
for Eg: ActionEvent, ItemEvent etc.

ⓑ EventSource → An object that generate the event. for Eg:
JButton, JTextField, JComboBox etc.

ⓒ EventListner → An object that get notified about the event and does the further processing. Every event source must register to the event Listener otherwise processing of event is not performed. one event source may generate more than one type of event.
Eg: ActionListener, ItemListner etc.

TextField
↗ ↘ FocusEvent
KeyEvent ActionEvents



ActionEvent
↓
ActionListener
↳ public void actionPerformed(ActionEvent e)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

Public class EventDemo implements
    ActionListener {
        JFrame f1;
        JTextField t1;
        JButton b1;
        EventDemo() {
        f1 = new JFrame("handling buttonclick");
        f1.setLayout(new FlowLayout());
        f1.setDefaultCloseOperation(JFrame.
            EXIT_ON_CLOSE);
        t1 = new JTextField(10);
        b1 = new JButton("click me");
        b1.addActionListener(this);
        b2 = new JButton("clear");
        b2.addActionListener(this);

        f1.add(t1);
        f1.add(b1);
        f1.add(b2);
        f1.setSize(200,200);
        f1.setVisible(true);
    }
    public static void main(String args[]) {
        new EventDemo();
    }
    public void actionPerformed(ActionEvent e){
        if(e.getSource() == b1) {
```

t1.setText("Button is clicked");
}
else if (e.getSource() == b2) {
t1.setText("");
}

t1(100,10,80,25)
t2(100,45,80,25)
t3(100,80,80,25)
b1(10,10,80,25)
b2(10,45,80,25)
bL(10,80,80,25)

Num1
Num2
Result

int a = Integer.parseInt(t1.getText());
int b = Integer.parseInt(t2.getText());
int c = a+b
t3.set(c+"");
or
t3.setText(String.valueOf(c));

15
25
10
50

# How to set size of frame
f1.setSize(width, height);

# How to set location of frame
f1.setLocation(left, TOP);

# How to set size and location of frame
f1.setBounds(left, TOP, width, height);

How to set title image icon of JFrame
//first we need to make object of imageIcon.
ImageIcon icon = new ImageIcon("HP_LOGO.JPG");
f1.setIconImage(icon.getImage());

How to set background color of JFrame
we need to make object of container.
Container c = frame.getContentPane();
c.setBackground(Color.RED);

# How to restrict to resize
f1.setResizable(false);

How to create JLabel

```
Container c = f4.getContentPane();
JLabel label = new JLabel("UserName");
c1.setLayout(null);
label.setBounds(100,100,80,25);
c1.add(label);
```

How to set text on JLabel on any time

```
label.setText("Password");
```

How to set font style and size :
use need to create object of Font.

```
Font f = new Font("FontName,constant,fontsize);
i.e
Font f = new Font("Arial", Font.PLAIN, 30);
Font f = new Font("Arial", Font.ITALIC, 30);
label.setFont(f);
```

How to set image in JLabel

```
ImageIcon icon = new ImageIcon("logo.png");
JLabel label = new label(icon);
label.setIconWidth();
label.setBounds(100,50,icon.getIconWidth(),30);
```

How to set JTextField

```
JTextField f1 = new JTextField();
Font f = f4.getFont();
How to change font size and style
Font f4 = new Font("Arial",Font.BOLD,25);
f4.setFont(f);
```

How to set Font and Background color of Textfield

```
f4.setBackground(Color.YELLOW);
f1.setForeground(color.RED);
or
Color c = new color(120,100,50); // rgb(120,100,50);
f1.setBackground(c);
```

How to set JPasswordField

```
JPasswordField pass = new JPasswordField();
pass.setBounds(100,150,100,30);
c.add(pass);
```

How to create JButton and Text in this

```
Static Text :
JButton b1 = new JButton("Click me");
Anonymously :
JButton btn = b1.setText("Don't Click");
```

How to set image in JButton

```
ImageIcon icon = new ImageIcon("photo.jpg");
JButton btn = new JButton(icon);
btn.setBounds(100,50, icon.getIconWidth(),
    icon.getIconHeight());
fl.add(btn);
```

How to chang font style & size

```
Font F=new Font("Aerial", FONT.BOLD, 25);
btn.setfont(F);
```

How to set Background and Foreground color in Button:

```
btn.setBackground(Color.RED);
btn.setForeground(Color.WHITE);
```

How to set Cursor for JButton

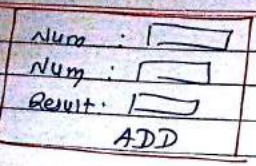
```
Cursor crsr = new Cursor(Cursor.
        HAND_CURSOR);
btn.setCursor(crsr);
```

// CROSSHAIR_CURSOR, WAIT_CURSOR

Event handling

* JButton Event

```
import javax.swing.*;
import java.awt.*;

class MyFrame extends JFrame {
    JButton btn = new JButton("click");
    Container c;
    MyFrame()
    {   c = this.getContentPane();
        c.setLayout(null);
        btn.setBounds(100,50,100,50);
        c.add(btn);
    }
}
class
```



#

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

Public class Addition extends WindowAdapter
implements ActionListener {
    Frame f1;
    Label l1, l2, l3;
    TextField t1, t2, t3;
    Button b1;
    Addition()
{   f1 = new Frame("
    f1.setLayout(new FlowLayout());
    f1.setSize(200, 200);
    f1.setVisible(true);
    l1 = new Label("Num1");
    l2 = new Label("Num2");
    l3 = new label("Num3");
    t1 = new TextField(10);
    t2 = new TextField(10);
    t3 = new TextField(10);
    f1.addWindowListener(this);
    b1 = addAction  b1 = new Button("Add");
    b1.addActionListener(this);
    f1.add(l1); f1.add(t1)
    f1.add(l2); f1.add(t2)
    f1.add(l3); f1.add(t3);
    f1.add(b1);
}
    f1.setS:
    Public static void main(String args[])
{       new Addition();
}

public void ActionPerformed(
            ActionEvent e)  {

    int num1 = Integer.parseInt(t1.getText());
    int num2 = Integer.parseInt(t2.getText());
    int result = num1+num2;
    l3.setText(result + " ");
}

public void windowClosing(WindowEvent e) {
        f1.dispose();
}
}
```

Another method

Outer class {

- b1.addActionListener(new ButtonClick());

class ButtonClick implements ActionListener {
public void actionPerformed (ActionEvent e) {
:
}
}

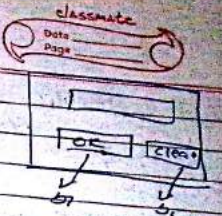
Anonymous inner class. for new event -

class {

b1.addActionListener (new ActionListener() {

public void actionlistener actionperformed
(ActionEvent e) {
int num1 = Integer.parseInt(t1.getText());
int num2 = Integer.parseInt(t2.getText());
int result = integer num1 + num2;
...

}

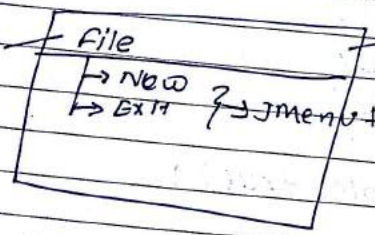


public class GUI extends
WindowAdapter
implements ActionListener {
Frame f1;
TextField t1;
Button b1, b2;
GUI() {
f1 = new Frame();
f1.setLayout(new FlowLayout());
f1.addWindowListener(this);
t1 = new TextField(10);
b1 = new Button("OK");
b2 = new Button("Clear");
b1.setActionCommand("OK");
b2.setActionCommand("clear");
b1.addActionListener(this);
b2.addActionListener(this);
f1.add(t1);
f2.add(b1);
f1.add(b2);
f1.pack();
f1.setVisible(true);
}
public static void main(String args[]) {
new GUI();
}

```
public void windowClosing (WindowEvent e){
    f1.dispose();
}

public void actionPerformed (ActionEvent e)
{
    String command = e.getActionCommand();
    if(command.equals("ok")) {
        t1.setText("Button is Clicked");
    }
    else if( command.equals(ignorecase("clear"))
    {
        t1.setText(" ");
    }
}
}
```

Creating a Meno ag.

Swing: JMenuBar
JMenu
JMenuItem




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MenuDemo {
    JFrame f1;
    JMenuBar mbar;
    JMenu FileMenu
    JMenuItem mi1, mi2;

    MenuDemo() {
        f1 = new JFrame("Using MenuBar");
        f1.setDefaultCloseOperation(JFrame.
                                    EXIT_ON_CLOSE);
        f1.setLayout(new BorderLayout());
        mbar = new JMenuBar();
        FileMenu = new JMenu("File");
        mi1 = new JMenuItem("New");
        mi2 = new JMenuItem("exit");
        Filemenu.setMnemonic('F');
        FileMenu.add(mi1);
        Filemenu.addSeparator();
        Filemenu.add(mi2);
        goto: Label;
        mbar.add(Filemenu);
    // f1.add(mbar, BorderLayout.NORTH);
        JMenu f1.setJMenuBar(mbar);
        f1.setSize(300, 300);
        f1.setVisible(true);
    }

    public static void main(String args[])
    {
        new MenuDemo();
    }
}
```

Label :

```
mi1.addActionListener(new ActionListener
                                    () {
public void actionPerformed (ActionEvent e)
{
    new GUI();
}
});
```

```
mi2.addActionListener(new ActionListener() {
    public void actionPerformed (ActionEvent e)
    {  f1.dispose();
    }
});
```

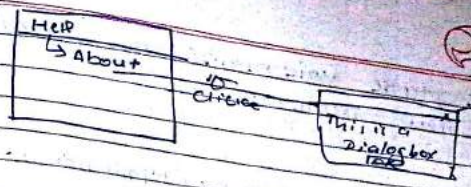
creating a dilog box :-

→ A dialog box is a window similar to Frame. However, it does not have Menubar.

→ Types of Dialog Bar

① Modal → Doesnot allow to perform other tasks without closing it.

② Non-Modal (Modeless) → other tasks can be performed without closing it.




```
import.awt.*;
import.awt.event.*
public class DialogExample extends WindowsAdapter
{
    Frame f1;
    MenuBar mbar;
    Menu help;
    MenuItem about;
    DialogExample(){
    f1= new Frame("creating dialogbox");
    f1.setLayout(new BorderLayout());
    f1.addWindowListener(this);
    mbar= new MenuBar();
    help= new Menu("Help");
    about= new MenuItem("About");
    help.add(about);
    mbar.add(help);
    f1.setMenuBar(mbar);
    about.addActionListener(new ActionListener
        () {
    public void actionPerformed(ActionEvent e){
        new DialogDemo();
    }
    });
    f1.setSize(300,300);
    f1.setVisible(true);
```

```
public static void main(String args[]){
    new DialogExample();
}

Public void windowClosing(WindowEvent e){
    d1.dispose();
}

class DialogDemo extends WindowAdapter {
    Dialog d1;
    Label l1;
    Button b1;
    DialogDemo() {
d1 = new Dialog(f1," Dialog Box", false)
d1.setLayout(new FlowLayout());
l1 = new Label("This is a dialog box");
b1 = new Button(" OK");
d1.add(l1)
d1.add(b1);
d1.setSize(200,200);
d1.addWindowListener(this);
d1.setVisible(true);
}

public void WindowClosing(WindowEvent e)
{
    d1.dispose();
}
```

True → Modal
False → Non-modal

JDBC (Java database connectivity)

→ JDBC is a java API used for connecting java with database.

→ It is made up of java.sql package the classes of this package to be used are

Connection
hold the connection to a database.

DriverManager
Establish the connection to a database

Statement
used to execute static sql statement

PreparedStatement
used to execute pre-compiled sql statement

CallableStatement
used to execute stored procedure.

ResultSet
used to hold the data returned by sql 'select query'.

Steps for JDBC

1) import required package.
2) Register with JDBC driver
3) create connection with database
4) perform required SQL operations
5) close the connection.

Drivers

JDBC-ODBC → Type 1

Type: 4 driver for mysql

SQL Queries

① insert

→ insert into tblname (col1, col2...) values (value1, value2, ...)

eg: insert into employee (id, name, salary) value (1, 'Ram', 2000)

② Delete

delete from tblname → It deletes all data from table

delete from tblname where id=1

(3) update:

update tblname set col1 = value, col2 = value... where id= value.

eg: update employee set salary = 30000 where id=2

④ Select:

→ select * from tblname

select * from tblname where id=2

library → Add library → mysql jdbc choose → Add library

```
import java.sql.*

public class staticInsert {
  public static void main(String args[])
  {
    try {
    Class.forName("com.mysql.jdbc.Driver");
    // register with jdbc driver
    Start-f
    Connection con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/demo","root","");
    String sql = "insert into employee(id,name,
    salary) values(3,"Rajan",3000000)
    Statement st = con.createStatement();
    st.executeUpdate(sql);
    System.out.println("1 row inserted");
    con.close();
    }
    catch (ClassNotFoundException | SQLException ex)
    {
      System.out.println("ex");
    }

  }
}
```

Display Records

```
import java.sql.*;
public class DisplayRecords {
  public static void main (String args[]) {
  try {
  Class.forName("com.mysql.jdbc.Driver");
  //register with jdbc com.mysql.Fixed

  Connection con = DriverManager.getConnection
  ("jdbc:mysql:///localhost:3306/demo","root","");
     //Create connection
  String sql = "select * from employee";
  //select value in sql employee.
  Statement st = con.createstatement();
   //Statement created
  ResultSet rs = st.executeQuery(sql);
   //for execute statement

  while(rs.next()) {
  System.out.println(rs.getString("id") + "\t");
     "     "     "    (rs.getString("name") + "\t");
     "     "     "    (rs.getString("salary"));
  }
  rs.close();
  con.close();
  } catch(SQLE                    ){
    .
  }
}
```

```
public void itemStateChanged (ItemEvent e)
{
    String item = ch.getSelectedItem();
    ta.setItem(item);
}
```

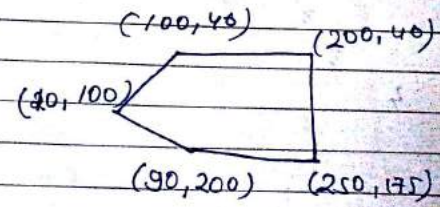
Graphics and Animation

java.awt

```
public void paint (Graphics g) {
    drawXXX(int xco1, int yco1, int xco2, int yco2)
    drawLine(int xco1, int yco1, int xco2, int yco2)
    drawRect(int xco, yco, width, height)
    fillRect( same)
    g.setColor(Color.BLUE)
    drawRect (int xco, yco, width, width)
                    ↳ Square
    drawOval (same as drawRect)
    fillOval (same).
    drawOval(int xco, yco, width, height)
    drawOval (int xco, yco, width, width)
                    ↳ Circle
```

g.drawString

```
# import java.awt.*;
import javax.swing.*;

public class DrawShapes extends JFrame {
    DrawShapes() {
        setSize(300, 300)
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
    }
    public static void main(String arg[]) {
        new DrawShapes();
    }
    public void pain(Graphics g) {
        g.setColor(Color.blue);
        g.drawLine(10, 40, 200, 40);
        g.setColor(Color.black);
        g.fillRect(10, 60, 200, 200);
        g.fillOval(10, 200, 200, 200);
        g.setColor(Color.white);
        g.setFont(new Font("TNR", Font.BOLD, 20));
        g.drawString("Test", 50, 250)
```



Creating a database using MS Access.

```
import java.awt.*;
import java.sql.*;

public class InsertRecords {
    public static void main(String args[]) {
        try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection"
                    jdbc:odbc:ncitDataSource");
        String sql = "insert into staff(names, salary)
                    values('hari', 30000)";
        Statement st = con.createStatement();
        st.executeUpdate(sql);
        System.out.println("Record inserted.");
        con.close();
        }
        catch(Exception ex) {
            System.out.println(ex);
        }
    }
}
```

Display Records from database using MS Access
JDBC-ODBC

```
import java.awt.*;
       java.sql.*;
       javax.swing.*;
       java.awt.event.*;
public class DisplayRecords implements ActionListener{
        JFrame f1;
        JButton b1;
        JTextArea t1;

    public DisplayRecords() {
        f1 = new JFrame("Display Records");
        f1.setLayout(new BorderLayout());
        f1.setSize(300, 300)
        f1.setVisible(true
        t1 = new JTextArea();
        b1 = new JButton("Load");
        f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f1.add(t1, BorderLayout.CENTRE);
        f1.add(b1, BorderLayout.SOUTH);
        b1.addActionListener(this);
    }
    public static void main(String[] args) {
        new DisplayRecords();
    }
    public void actionPerformed(ActionEvent e) {
        try {
        Class.forName("sun.jdbc.odbc.JdbcOdbc
                Driver");
```

```
Connection con = DriverManager.getConnection("
        jdbc:odbc:nciDataSource");
String sql = "select * from staff";
Statement st = con.createStatement();
~~Result st = con.create~~
ResultSet rs = st.executeQuery(sql);
String str = "";

While(rs.next()) {
   str += rs.getString("id") + "\t";
   str += rs.getString("name") + "\t";
   str += rs.getString("salary") + "\n";
}
String title = "id\t name \t salary \n.";
t1.setText(title + str);
con.close();
} catch (Exception ex) {
    System.out.println(ex);
}
```

To display data in a table:

→ To display data in a table:
→ JTable class is used.
→ its constructor takes two arguments
JTable(Object [][]data, String []cols)

eg:

```
Object [][] data = {{1, "Ram", 2000}, {2, "Hari",
        3000}, {3, "sita", 4000}};
String cols[] = {"id", "Name", "salary"};
JTable table = new JTable(data, cols);
JScrollpane jp = new JScrollPane(table);
                ↳ add it to frame
```

// program to retrieve data from database and display in a tabular form.

```
import java.awt.*;
import javax.swing.*;

public class TableDemo {
    JFrame f1;
    JTable table;
    TableDemo() {
      f1 = new JFrame("using table");
      f1.setLayout(new BorderLayout());
      f1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Object data[][] = {{1, "ram", 2000}, {2, "hari",
        3000}};
String cols[] = {"Id", "Name", "Salary"};
table = new JTable(data, cols);
```

netbeans.com

netbeans_jdkhome...

```
bl = new JButton("Load");
fl.add(bl, BorderLayout.SOUTH);

bl.addActionListener(this);
fl.setSize(300,300);
fl.setVisible(true);
}

public static void main(String args[]) {
new DisplayRecords();
}
public void actionPerformed(ActionEvent e) {
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbc
            Driver");
Connection con = DriverManager.getConnection
    ("jdbc:odbc:ncitDataSource");
String sql = "select * from staff";
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(sql);
Object data[][] = new Object[10][3];
int i = 0;
while(rs.next()) {
  data[i][0] = rs.getString("id");
  data[i][1] = rs.getString("name");
  data[i][2] = rs.getString("salary");
  i++;
}
String title[] = {"id","name","salary"};
tl = new JTable(data,title);
JScrollPane jp = new JScrollPane(tl);
fl.add(jp);
```

```
f1.pack();
con.close();
}
catch(Exception ex) { System.out.println(ex);
}
```

BLOB
↳ Binary Large object.

writing BLOB

nexit → database
tbltest
↳ id INT A.I PK
name VARCHAR 40
pic BLOB

Program for image inserting on database

```
Package jdbc
import java.io.*;
import java.sql.*;

public class WriteBLOB {

    public void main(String args[]) {
try {
    Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection
    ("jdbc:mysql://localhost:3306/nexit","root",
    "");
String sql = "insert into tbltest(name,pic)
    - value(?,?)";
Prepared statement pstmt = con.prepareStatement
                    (sql);
String fileName = "f:\\images\\1.imj.jpg";
File file = new File(fileName);
FileInputStream input = new InputStream(file);
pstmt.setString(1,"hari");
pstmt.setBinaryStream(2,input);
pstmt.executeUpdate();
con.close();

System.out.println("inserted in the database.");
}
    catch(exception e) {
        System.out.println(e.getMessage
    }
}
```

Applet

Container
- Window
  - Frame
    - JFrame
- Panel
  - Applet
    - JApplet

local applette
Remote Applette

~~life~~

Applet:

--> Applet is a small java program primarily used in Internet computing.

--> An applet can play sounds, display graphics, take user input, play interactive games and so on.

--> An applet which is stored on our computer system and when the browser tries to access the applet, it is not necessary for our computer to be connected to the internet is called local applet.

--> A remote applet is the one which is not stored in our computer system and we are required to connect to the internet to access it.

applet viewer

Difference between an applet and a program

--> Applets do not have a main() method

--> Applets cannot run independently. They are run from within a browser or tested by using "appletviewer" tool.

--> Applets cannot read from or write to files on the local computer

--> Applets are restricted from using libraries of other languages.

--> Applets cannot communicate with other computers on the ~~net~~ network.

Applet life cycle:

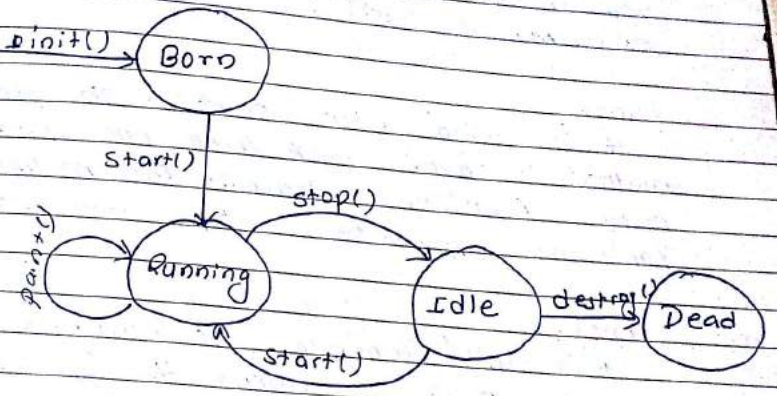
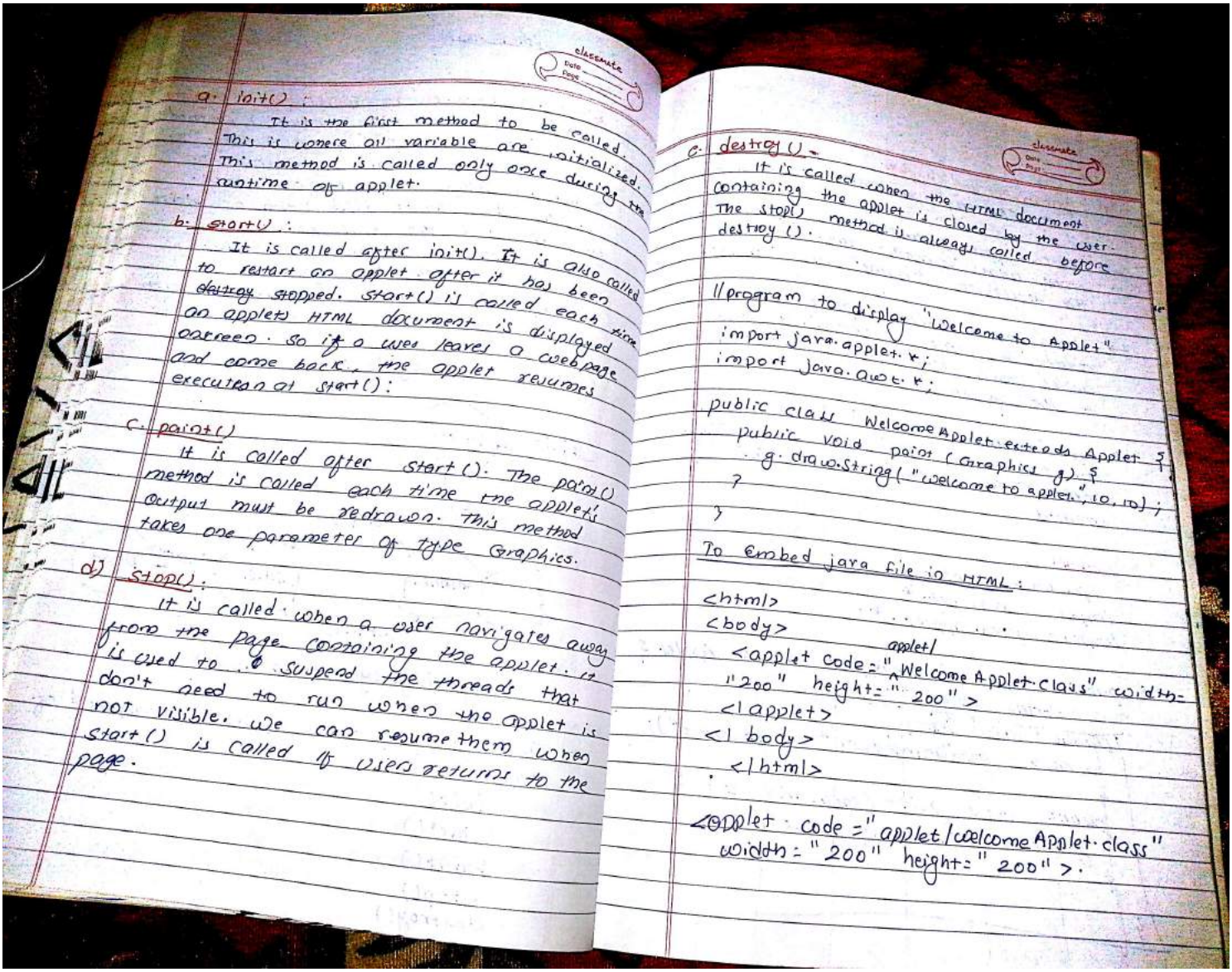


Fig: Life cycle of an applet

* The life cycle of an applet is composed of five methods:
  - init()
  - start()
  - paint()
  - stop()
  - destroy()



a. init() :
It is the first method to be called. This is where all variable are initialized. This method is called only once during the runtime of applet.

b. start() :
It is called after init(). It is also called to restart an applet after it has been destroy stopped. start() is called each time an applet's HTML document is displayed onscreen. So if a user leaves a webpage and come back, the applet resumes execution at start().

c. paint()
It is called after start(). The paint() method is called each time the applet's output must be redrawn. This method takes one parameter of type Graphics.

d) stop() :
It is called when a user navigates away from the page containing the applet. It is used to suspend the threads that don't need to run when the applet is not visible. We can resume them when start() is called if user returns to the page.

c. destroy() :
It is called when the HTML document containing the applet is closed by the user. The stop() method is always called before destroy().

```
//program to display "Welcome to Applet"
import java.applet.*;
import java.awt.*;

public class WelcomeApplet extends Applet {
    public void paint (Graphics g) {
        g.drawString("welcome to applet", 10, 10);
    }
}
```

To Embed java file in HTML :

```
<html>
<body>
<applet code="Welcome Applet.class" width="200" height="200">
</applet>
</body>
</html>
```

applet/

```
<applet code="applet/welcomeApplet.class"
width="200" height="200">
```

Parameterized Applet
(Passing parameter to Applet)

→ We can pass parameters to applet from HTML using <param> tag. It takes the attributes "name" and "value".

→ Using parameterized applet, we can reduce the task of compiling the applet each time changes are made to the applet code.

→ getParameter (String) is used to get the parameter value in applet. This method must be used inside init() method

Example:

```
package applet
import java.awt.*;
import java.applet.*;

public class ParamApplet extends Applet {
    String msg = "";
public void init() {
    msg = getParameter("applet_msg");
}
public void paint(Graphics g) {
    g.drawString(msg, 10,20);
}
}
```

```
public void init() {
    msg = getParameter("applet_msg");
    if(msg == NULL)
        msg = "Starting message";
}
public void paint(Graphics g) {
    g.drawString(msg, 10,20)
}
}
```

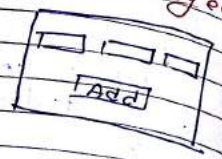
HTML:

```
<html>
<body>
<applet code="applet/ParamApplet.class" width="200" height="200">
<param name="applet_msg" value="Hello Applet" />
</applet>
</body>
</html>
```

//we have to enable
control panel > java > Security > Exception site list > Edit site list > add > paste the path of html ie file:///C:/Users/Nabin/Desktop/applet/test1.html > add > continue >> ok > ok

Run File (.html)

create an applet that contain 3 text field and 1 button. When the button is clicked, sum of values in first two textfields should be displayed in the third text field.




```
package applet;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Addition extends Applet implements
ActionListener {
    TextField t1, t2, t3;
    Button b1;
public void init() {
    setLayout(new FlowLayout());
    t1 = new TextField(10);
    t2 = new TextField(10);
    b1 = new Button("Add");
    add(t1);
    add(t2);
    add(t3);
    add(b1);
    b1.addActionListener(this);
}
public void actionPerformed(ActionEvent e){
    int num1 = Integer.parseInt(t1.getText());
    int num2 = Integer.parseInt(t2.getText());
    int num3 = num1 + num2;
    t3.setText(result + "");
}
}
```

Converting applications to applet:

Steps:

1) Make an html page with appropriate tag to load the applet code.
2) Supply a subclass of Applet (JApplet class. Make this public.
3. Eliminate the main method in the Application. Do not construct a frame window for the application. Your applet will be displayed inside the browser.
4. Move any initialization code from the frame window constructor to the init() method of the applet.
5. Remove the call to setSize; for applets, sizing is done with the width and height parameter in the HTML file.
6. Remove the call to setDefaultCloseOperation / windowClosing. An applet cannot be closed & it terminates when browser exits.
7. If the application calls setTitle, eliminate the call to the method. Applets cannot have title bars.
8. Dont call setVisible(true). The applet is displayed automatically.

SandBox security:

→ Whenever code is loaded from a remote site then executed locally, security becomes vital.

→ Visiting a webpage automatically starts all applets on the page

→ If clicking a link or visiting page could install arbitary code on the user's computer, there would be greater chance of losing confidental information.

→ To prevent it, java uses a security model "sandbox security model" to allow only those operations that are harmless.

→ To allow additional operations, the code must be digitally signed and the user must approve the signing certificate

→ programs in "sandbox security mode" have the following restrictions:

1) They can never run any local executable programs.
2) They cannot read from or write to the local computer's file system.
3) They cannot find out any information about the local computer, except for the java version used on a few harmless operating system detail's
4. Remotely loaded programs need user consent to communicate with any host other than the server from which they were downloaded.

getCodeBase(), "Ring09.wav");

5) All pop-up windows carry a warning message:- This message is a security feature to ensure that users do not mistake the windows for a local application

Playing sound in an Applet:-
.au, .wav, .midi

```
import java.awt.event.*
import javax.swing.*;
import java.awt.*;
import java.applet.*;

public class PlaySound Applet extends Applet
implements ActionListener {
    Button play, stop;
    AudioClip audioClip
    public void init() {
    play = new Button("play");
    add (play);
    play.addActionListener(this);
    stop = new Button("stop");
    add (stop);
    stop.addActionListener(this);
    audioClip. = getAudioClip(getCodeBase(),
                "Ring09.wav");
}

public void actionPerformed(ActionEvent e){
    if (e.getSource()==play) {
        audio clip.play();
    }
    else if(e.getSource == stop) {
        audioClip.stop();
} } }
```

Java Network Programming :-

```
java.net.*;
```

Classes on .net :

URL → represent any a url

URLConnection → to connect with URL

```
URL url = new URL("http://www.sanjayadhikari.com.np/index.php");
```

↳ MalformedURLException

//Analyzing URL :

```
import java.net.*;

public class URLDemo
{
public static void main(String args[]) {
    try {
        URL url = new URL("http://www.ncit.edu.np:80/download");
System.out.println(url.getProtocol());
System.out.println(url.getHost());
System.out.println(url.getPort());
System.out.println(url.getFile());
```

```
package network;

import java.net.*;
public class URLConnectionDemo {
    public static void main(String args[])
    throws Exception {
URL url = new URL("http://www.online
    khabar.com");
URLConnection urlCon = url.openConnection
                        ();
InputStream in = urlCon.getInputStream();
    int ch;
    while((ch = in.read()) != -1)
        System.out.print((char)ch);
    }
}
```

preserved order of msg.

User Datagram Protocol

TCP vs UDP

TCP	UDP
connection oriented	Transfer Layer protocol
always hold connection	connect whenever needed
Ordering of data packets	no ordering of data packets.
reliable (retransmit)	unreliable transport prota
comparitively slow (HTTP---: (Socket)	fast: (ted) multimeldia application

Connection less

Socket
→ server socket
→ client socket

TCPserver
TCPClient → socket

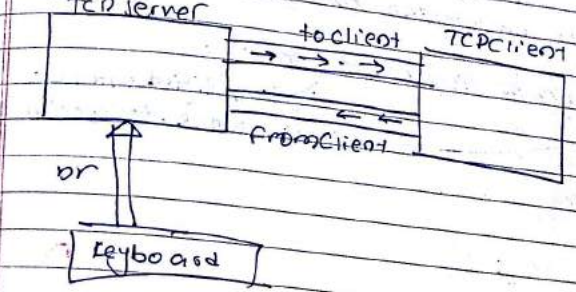
ServerSocket
socket
server

```
ServerSocket serverSock = new ServerSocket(9999);
Socket client_socket = serverSock.accept();
client :
Socket sock = new Socket("abc", 9999);
```

Client: Read from keyboard
send to server.
Read from server
Display msg from server

Server: Read from client
Display msg from client
Read from server keyboard
Send to client

Server:




```
import java.io.*;
import java.net.*;

class TCPServer
{ public static void main(String argv[])
  throws Exception {
  String clientsentence, modifiedsentence;
  ServerSocket WelcomeSocket = new
  serverSocket(6787); Socket connectionsocket = welcomeSocket.
accept(); BufferedReader fromClient = new BufferedReader(
  new InputStreamReader(connectionSocket.
  getInputStream()));
```

clientSentence = fromClient.readLine();
System.out.println("From client: " + clientSentence);
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
modifiedSentence = br.readLine();
DataOutputStream toClient = new DataOutputStream(connectionSocket.getOutputStream());
toClient.writeBytes(modifiedSentence + "\n");
connectionSocket.close();
}
}

modifiedSentence = clientSentence.toUpperCase();

Clientside:

```
import java.io.*;
import java.net.*;

class TCPClient {
public static void main(String args[])
throws Exception {
String sentence, modifiedsentence;
Socket clientSocket = new Socket("sa" 6789);
BufferedReader br = new BufferedReader(
new InputStreamReader(System.in));
sentence = br.readLine();
```

```
DataOutputStream toServer = new DataOutputStream(
clientSocket.getOutputStream());
toServer.writeBytes(sentence + "\n");
BufferedReader fromServer = new BufferedReader(
new InputStreamReader(clientSocket.getInput
Stream()));
modifiedSentence =
modifiedSentence = fromServer.readLine();
System.out.println("From server: " +
modifiedSentence);
clientSocket.close();
}
}
```

To communicate until exist

```
public static void main(String args[])
throws Exception
{ String clientSentence, serverSentence;
ServerSocket welcomeSocket = new ServerSocket
(6789);
Socket connectionSocket = welcomeSocket.acce
pt();
BufferedReader fromClient = new BufferedR-
eader(new InputStreamReader(connectionS
ocket.getInputStream()));
```

```
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

DataOutputStream toClient = new DataOutput
Stream(connectionSocket.getOutputStream
());

while(true) {
clientSentence = fromClient.readLine();
if(clientSentence.equalsIgnoreCase("Bye"))
break;

System.out.println("From Client: "+clientSentence);
System.out.print("Server: ");
serverSentence = br.readLine();
toClient.writeBytes(serverSentence + "\n");
}
welcomeSocket.close();
connectionSocket.close();
fromClient.close();
toClient.close();
}
```

Client to communicate until exit.

```
public static void main(String args[])
throws Exception {
String clientSentence, serverSentence;
Socket clientSocket = new Socket("nd", 6789);
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

DataOutputStream toServer = new DataOutputStream
(clientSocket.getOutputStream());
BufferedReader fromServer = new BufferedReader
(new InputStreamReader(clientSocket.getInput
Stream()));

while(true) {
System.out.print("Client: ");
clientSentence = br.readLine();
toServer.writeBytes(clientSentence + "\n");
serverSentence = fromServer.readLine();
if(serverSentence.equalsIgnoreCase("Bye"))
break;

System.out.println("From Server: " + serverSentence);
}
clientSocket.close();
fromServer.close();
toServer.close();
}
```

Serving multiple clients

```
package network
import java.net.*;
```

Reflection :

Reflection is a java API to deal with java class at runtime. It can be used to load the java class, call its methods or analyze the class at runtime

Reflection is a java language ability to inspect and dynamically calless, methods attributes at runtime.

```
package reflection;

import java.lang.reflection.*;

Public class ReflectionDemo {
    public static void main(String args[])
{
    class reflectClass = ReflectionTest.class
    String ClassName = Reflect class.getName();

    int modifier = Reflect class.getModified
                    ("Modifiers : ispublic(modifier);

    ~~Method[] method~~
    Method[] classMethod = reflectClass.getMethod
for (Method method : classMethod)
{   System.out.println ("Method name:" method.
                        getName());
System.out.println("Return Type:"Method.
                    getReturnType());
```