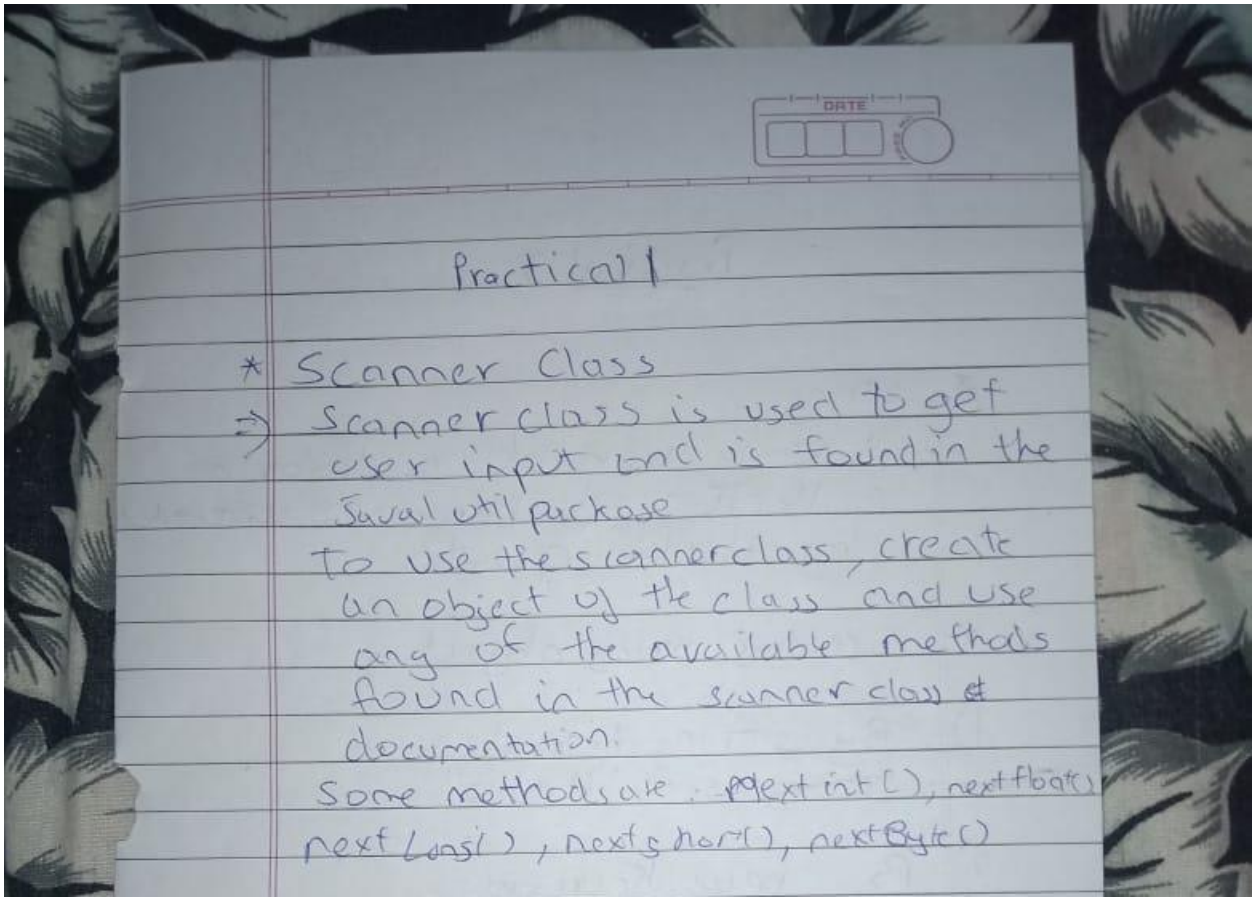
Practical 1



Practical 1

* Scanner Class

=> Scanner class is used to get user input and is found in the Java util package

To use the scanner class, create an object of the class and use any of the available methods found in the scanner class documentation.

Some methods are : nextInt(), nextFloat() nextLong(), nextShort(), nextByte()

Practical 1

Write a java program to demonstrate Scanner class.

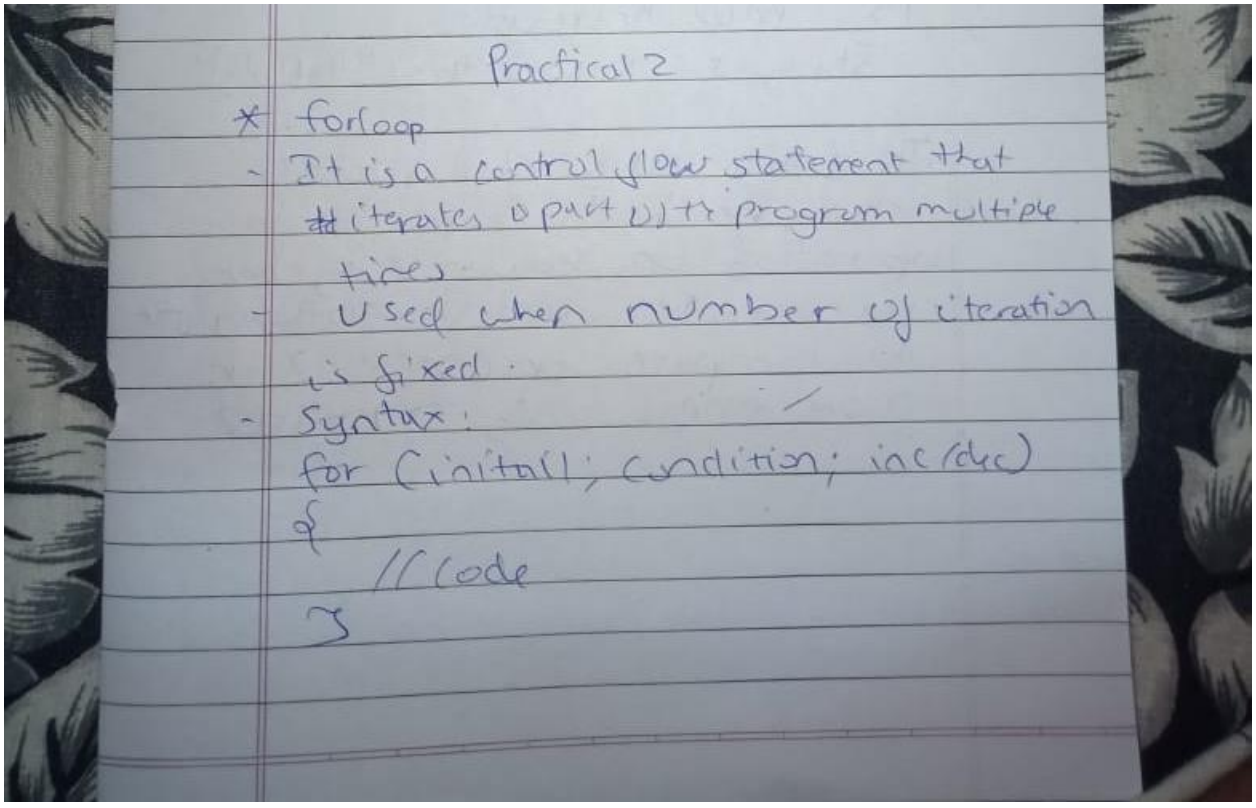
Input:

```
import java.util.Scanner;
public class ScannerDemo{
 public static void main(String[] args) {
   Scanner sc= new Scanner(System.in);
System.out.println("Enter First number");
   int a= sc.nextInt();
System.out.println("Enter Second number ");
   int b=sc.nextInt();
int c=a+b;
System.out.println("Sum of  "+a+" "+"and "+b+"  is  "+c);
}
}
```

Output:-

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac ScannerDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java ScannerDemo
Enter First number
10
Enter Second number
20
Sum of  10 and 20  is  30
```

Practical 2


Practical 2
* forloop
- It is a control flow statement that iterates a part of the program multiple times
- Used when number of iteration is fixed.
- Syntax:
for (initall; condition; inc/dec)
{
//code
}

Practical 2

Write a java program to display 3x3 matrixes. Find the sum, multiplication and transpose operation.

Input:

```
import java.util.Scanner;
public class MatrixAddition {
   public static void main(String[] args) {
      @SuppressWarnings("resource")
      Scanner scanner = new Scanner(System.in);
      int[][] matrix1 = new int[3][3];
      int[][] matrix2 = new int[3][3];
      System.out.println("Enter the elements in first matrix :");
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            matrix1[i][j] = scanner.nextInt();
         }
      }
      System.out.println("Enter the elements in second matrix :");
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            matrix2[i][j] = scanner.nextInt();
         }
      }
      int[][] sumMatrix = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
         }
      }
      int transpose1[][] = new int[3][3];
      int transpose2[][] = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            transpose1[j][i] = matrix1[i][j];
            transpose2[j][i] = matrix2[i][j];
         }
      }
      int[][] productMatrix = new int[3][3];
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
               productMatrix[i][j] = productMatrix[i][j] + matrix1[i][k] * matrix2[k][j];
            }
```

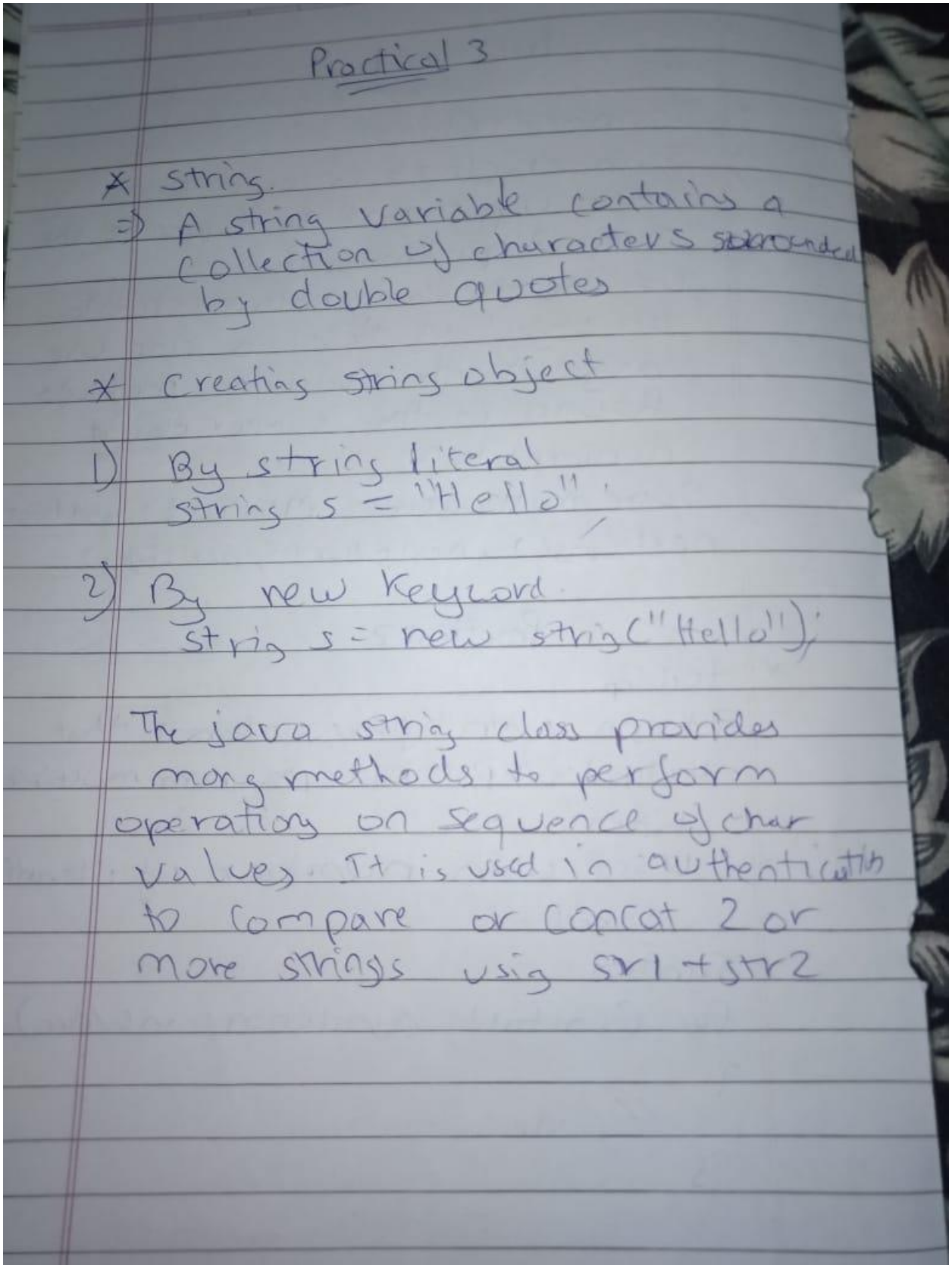
```
    }
  }

  System.out.println("\nFirst matrix is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      System.out.print(matrix1[i][j] + " ");
    }
    System.out.println();
  }
  System.out.println("\nSecond matrix is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      System.out.print(matrix2[i][j] + " ");
    }
    System.out.println();
  }
  System.out.println("\nThe sum of the two matrices is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      System.out.print(sumMatrix[i][j] + " ");
    }
    System.out.println();
  }
  System.out.println("\nThe product of the two matrices is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      System.out.print(productMatrix[i][j] + " ");
    }
    System.out.println();
  }
  System.out.println("\nThe transpose of the matrix 1 is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      System.out.print(transpose1[i][j] + " ");
    }
    System.out.println();
  }

  System.out.println("\nThe transpose of the matrix 2 is : ");
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
```

```
            System.out.print(transpose2[i][j] + " ");
        }
        System.out.println();
    }
  }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac MatrixAddition.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MatrixAddition
Enter the elements in first matrix :
2 5 1
2 7 5
4 5 4
Enter the elements in second matrix :
1 8 6
4 2 5
4 9 9

First matrix is :
2 5 1
2 7 5
4 5 4

Second matrix is :
1 8 6
4 2 5
4 9 9

The sum of the two matrices is :
3 13 7
6 9 10
8 14 13

The product of the two matrices is :
26 35 46
50 75 92
40 78 85

The transpose of the matrix 1 is :
2 2 4
5 7 5
1 5 4

The transpose of the matrix 2 is :
1 4 4
8 2 9
6 5 9
```

Practical 3



Practical 3

* String.
=> A string variable contains a collection of characters surrounded by double quotes

* Creating strings object

1) By string literal
String s = "Hello";

2) By new keyword
String s = new String("Hello");

The java string class provides many methods to perform operations on sequence of char values. It is used in authentication to compare or concat 2 or more strings using str1+str2

Practical 3.1

Write a java program accept n strings and sort them into ascending order.

Input:

```
import java.util.Scanner;
class SortString
{
public static void main(String args[])
{
        String temp;
        int diff=0,i,j;
        Scanner input = new Scanner(System.in);
        System.out.println("enter n string");
        int n = input.nextInt();
        Scanner input1 = new Scanner(System.in);
        String str[] = new String[n];
        System.out.println("enter string one by one");
        for(i=0;i<n;i++)
        {
           str[i] = input1.nextLine();
        }
        input.close();
        input1.close();
        for(i=0;i<n;i++)
            for(j=i+1;j<n;j++)
              if(str[i].compareTo(str[j])>0)
               {
                   temp = str[i];
                    str[i] = str[j];
                   str[j] = temp;
               }
        System.out.print("Strings in Sorted Order:");
        for (i = 0; i<n; i++)
             System.out.print(str[i] + ", ");
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac SortString.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java SortString
Enter n string
5
Enter string one by one
gaurav
sanjeev
akshay
nikhil
rohit
string in sorted order :akshay,gaurav,nikhil,rohit,sanjeev,
```

Practical 3.2

Write a java program to demonstrate string and string buffer methods.

Input:

```
//Aim : string and stringbuffer class
class StringDemo{

    public static void main(String[] args) {
        String s1="nikhil singh",s2="SYCS 45";
        System.out.println("strings s1 = "+s1+" s2 = "+s2);
        System.out.println("conversion to uppercase: "+s1.toUpperCase());
        System.out.println("conversion to lowercase: "+s2.toLowerCase());
        System.out.println("s1==s2 ? "+s1.equals(s2));
        System.out.println("s1==s2 ignore casing ? "+s1.equalsIgnoreCase(s2));
        System.out.println("s1+s2 ="+s1.concat(s2));
        System.out.println("s1 compare s2 ?"+s1.compareTo(s2));
        System.out.println("s1 charat index 5 :"+s1.charAt(5));
        System.out.println("s1 substring from 3 to 5 :"+s1.substring(3,5));
        System.out.println("s1 valueof 12 :"+s1.valueOf(12));
        System.out.println("s2 index of C :"+s2.indexOf("C"));
        System.out.println("s1 trim :"+s1.trim());
        System.out.println("s1 replace i with z :"+s1.replace('i','z'));
        System.out.println("s1 length:"+s1.length());
        System.out.println("s2 length:"+s2.length());
        //string buffer
        StringBuffer sb=new StringBuffer(s1);
        int pos =sb.indexOf("s");
        System.out.println("string buffer: "+sb);
        sb.insert(pos,s2);
        System.out.println("insert s2 at s "+sb);
        sb.setCharAt(5,'-');
        System.out.println("stcharat 5 as - :"+sb);
        sb.append("hii");
        System.out.println("append hii :"+sb);
        sb.setLength(30);
        System.out.println("updated length:"+sb.length());
    }
}
```

Output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac StringDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java StringDemo
strings s1 = nikhil singh s2 = SYCS 45
conversion to uppercase: NIKHIL SINGH
conversion to lowercase: sycs 45
s1==s2 ? false
```

```
s1==s2 ignore casing ? false
s1+s2 =nikhil singhSYCS 45
s1 compare s2 ?27
```

Practical 3.3

Write a java program to demonstrate tokenizer.

Input:

```
import java.util.StringTokenizer;
class TokenizerDemo{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my college is bhavans");
   System.out.println("total no of token: "+st.countTokens());
     while (st.hasMoreTokens()) {
        System.out.println(st.nextToken());
     }

   }
}
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac TokenizerDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java TokenizerDemo
total no of token: 4
my
college
is
bhavans
```

Practical 3.4

Write a java program to print the pattern

input

```
class Triangle{
    public static void main(String[] args) {
        int num= Integer.parseInt(args[0]),i,j;
        for(i=0;i<=num;i++){
            for(j=1;j<=i;j++){
                System.out.print(" ");
                System.out.print(i);
            }
            System.out.println("\n");
        }
        System.out.println("done");
    }
}
```

output

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Triangle 6

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

done

Practical 3.5

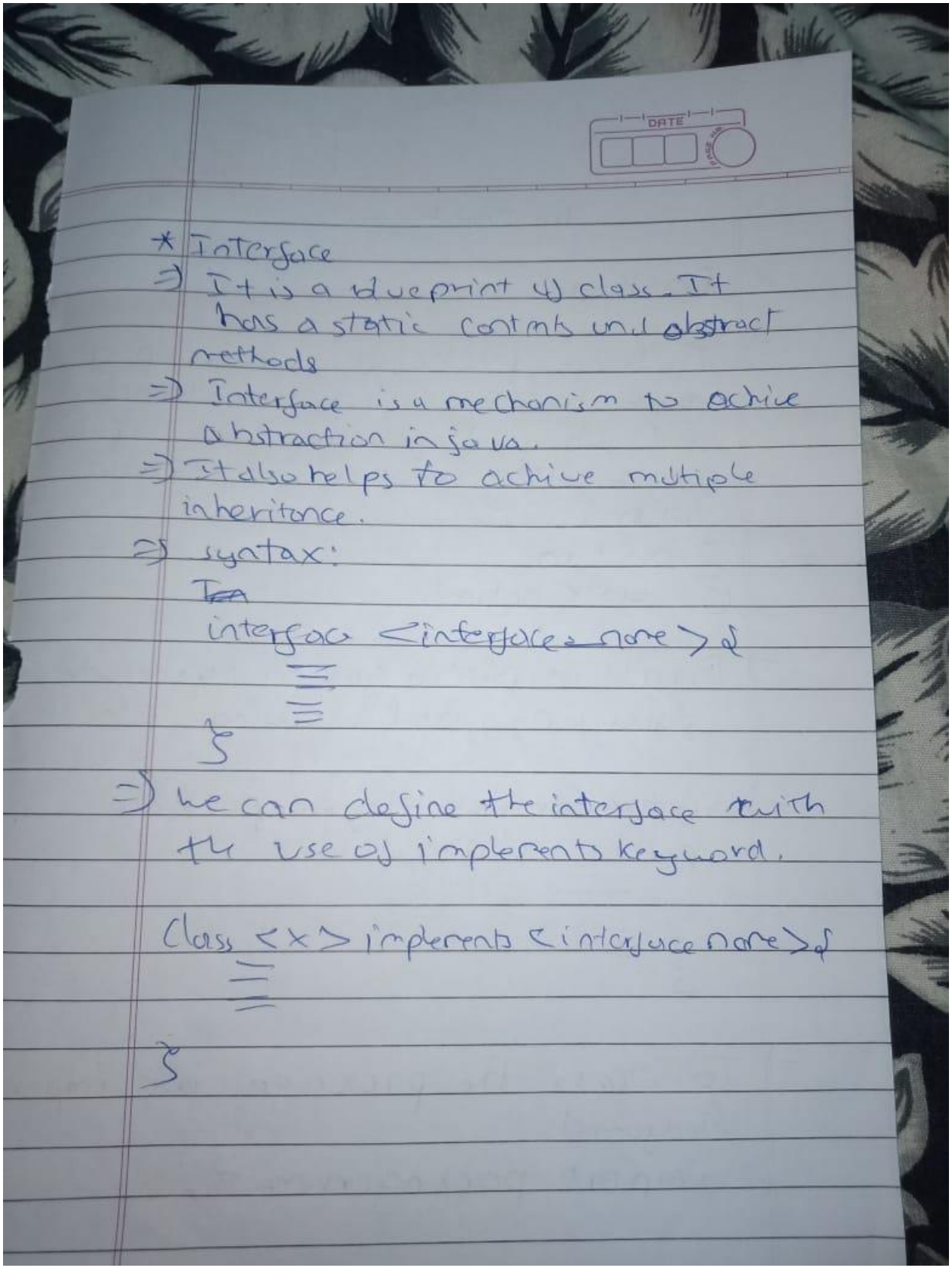
Write a java program to sort a given array

```
//Input
class Sort{
    int number[]={55,65,23,45,66};
    int n=number.length;
    void display(){
        for(int i=0;i<n;i++){
            System.out.println(" "+number[i]);
        }
    }
    void sort(){
        for(int i=0;i<n-1;i++){
            for(int j=1;j<n;j++){
                if(number[j]<number[i]){
                    int temp= number[i];
                    number[i]=number[j];
                    number[j]=temp;
                }
            }
        }
    }
    public static void main(String[] args) {
        Sort s1=new Sort();
        System.out.println("given list");
        s1.display();
        s1.sort();
        System.out.println("sorted list");
        s1.display();
    }
}
```

output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3\java>javac Sort.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3\java>java Sort
given list
 55
 65
 23
 45
 66
sorted list
 23
 45
 55
 65
 66
```

Practical 4



* Interface
=> It is a blueprint of class. It has a static constants and abstract methods
=> Interface is a mechanism to achive abstraction in java.
=> It also helps to achive multiple inheritance.
=> syntax:

```
interface <interface_name> {
    ==
    ==
}
```

=> we can define the interface with the use of implements keyword.

```
Class <x> implements <interface name> {
    ==
    ==
}
```

Practical 4

Create a package: Animals. In package animals create interface Animal with suitable behaviors. Implement the interface Animal in the same package animals.

INPUT:

```
package Animals;
public interface Animal
{
  void type();
  void food();
}

import Animals.*;
class Elephant implements Animal
{
  public void type()
    {
       System.out.println("\n Elephant is viviparious");
    }
    public void food()
    {
      System.out.println("\n Elephant is herbivorious");
     }
}


class DemoPackage
{
  public static void main(String[] args)
  {
     Elephant e1 = new Elephant();
     e1.type();
     e1.food();
  }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java\Animals>javac Animal.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java\Animals>cd ..

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac DemoPackage.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java DemoPackage
 Elephant is viviparious

 Elephant is herbivorious
```

Practical 4.1
Create a java package to illustrate the interface and implement in a java program.
Input:

```
interfaceCallback {

voidcallback(intparam);

}

class Client implements Callback {

// Implement Callback's interface

public void callback(int p) {

System.out.println("callback called with " + p);

}

}

classTestIface {

public static void main(String args[]) {

Callback c = new Client();

c.callback(42);

}

}
```

OUTPUT;

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Callback.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac TestIface.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java TestIface

callback called with 42
```

Practical 4.2

Write a java program to demonstrate classes that implements interfaces may also define other members too.

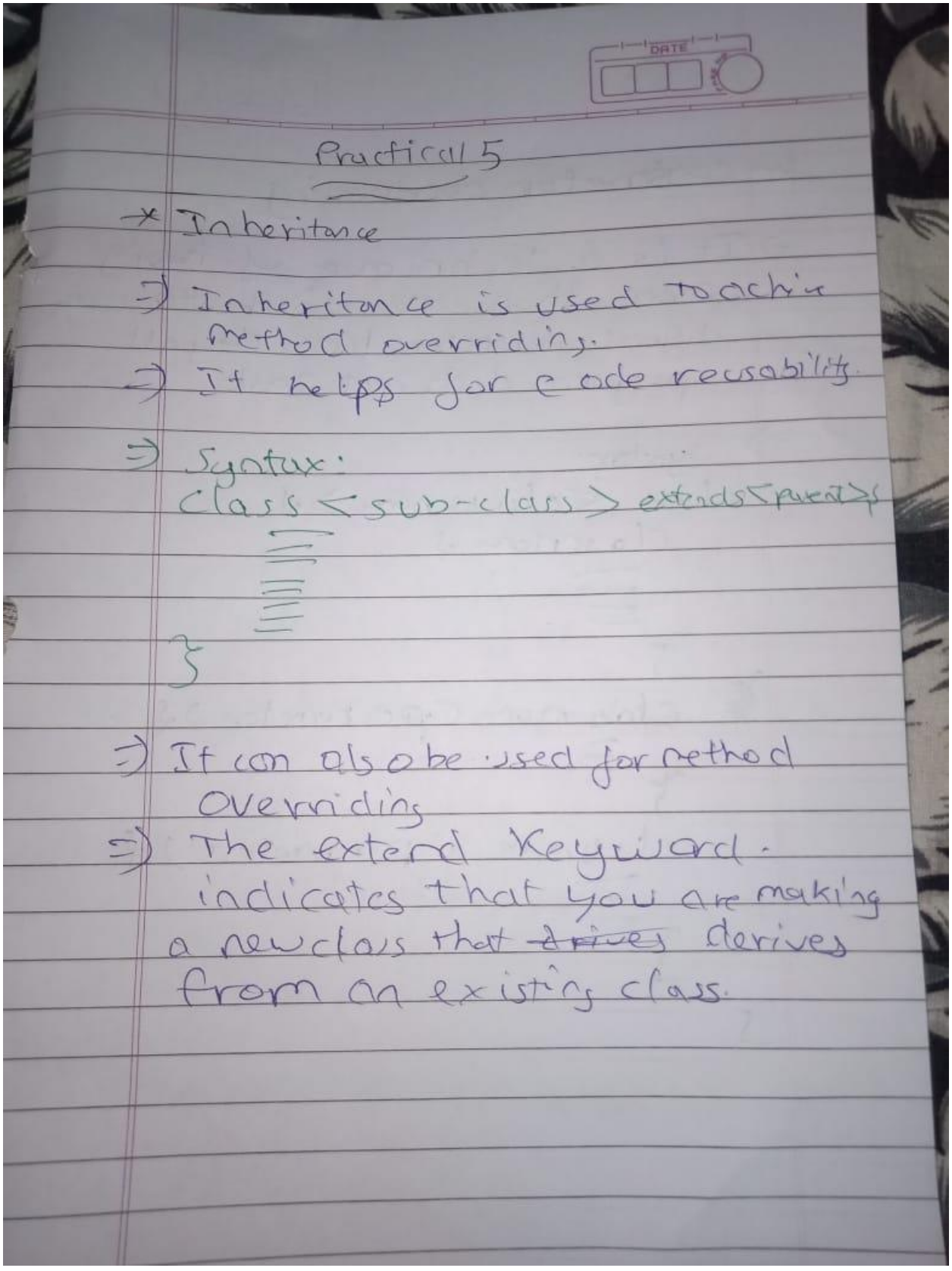
Input:

```
interfaceCallback {
voidcallback(intparam);
}
classAnotherClient implements Callback {
// Implement Callback's interface
public void callback(int p) {
System.out.println("Square is " + (p*p));
}
public void otherMember(int a, int b){
System.out.println("Addition is: " +(a+b));
}
}
class TestIface2 {
public static void main(String args[]) {
AnotherClient a=new AnotherClient();
a.callback(2);
a.otherMember(10,5);
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac TestIface2.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java TestIface2
Square is 4
Addition is: 15
```

Practical 5



Practical 5

* Inheritance

⇒ Inheritance is used to achieve method overriding.

⇒ It helps for code reusability.

⇒ Syntax:

class <sub-class> extends <parent> {

}

⇒ It can also be used for method overriding

⇒ The extend keyword indicates that you are making a new class that ~~drives~~ derives from an existing class.

Practical 5

Demonstrate Java inheritance using extends keyword.

Input:

```
import java.util.Scanner;
class Inheritance{
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int k=0;
        while(k<5){
            System.out.println("\nSelect the Operation :\n");
            System.out.println("1.Display array elements");
            System.out.println("2.Display array sum");
            System.out.println("3.Display array average");
            System.out.println("4.Display array maximum");
            System.out.println("5.break");
            k=sc.nextInt();
            Menu m=new Menu(k);
        }
    }
}
class Array{
    public  int[] a = new int[]{ 1,2,3,4,5,6,7,8,9,10 };
}
class Menu extends Array{
    int i,sum=0,avg=0,max=0;
    Menu(int k){
        switch (k) {
            case 1:
                for(i=0;i<a.length;i++){
                    System.out.print(a[i]);
                }
                break;
            case 2:
                for(i=0;i<a.length;i++){
                    sum+=a[i];
                }
                System.out.println("sum :"+sum);
                break;
            case 3:
                for(i=0;i<a.length;i++){
                    sum+=a[i];
                }
                avg=sum/a.length;
                System.out.println("average: "+avg);
                break;
```

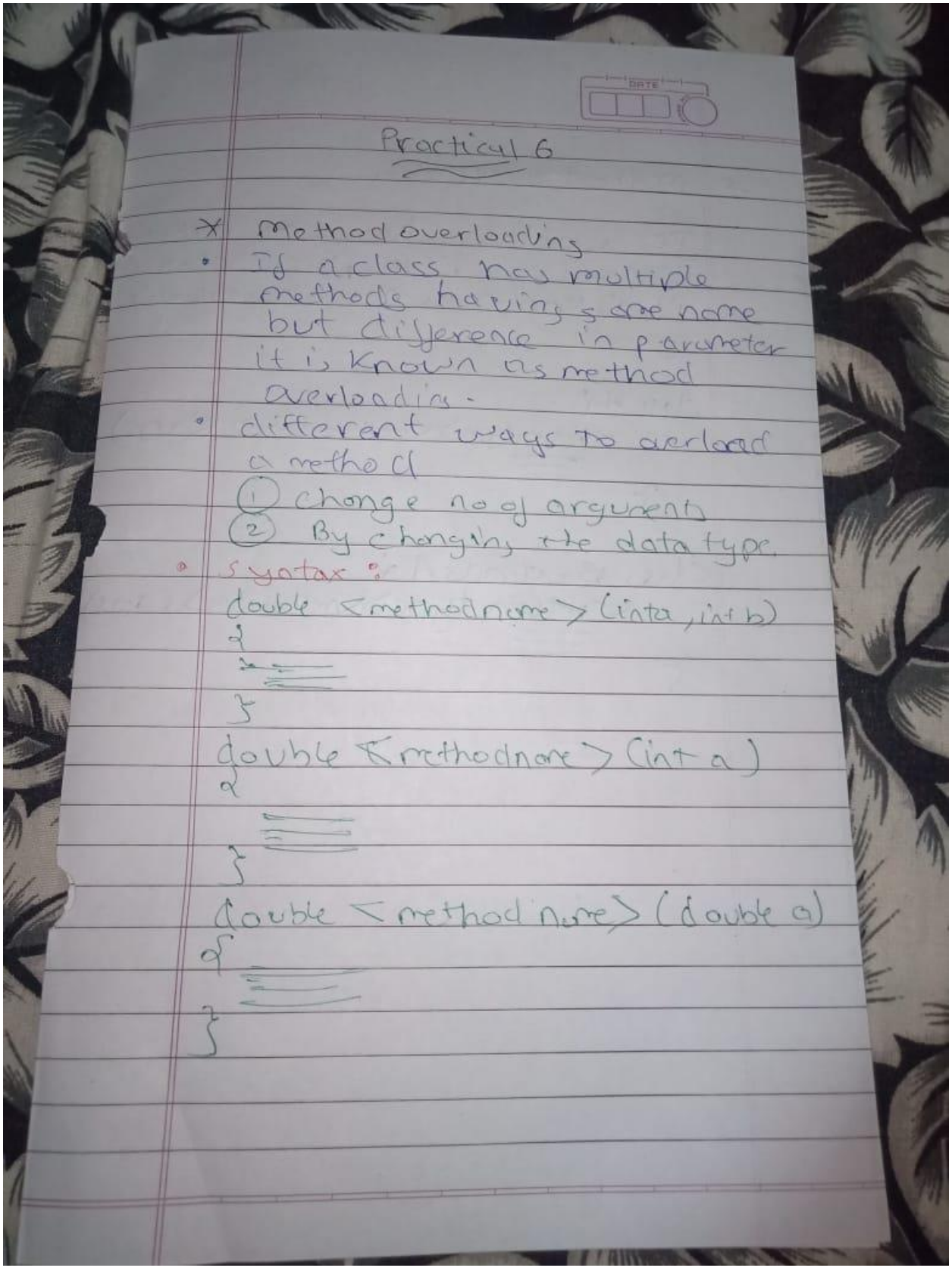
```
            case 4:
                for(i=0;i<a.length;i++){
                    if(a[i]>max){
                        max=a[i];
                    }
                }
                System.out.println("max "+max);
                break;
            default:

                break;
        }
    }
}
```

Output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Inheritance.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Inheritance

Select the Operation :

1.Display array elements
2.Display array sum
3.Display array average
4.Display array maximum
5.break
1
12345678910
Select the Operation :

1.Display array elements
2.Display array sum
3.Display array average
4.Display array maximum
5.break
2
sum :55

Select the Operation :

1.Display array elements
2.Display array sum
3.Display array average
4.Display array maximum
5.break
```

```
3
average: 5

Select the Operation :

1.Display array elements
2.Display array sum
3.Display array average
4.Display array maximum
5.break
4
max 10

Select the Operation :

1.Display array elements
2.Display array sum
3.Display array average
4.Display array maximum
5.break
5
```

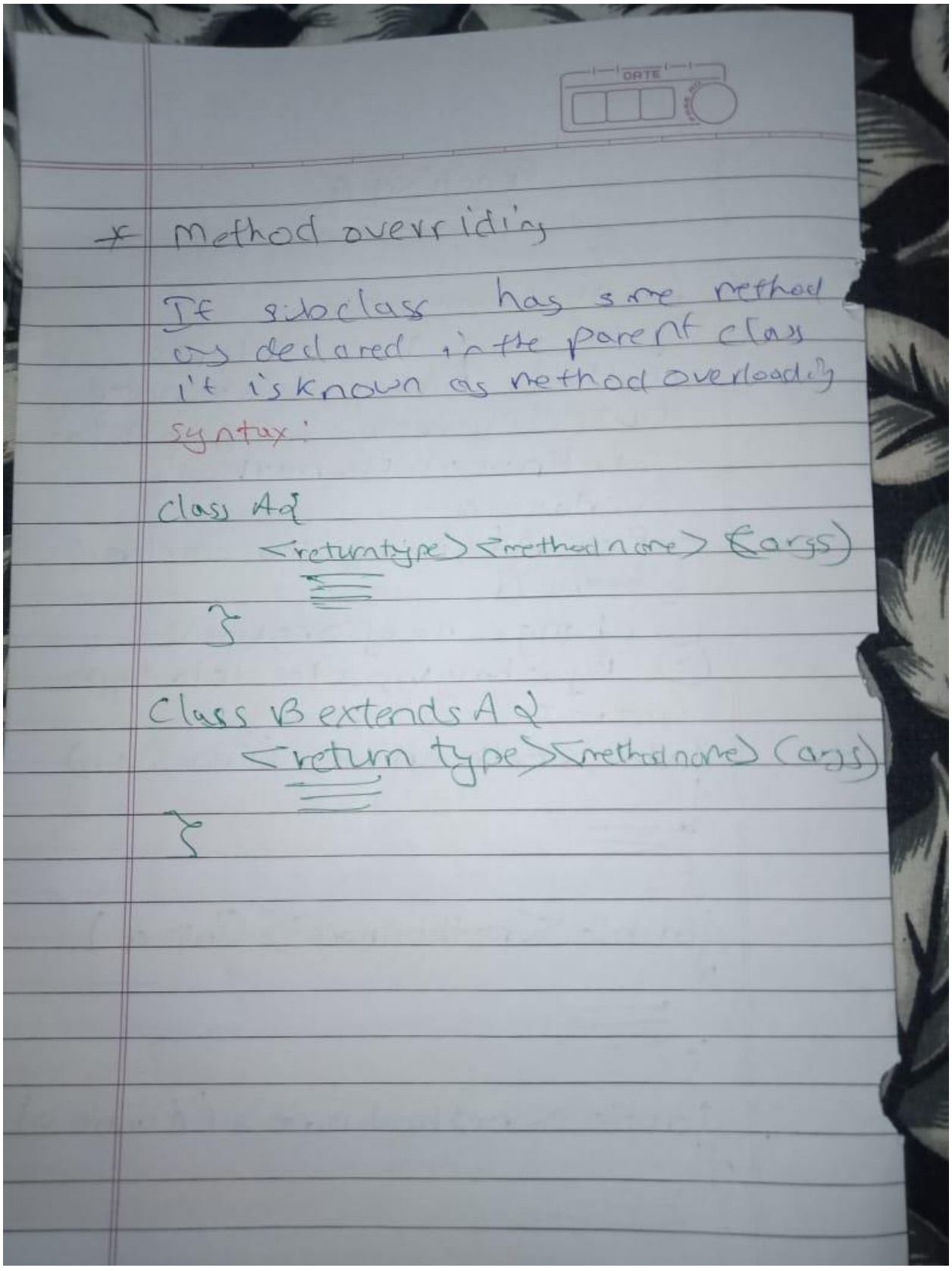
Practical 6



Practical 6

* Method overloading
- If a class has multiple methods having same name but difference in parameter it is known as method overloading.
- different ways to overload a method
  1. change no of arguments
  2. By changing the data type.
- Syntax :

```
double <methodname> (int a, int b)
{
}
double <methodname> (int a)
{
}
double <methodname> (double a)
{
}
```

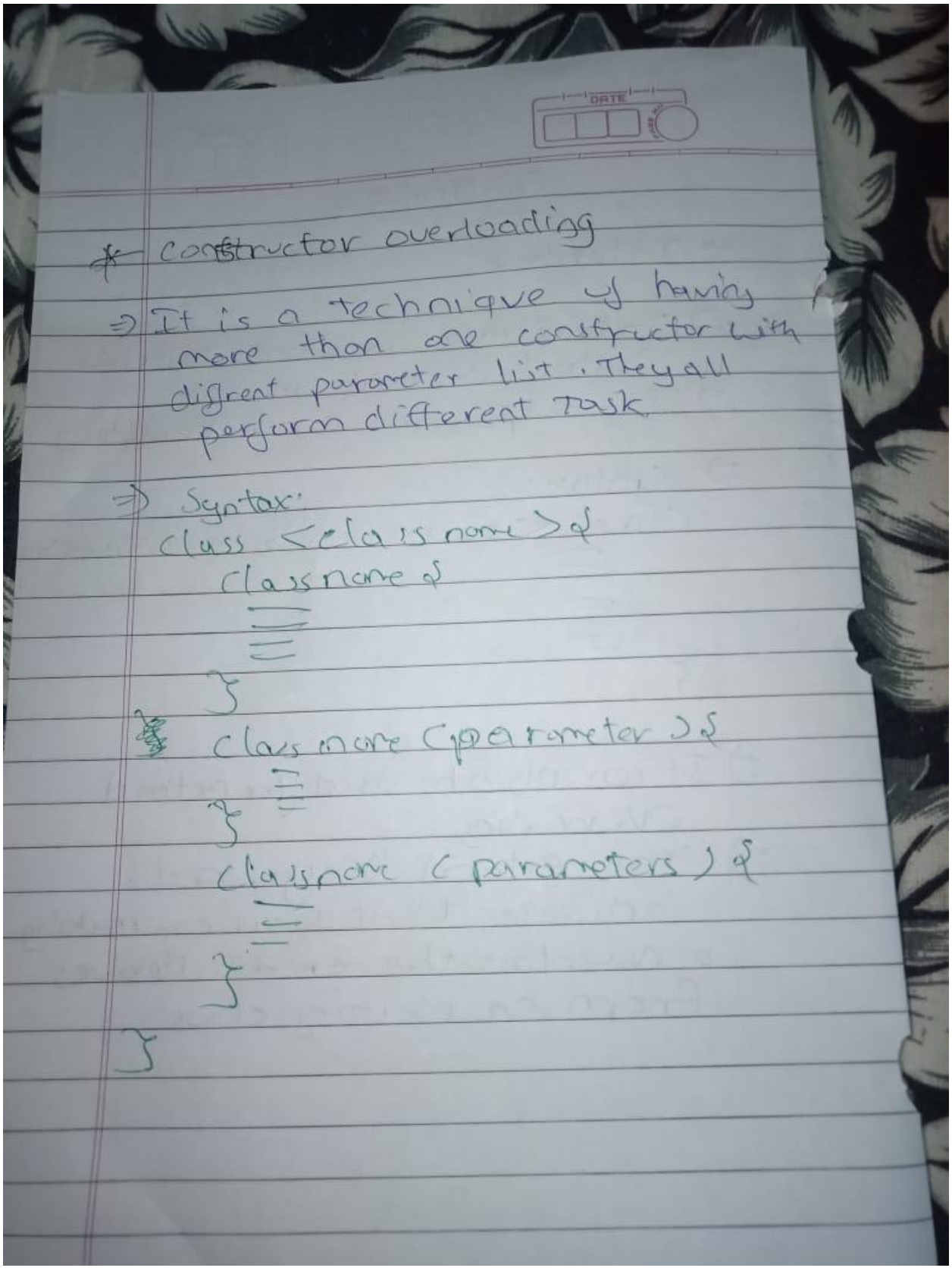


Method overriding

If subclass has same method as declared in the parent class it is known as method overloading

syntax:

```
class A{
    <returntype> <methodname> (args)
    {
    }

Class B extends A{
    <return type> <method name> (args)
    {
    }
```

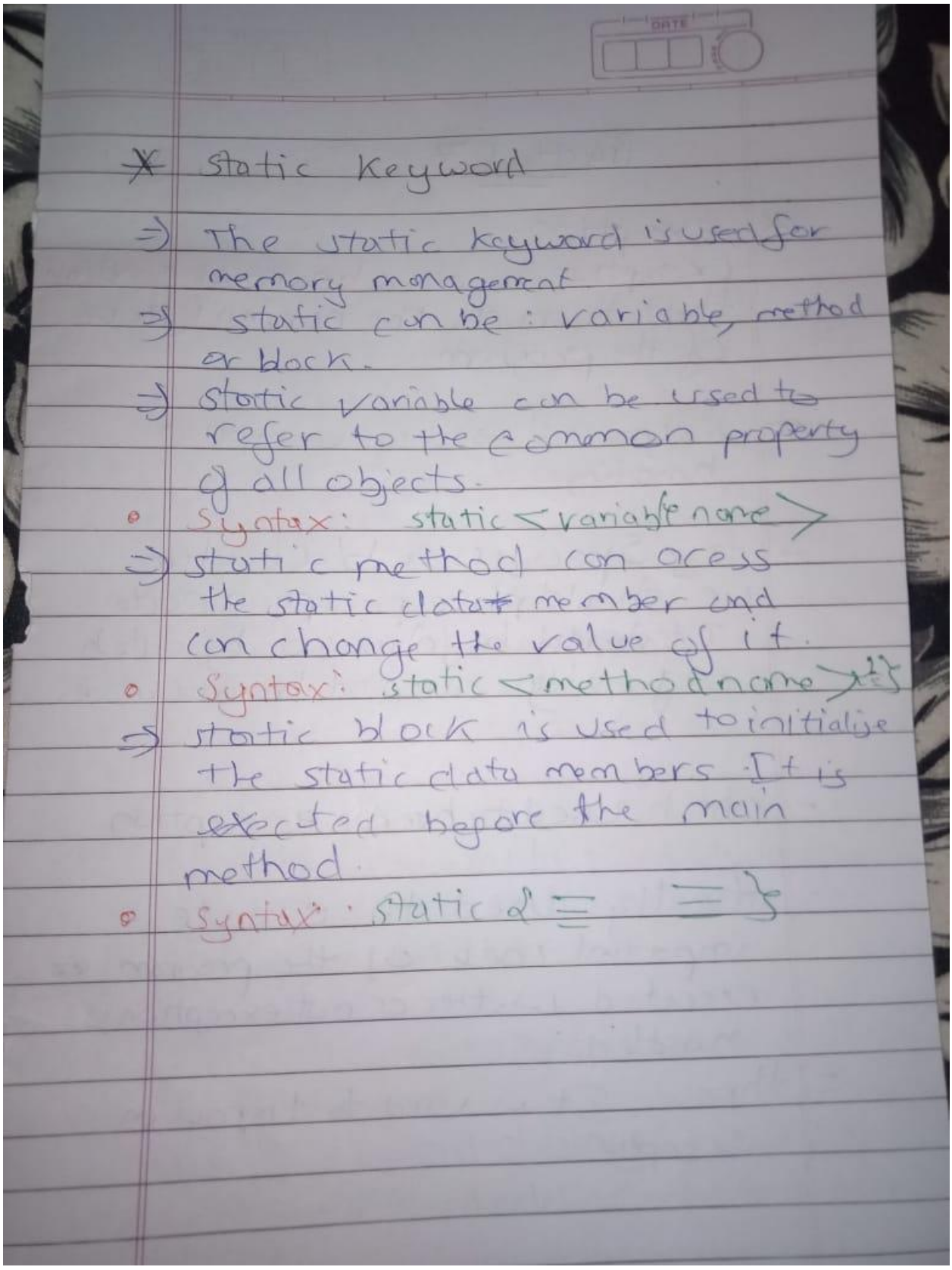


* Constructor overloading

⇒ It is a technique of having more than one constructor with diffrent parameter list. They all perform different task.

⇒ Syntax:

```
class <class name>{
    classname(){
    ==
    ==
    }
    classname(parameter){
    ==
    }
    classname(parameters){
    ==
    }
}
```



Static Keyword

- The static keyword is used for memory management
- static can be : variable, method or block.
- static variable can be used to refer to the common property of all objects.
  - Syntax: static <variable name>
- static method can access the static data member and can change the value of it.
  - Syntax: static <methodname>{}
- static block is used to initialise the static data members. It is executed before the main method.
  - Syntax: static {}

Practical 6

Aim:

Demonstrate method overloading and method overriding in Java.

INPUT:

```
//aim: show method Overriding
class Override{
    public static void main(String[] args) {
        Derived d= new Derived(5,6);
        d.Display();
    }
}
class Base{
    int x;
    Base(int p){
        this.x=p;
    }
    void Display(){
        System.out.println("inside base x= "+ x);
    }
}
class Derived extends Base{
    int y;
    Derived(int x,int y){
        super(x);
        this.y=y;

    }
    void Display()
    {
        System.out.println("Base x = "+x);
        System.out.println("Derived y ="+y);
    }
}
```

OUTPUT:

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Override.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Override

Base x = 5

Derived y =6

Practical 6.1

Aim:

Write a java program to demonstrate function overloading to compute volume

INPUT:

```
//Aim : show polymorphism
//method overloading
class Volume{
    double myv(double x){
        return x*x*x;
    }
    int myv(int x,int y,int z){
        return x*y*z;
    }
    double myv(int r){
        return (4/3)*3.14*r*r*r;
    }
}
class VolumeDemo{
    public static void main(String[] args) {
        Volume v1=new Volume();
        System.out.println("volume of cube of length 5.1 is " +v1.myv(5.1));
        System.out.println("volume of cuboid of dimensions 5*5*5 is
"+v1.myv(5,5,5));
        System.out.println("volume of sphere of radius 5 is "+v1.myv(5));
    }
}
```

Output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac VolumeDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java VolumeDemo
volume of cube of length 5.1 is 132.65099999999998
volume of cuboid of dimensions 5*5*5 is 125
volume of sphere of radius 5 is 392.5
```

Practical 6.2

Aim:

Write a java program to demonstrate function recursion and compute factorial of a given number.

INPUT:

```
import java.util.Scanner;
class FactorialDemo {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number:");
      int num = sc.nextInt();
      int factorial = fact(num);
      System.out.println("Factorial of entered number is: " + factorial);
   }
   static int fact(int n) {
      int output;
      if (n == 1 || n == 0) {
         return 1;
      }
      output = fact(n - 1) * n;
      return output;
   }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac FactorialDemo.java


C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java FactorialDemo

Enter the number:

7

Factorial of entered number is: 5040
```

Practical 6.3

Aim:

Write a java program to accept height and weight of ten different people and find person whose weight is less than 50 and height is more than 170.

input

```
class Counting{
    public static void main(String[] args) {
        int i,count=0,count1=0,count2=0;
        double[]
        wieght={45.2,  42  ,51.2 ,30.3 ,21   ,67 ,70.2 ,20.1 ,86.6 ,70.10},
        height={123.1,176.2,160.1,170.5,176.4,150,128.6,182.1,193.2,180};
        for(i=0;i<=9;i++){
            if(wieght[i]<50 && height[i]>170){
                count1++;
            }
            count++;
        }
        count2=count-count1;
        System.out.println("num of people with");
        System.out.println("hieght>170 and wieght <50 is "+count1 );
        System.out.println("others is "+count2);

    }
}
```

output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Counting
num of people with
hieght>170 and wieght <50 is 4
others is 6
```

Practical 6.4

Aim:

Write a java program to input a telephone no. and number of calls calculate and display bill amount which includes a fix rent of rupees 400, the first 150 cost are free with excess calls charge 80 paise each.

```
//Input

class BillRate{
    double rate=400;
    double check(int calls){

        if(calls>150){
            rate+=(calls-150)*0.80;
        }
        return rate;
    }
}
class BillDemo{
    static int phno,calls;
    public static void main(String[] args) {//command line argument
        phno=Integer.parseInt(args[0]);
        calls=Integer.parseInt(args[1]);
        BillRate obj=new BillRate();
        double amount=obj.check(calls);
        System.out.println("total bill for "+phno+" for "+calls +" calls is "+amo
unt);
    }
}
```

output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3\java>javac BillDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3\java>java BillDemo 8080 150
total bill for 8080 for 150 calls is 400.0

C:\Users\Nikhil\Desktop\bsc cs\sem 3\java>java BillDemo 8080 180
total bill for 8080 for 180 calls is 424.0
```

Practical 6.5

Aim:

Write a java program to demonstrate constructor overloading.

INPUT:

```
//Aim : show polymorphism
//constructor overloading
class StudentData
{
   int stuID;
   String Name;
   int Age;
   StudentData()
   {
       stuID = 100;
       Name = "nikhil singh";
       Age = 18;
   }
   StudentData(int num1, String str, int num2)
   {

       stuID = num1;
       Name = str;
       Age = num2;
   }

   public int getStuID() {
       return stuID;
   }
   public void setStuID(int stuID) {
       this.stuID = stuID;
   }
   public String getName() {
       return Name;
   }
   public void setName(String Name) {
       this.Name = Name;
   }
   public int getAge() {
       return Age;
   }
   public void setStuAge(int Age) {
       this.Age = Age;
   }

   public static void main(String args[])
```

```
   {
        //This object creation would call the default constructor
        StudentData myobj = new StudentData();
        System.out.println("Student Name is: "+myobj.getName());
        System.out.println("Student Age is: "+myobj.getAge());
        System.out.println("Student ID is: "+myobj.getStuID());

        /*This object creation would call the parameterized
         * constructor StudentData(int, String, int)*/
        StudentData myobj2 = new StudentData(555, "shivam", 25);
        System.out.println("Student Name is: "+myobj2.getName());
        System.out.println("Student Age is: "+myobj2.getAge());
        System.out.println("Student ID is: "+myobj2.getStuID());
  }
}
```

Output

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac StudentData.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java StudentData
Student Name is: nikhil singh
Student Age is: 18
Student ID is: 100
Student Name is: shivam
Student Age is: 25
Student ID is: 555
```

Practical 6.6

Aim:

Write a java program to demonstrate static data(variable).

INPUT:

```
class stateg
{
	static int num=0;
	stateg()
	{
		num++;
		System.out.println(num);
	}
	public static void main(String args[])
	{
		stateg c1=new stateg();
		stateg c2=new stateg();
		stateg c3=new stateg();
	}
}
```

Output:

```
1
2
3
```

Practical 6.7

Aim:

Write a java program to demonstrate static method.

INPUT:

```
class Employee
{
   int eid;
   String name;
   static String company = "JIO";
         static void change()
         {
                  company = "Airtel";
   }
   Employee(int x, String y)
         {
                  eid = x;
                  name = y;
   }
   void display()
         {
                  System.out.println(eid+" "+name+" "+company);
         }
}
public class staticmethod{
   public static void main(String args[])
         {
                  Employee.change();
                  Employee e1 = new Employee(1,"XYZ");
                  Employee e2 = new Employee(2,"PQR");
                  Employee e3 = new Employee(3,"ABC");
                  e1.display();
                  e2.display();
                  e3.display();
   }
}
```

Output:

```
1 XYZ Airtel
2 PQR Airtel
3 ABC Airtel
```

Practical 6.8

Aim:

Write a java program to demonstrate static block.

INPUT:

Code:

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

Output:

```
This is static block
This is main block
```

Practical 6.9

Aim:

Write a java program having customer number customer name and city as its class variable design getter and setter methods to function on customer class.

INPUT:

Code:

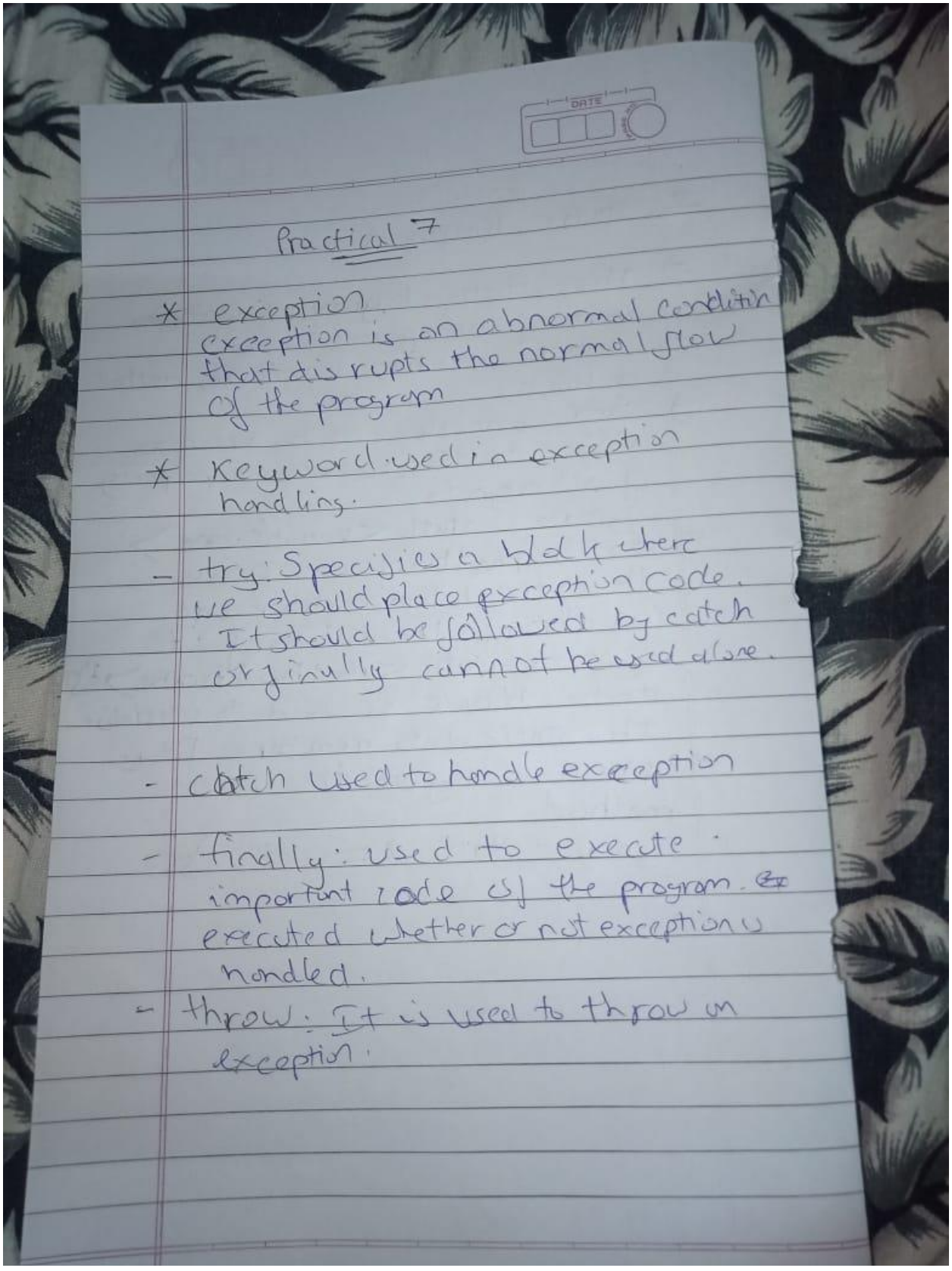
```
import java.util.*;
class customer
{
	void numb(int n)
	{
		int x;
		x=n;
		System.out.println("Customer number is "+x);
	}
	void name(String s)
	{
		String na;
		na=s;
		System.out.println("Customer name is "+na);
	}
	void city(String c)
	{
		String ci;
		ci=c;
		System.out.println("Customer name is "+ci);
	}
}
class customerDemo
{
	public static void main(String arg[])
	{
		customer c=new customer();
		customer c1=new customer();
		System.out.println("Getter Method");
		Scanner in=new Scanner(System.in);
		System.out.println("Enter Customer number");
		int z=in.nextInt();
		Scanner i=new Scanner(System.in);
		System.out.println("Enter Customer name");
		String a;
		a=i.nextLine();
```

```
                Scanner ix=new Scanner(System.in);
                System.out.println("Enter Customer city");
                String f;
                f=ix.nextLine();
                c.numb(z);
                c.name(a);
                c.city(f);
                System.out.println("-------------------------------");
                System.out.println("Setter Method");
                c1.numb(50);
                c1.name("Shubham");
                c1.city("Mum");
        }
}
```

Output:

```
Getter Method
Enter Customer number
20
Enter Customer name
ABC
Enter Customer city
LMN
Customer number is 20
Customer name is ABC
Customer name is LMN
-------------------------------
Setter Method
Customer number is 50
Customer name is XYZ
Customer name is PQR
```

Practical 7



Practical 7

* exception
exception is an abnormal condition that disrupts the normal flow of the program

* Keyword used in exception handling.

- try: Specifies a block where we should place exception code. It should be followed by catch or finally cannot be used alone.

- catch used to handle exception

- finally: used to execute important code of the program. Ex executed whether or not exception is handled.

- throw: It is used to throw an exception.

Practical 7

Aim:

Demonstrate creating your own exception in Java.

INPUT:

```
class ItemNotFound extends Exception {
   public ItemNotFound(String s) {
      super(s);
   }
}
class Demo {
   static void find(int arr[], int item) throws ItemNotFound {
      boolean flag = false;
      for (int i = 0; i < arr.length; i++) {
         if (item == arr[i])
            flag = true;
      }
      if (!flag) {
         throw new ItemNotFound("Item Not Found"); //calling constructor of user-defined exception
class
      } else {
         System.out.println("Item Found");
      }
   }
   public static void main(String[] args) {
      try {
         find(new int[] {1,2,3}, 4);
      } catch (ItemNotFound i) {
         System.out.println(i);
      }
   }
}
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Demo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Demo
ItemNotFound: Item Not Found
```

Practical 7.1

Aim:

Exception handling using try and catch

INPUT:

Aim:
Exception handling using try and catch
INPUT:

```
public class TryCatch {

   public static void main(String[] args) {
      try
      {
      int data=50/0;
      }
      catch(ArithmeticException e)
      {
         System.out.println(e);
      }
      System.out.println("rest of the code");
   }

}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac TryCatch.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java TryCatch
java.lang.ArithmeticException: / by zero
rest of the code
```

Practical 7.2

Aim:

Write a java program for Error handling.

INPUT:

```
class Main {

  public static void main(String[] args) {



    try {

      int divideByZero = 2 / 0;

System.out.println("Dividing");

    }

    catch (ArithmeticException ae) {

System.out.println("ArithmeticException => " + ae.getMessage());

    }

  }

}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Main.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Main
ArithmeticException => / by zero
```

Practical 7.3

Aim:

Write a java program to demonstrate Multiple catch clause

INPUT:

```
class MultipleTryCatchDemo{
static void ProcedureA(){
try{
System.out.println("Inside procedure A");
throw new RuntimeException("Demo");
}
finally{
System.out.println("Inside procedure A finally");
}

}static void ProcedureB(){
try{
System.out.println("Inside procedure B");
return;
}
finally{
System.out.println("Inside procedure B finally");
}

}
static void ProcedureC(){
try{
System.out.println("Inside procedure C");

}
finally{
System.out.println("Inside procedure C finally");
}

}
public static void main(String args[]){
try{
ProcedureA();
}
catch (Exception e){

System.out.println("Caught Exception" +e);

  }
ProcedureB();
```

```
ProcedureC();
}
}
```

Output

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac MultipleTryCatchDEmo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MultipleTryCatchDemo

Inside procedure A

Inside procedure A finally

Caught Exceptionjava.lang.RuntimeException: Demo

Inside procedure B

Inside procedure B finally

Inside procedure C

Inside procedure C finally

Practical 7.4

Aim:

Write a program to demonstrate the use of throwable class, by illustrating that the object of class that extends throwable can be thrown and caught. (Hint: Note that Exception is subclass of Throwable and thus create a user defined class MyException that will extends
Throwable class.)

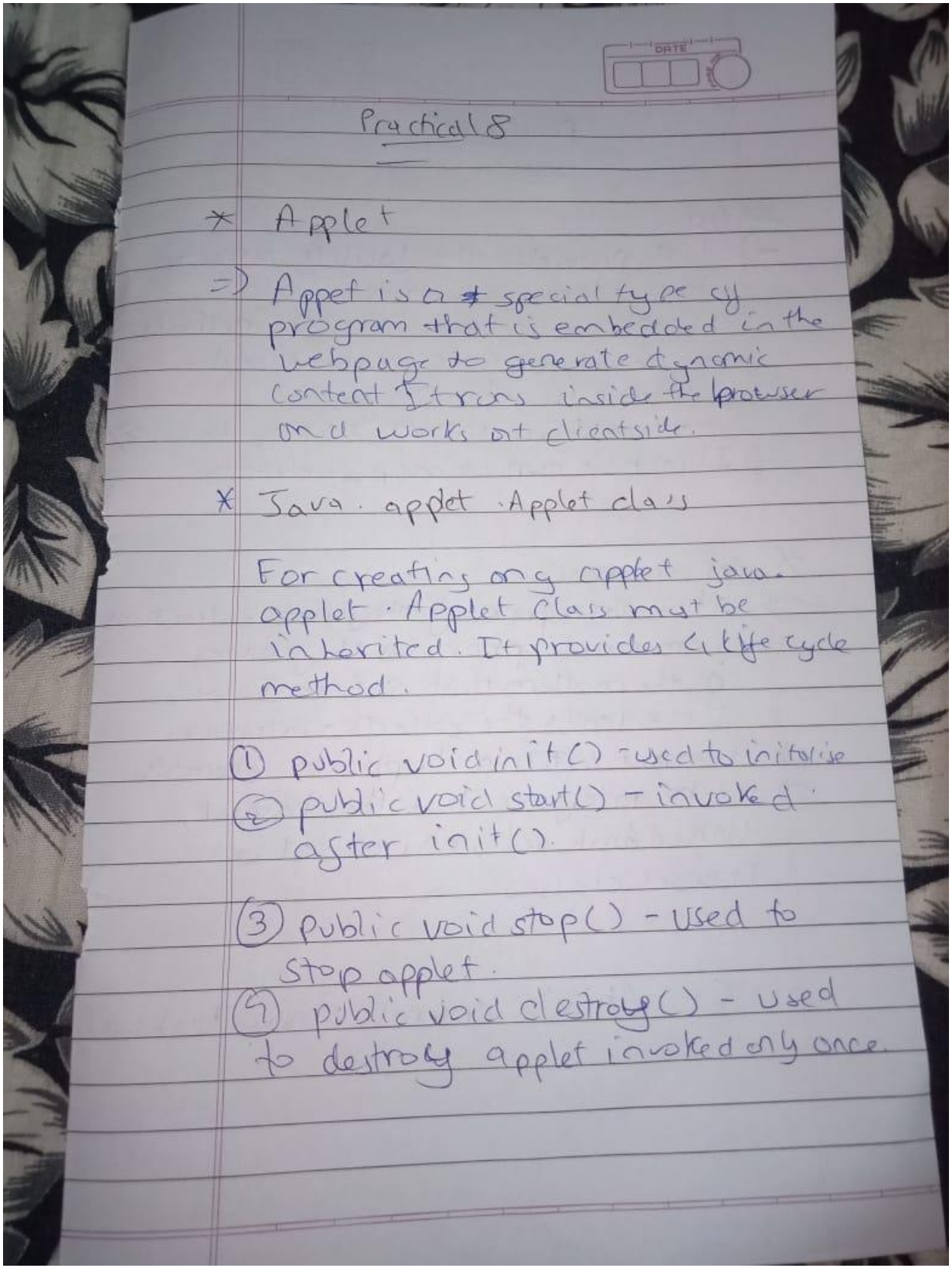
INPUT:

```
// write a a java program to demonstrate the use of throwable class bt ,illustrat
ing the object
//of that class extends throwable can be thrown and caught
import  java.lang.Exception;
class MyException extends Exception
{
    MyException(String msg)
    {
        super(msg);
    }
}
class TestException{
    public static void main(String args[]){
    int x=5,y=1000;
    try{
    float z=(float) x/ (float) y;
    if(z<0.01){
    throw new MyException ("Number is too small");
    }
    }
    catch(MyException e){
    System.out.println("Caught my Exception");
    System.out.println(e.getMessage());
    }
    finally{

    System.out.println("I am executed always");
    }
}
}
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac TestException.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java TestException
Caught my Exception
Number is too small
I am executed always
```

Practical 8



Practical 8

* Applet

=> Appet is a special type of program that is embedded in the webpage to generate dynamic content. It runs inside the browser and works at clientside.

* Java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 6 life cycle method.

(1) public void init() - used to initialise
(2) public void start() - invoked after init().

(3) Public void stop() - used to stop applet.
(4) public void destroy() - used to destroy applet invoked only once.

Practical 8

Aim:

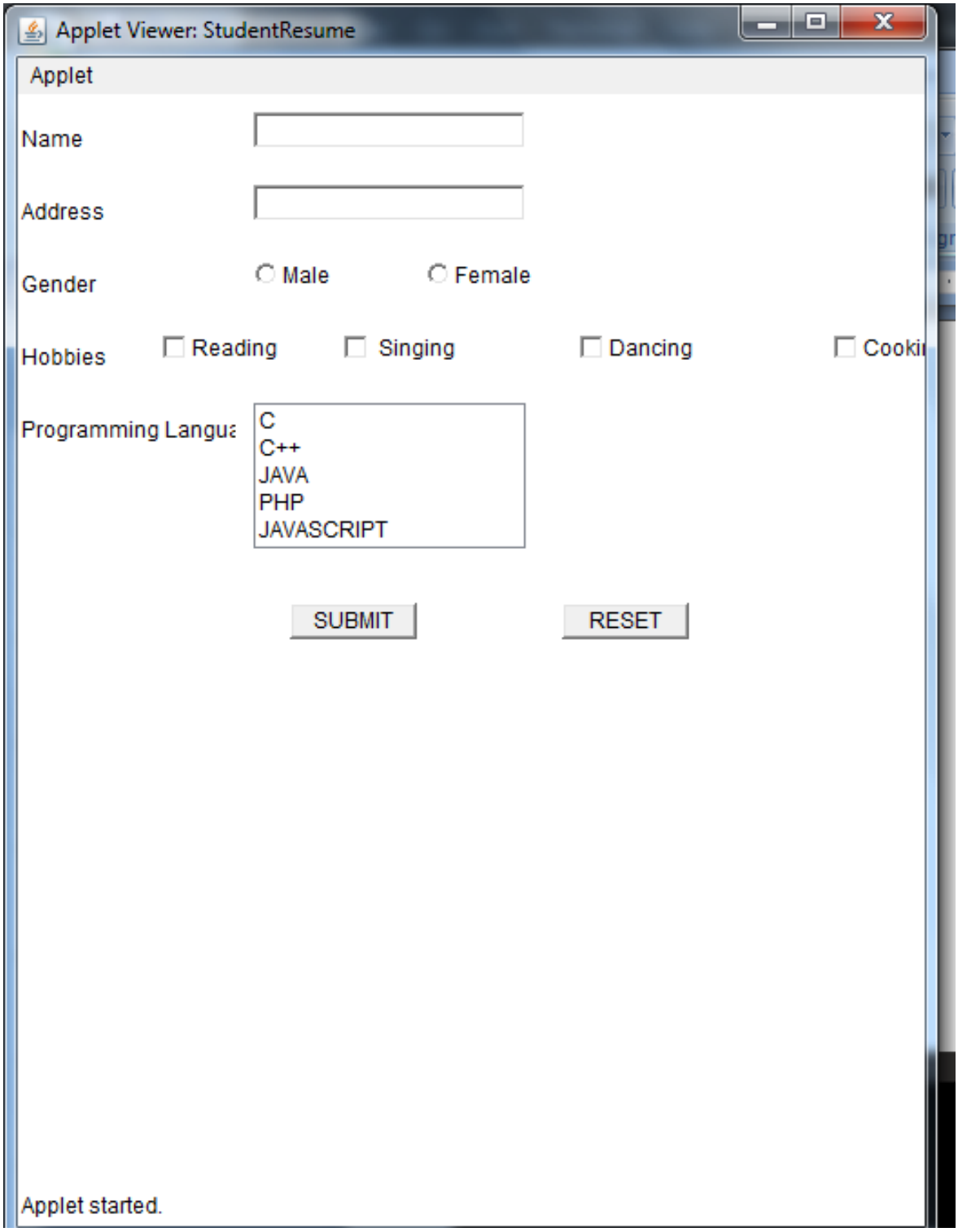
Using various swing components design Java application to accept a student's resume. (Design form)

INPUT:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/* <applet code="StudentResume" width=500 height=600>
</applet> */

public class StudentResume extends Applet implements ActionListener, ItemListener {
   TextField txtname, txtaddr;
   Button btnsubmit, btnreset;
   Checkbox c1, c2, c3, c4, m, f;
   CheckboxGroup cbg;
   List l1;
   TextArea txt;
   Label lblname, lbladdr, lblgender, lbllang, lblhobbies;

   public void init() {

      setLayout(null);
      lblname = new Label("Name");
      lblname.setBounds(0, 0, 50, 50);
      add(lblname);
      txtname = new TextField(20);
      txtname.setBounds(130, 10, 150, 20);
      add(txtname);

      lbladdr = new Label("Address");
      lbladdr.setBounds(0, 40, 70, 50);
      add(lbladdr);
      txtaddr = new TextField(20);
      txtaddr.setBounds(130, 50, 150, 20);
      add(txtaddr);

      lblgender = new Label("Gender");
      lblgender.setBounds(0, 80, 70, 50);
      add(lblgender);
      cbg = new CheckboxGroup();
      m = new Checkbox("Male", false, cbg);
      m.setBounds(130, 90, 75, 20);
      add(m);
      m.addItemListener(this);
      f = new Checkbox("Female", false, cbg);
      f.setBounds(225, 90, 75, 20);
      add(f);
      f.addItemListener(this);
```

```
        lblhobbies = new Label("Hobbies");
        lblhobbies.setBounds(0, 120, 70, 50);
        add(lblhobbies);
        c1 = new Checkbox("Reading");
        c1.setBounds(80, 130, 100, 20);
        add(c1);
        c1.addItemListener(this);
        c2 = new Checkbox(" Singing");
        c2.setBounds(180, 130, 130, 20);
        add(c2);
        c2.addItemListener(this);
        c3 = new Checkbox("Dancing");
        c3.setBounds(310, 130, 130, 20);
        add(c3);
        c3.addItemListener(this);
        c4 = new Checkbox("Cooking");
        c4.setBounds(450, 130, 130, 20);
        add(c4);
        c4.addItemListener(this);
        lbllang = new Label("Programming Languages Known");
        lbllang.setBounds(0, 160, 120, 50);
        add(lbllang);
        l1 = new List(4, true);
        l1.add("C");
        l1.add("C++");
        l1.add("JAVA");
        l1.add("PHP");
        l1.add("JAVASCRIPT");
        l1.setBounds(130, 170, 150, 80);
        add(l1);

        btnsubmit = new Button("SUBMIT");
        btnsubmit.setBounds(150, 280, 70, 20);
        add(btnsubmit);
        btnsubmit.addActionListener(this);
        btnreset = new Button("RESET");
        btnreset.setBounds(300, 280, 70, 20);
        add(btnreset);
        btnreset.addActionListener(this);
    }


    public void actionPerformed(ActionEvent e) {
        repaint();
    }
    public void itemStateChanged(ItemEvent e) {
        repaint();
    }
}
```

OUTPUT:


Applet Viewer: StudentResume
Applet
Name
Address
Gender
Male
Female
Hobbies
Reading
Singing
Dancing
Programming Langua
C
C++
JAVA
PHP
JAVASCRIPT
SUBMIT
RESET
Applet started.

Practical 8.2

Aim:

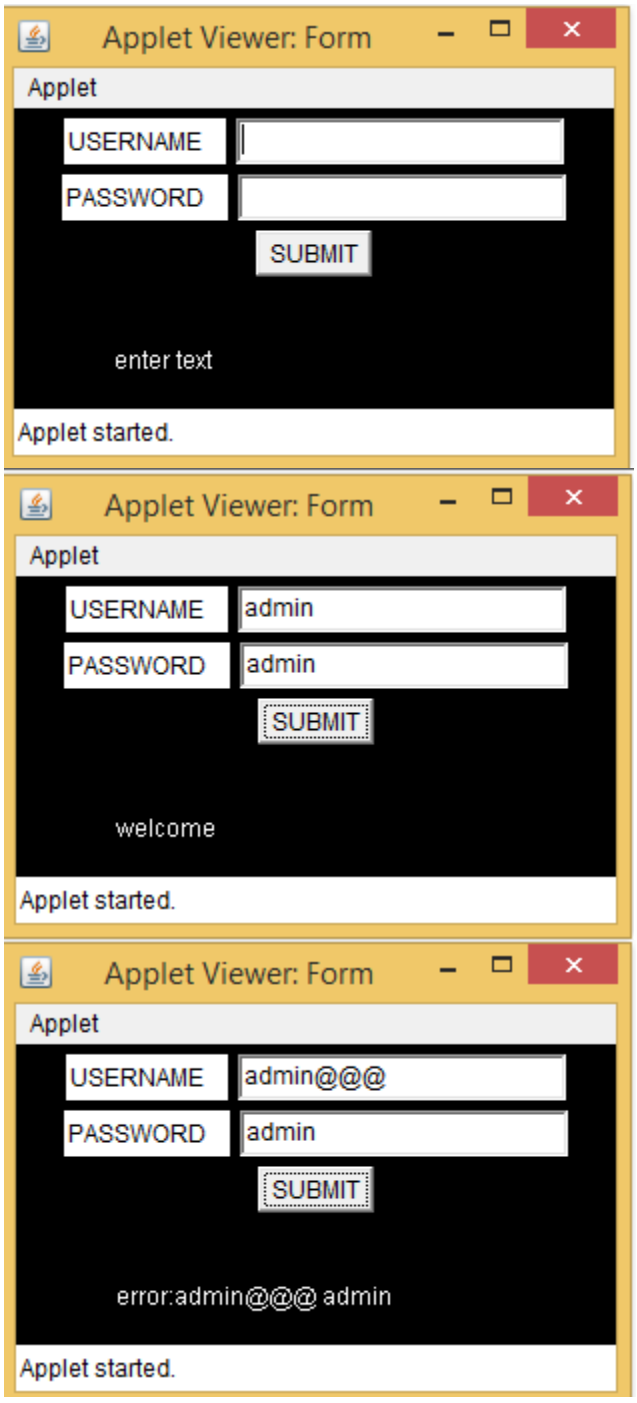
Write a java applet to Design a Login Form containing User Name and password. When user name and password matches with predefined values, a welcome message should be displayed otherwise error message should be displayed. If wrong passwords is entered, the application should end.

INPUT:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="Form" width=300 height=150>
</applet>
*/
public class Form extends Applet implements ActionListener {
   TextField n1, n2,re1;
   Label n11, n22;
   Button btn;
   public void init()
   {
      n1= new TextField(20);
      n2= new TextField(20);
      n11 = new Label("USERNAME",Label.LEFT);
      n22 = new Label("PASSWORD");
      btn = new Button("SUBMIT");
      add(n11);
      add(n1);
      add(n22);
      add(n2);
      add(btn);
      n1.addActionListener(this);
      n2.addActionListener(this);
                  btn.addActionListener(this);
                  setBackground(Color.black);
                  setForeground(Color.white);
   }
   public void actionPerformed(ActionEvent ae) {
                  repaint();
   }
         public void paint(Graphics g)
         {
                        String name=n1.getText();
                        String password=n2.getText();
                  if (name.trim().equals("admin") && password.trim().equals("admin")){
                        g.drawString("welcome" , 50, 130);
                  }
                  else if(name.trim().equals("") || password.trim().equals("")) {
                        g.drawString("enter text" , 50, 130);
```

```
                }
            else if((name.equals("admin")!=true && name.equals("admin")!=true)){
                g.drawString("error:"+n1.getText()+" "+n2.getText() , 50, 130);
            }
        }
}
```

OUTPUT:



Practical 8.3

Aim:

A bank wants to keep an enquiry page for inquiring about home loan. The rates of these loans are 7.8, 7.9 AND 9.8 for various span of years (5/10/15) respectively. The user types the loan amount & selects the number of years for repayment of loan (5/10/15) from combo box. In return, the user gets information of the rate of loan, monthly installment for the specified number of years.

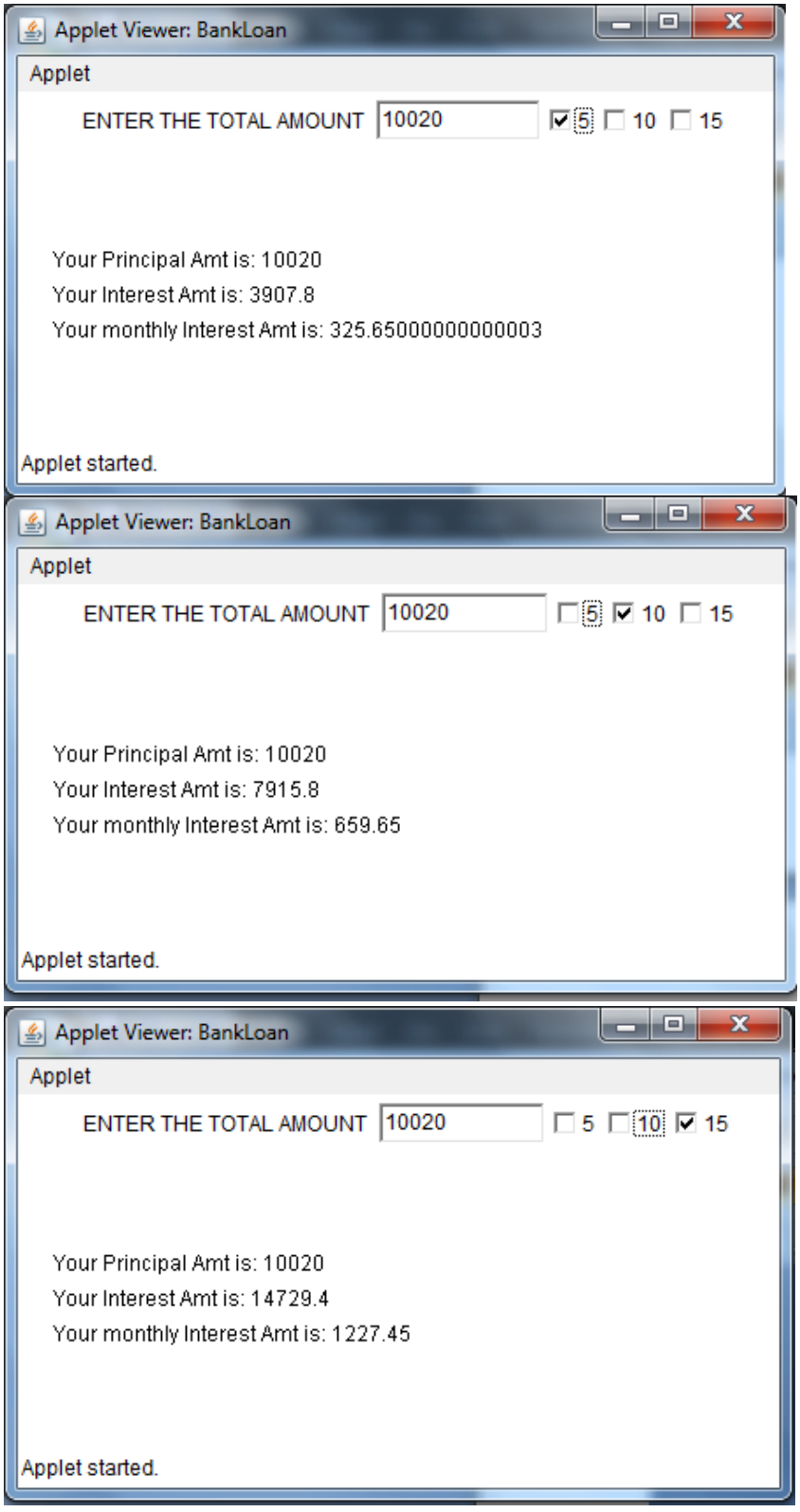
INPUT:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.applet.Applet;
import java.awt.Choice;
/*
<applet code="BankLoan" width=200 height=200>
</applet>
*/
public class BankLoan extends Applet implements ActionListener, ItemListener {
	TextField txtamt;
	Checkbox five, ten, fifteen;
	public void init() {
		Label lblamt = new Label("ENTER THE TOTAL AMOUNT", Label.RIGHT);
		txtamt = new TextField(10);
		five = new Checkbox("5");
		ten = new Checkbox("10");
		fifteen = new Checkbox("15");
		add(lblamt);
		add(txtamt);
		add(five);
		add(ten);
		add(fifteen);
		txtamt.addActionListener(this);
		five.addItemListener(this);
		ten.addItemListener(this);
		fifteen.addItemListener(this);
	}
	public void actionPerformed(ActionEvent ae) {
		repaint();
	}
	public void itemStateChanged(ItemEvent ie) {
		repaint();
	}
	public void paint(Graphics g) {
```

```
                double interest = 0.0;
                g.drawString("Your Principal Amt is: " + txtamt.getText(), 20, 100);
                int amt = Integer.parseInt(txtamt.getText());
                if(five.getState() == true) {
                        interest = (amt * 5 * 7.8) / 100;
                }
                if(ten.getState() == true) {
                        interest = (amt * 10 * 7.9) / 100;
                }
                if(fifteen.getState() == true) {
                        interest = (amt * 15 * 9.8) / 100;
                }
                g.drawString("Your Interest Amt is: " + interest, 20, 120);
                g.drawString("Your monthly Interest Amt is: " + interest/12, 20, 140);
        }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac BankLoan.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java BankLoan
```


Applet Viewer: BankLoan
Applet
ENTER THE TOTAL AMOUNT
10020
5
10
15
Your Principal Amt is: 10020
Your Interest Amt is: 3907.8
Your monthly Interest Amt is: 325.65000000000003
Applet started.
Applet Viewer: BankLoan
Applet
ENTER THE TOTAL AMOUNT
10020
5
10
15
Your Principal Amt is: 10020
Your Interest Amt is: 7915.8
Your monthly Interest Amt is: 659.65
Applet started.
Applet Viewer: BankLoan
Applet
ENTER THE TOTAL AMOUNT
10020
5
10
15
Your Principal Amt is: 10020
Your Interest Amt is: 14729.4
Your monthly Interest Amt is: 1227.45
Applet started.

Practical 8.4

Aim:

Design ticket enquiry form for a theater. Select location (Mumbai/Pune) using radio button. When user clicks a location, it fills a combo box with names of theaters in that city. When user selects the name of the theatre, the list of films currently shown and their show timings should be displayed.

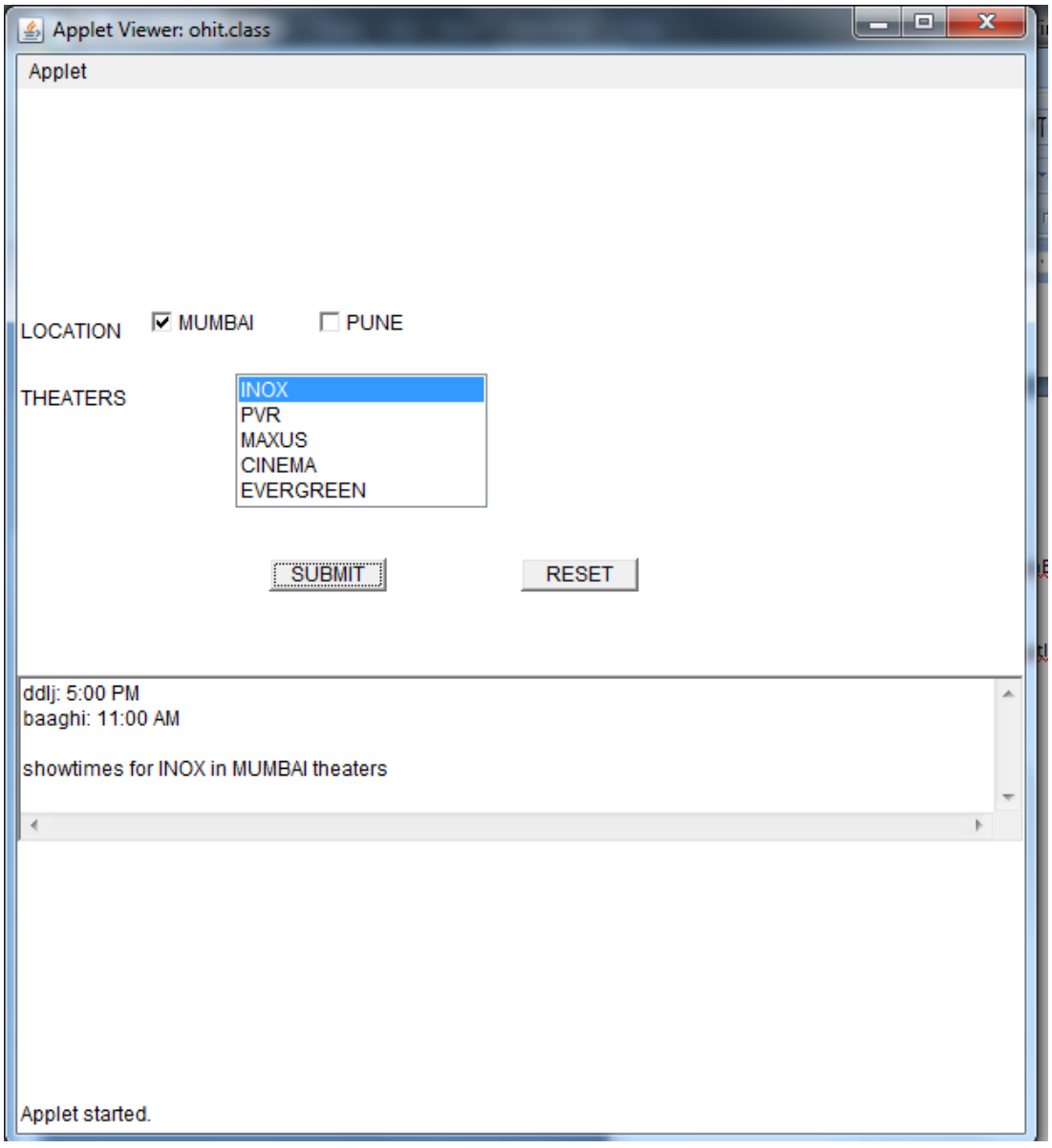
INPUT:

```
import java.util.Random;
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/* <applet code="ohit.class" width=600 height=600>
</applet> */

public class ohit extends Applet implements ActionListener, ItemListener {

   TextField t3, t4, t5, t6, t7;
   Button b1, b2;
   Checkbox c1, c2, c3, c4, m, f;
   CheckboxGroup cbg;
   List l1;
   Label l2, l3, l4, l5;
   TextArea tx1;

   public void init() {
      setLayout(null);
      l5 = new Label("LOCATION");
      l5.setBounds(0, 120, 70, 50);
      add(l5);
      c1 = new Checkbox("MUMBAI");
      c1.setBounds(80, 130, 100, 20);
      add(c1);
      c1.addItemListener(this);
      c2 = new Checkbox("PUNE");
      c2.setBounds(180, 130, 130, 20);
      add(c2);
      c2.addItemListener(this);
      l5 = new Label("THEATERS");
      l5.setBounds(0, 160, 120, 50);
      add(l5);
      l1 = new List(4, true);
      l1.add("INOX");
      l1.add("PVR");
```

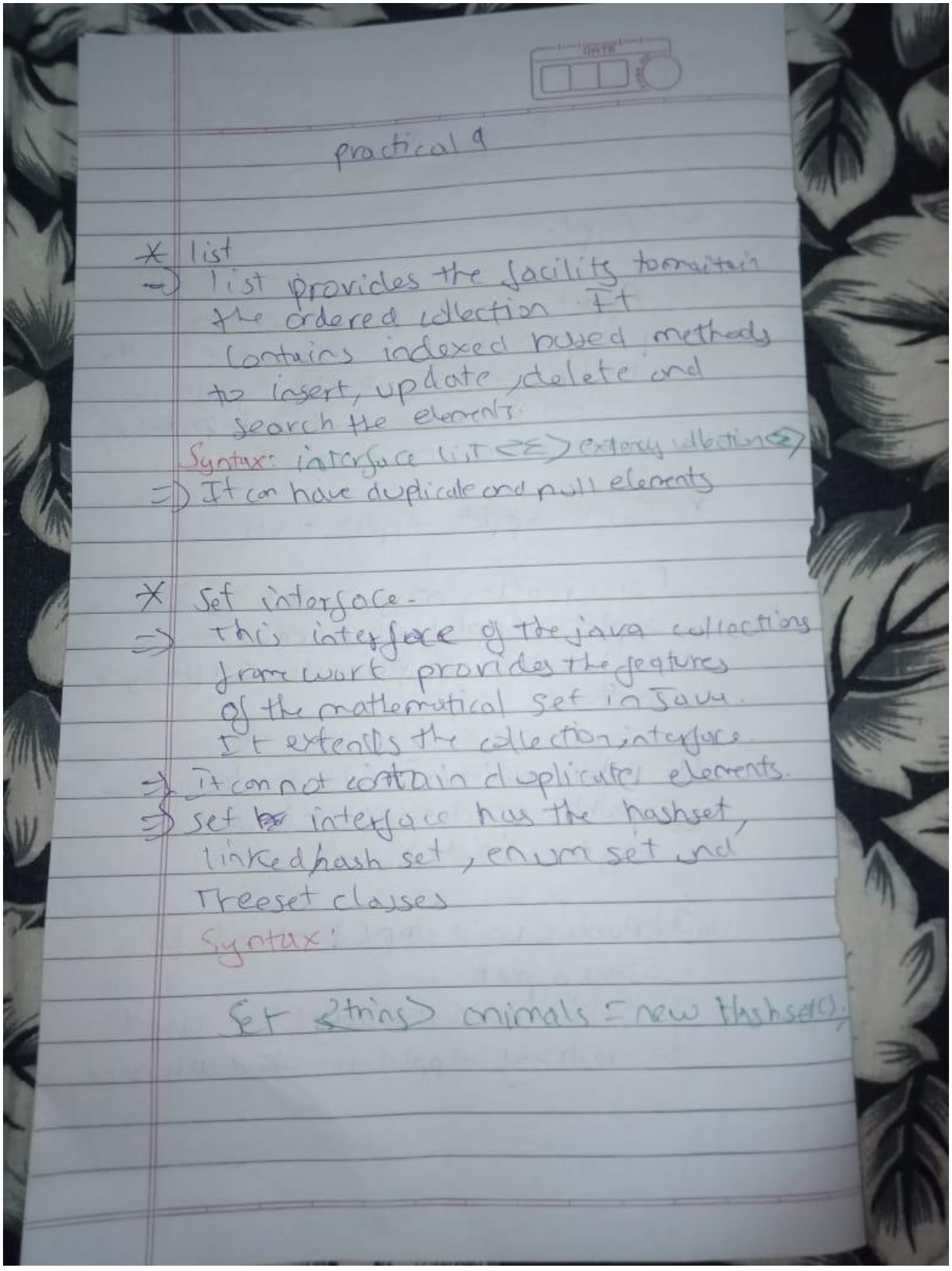
```
    l1.add("MAXUS");
    l1.add("CINEMA");
    l1.add("EVERGREEN");
    l1.setBounds(130, 170, 150, 80);
    add(l1);
    b1 = new Button("SUBMIT");
    b1.setBounds(150, 280, 70, 20);
    add(b1);
    b1.addActionListener(this);
    b2 = new Button("RESET");
    b2.setBounds(300, 280, 70, 20);
    add(b2);
    b2.addActionListener(this);
    tx1 = new TextArea("", 10, 20, TextArea.SCROLLBARS_BOTH);
    tx1.setBounds(0, 350, 600, 100);
    add(tx1);

  }
  public void paint(Graphics g){

  }
  String selections[];
  public void actionPerformed(ActionEvent e) {
    Random rn = new Random();
    int range = 12 - 1 + 1;
    int randomNum =  rn.nextInt(range) + 1;
    repaint();
    if (e.getSource() == b1) {
      //SUBMIT
      selections=l1.getSelectedItems();
      tx1.insert("showtimes for "+selections[0], 0);
      tx1.insert( "\n", 0);
      tx1.insert("baaghi: "+ (rn.nextInt(range) + 1)+":00 AM \n",0);
      tx1.insert("ddlj: "+ (rn.nextInt(range) + 1)+":00 PM \n",0);
    }

    String msg = new String("");
    if (e.getSource() == b2) {
      //RESET
      tx1.setText(msg);

    }
```

```
    }
    public void itemStateChanged(ItemEvent e) {
        //checkbox
        tx1.setText("");
        tx1.insert(" in "+((Checkbox) e.getItemSelectable()).getLabel() + " theaters", 0);

    }
}
```

output



Practical 9

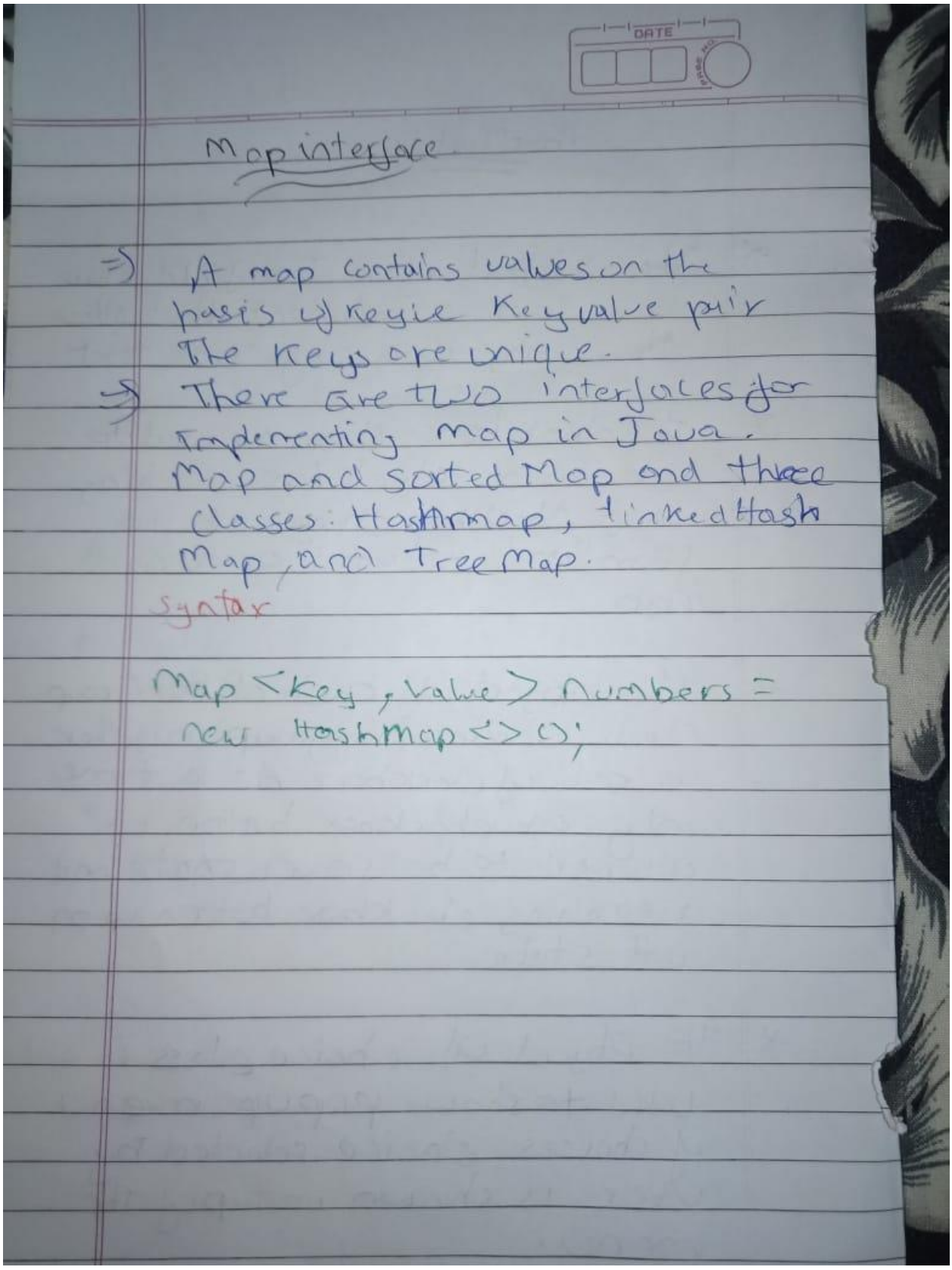


practical 9

* list
=> list provides the facility to maintain the ordered collection. It contains indexed based methods to insert, update, delete and search the elements.
Syntax: interface List<E> extends Collection<E>
=> It can have duplicate and null elements

* Set interface.
=> This interface of the java collections framework provides the features of the mathematical set in Java. It extends the collection interface.
=> It cannot contain duplicate elements.
=> Set interface has the hashset, linkedhash set, enum set and Treeset classes
Syntax:

Set<String> animals = new HashSet();


DATE
Map interface
A map contains values on the basis of key i.e Key value pair The Keys are unique.
There are two interfaces for Implementing map in Java. Map and sorted Map and three classes: HashMap, LinkedHash Map, and TreeMap.
Syntax
Map<Key, Value> Numbers = new HashMap<>();

Practical 9.1

Aim:

Write a Java Program to demonstrate List interface and its methods.

INPUT:

```
import java.util.*;
public class CollectionDemo {
   public static void main(String[] args) {
      List a1 = new ArrayList();
      a1.add("Anand");
      a1.add("Shivam");
      a1.add("Akash");
      a1.add(2, "Nikhil");
      a1.set(1, "Sophia");
      System.out.println("ArrayList Elements");
      System.out.print("\t" + a1);
      System.out.println("\nValue at 1st position is:  " + a1.get(1));

      List l1 = new LinkedList();
      l1.add("Apple");
      l1.add("Banana");
      l1.add("Papaya");
      l1.add("Orange");
      l1.add("Apple");
      l1.add("Grapes");
      l1.addAll(3, a1);
      System.out.println();
      System.out.println("LinkedList Elements");
      System.out.print("\t" + l1);
      System.out.println("\nIndex of Banana is:  " + l1.indexOf("Banana"));
      System.out.println("Last index of Apple is:  " + l1.lastIndexOf("Apple"));
      l1.remove(8);
      System.out.println("Removed element from 8th position");
      System.out.print("\nNew Linked List is...\t" + l1);

      System.out.println("\nSublist is..." + l1.subList(4, 7));
   }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java CollectionDemo
ArrayList Elements
      [Anand, Sophia, Nikhil, Akash]
Value at 1st position is:  Sophia
```

LinkedList Elements
[Apple, Banana, Papaya, Anand, Sophia, Nikhil, Akash, Orange, Apple, Grapes]
Index of Banana is: 1
Last index of Apple is: 8
Removed element from 8th position

New Linked List is... [Apple, Banana, Papaya, Anand, Sophia, Nikhil, Akash, Orange, Grapes]
Sublist is...[Sophia, Nikhil, Akash]

Practical 9.2

Aim:

Write a Java Program to demonstrate Set interface and its methods.

INPUT:

```
import java.util.*;
public class SetDemo {

   public static void main(String args[]) {
      int count[] = {34, 22,10,60,30,22};
      Set<Integer> set = new HashSet<Integer>();
      try {
        for(int i = 0; i < 5; i++) {
          set.add(count[i]);
        }
        System.out.println(set);

        TreeSet   sortedSet = new TreeSet<Integer>(set);
        System.out.println("The sorted list is:");
        System.out.println(sortedSet);
        System.out.println("The size of the set is:  "+sortedSet.size());
        System.out.println("The First element of the set is: "+sortedSet.first());
        System.out.println("The last element of the set is: "+ sortedSet.last());
      }
      catch(Exception e) {}
   }
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac SetDemo.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java SetDemo
[34, 22, 10, 60, 30]
The sorted list is:
[10, 22, 30, 34, 60]
The size of the set is:  5
The First element of the set is: 10
The last element of the set is: 60
```

Practical 9.3

Aim:

Write a Java Program to demonstrate Map interface and its methods.

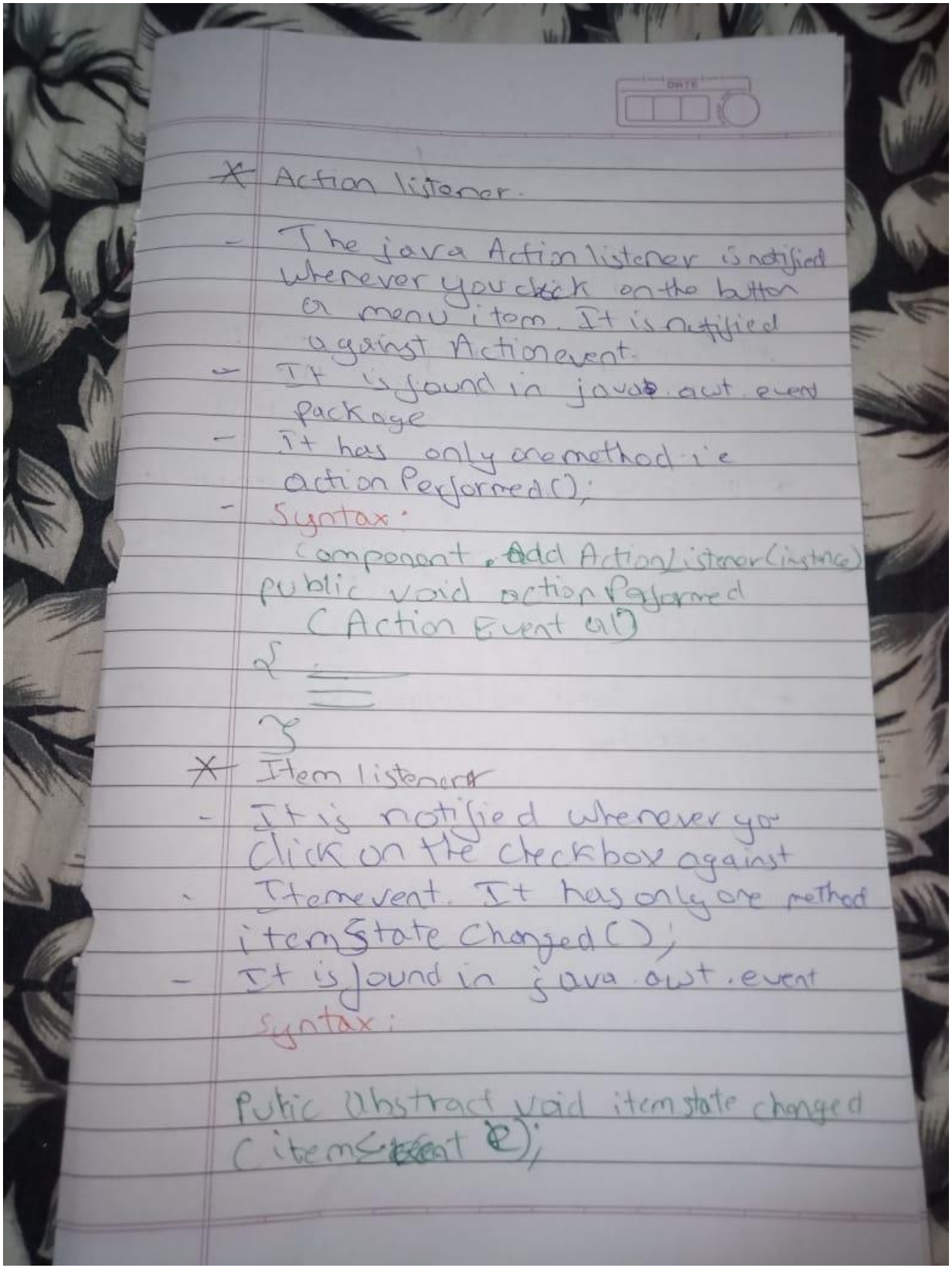
INPUT:

```
import java.util.*;
public class MapDemo {
   public static void main(String[] args) {
      Map<Integer, String> hm1 = new HashMap<>();
      hm1.put(1, "This");
      hm1.put(2, "is");
      hm1.put(3, "Map");
      hm1.put(4,"Demo") ;
      System.out.println(hm1);
      String rmd=hm1.remove(new Integer(2));
      System.out.println("Map after removing element at key 2 "+hm1);
      hm1.put(2,"is");
      System.out.println("Map after adding 'is' back at key 2 "+hm1);
      System.out.println("Size of the map is "+ hm1.size());
      System.out.println("Element at key 4 is "+hm1.get(4));
      System.out.println("Is Map empty  "+hm1.isEmpty());
      hm1.clear();
      System.out.println("Clearing all elements and value in map "+hm1);
      System.out.println("Is Map empty  "+hm1.isEmpty());
   }
   }
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MapDemo
{1=This, 2=is, 3=Map, 4=Demo}
Map after removing element at key 2 {1=This, 3=Map, 4=Demo}
Map after adding 'is' back at key 2 {1=This, 2=is, 3=Map, 4=Demo}
Size of the map is 4
Element at key 4 is Demo
Is Map empty  false
Clearing all elements and value in map {}
Is Map empty  true
```

Practical 10



* Action listener.

- The java Action listener is notified whenever you click on the button or menu item. It is notified against Actionevent.
- It is found in java.awt.event package
- It has only one method i.e actionPerformed();
- Syntax:

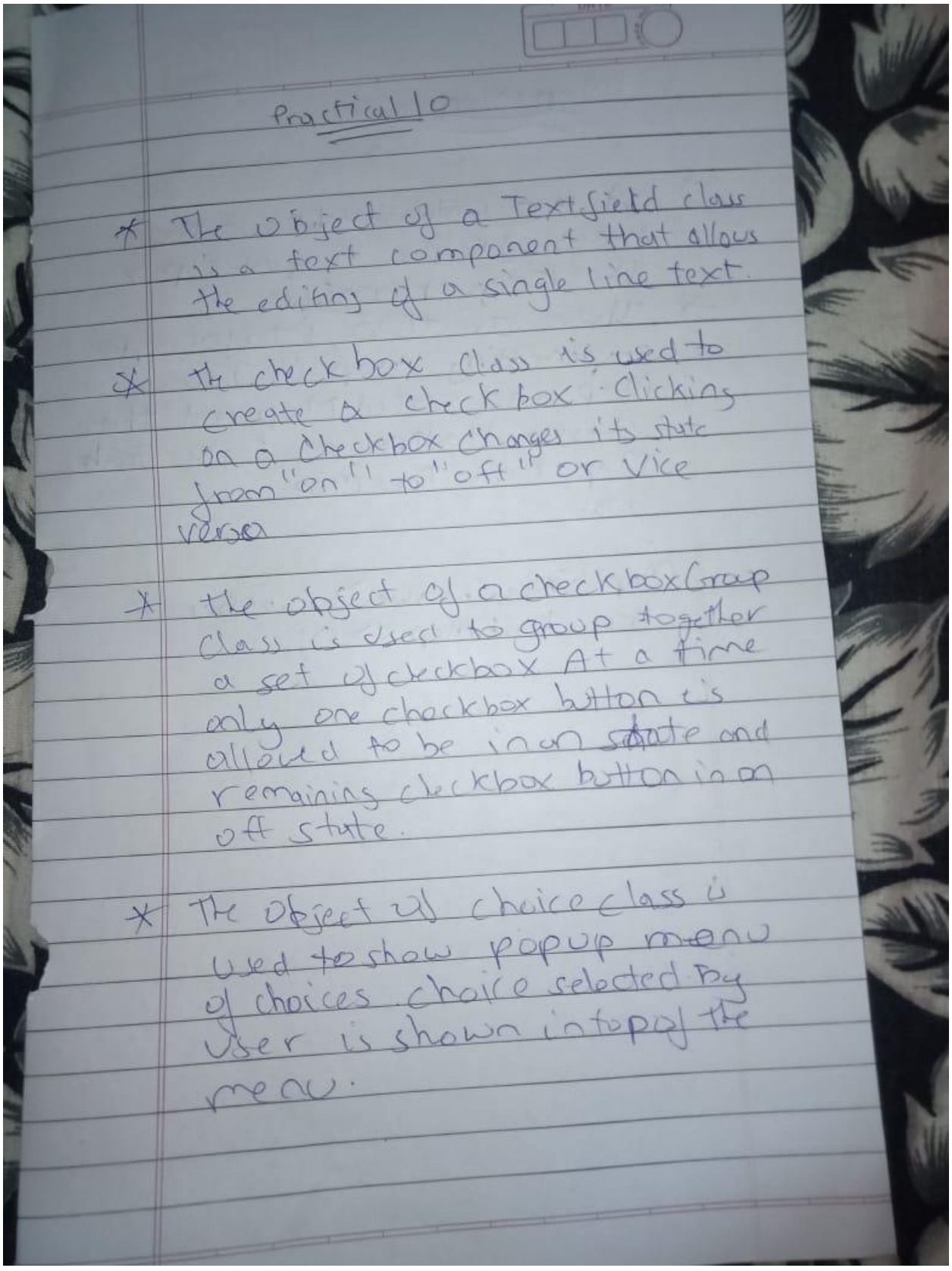
Component.AddActionListener(instance)
public void actionPerformed
(ActionEvent e)
{
}

* Item listener

- It is notified whenever you click on the checkbox against Itemevent. It has only one method itemStateChanged();
- It is found in java.awt.event

Syntax:

Putic Abstract void itemstate changed (ItemEvent e);



Practical 10

* The object of a Textfield class is a text component that allows the editing of a single line text.

* The checkbox class is used to create a checkbox. Clicking on a checkbox changes its state from "on" to "off" or vice versa

* The object of a checkboxGroup class is used to group together a set of checkbox. At a time only one checkbox button is allowed to be in on state and remaining checkbox button in on off state.

* The object of choice class is used to show popup menu of choices. Choice selected by user is shown in top of the menu.

Practical 10.2

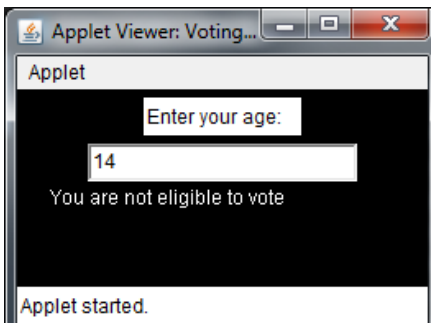
Aim:

Write a program to check eligibility for voting using Java Applet.

INPUT:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="VotingEligibility" width=250 height=120>
</applet>
*/
public class VotingEligibility extends Applet implements ActionListener {
   TextField age;
   Label lblage;
   public void init() {
      lblage = new Label("Enter your age:");
      age = new TextField(20);
      add(lblage);
      add(age);
      setBackground(Color.black);
      setForeground(Color.white);
      age.addActionListener(this);
   }
   public void actionPerformed(ActionEvent e) {
      repaint();
   }
   public void paint(Graphics g) {
      int numage;
      numage = Integer.parseInt(age.getText());
      if (numage >= 18) {
         g.drawString("You are eligible to vote", 20, 70);
      } else {
         g.drawString("You are not eligible to vote", 20, 70);
      }
   }
}
```

Output:



Practical 10.3

Aim:

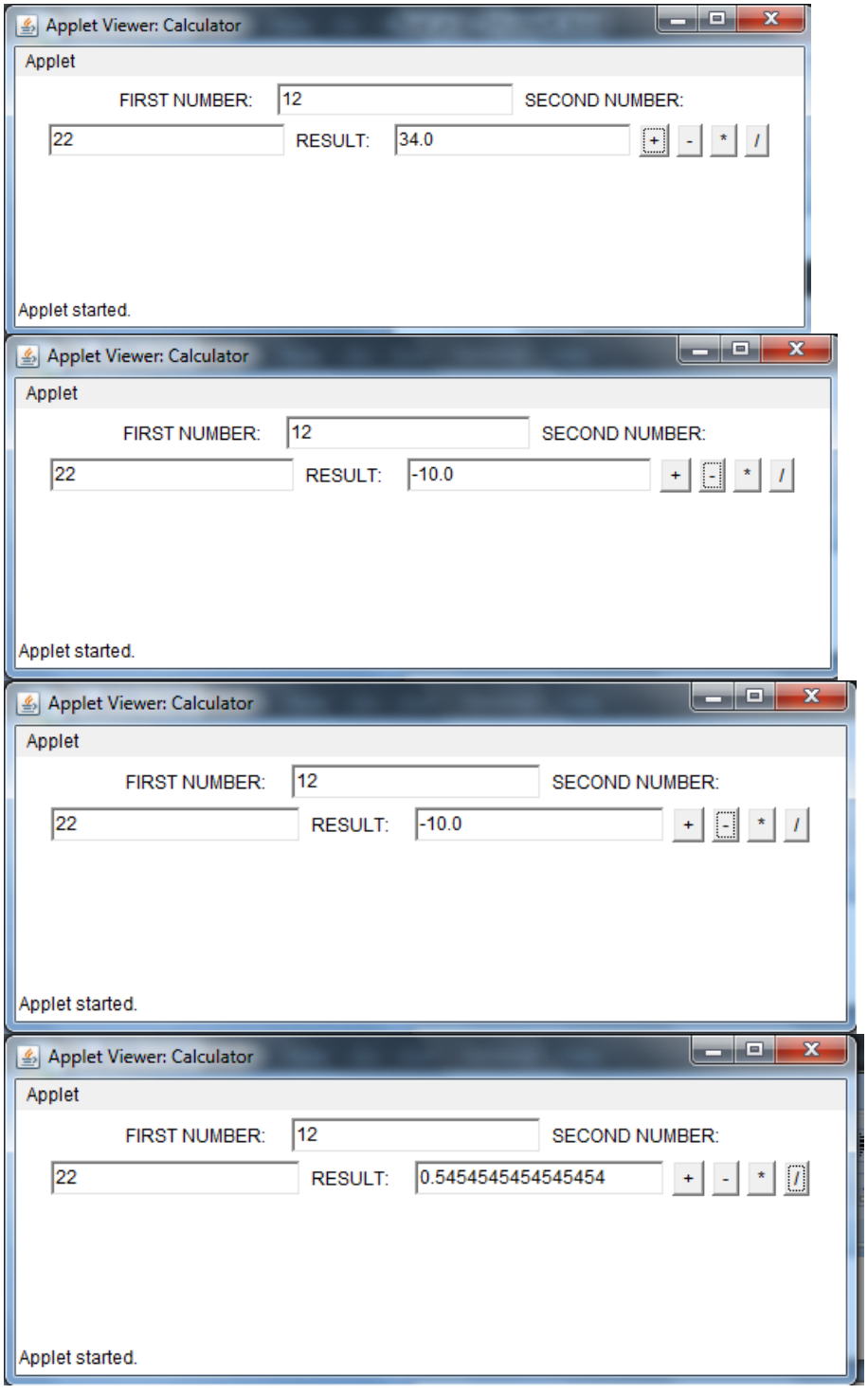
Write a Java program to perform arithmetic operations using Java Applet.

INPUT:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="Calculator" width=250 height=150>
</applet>
*/
public class Calculator extends Applet implements ActionListener {
   double num1, num2, res;
   TextField n1, n2, re;
   Label n11, n22, re1;
   Button sum, sub, mul, div;
   public void init() {
      n11 = new Label("FIRST NUMBER:");
      n22 = new Label("SECOND NUMBER:");
      re1 = new Label("RESULT:");
      n1 = new TextField(20);
      n2 = new TextField(20);
      re = new TextField(20);
      sum = new Button("+");
      sub = new Button("-");
      mul = new Button("*");
      div = new Button("/");
      add(n11);
      add(n1);
      add(n22);
      add(n2);
      add(re1);
      add(re);
      add(sum);
      add(sub);
      add(mul);
      add(div);
      sum.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            num1 = Double.parseDouble(n1.getText());
            num2 = Double.parseDouble(n2.getText());
            res = num1 + num2;
```

```
            re.setText(Double.toString(res));
        }
    });

    sub.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            num1 = Double.parseDouble(n1.getText());
            num2 = Double.parseDouble(n2.getText());
            res = num1 - num2;
            re.setText(Double.toString(res));
        }
    });
    mul.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            num1 = Double.parseDouble(n1.getText());
            num2 = Double.parseDouble(n2.getText());
            res = num1 * num2;
            re.setText(Double.toString(res));
        }
    });
    div.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            num1 = Double.parseDouble(n1.getText());
            num2 = Double.parseDouble(n2.getText());
            if (num2 == 0) {
                re.setText("ERROR");
            } else {
                res = num1 / num2;
                re.setText(Double.toString(res));
            }
        }
    });
  }
  public void actionPerformed(ActionEvent e) {}
}
```

Output:


Applet Viewer: Calculator
Applet
FIRST NUMBER: 12 SECOND NUMBER:
22 RESULT: 34.0 + - * /
Applet started.
Applet Viewer: Calculator
Applet
FIRST NUMBER: 12 SECOND NUMBER:
22 RESULT: -10.0 + - * /
Applet started.
Applet Viewer: Calculator
Applet
FIRST NUMBER: 12 SECOND NUMBER:
22 RESULT: -10.0 + - * /
Applet started.
Applet Viewer: Calculator
Applet
FIRST NUMBER: 12 SECOND NUMBER:
22 RESULT: 0.5454545454545454 + - * /
Applet started.

Practical 10.4

Aim:

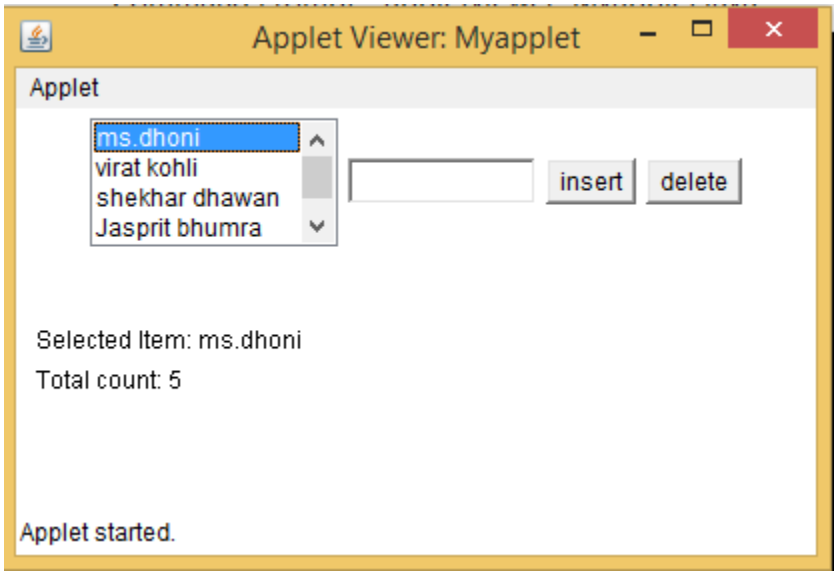
Write an applet program to Create a form containing a listbox with names of Indian cricket players. The program
should perform following operations –
insertion of new elements in to the list,
deletion of selected element from the list,
show the currently selected element,
show total count

INPUT:

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

/* <applet code="Myapplet" width=400 height=200>
</applet>*/

public class Myapplet extends Applet implements ActionListener
{
Button b,b1;
List player;
TextField t1;
String a;
        public void init()
        {
                player= new List(4);
                b=new Button("insert");
                t1 = new TextField(10);
                b.addActionListener(new insrt());
                b1=new Button("delete");
                b1.addActionListener(new insrt());
                player.add("ms.dhoni");
                player.add("virat kohli");
                player.add("shekhar dhawan");
                player.add("bumra");
                player.add("Rohit sharma");
                add(player);
                add(t1);
                add(b);
                add(b1);

                // add listner
                player.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae)
        {
                // user presses enter key
```

```
                repaint();
        }
        public void paint(Graphics g)
        {
                g.drawString("Selected Item: "+ player.getSelectedItem(), 10, 120);
                g.drawString("Total count: "+ player.getItemCount(), 10, 140);
        }
        class insrt implements ActionListener
        {
                public void actionPerformed(ActionEvent e)
                {
                        if(e.getSource()==b)
                        {
                                player.add(t1.getText());
                        }
                        if(e.getSource()==b1)
                        {
                                a=player.getSelectedItem();
                                player.remove(a);
                        }
                }
        }
}
```

OUTPUT:



Practical 10.5

Aim:

Write an applet program that accepts Principle Amount, No. of Years & Rate of Interest from 3 text fields, when you click “Calculate Interest” button, the data is sent to a function that returns the simple interest. When you click on “Final Amount” button, the final amount by adding principle amount and interest should be displayed.

INPUT:

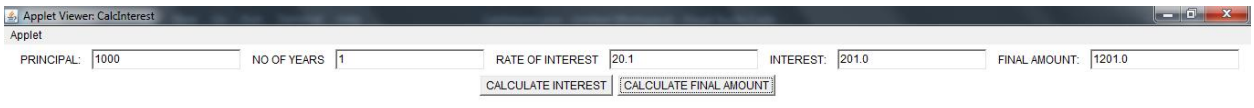
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="CalcInterest" width=250 height=150>
</applet>
*/
public class CalcInterest extends Applet implements ActionListener{
Double fin,inter,prin,rate,year;
String str,str1="";
Label pri,nof,roi,f,i;
TextField p,n,r,f1,i1;
Button calc_i,calc_f;
public void init(){
pri=new Label("PRINCIPAL:",Label.CENTER);
nof=new Label("NO OF YEARS",Label.CENTER);
roi=new Label("RATE OF INTEREST",Label.CENTER);
f=new Label("FINAL AMOUNT:",Label.CENTER);
i=new Label("INTEREST:",Label.CENTER);
i1=new TextField(20);
f1=new TextField(20);
p=new TextField(20);
n=new TextField(20);
r=new TextField(20);
calc_i=new Button("CALCULATE INTEREST");
calc_f=new Button("CALCULATE FINAL AMOUNT");
add(pri);
add(p);
add(nof);
add(n);
add(roi);
add(r);
add(i);
add(i1);
add(f);
```

```
add(f1);
add(calc_i);
add(calc_f);
calc_i.addActionListener(new ActionListener(){public void actionPerformed(ActionEvent e){
prin=Double.parseDouble(p.getText());
rate=Double.parseDouble(r.getText());
year=Double.parseDouble(n.getText());
inter=(prin*rate*year)/100.0;
str=Double.toString(inter);
i1.setText(str);
}});
calc_f.addActionListener(new ActionListener(){public void actionPerformed(ActionEvent e){
prin=Double.parseDouble(p.getText());
rate=Double.parseDouble(r.getText());
year=Double.parseDouble(n.getText());
inter=(prin*rate*year)/100.0;
fin=prin+inter;
str1=Double.toString(fin);
f1.setText(str1);
}});
f1.addActionListener(this);
i1.addActionListener(this);
}
public void actionPerformed(ActionEvent e){
}
}
```

Output:

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac CalcInterest.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>appletviewer CalcInterest.java



Practical 10.6

Aim:

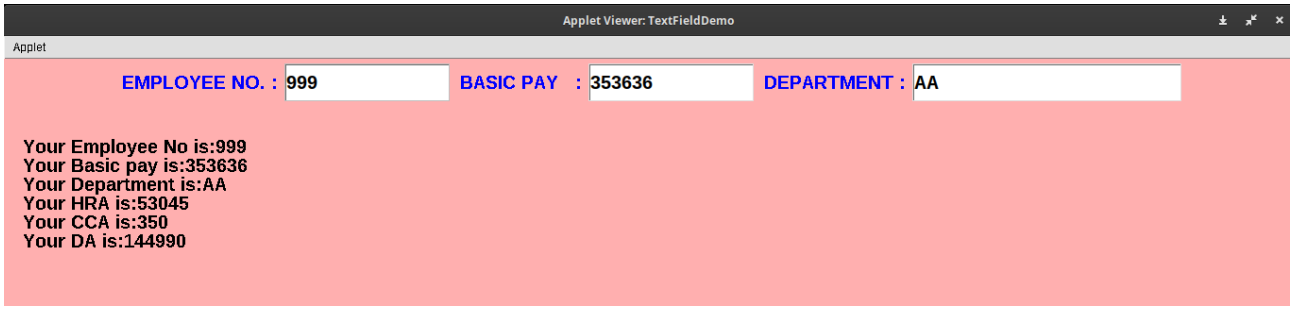
Write a program to accept the elements of the classes i.eEmp_no,Basic_pay,Dept.
These elements are passed through the applet using TextField. The salary slips along with calculation of DA,HRA and CCA should be printed on the applet. Where
DA=81% of Basic Salary if Basic Salary<5000
DA=51% of Basic Salary if Basic Salaryis in therange of 5000 to 7000
DA=41% of Basic Salary if Basic Salary>7000
HRA=15% of Basic Salary
CCA=350/-

INPUT:

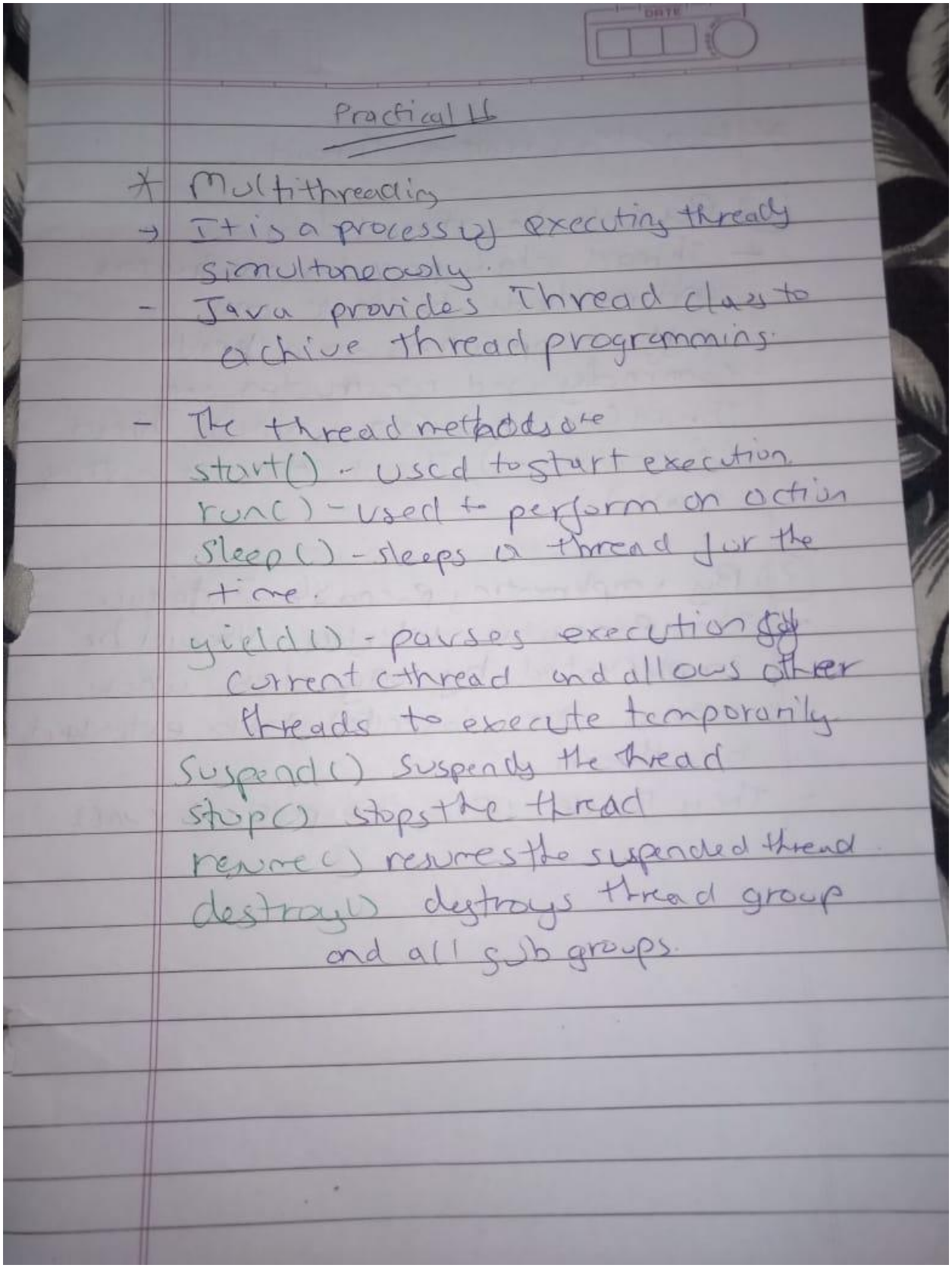
```
// Recognize Button objects.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="TextFieldDemo" width=250 height=150>
</applet>
*/
public class  TextFieldDemo extends Applet implements ActionListener
{
TextField Emp_no,Basic_pay,Dept;
public void init(){
Label eno=new Label("EMPLOYEE NO. :",Label.RIGHT);
Label bp=new Label("BASIC PAY   :",Label.RIGHT);
Label dpt=new Label("DEPARTMENT :",Label.RIGHT);

Emp_no=new TextField(15);
Basic_pay=new TextField(15);
Dept=new TextField(25);

add(eno);//label
add(Emp_no);//textfield
add(bp);
add(Basic_pay);
add(dpt);
add(Dept);

// register the receive and action event
Basic_pay.addActionListener(this);
Emp_no.addActionListener(this);
Dept.addActionListener(this);
```

```
setBackground(Color.pink);
setForeground(Color.blue);
Font f=new Font("Tahoma",Font.BOLD,20);
setFont(f);
}
public void actionPerformed(ActionEvent ae)
{
//USER PRESSES ENTER KEY
repaint();
}
public void paint(Graphics g){
g.drawString("Your Employee No is:"+Emp_no.getText(),20,100);
g.drawString("Your Basic pay is:"+Basic_pay.getText(),20,120);
g.drawString("Your Department is:"+Dept.getText(),20,140);
int BS=Integer.parseInt(Basic_pay.getText());
int DA,HRA,CCA;
CCA=350;
HRA=(15*BS)/100;
if(BS<5000)
DA=(81*BS)/100;
else if(BS>5000 && BS<7000)
DA=(51*BS)/100;
else if(BS>7000)
DA=(41*BS)/100;
else
DA=0;
g.drawString("Your HRA is:"+HRA, 20,160);
g.drawString("Your CCA is:"+CCA, 20,180);
g.drawString("Your DA is:"+DA, 20, 200);
}
}
```

OUTPUT:



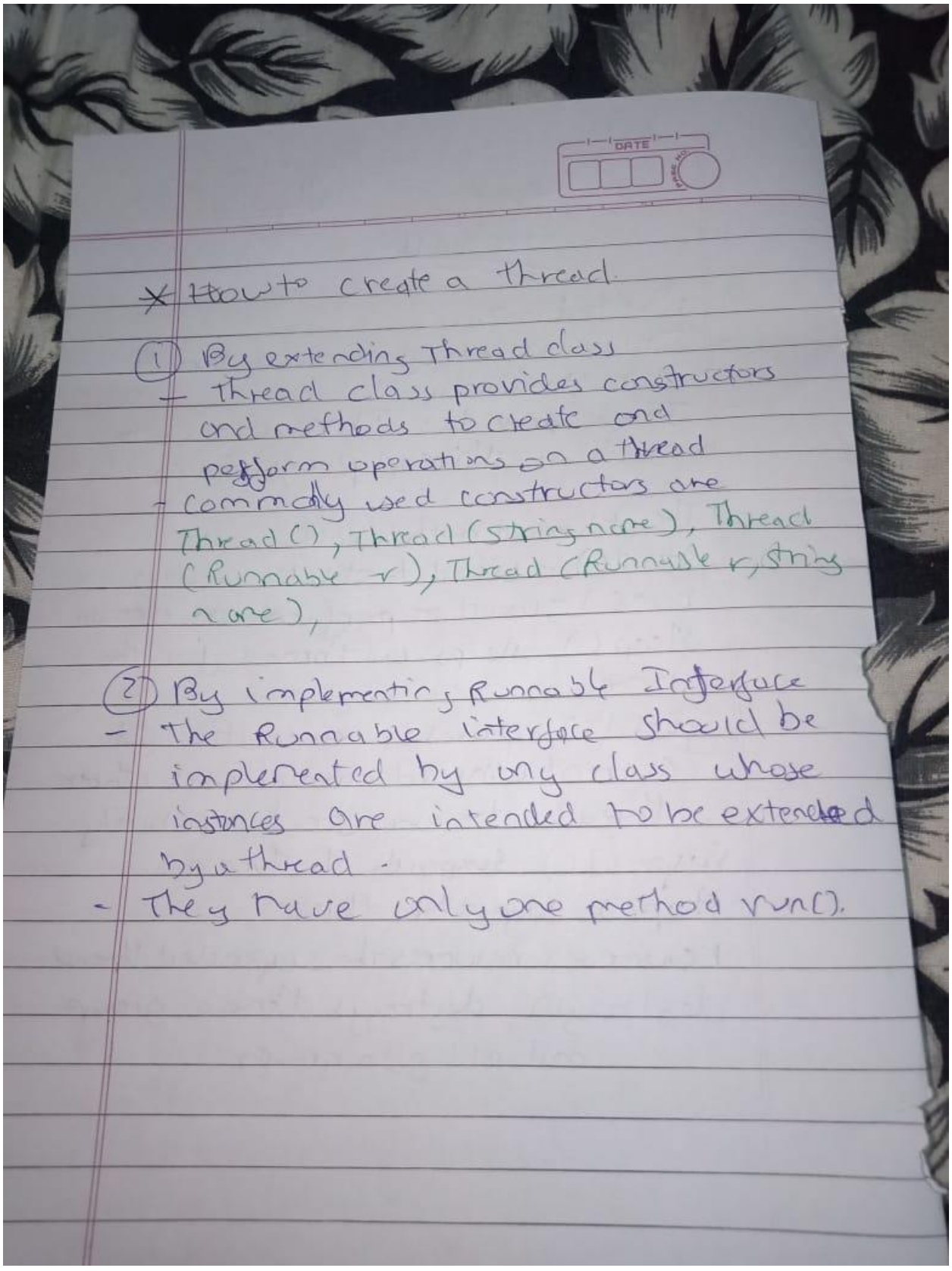
Practical 11



Practical 11

* Multithreading
→ It is a process of executing threads simultaneously.
- Java provides Thread class to achive thread programming.

- The thread methods are
start() - used to start execution.
run() - used to perform an action
sleep() - sleeps a thread for the time.
yield() - pauses execution of current thread and allows other threads to execute temporarily.
Suspend() suspends the thread
stop() stops the thread
resume() resumes the suspended thread
destroy() destroys thread group and all sub groups.



* How to create a thread.

(1) By extending Thread class
- Thread class provides constructors and methods to create and perform operations on a thread
- Commonly used constructors are Thread(), Thread(String name), Thread(Runnable r), Thread(Runnable r, String name),

(2) By implementing Runnable Interface
- The Runnable interface should be implemented by any class whose instances are intended to be extended by a thread.
- They have only one method run().

Practical 11.1

Aim:

Write a java program to demonstrate java thread.

Java Thread Example by extending Thread class

INPUT:

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

OUTPUT:

```
thread is running...
```

Java Thread Example by implementing Runnable interface

INPUT:

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}

public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
 }
}
```

OUTPUT:

```
thread is running...
```

Practical 11.2

Aim:

Create a thread by extending thread

INPUT:

```
class MyNewThread implements Runnable {
    Thread t;
    MyNewThread() {
    // Create a new, second thread
    t = new Thread(this, "Demo Thread");
    System.out.println("Child thread: " + t);
    t.start(); // Start the thread
    }
    // This is the entry point for the second thread.
    public void run() {
    try {
    for(int i = 5; i > 0; i--) {
    System.out.println("Child Thread: " + i);
    Thread.sleep(500);
    }
    } catch (InterruptedException e) {
    System.out.println("Child interrupted.");
    }
    System.out.println("Exiting child thread.");
    }
    }
    class ThreadDemo {
    public static void main(String args[]) {
    new MyNewThread();
    try {
    for(int i = 5; i > 0; i--) {
    System.out.println("Main Thread: " + i);
    Thread.sleep(5000);
    }
    } catch (InterruptedException e) {
    System.out.println("Main thread interrupted.");
    }
    System.out.println("Main thread exiting.");
    }
    }
```

Output:

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java ThreadDemo

```
Child thread: Thread[Demo Thread,5,main]
Main Thread: 5
Child Thread: 5
Child Thread: 4
Child Thread: 3
Child Thread: 2
Child Thread: 1
Exiting child thread.
Main Thread: 4
Main Thread: 3
Main Thread: 2
Main Thread: 1
Main thread exiting.
```

Practical 11.3

Aim:

Create thread by implementing runnable

INPUT:

```
class MyNewThread implements Runnable {
    Thread t;
    MyNewThread() {
    // Create a new, second thread
    t = new Thread(this, "Demo Thread");
    System.out.println("Child thread: " + t);
    t.start(); // Start the thread
    }
    // This is the entry point for the second thread.
    public void run() {
    try {
    for(int i = 5; i > 0; i--) {
    System.out.println("Child Thread: " + i);
    Thread.sleep(500);
    }
    } catch (InterruptedException e) {
    System.out.println("Child interrupted.");
    }
    System.out.println("Exiting child thread.");
    }
    }
    class ThreadDemo {
    public static void main(String args[]) {
    new MyNewThread();
    try {
    for(int i = 5; i > 0; i--) {
    System.out.println("Main Thread: " + i);
    Thread.sleep(5000);
    }
    } catch (InterruptedException e) {
    System.out.println("Main thread interrupted.");
    }
    System.out.println("Main thread exiting.");
    }
    }
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java ThreadDemo

Child thread: Thread[Demo Thread,5,main]

Main Thread: 5

Child Thread: 5

Child Thread: 4

Child Thread: 3

Child Thread: 2

Child Thread: 1

Exiting child thread.

Main Thread: 4

Main Thread: 3

Main Thread: 2

Main Thread: 1

Main thread exiting.
```

Practical 11.4

Aim:

Write a java program to demonstrate unsynchronized thread.

INPUT:

```
// This program is not synchronized.
classCallme {
void call(String msg) {
System.out.print("[" + msg);
try {
Thread.sleep(1000);
} catch(InterruptedException e) {
System.out.println("Interrupted");
}
System.out.println("]");
}
}
class Caller implements Runnable {
String msg;
Callme target;
Thread t;
public Caller(Callmetarg, String s) {
target = targ;
msg = s;
t = new Thread(this);
t.start();
}
public void run() {
target.call(msg);
}
}
class Synch {
public static void main(String args[]) {
Callme target = new Callme();
Caller ob1 = new Caller(target, "Hello");
Caller ob2 = new Caller(target, "Synchronized");
Caller ob3 = new Caller(target, "World");
// wait for threads to end
try {
ob1.t.join();
```

```
ob2.t.join();
ob3.t.join();
} catch(InterruptedException e) {
System.out.println("Interrupted");
}
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Synch.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Synch

[Hello[Synchronized[World]

]

]
```

Practical 11.5

Aim:

Write a java program to demonstrate synchronized thread

INPUT:

```
classCallme {

synchronized void call(String msg) {

System.out.print("[" + msg);

try {

Thread.sleep(1000);

} catch(InterruptedException e) {

System.out.println("Interrupted");

}

System.out.println("]");

}

}

class Caller implements Runnable {

String msg;

Callme target;

Thread t;

public Caller(Callmetarg, String s) {

target = targ;

msg = s;

t = new Thread(this);

t.start();

}

public void run() {
```

```
target.call(msg);
}
}
class Synchronized {
public static void main(String args[]) {
Callme target = new Callme();
Caller ob1 = new Caller(target, "Hello");
Caller ob2 = new Caller(target, "Synchronized");
Caller ob3 = new Caller(target, "World");
// wait for threads to end
try {
ob1.t.join();
ob2.t.join();
ob3.t.join();
} catch(InterruptedException e) {
System.out.println("Interrupted");
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Synchronized.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Synchronized
[Hello]
[Synchronized]
[World]
```

Practical 11.6
Aim:
Write a java program to demonstrate block synchronized thread.
INPUT:

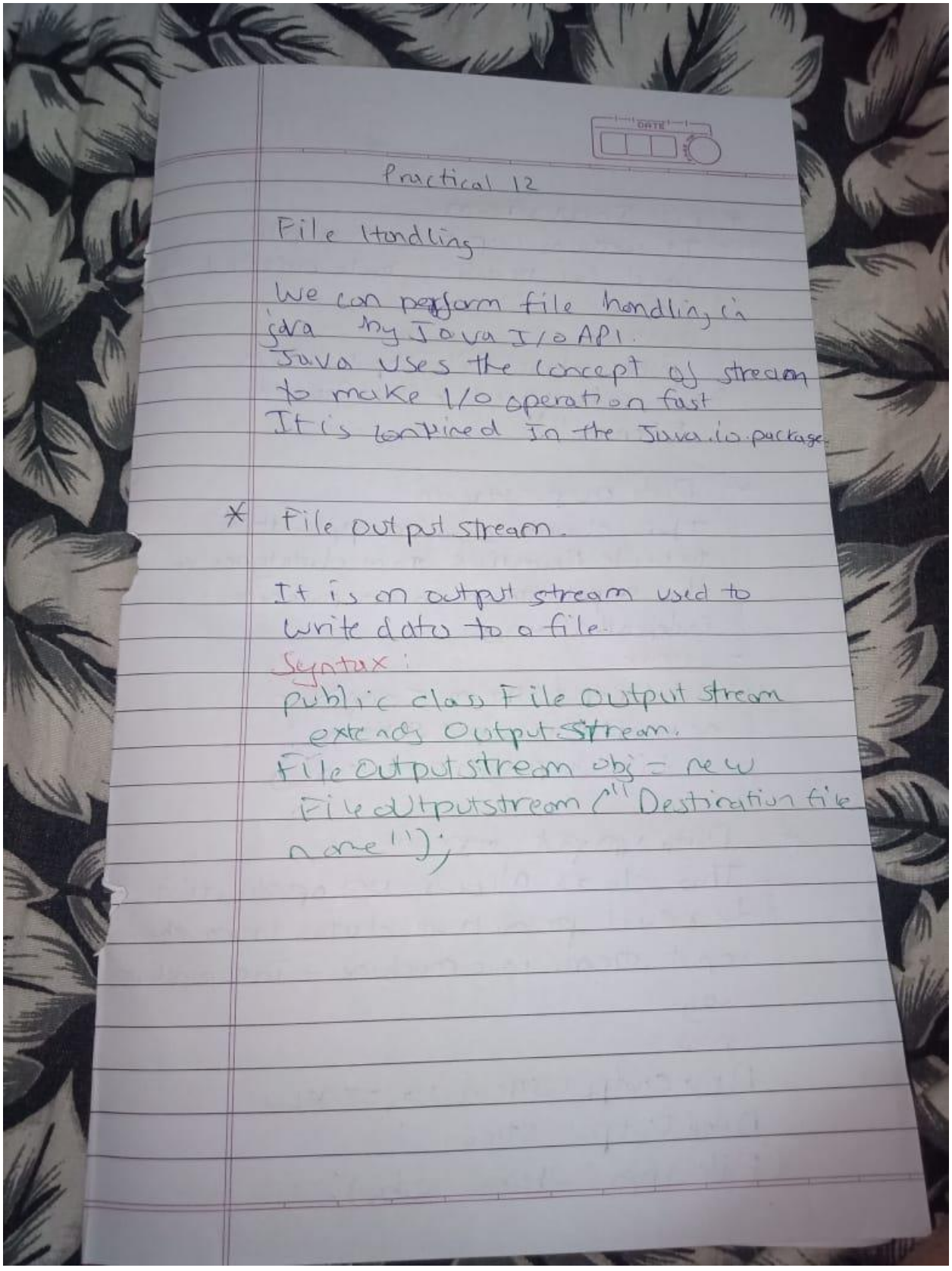
```
// This program uses a synchronized block.

classCallme {

void call(String msg) {

System.out.print("[" + msg);

try {Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Interrupted");

}

System.out.println("]");

}

}

class Caller implements Runnable {

String msg;

Callme target;

Thread t;

public Caller(Callmetarg, String s) {

target = targ;

msg = s;

t = new Thread(this);

t.start();

}
```

```
// synchronize calls to call()
public void run() {
synchronized(target) { // synchronized block
target.call(msg);
}
}
}
class Synch1 {
public static void main(String args[]) {
Callme target = new Callme();
Caller ob1 = new Caller(target, "Hello");
Caller ob2 = new Caller(target, "Synchronized");
Caller ob3 = new Caller(target, "World");
// wait for threads to end
try {
ob1.t.join();
ob2.t.join();
ob3.t.join();
} catch(InterruptedException e) {
System.out.println("Interrupted");
}
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac Synch1.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java Synch1
[Synchronized]
[World]
[Hello]
```

Practical 12



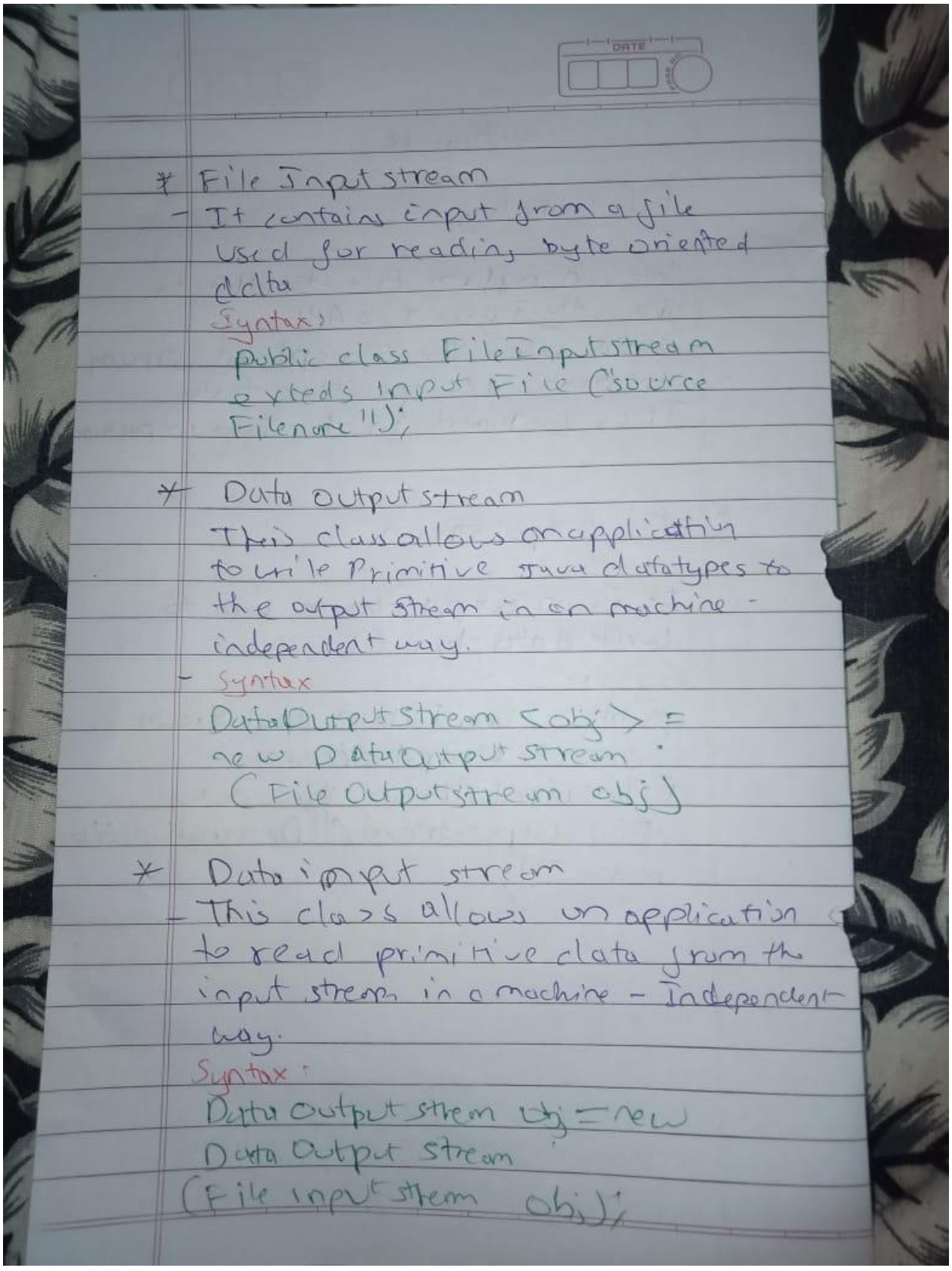
Practical 12

File Handling

We can perform file handling in java by Java I/O API.
Java uses the concept of stream to make I/O operation fast
It is contained in the Java.io package.

* File output stream.

It is an output stream used to write data to a file.
Syntax:
public class FileOutputStream extends OutputStream.
FileOutputStream obj = new FileOutputstream("Destination file name");



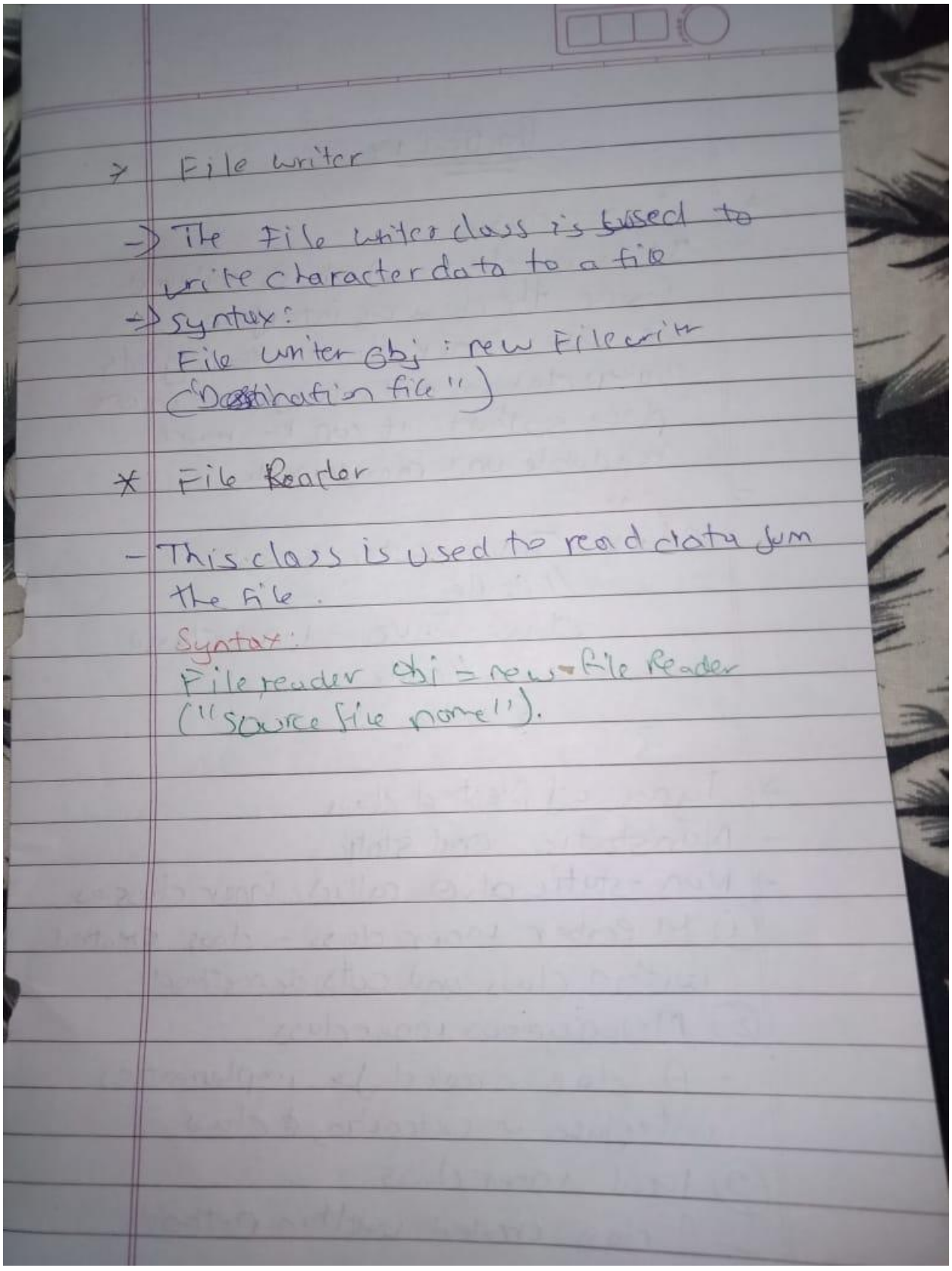
* File Input stream
- It contains input from a file used for reading byte oriented data

Syntax:
public class FileInputStream extends InputFile ("source Filename");

* Data Output stream
This class allows an application to write Primitive Java datatypes to the output stream in an machine independent way.
- Syntax
DataOutputStream <obj> = new DataOutputStream (FileOutputStream obj)

* Data input stream
- This class allows an application to read primitive data from the input stream in a machine - Independent way.

Syntax:
Data Output stream obj = new Data Output Stream (File input stream obj);



* File writer

-> The File writer class is ~~used~~ used to write character data to a file

-> Syntax:
File writer obj : new Filewriter ("Destination file")

* File Reader

- This class is used to read data from the file.

Syntax:
File reader obj = new File Reader ("source file name").

Practical 12.1

Aim:

Write a java program to read data from file.

INPUT:

```
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files
public class ReadFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

OUTPUT:

Input file

Practical 12.2

Aim:

Write a java program to write data to the file.

INPUT:

```
package com.javatpoint;
import java.io.FileWriter;
public class FileWriterExample {
   public static void main(String args[]){
       try{
         FileWriter fw=new FileWriter("D:\\testout.txt");
         fw.write("output");
         fw.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

OUTPUT:

Output

Practical 12.3

Aim:

Write a java program to read file from another drive.

INPUT:

```
import java.io.*;
import java.util.*;
class myclass123
{
	public static void main(String[] args) throws IOException{
		File f = new File("D:/Utilities/Ditto/hey.txt");
		Scanner s;
		try {
BufferedReader reader = new BufferedReader(new FileReader(f));
			String line=reader.readLine();
			while (line != null) {
System.out.println(line);
				line=reader.readLine();
			}
reader.close();

		} catch (FileNotFoundException e) {
System.out.println("File does not exist");
		}
	}
}
```

OUTPUT:

```
Hello I am nikhil
```

Practical 12.4

Aim:

Write a java program for reading data from one file and copy it into another file

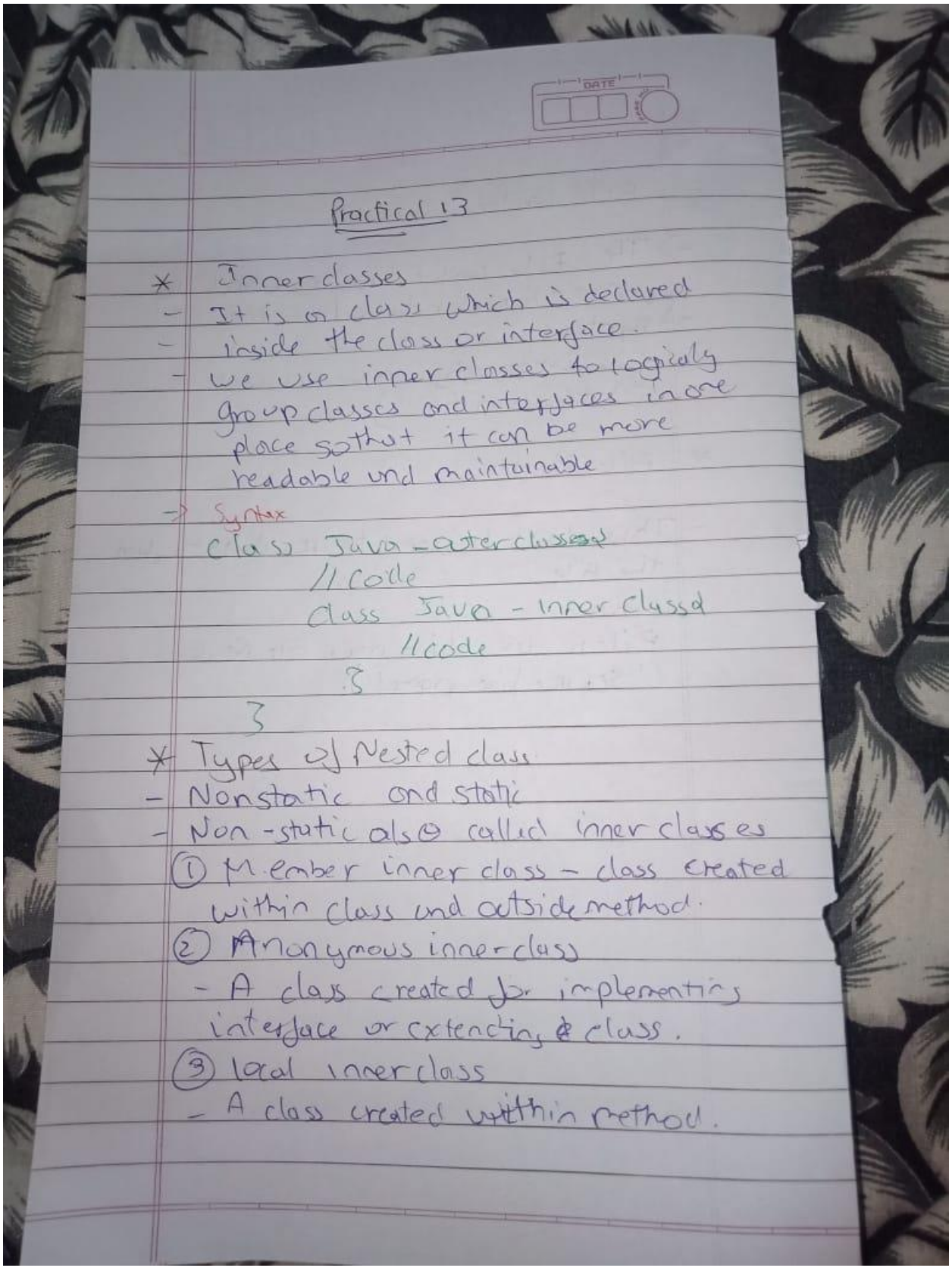
INPUT:

```
//wap to read from a file and copy into another file
import java.io.*;
class CopyFile{
    public static void main(String[] args) {
        File FI =new File("input.txt");
        File FO =new File("output.txt");

        FileReader fr=null;
        FileWriter fw=null;

    try{
        fr=new FileReader(FI);
        fw=new FileWriter(FO);
        int ch;
        while((ch=fr.read())!=-1)
        { char c=(char)ch;

            fw.write(ch);
            System.out.print(" "+c);
        }
    }
    catch(IOException e){
        System.out.println(e);
        System.exit(-1);
    }
    finally{
        try{
            fr.close();
            fw.close();

        }
        catch(IOException e){}
    }
}
}
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac CopyFile.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java CopyFile
 h e l l o  i  a m  n i k h i l
```

PRACTICAL 13



Practical 13

* Inner classes
- It is a class which is declared
- inside the class or interface.
- We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable

-> Syntax

```
class Java - outer class
    // code
    class Java - inner class
        // code
    }
}
```

* Types of Nested class
- Non static and static
- Non-static also called inner classes

(1) Member inner class - class created within class and outside method.

(2) Anonymous inner class
- A class created for implementing interface or extending a class.

(3) Local inner class
- A class created within method.

13.1 Write a java program to demonstrate Inner classes

```
class Outer_Demo {
int num;
private class Inner_Demo {
public void print() {
System.out.println("This is an inner class");
}
}
void display_Inner() {
Inner_Demo inner = new Inner_Demo();
inner.print();
}
}
public class InnerDemo{
public static void main(String args[]) {
Outer_Demo outer = new Outer_Demo();
outer.display_Inner();
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac  InnerDemo.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java InnerDemo
This is an inner class
```

13.2 Write a java program to demonstrate getNum() method of the inner class

```
class Outer_Demo {
private int num = 170;
public class Inner_Demo {
public int getNum() {
System.out.println("This is the getnum method of the inner class");
return num;
}
}
}
public class GetNumDemo{
public static void main(String args[]) {
Outer_Demo outer = new Outer_Demo();
Outer_Demo.Inner_Demo inner = outer.new Inner_Demo();
System.out.println(inner.getNum());
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac  GetNumDemo.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java GetNumDemo
This is the getnum method of the inner class
170
```

13.3 Write a java program to access the method inner class

```
public class MethodInner_Demo{
void my_Method(){
int num = 23;
class InnerDemo{
public void print() {
System.out.println("This is method inner class "+num);
}
}
InnerDemo inner = new InnerDemo();
inner.print();
}
public static void main(String args[]) {
MethodInner_Demo outer = new MethodInner_Demo();
outer.my_Method();
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac MethodInner_Demo.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MethodInner_Demo
This is method inner class 23
```

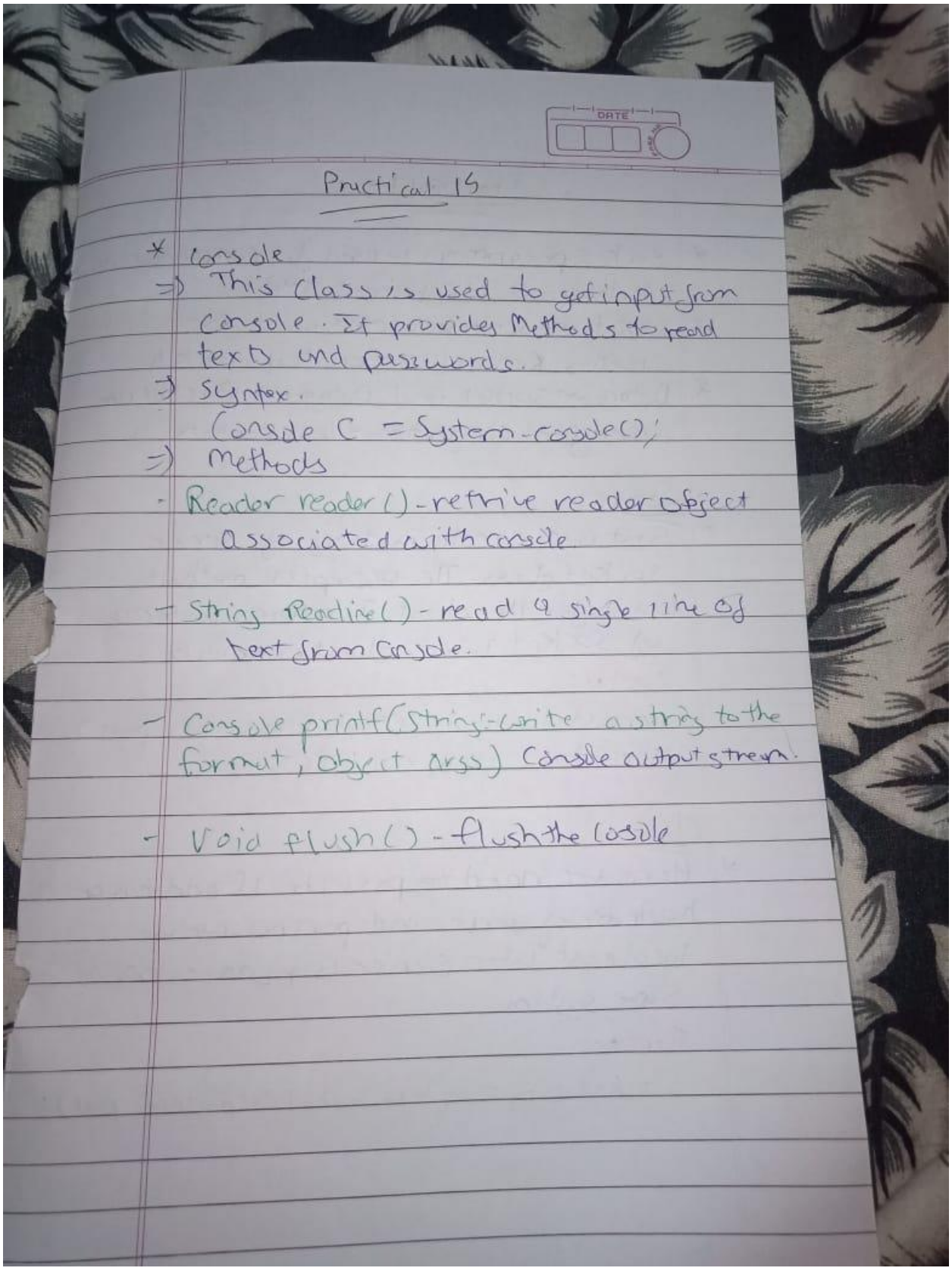
13.4 Write a java program to demonstrate anonymous inner class.

```
abstract class AnonymousInner {
public abstract void mymethod();
}
public class AnonymousInnerClass {
public static void main(String args[]) {
AnonymousInner inner = new AnonymousInner() {
public void mymethod() {
System.out.println("This is an example of anonymous inner class");
}
};
inner.mymethod();
}
}
```

OUTPUT:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac AnonymousInnerClass.java
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java AnonymousInnerClass
This is an example of anonymous inner class
```

Practical No: 14



Practical 14

* Console
  - This class is used to get input from console. It provides Methods to read texts and passwords.
  - Syntax.
    Console C = System.console();
  - Methods
    - Reader reader() - retrive reader object associated with console
    - String Readline() - read a single line of text from console.
    - Console printf(String - write a string to the format, object args) Console output stream.
    - Void flush() - flush the console

Practical No: 14 Write a java program for reading an console input

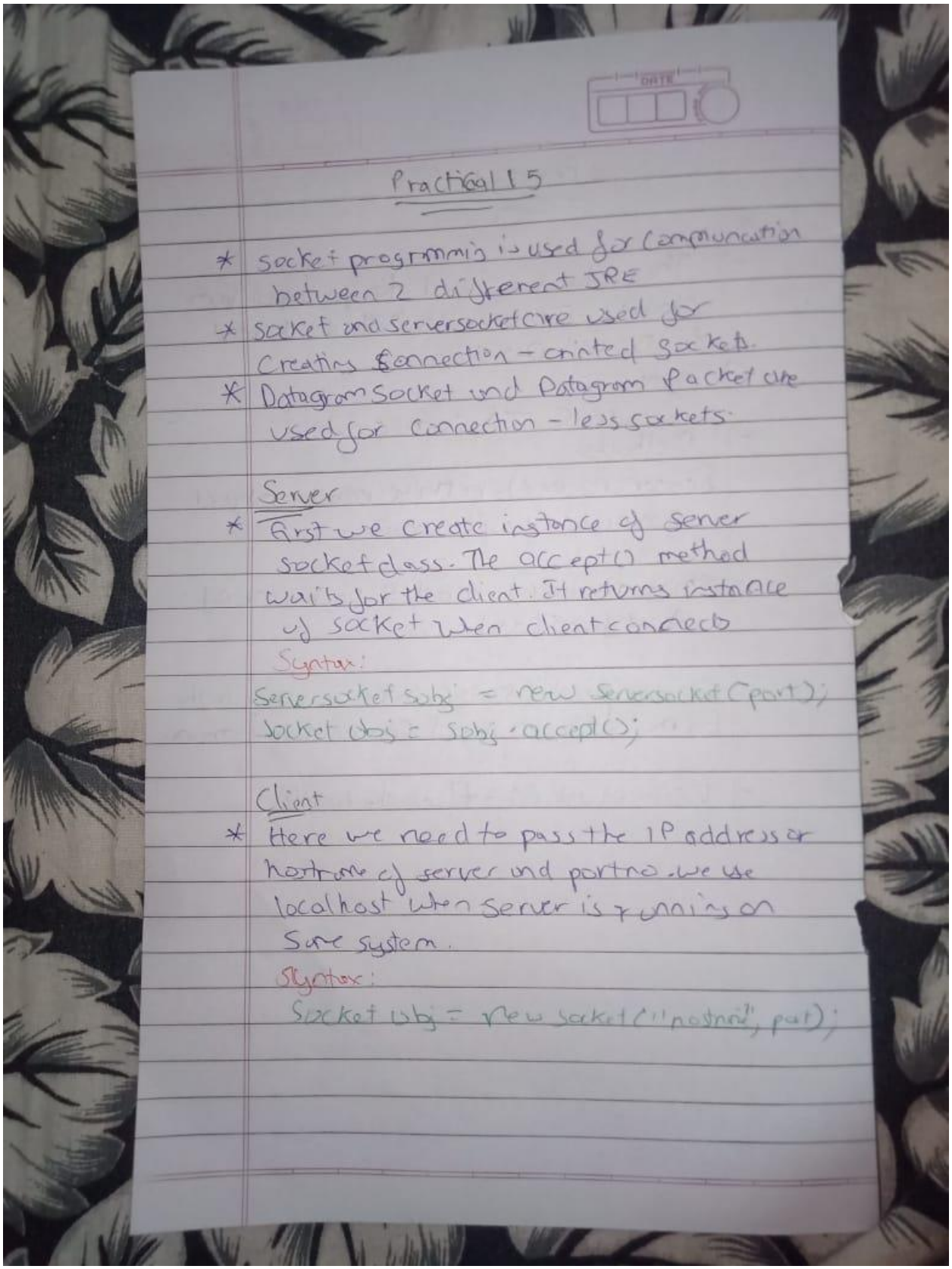
Input:

```
import java.io.*;
class ConsoleInput
{

   public static void main( String args[]) throws IOException{
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      String str[]= new String[100];
      System.out.println("enter lines of code");
      System.out.println("enter 'stop' to quit");
      for(int i=0;i<100;i++){
         str[i]=br.readLine();
         if(str[i].equals("stop"))
            break;

      }
      System.out.println("\n\n her is your data...\n\n");
      for(int i=0;i<100;i++){
         if(str[i].equals("stop"))
            break;
         System.out.println(str[i]);
      }
   }
}
```

Output:

```
C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac ConsoleInput.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java ConsoleInput
enter lines of code
enter 'stop' to quit
hello
i am nikhil
stop


 her is your data...


hello
i am Nikhil
```

Practical No: 15



Practical 15

* Socket programming is used for communication between 2 different JRE
* Socket and serversocket are used for Creating Connection - oriented Sockets.
* Datagram Socket and Datagram Packet are used for Connection - less sockets.

Server

* First we create instance of server socket class. The accept() method waits for the client. It returns instance of socket when client connects

Syntax:

```
ServerSocket sobj = new ServerSocket(port);
Socket obj = sobj.accept();
```

Client

* Here we need to pass the IP address or hostname of server and portno. We use localhost when server is running on same system.

Syntax:

```
Socket obj = new Socket("hostnm", put);
```

Practical No: 15 Write a java program to develop client server chat application using socket programming.

Input:

Server Code:

```
import java.net.*;

import java.io.*;

class MyServer{

public static void main(String args[])throws Exception{

ServerSocket ss=new ServerSocket(3333);

Socket s=ss.accept();

DataInputStream din=new DataInputStream(s.getInputStream());

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

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


String str="",str2="";

while(!str.equals("stop")){

str=din.readUTF();

System.out.println("client says: "+str);

str2=br.readLine();

dout.writeUTF(str2);

dout.flush();

}

din.close();

s.close();

ss.close();

}}
```

Client Code:

```
import java.net.*;
import java.io.*;
class MyClient{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=br.readLine();
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Server says: "+str2);
}

dout.close();
s.close();
}}
```

Output:

Server output

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac MyServer.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MyServer

client says: hello sycs45 is here

welcome SYCS45

client says: how is java

its very intresting

client says: ok bye

bye

client says: stop

stop

Client output:

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>javac MyClient.java

C:\Users\Nikhil\Desktop\bsc cs\sem 3 practical\java>java MyClient

hello sycs45 is here

Server says: welcome SYCS45

how is java

Server says: its very intresting

ok bye

Server says: bye

stop

Server says: stop