LOKMANYA TILAK COLLEGE OF ENGINEERING

Sector No. 4., Vikas Nagar, Koparkhairane, Navi Mumbai -400 709.

Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

AY 2021-22 (Odd)

(SEM. III)

Practical file

for

Skill Skill base Lab course

(Object Oriented Programming with Java)

(CSL-304)

Name of Student: **SINGH SUDHAM DHARMENDRA**

Roll no. : **AIMLD50**

Faculty Incharge: Dr. Monika Mangla

Subject Incharge:
Dr. Monika Mangla

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (AI & ML)
Year 2021-22 (ODD SEM)
Object Oriented Programming with JAVA

1. **Programs on Basic programming constructs like branching and looping**

   1. Simple print function
   2. Larger of 2 numbers
   3. Largest of 3 numbers
   4. Factorial of a number

2. **Program on accepting input through keyboard.**

   Pattern *(Accept input from user)*
   Calculation of Simple Interest *(Accept input from user)*

3. **Programs on class and objects**

   Constructor overloading for Rectangle Class

4. **Program on method and constructor overloading.**

   Find the area and perimeter of a circle. Also use multiple constructors for the circle class.
   Declare a Student class with different data members. Also put the constructor(default and parameterized), getdata() and showdata() function.

5. **Program on 2D array, strings functions**

   Input array from user & sum of all elements of array
   Smallest & largest element in an array
   Simple 2D matrix
   Transpose
   Addition of 2 matrix
   Sum of diagnoal elements of matrix

Sorting of elements in descending order

6. **Program on StringBuffer and Vectors**

Vowels & consonants(string)

7. **Program on types of inheritance**

Default constructor inheritance.
Parameterized constructor inheritance.

8. **Program on abstract class and abstract methods.**

Define an abstract shape class and have abstract method area. Extend this class into circle and rectangle.

9. **Program using super and final keyword**

Use Final keyword with function, class, variable and study the difference when you try to change the value.

10. **Program on Exception handling**

Write a simple division program using try and catch block to study different types of exceptions
(Accept number input from user)
Use finally block in same program

11. **Program on user defined exception**

Throwing a user defined exception

12. **Program on Multithreading**

13. **Program on applet class**
14. **ASSIGNMENT 1 & 2**
15. **Mini Project based on the content of the syllabus(Group of 2-3 students)**

TOPIC: CURRENCY CONVERTER

Experiment no. 1

AIM :- Programs on basic programming constructs like branching and looping.

1. Simple print function.

2. Larger of 2 numbers

3. Largest of 3 numbers

4. Factorial of a number.

Theory :- In computer programming, loops are used to repeat a block of code. For example, if you want to show a message 100 times, then rather than typing the same code 100 times, you can use a loop.

In Java, there are three types of loops:-

1) for loop

2) while loop

3) do...while

Branching statements are the statements used to jump the flow of execution from one part of a program to another. The branching statements are mostly used insidethe control statements. Java has mainly three branching statements, i.e., continue, break, and return. The branching statements allow us to exit from a control statement when a certain condition meet.

Branching statements are of 3 types :

a) Break statements : The break statement is used for terminating a loop based on acertain condition. it is of two types:

1) Labeled break statement

2) Unlabeled break statement

b) Continue statement : The continue statement is another branching statement used to immediately jump to the next iteration of the loop. It is a special type of loop which breakscurrent iteration when the condition is met and start the loop with the next iteration. In simple words, it continues the current flow of the program and stop executing the remaining code at the specified condition.

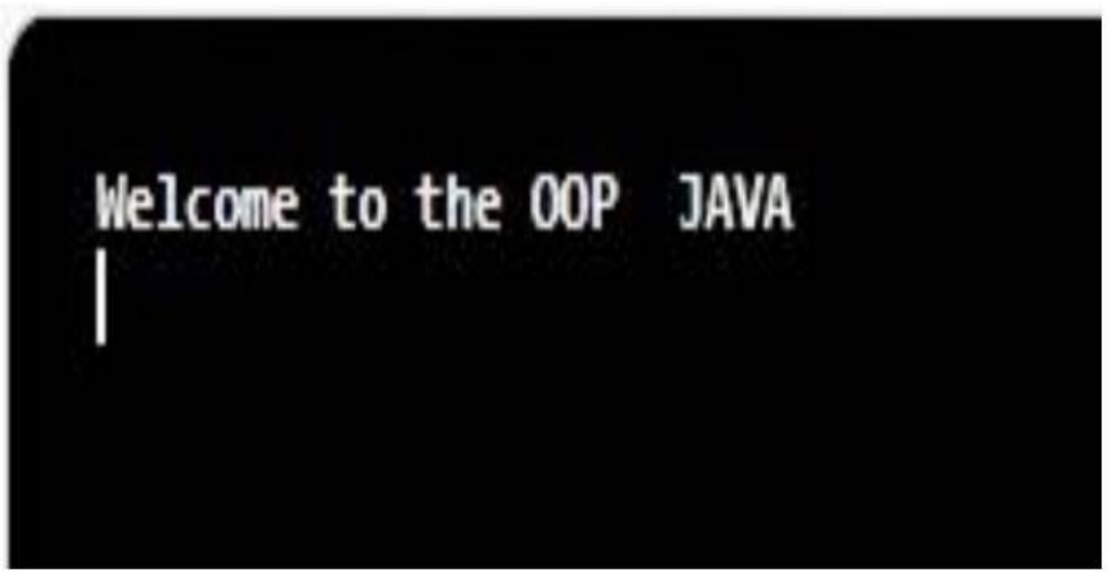
c) Return statements : The return statement is also a branching statement, which allows us to explicitly return value from a method. The return statement exits us from the calling method and passes the control flow to where the calling method is invoked. Just like the break statement, the return statement also has two forms, i.e., one that passes some value with control flow and one that doesn't.

1. Simple print function.

Program :

```
import java.util.*;
import java.io.*;
public class welcome
{
public static void main(String args[])
{
System.out.println("Welcome to the OOP JAVA");
}
}
```

OUTPUT:-



2. Larger of 2 numbers.

Program :-

```
import java.lang.*;
import java.io.*;
public class largernumber
{
 public static void main(String args[])
{
int a,b;
a=45;
b=80; if (a>b)
System.out.println("the number "+a+" is the larger than "+b+".");
```

```
 else
 System.out.println("the number "+b+" is the larger than "+a+".");
 }

}
```

OUTPUT :-

```
the number 80 is the larger than 45.
```

3. Largest of 3 numbers.

Program :-

```
import java.lang.*;
import java.io.*;
public class largest
{
 public static void main(String args[])
 {
 int a,b,c;
 a=25;
 b=65;
 c=80;
 if((a>b)&&(a>c))
 System.out.println("a is largest");
 else if ((b>a)&&(b>c))
 System.out.println("b is largest");
 else if ((c>a)&&(c>b))
 System.out.println("c is largest");
```

```
 }
}
```

OUTPUT :-

```
80 is largest.
```

4. Factorial of a number.

Program :-

```
import java.lang.*;
import java.io.*;
public class factorial
{
 public static void main(String args[])
 {


int n;
n = 7;
int fact,i;
fact = 1;
if(n<0)
{
System.out.println("ERROR! No factorial for negative integer.");
}
else
{
for(i = 1; i <= n; ++i)
{
```

```
fact *= i;
}
System.out.println("The factorial of "+n+" is " +fact);
}
}
}
```

OUTPUT:-

```
The factorial of 7 is 5040
```

EXPERIMENT NO. 2

AIM :- Program on accepting input through keyboard.

1. Pattern (Accept input from user)

2. Calculation of simple interest (Accept input from user) .

THEORY :-

Java Scanner Class:

Java Scanner class allows the user to take input from the console. It belongs to java.util package. It is used to read the input of primitive types like int , double, long, short, float, and byte. It is the easiest way to read input in Java program.

Syntax :

Scanner sc = new Scanner(System.in);

The above statement creates a constructor of the Scanner class having System.in as an argument. It means it is going to read from the standard input stream of the program. The java.util package should be import while using Scanner class. It also converts the Bytes (from the input stream) into characters using the platform's default charset.

Methods of Java Scanner Class:

1) int nextInt() = It is used to scan the next token of the input as an integer.

2) float nextFloat() = It is used to scan the next token of the input as a float.

3) double nextDouble() = It is used to scan the next token of the input as a double.

4) byte nextByte() = It is used to scan the next token of the input as a byte.

5) String nextLine() = Advances this scanner past the current line.

6) boolean nextBoolean() = It is used to scan the next token of the input into a boolean value.

7) long nextLong() = It is used to scan the next token of the input as a long.

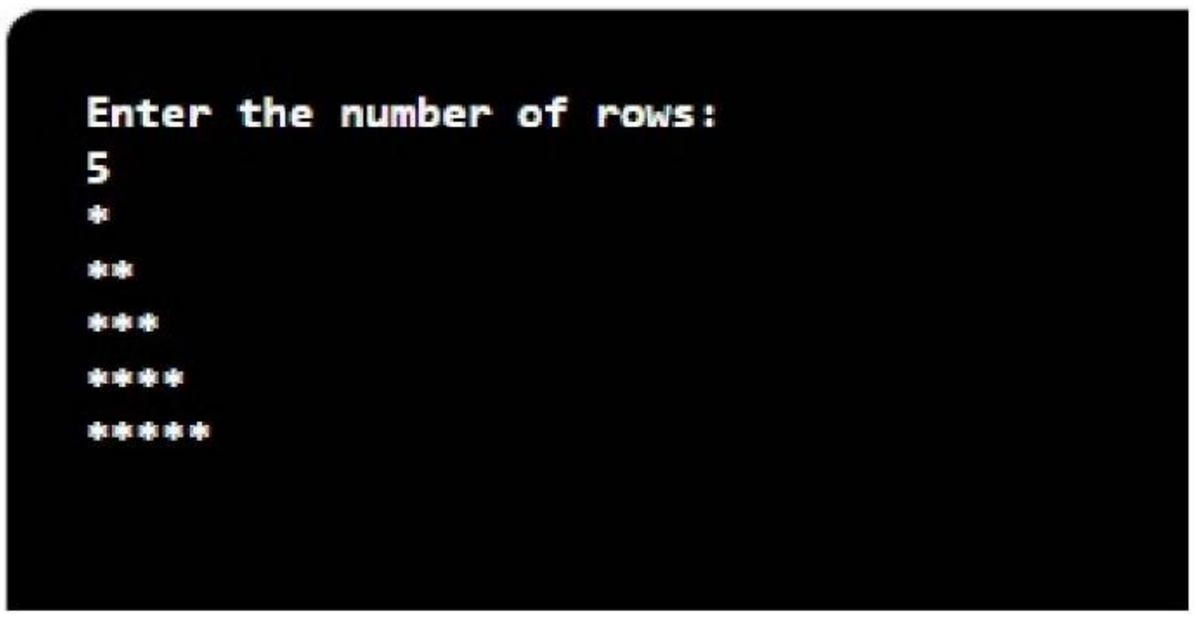
1. Pattern (Accepts from user)

Program :-

```
import java.lang.*;
import java.io.*;
import java.util.*;
public class pattern
```

```
{
 public static void main(String args[])
 {
 int i;
 int j;
 int rows;
Scanner s = new Scanner(System.in);
System.out.println("Enter the number of rows: ");
rows = s.nextInt();
for(i=1;i<=rows;i++)
{
for(j=1;j<=i;j++)
System.out.print("*");
System.out.println();
}
}
}
```

OUTPUT:-



2. Calculation of Simple Interest

Program :-

```
import java.util.Scanner;
public class JavaExample
```

```
{
 public static void main(String args[])
 {
 float p, r, t, sinterest;
 Scanner scan = new Scanner(System.in);
 System.out.print("Enter the Principal : ");
 p = scan.nextFloat();
 System.out.print("Enter the Rate of interest : ");
 r = scan.nextFloat();
 System.out.print("Enter the Time period : ");
 t = scan.nextFloat();
 scan.close();
 sinterest = (p * r * t) / 100;
 System.out.print("Simple Interest is: " +sinterest);
 }
}
```

OUTPUT:-

```
Enter the Principal : 33
Enter the Rate of interest : 31
Enter the Time period : 38
Simple Interest is: 388.74
```

EXPERIMENT NO. 3

AIM :- Programs on class and object.

1. Constructor overloading for rectangle class

THEORY :-

OBJECT : An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table,

car, etc. It can be physical or logical (tangible and intangible). The example of an intangible An object has

three characteristics:

a) State: represents the data (value) of an object.

b) Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.

c) Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible

to the external user. However, it is used internally by the JVM to identify each object uniquely. For

Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so

writing is its behavior. An object is an instance of a class. A class is a template or blueprint from which

objects are created. So, an object is the instance(result) of a class.

Object Definitions:

1) An object is a real-world entity.

2) An object is a runtime entity.

3) The object is an entity which has state and behavior.

4) The object is an instance of a class. object is the banking system.

CLASS :

A class is a group of objects which have common properties. It is a template or blueprint from which

objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

1) Fields

2) Methods

3) Constructors

4) Blocks

5) Nested class and interface

1. Constructor overloading for rectangle class

Program :-

```
public class Rectangle
{
 int length;
 int breadth;
 Rectangle()
 {
System.out.println("Testing the execution of constructor:");
length=breadth=0;
}
Rectangle(int x)
{
System.out.println("Testing the execution of parametrized constructor with 1 argument");
length = x;
breadth = x;
}
Rectangle(int x, int y)
{
System.out.println("Testing the execution of parametrized constructor");
length = x;
breadth = y;
}
void area()
{
int a = length * breadth;
System.out.println("The area of rectangle is "+a);
}
public static void main(String args[])
{
Rectangle r1 = new Rectangle();
```

```
Rectangle r2 = new Rectangle();
Rectangle r3 = new Rectangle(6,8);
r3.area();
Rectangle r4 = new Rectangle(4);
}
}
```

OUTPUT:-

```
Testing the execution of constructor:
Testing the execution of constructor:
Testing the execution of parametrized constructor
The area of rectangle is 63
Testing the execution of parametrized constructor with 1 argument
```

EXPERIMENT NO. 4

AIM :- Program on method and constructor overloading.

1. Find the area and perimeter of a circle. Also use multiple constructors for the circle class.

2. Declare a student class with different data members. Also put the constructor(default and parametrized), getdata() and showdata() function.

THEORY :-

Constructor Overloading:

1) Writing more than 1 constructor in a class with a unique set of arguments is called as Constructor Overloading.

2) All constructors will have the name of the class.

3) Overloaded constructor will be executed at the time of instantiating an object.

4) An overloaded constructor cannot be static as a constructor relates to the creation of an object.

5) An overloaded constructor cannot be final as constructor is not derived by subclasses it won't make

sense.

6) An overloaded constructor can be private to prevent using it for instantiating from outside of the class.

Method Overloading:

1) Writing more than one method within a class with a unique set of arguments is called as method overloading.

2) All methods must share the same name.

3) An overloaded method if not static can only be called after instantiating the object as per the requirement.

4) Overloaded method can be static, and it can be accessed without creating an object.

5) An overloaded method can be final to prevent subclasses to override the method.

6) An overloaded method can be private to prevent access to call that method outside of the class.

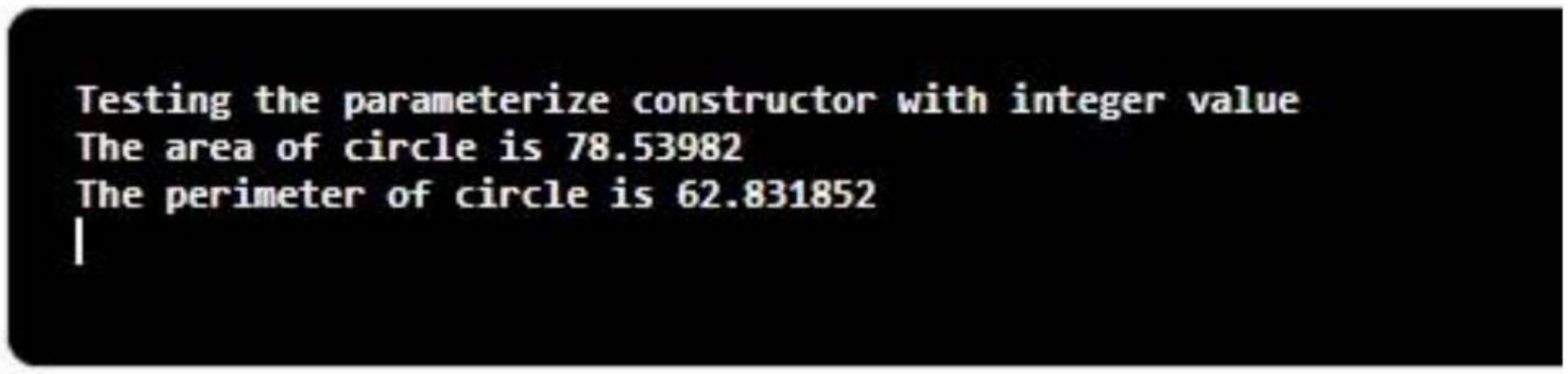
1. Find the area and perimeter of a circle. Also use multiple constructors for the circle class.

Program :-

```
public class circle
{
 float r;
```

```
circle() {
r = 0;
System.out.println("Testing the default constructor");
}
circle(int x)
{
r = x;
System.out.println("Testing the parameterize constructor with integer value");
}
circle(float x)
{
r = x;
System.out.println("Testing the parameterize constructor with float value");
}
void area()
{
float a = (float) (Math.PI * r * r);
System.out.println("The area of circle is " + a);
}
void perimeter()
{
float a = (float) (Math.PI * 2 * r);
System.out.println("The perimeter of circle is " + a);
}
public static void main(String[] args) {
circle c1 = new circle(5);
 c1.area();
 c1.perimeter();
 }
}
```

OUTPUT:-



2.Declare a student class with different data members. Also put the constructor(default and parametrized), getdata() and showdata() function.

Program :-

```
import java.lang.*;
import java.util.Scanner;
public class student {
 String name;
 int roll_no;
 String branch;
 student() {
 name = "SINGH SUDHAM DHARMENDRA";
 roll_no = 50;
 branch = "CSE(AIML)";
 }
void getData() {
 Scanner in = new Scanner(System.in);
 System.out.println("Enter the name: ");
 name = in.nextLine();
 System.out.println("Enter the Branch: ");
 branch = in.nextLine();
 System.out.println("Enter the Roll No.: ");
 roll_no = in.nextInt();
 in.close();
 }
```

```
void showData() {
System.out.println("Name: " + name);
System.out.println("Roll no.: " + roll_no);
System.out.println("Branch: " + branch);
}
public static void main(String[] args) {
 student s1 = new student();
 s1.getData();
 s1.showData();
 }
}
```

OUTPUT:-

```
Enter the name:
Singh Sudham Dharmendra
Enter the Branch:
CSE(AIML)
Enter the Roll No.:
50
Name: Singh Sudham Dharmendra
Roll no.: 50
Branch: CSE(AIML)
```

EXPERIMENT NO. 5

AIM :- Program on 2D arrays, strings, functions.

1. Input array from user and sum of all elements of array.

2. Smallest and largest number in array.

3. Simple 2D matrix.

4. Transpose

5. Addition of 2 matrix.

6. Sum of diagonal elements of matrix.

7. Sorting of elements in descending order.

THEORY :-

A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself.

For example,

Syntax for 2D array:-

int[][] a = new int[3][4];

A 2D-array can hold maximum of 12 elements.

String functions:- In Java, a string is an object that represents a sequence of characters or char values. The java.lang.String class is used to create a Java string object.

There are two ways to create a String object:

1. By string literal : Java String literal is created by using double quotes.

For Example: String s="Welcome";

2. By new keyword : Java String is created by using a keyword "new".

For example: String s=new String("Welcome");

It creates two objects (in String pool and in heap) and one reference variable where the variable

‘s’ will refer to the object in the heap.

3. Some string functions are:-

a) Java String endsWith()

b) Java String IsEmpty()

c) Java StringGetBytes()

d) Java String replace()

e) Java String toUpper() etc.

1.Input array from user and sum of all elements of array.

Program :-

```
import java.util.Scanner;

public class array

{

 public static void main(String[] args)

 {

 Scanner in = new Scanner(System.in);

 System.out.println("Enter the size of array: ");

int n = in.nextInt();

int arr[] = new int[n];

System.out.println("Enter the elements of array: ");

for (int i = 0; i < n; i++)

{

arr[i] = in.nextInt();

}

for (int i = 0; i < n; i++)

{

System.out.print(arr[i] + " ");

}

System.out.println();

int sum = 0;

for (int i = 0; i < n; i++)

{
```

```
 sum += arr[i];
 }
 System.out.println("Sum of array " + sum);
 }
}
```

OUTPUT :-

```
Enter the size of array:
5
Enter the elements of array:
21
56
43
45
76
21 56 43 45 76
Sum of array 241
```

2. Smallest and largest number in array.

Program :-

```
import java.util.Scanner;
public class minNmax
{
 public static void main(String[] args)
 {
 Scanner in = new Scanner(System.in);
 System.out.println("Enter the size of array: ");
 int n = in.nextInt();
 int arr[] = new int[n];
 System.out.println("Enter the elements of array: ");
 for (int i = 0; i < n; i++)
 {
 arr[i] = in.nextInt();
 }
```

```
int min_ele = arr[0], max_ele = arr[0];
for (int i = 1; i < n; i++)
{
if (arr[i] < min_ele)
{
min_ele = arr[i];
}
}
for (int i = 1; i < n; i++)
{
if (arr[i] > max_ele)
{
max_ele = arr[i];
}
}
System.out.println("Minimum Element: " + min_ele);
 System.out.println("Maximum Element: " + max_ele);
 }
}
```

OUTPUT :-

```
Enter the size of array:
3
Enter the elements of array:
1 2 3
Minimum Element: 2
Maximum Element: 123
```

3. Simple 2D matrix.

Program :-

```
import java.util.Scanner;
public class twoDarray {
 public static void main(String[] args) {
 int i, j, r, c;
 int a[][] = new int[5][5];
 Scanner in = new Scanner(System.in);
 System.out.println("Enter the number of rows in the matrix");
 r = in.nextInt();
 System.out.println("Enter the number of columns in the matrix");
 c = in.nextInt();
 System.out.println("Enter the elements of matrix");
 for (i = 0; i < r; i++) {
 for (j = 0; j < c; j++) {
 a[i][j] = in.nextInt();
 }
 }
 System.out.println("The elements of matrix are ");
 for (i = 0; i < r; i++) {
 for (j = 0; j < c; j++) {
 System.out.print(a[i][j] + " ");
 }
 System.out.println();
 }


 }
}
```

OUTPUT :-

```
Enter the number of rows in the matrix
4
Enter the number of columns in the matrix
5
Enter the elements of matrix
1 2 3 4 5 6 7 8 9
10
11
23
45
65
76
75
875
334
566
77664

The elements of matrix are
1 2 3 4 5
6 7 8 9 10
11 23 45 65 76
75 875 334 566 77664
```

4. Transpose.

Program :-

```
import java.util.Scanner;

public class twoDarray {

 public static void main(String[] args) {

 int i, j, r, c;

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

 Scanner in = new Scanner(System.in);

 System.out.println("Enter the number of rows in the matrix");

 r = in.nextInt();

System.out.println("Enter the number of columns in the matrix");

c = in.nextInt();

System.out.println("Enter the elements of matrix");

for (i = 0; i < r; i++) {

for (j = 0; j < c; j++) {
```

```
a[i][j] = in.nextInt();
}
}
System.out.println("The elements of matrix are ");
for (i = 0; i < r; i++) {
for (j = 0; j < c; j++) {
System.out.print(a[i][j] + " ");
}
System.out.println();
}
System.out.println("The transpose of matrix are ");
for (i = 0; i < r; i++) {
for (j = 0; j < c; j++) {
System.out.print(a[j][i] + " ");
}
System.out.println();
 }
 }
}
```

OUTPUT :-

```
Enter the size of array:
3
Enter the elements of array:
1 2 3
Minimum Element: 2
Maximum Element: 123
```

---

5. Addition of 2 matrix.

Program :-

```
import java.util.Scanner;
public class sumOfTwoMatrix {
 public static void main(String[] args) {
 int i, j, r1, c1, r2, c2;
 int a[][] = new int[5][5];
 int b[][] = new int[5][5];
 Scanner in = new Scanner(System.in);
 System.out.println("Enter the number of rows of 1st matrix");
 r1 = in.nextInt();
 System.out.println("Enter the number of columns of 1st matrix");
 c1 = in.nextInt();
 System.out.println("Enter the elements of 1st matrix");
 for (i = 0; i < r1; i++) {
 for (j = 0; j < c1; j++) {
 a[i][j] = in.nextInt();
 }
 }
 System.out.println("The elements of matrix are ");
 for (i = 0; i < r1; i++) {
 for (j = 0; j < c1; j++) {
 System.out.print(a[i][j] + " ");
 }
 System.out.println();
 }
 System.out.println("Enter the number of rows of 2st matrix");
 r2 = in.nextInt();
 System.out.println("Enter the number of columns of 2st matrix");
 c2 = in.nextInt();
 System.out.println("Enter the elements of 2st matrix");
 for (i = 0; i < r2; i++) {
```

```
for (j = 0; j < c2; j++) {
b[i][j] = in.nextInt();
}
}
System.out.println("The elements of matrix are ");
for (i = 0; i < r2; i++) {
for (j = 0; j < c2; j++) {
System.out.print(b[i][j] + " ");
}
System.out.println();
}
if (r1 == r2 && c1 == c2) {
System.out.println("Sum of two matrix is: ");
for (i = 0; i < r1; i++) {
for (j = 0; j < c1; j++) {
System.out.print(a[i][j] + b[i][j] + " ");
 }
 System.out.println();
 }
 } else {
 System.out.println("Sum of the matrix is not possible");
 }
 }
}
```

OUTPUT :-

```
Enter the number of rows of 1st matrix
2
Enter the number of columns of 1st matrix
2
Enter the elements of 1st matrix
1 2 3 4
The elements of matrix are
1 2
3 4
Enter the number of rows of 2st matrix
2
Enter the number of columns of 2st matrix
2
Enter the elements of 2st matrix
4 3 2 1
The elements of matrix are
4 3
2 1
Sum of two matrix is:
5 5
5 5
```

6. Sum of diagonals element of matrix.

Program:-

```
import java.util.Scanner;

public class diagonalSum {

public static void main(String[] args) {

int i, j, r, c;

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

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of rows in the matrix");

r = in.nextInt();

System.out.println("Enter the number of columns in the matrix");

c = in.nextInt();

System.out.println("Enter the elements of matrix");

for (i = 0; i < r; i++) {

for (j = 0; j < c; j++) {
```

```
a[i][j] = in.nextInt();
}
}
int sum = 0;
System.out.println("The elements of matrix are ");
for (i = 0; i < r; i++) {
sum += a[i][i];
}
System.out.println("Sum of Diagonal Sum: " + sum);
}
}
```

OUTPUT:-

```
Enter the number of rows in the matrix
2
Enter the number of columns in the matrix
2
Enter the elements of matrix
1
2
3
4
The elements of matrix are
Sum of Diagonal Sum: 127
```

7. Sorting of elements in descending order.

Program :-

```
import java.util.Scanner;
public class sorting {
 public static void main(String[] args) {
 Scanner in = new Scanner(System.in);
 System.out.println("Enter the size of array: ");
```

```
int n = in.nextInt();

int arr[] = new int[n];

System.out.println("Enter the elements of array: ");

for (int i = 0; i < n; i++) {

arr[i] = in.nextInt();

}

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (arr[i] < arr[j]) {

arr[i] ^= arr[j];

arr[j] ^= arr[i];

arr[i] ^= arr[j];

}

}

}

System.out.print("Sorted array in descending order: ");

for (int i = 0; i < n; i++) {

System.out.print(arr[i] + " ");

}

in.close();

}
```

OUTPUT:-

EXPERIMENT NO. 6

AIM :- Program on string buffer and vectors.

1. Vowels and consonants (string)

THEORY :-

String Buffer:-

java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

Constructors of String Buffer class:

1.StringBuffer() :- it reserves room for 16 characters without allocation.

StringBuffer s = new Stringbuffer();

2. StringBuffer(int size):-It accepts integer argument that explicitly sets the size of the buffer.

StringBuffer s = new Stringbuffer(10);

3. StringBuffer(String str):-It accepts String argument that sets the initial content of the string buffer object and reserves room for 16 characters without allocation.

StringBuffer s = new Stringbuffer("hello world")

Methods of String Buffer class:-

append() Used to add text at the end of the existing text.

length() The length of a StringBuffer can be found by the length( ) method

capacity() the total allocated capacity can be found by the capacity( ) method

charAt()

delete() Deletes a sequence of characters from the invoking object

deleteCharAt() Deletes the character at the index specified by loc

ensureCapacity() Ensures capacity is at least equals to the given minimum.

insert() Inserts text at the specified index position

length() Returns length of the string

reverse() Reverse the characters within a StringBuffer object

replace()

Replace one set of characters with another set inside a StringBuffer

object

Vector s:- Vector is like the dynamic array which can grow or shrink its size. Unlike array, we can store n-number of elements in it as there is no size limit.

It is similar to the ArrayList, but with two differenceso Vector is synchronized.

o Java Vector contains many legacy methods that are not the part of a

collections framework.

o Some of the java vector constructors are vector(),vector((int

initialCapacity),vector(int initialCapacity, int capacityIncrement),vector(Collection<? extends E>

c).

Java vector methods are

add(),addAll(),addElement(),capacity(),clear(),contains(),clone() etc.,

1. Vowels and consonants (strings)

Program :-

```
public class numofvowels {
 public static boolean vowel(char a) {
 if (a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'u') {
 return true;
 }
 return false;
 }
 public static void main(String[] args) {
 String s = "ishita";
 s = s.toLowerCase();
 int n = s.length();
 int cnt = 0;
 for (int i = 0; i < n; i++) {
 if (vowel(s.charAt(i)))
 cnt++;
 }
 System.out.println("Number of vowels : " + cnt);
 System.out.println("Number of consonants : " + (n -cnt));
 }
}
```

OUTPUT:-

EXPERIMENT NO. 7

AIM :- Programs on type of inheritance.

1. Default constructor inheritance

2. Parametrized constructor inheritance.

THEORY :-

Types of Inheritance in java:

1)Single inheritance : When a class inherits another class, it is known as a single inheritance.

2) multi-level inheritance: When there is a chain of inheritance, it is known as multilevel inheritance

3)Hierarchial Inheritance : When two or more classes inherits a single class, it is known as hierarchical inheritance.

4) Multiple inheritance : one class have more than one super classes and inherits features from all parent classes . please note that java does not support multiple inheritances with classes , in java it can achieved only by interfaces.

1. Default constructor inheritance.

Program :-

```
public class Rectangle {
Rectangle() {
System.out.println("Testing the default constructor.");
}
public static void main(String args[])
{
Rectangle r1 = new Rectangle();
Rectangle r2 = new Rectangle();
}
}
```

OUTPUT:-

```
Testing the default constructor.
Testing the default constructor.
```

2. Parametrized constructor inheritance.

Program :-

```
public class Rectangle {
int length;
int breadth;
Rectangle() {
System.out.println("Testing the execution of constructor:");
length = breadth = 0;
}
Rectangle(int x) {
System.out.println("Testing the execution of parametrized constructor with : argument");
length = x;
breadth = x;
}
Rectangle(int x, int y) {
System.out.println("Testing the execution of parametrized constructor");
length = x;
breadth = y;
}
void area() {
int a = length * breadth;
System.out.println("The area of rectangle is " + a);
}
public static void main(String args[]) {
```

```
Rectangle r1 = new Rectangle();

Rectangle r2 = new Rectangle();

Rectangle r3 = new Rectangle(3, 12);

r3.area();

Rectangle r4 = new Rectangle(4);

}
```

OUTPUT:-

```
Testing the execution of constructor:
Testing the execution of constructor:
Testing the execution of parametrized constructor
The area of rectangle is 48
Testing the execution of parametrized constructor with : argument
```

EXPERIMENT NO. 8

AIM :- Program on abstract class and abstract methods.

1. Define an abstract shape class and have abstract method area. Extend this class into circle and rectangle.

THEORY :-

Abstraction in Java :-

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Abstract class in Java A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

o An abstract class must be declared with an abstract keyword.

o It can have abstract and non-abstract methods.

o It cannot be instantiated.

o It can have constructors and static methods also.

o It can have final methods which will force the subclass not to change the body of the method.

Abstract Method in Java :-

A method which is declared as abstract and does not have implementation is known as an abstract method. Example of abstract method abstract void printStatus();//no method body and abstract.

1. Define an abstract shape class and have abstract method area. Extend this class into circle and rectangle.

Program :-

```
import java.lang.*;
import java.util.*;
import java.io.*;
abstract class shape {
abstract void area();
}
class rectangle extends shape {
void area() {
Scanner a = new Scanner(System.in);
```

```
System.out.println("Enter a the length :");
int l = a.nextInt();
Scanner c = new Scanner(System.in);
System.out.println("Enter a breadth :");
int b = c.nextInt();
int area = l * b;
System.out.println("Area of rectangle :" + area);
}
}
class circle extends shape {
void area() {
Scanner d = new Scanner(System.in);
System.out.println("Enter a the radius :");
int r = d.nextInt();
int area = (22 / 7) * (r * r);
System.out.println("Area of circle is :" + area);
}
}
public class abstractshape {
public static void main(String args[]) {
System.out.println("RECTANCLE");
rectangle r1 = new rectangle();
r1.area();
System.out.println("CIRCLE");
circle c1 = new circle();
c1.area();
}
}
```

OUTPUT:-

```
RECTANCLE
Enter a the length :
4
Enter a breadth :
2
Area of rectangle :8
CIRCLE
Enter a the radius :
2
Area of circle is :12
```

EXPERIMENT NO. 9

AIM :- Program using super and final keyword.

1. Use final keyword with function, class, variable and study the difference when you try to change the value.

THEORY :-

Super Keyword in Java :-

The super keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword :-

1. Super can be used to refer immediate parent class instance variable. 2. Super can be used to invoke immediate parent class method.

3. super can be used to invoke immediate parent class constructor.

final Keyword in Java

The final keyword in java is used to restrict the user. The java final keyword can be used in many context.

Final can be:

1. variable

2. method

3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only.

1. Use final keyword with function, class, variable and study the difference when you try to change the value.

Program :-

```
import java.util.*;
class Person {
 String name;
 String DOB;
 Person() {
```

```
name = "SINGH SUDHAM DHARMENDRA";
DOB = "18/03/2002";
}
Person(String x, String y) {
name = x;
DOB = y;
}
void getdata() {
Scanner in = new Scanner(System.in);
System.out.println("Enter name: ");
name = in.nextLine();
System.out.println("Enter DOB: ");
DOB = in.nextLine();
}
void showdata() {
System.out.println("Name: " + name);
 System.out.println("DOB: " + DOB);
 }
}
class Student extends Person {
 int roll_no;
 Student() {
 super();
 roll_no = 20;
 }
 Student(String x, String y, int z) {
 super(x, y);
 roll_no = z;
 }
 void getdata() {
 Scanner in = new Scanner(System.in);
```

```
super.getdata();
System.out.println("Enter Roll No.: ");
roll_no = in.nextInt();
}
void showdata() {
super.showdata();
 System.out.println("Roll No.: " + roll_no);
 }
}
class Faculty extends Person {
 int faculty_ID;
 Faculty() {
 super();
 faculty_ID = 64;
 }
 Faculty(String x, String y, int z) {
 super(x, y);
 faculty_ID = z;
 }
 void getdata() {
 Scanner in = new Scanner(System.in);
 super.getdata();
 System.out.println("Enter Faculty ID: ");
 faculty_ID = in.nextInt();
 }
 void showdata() {
  super.showdata();
  System.out.println("Faculty ID: " + faculty_ID);
  }
}
public class Main {
```

```
public static void main(String[] args) {
Person p1 = new Person();
Person p2 = new Person("Jeff Bezos", "4/10/1975");
p1.showdata();
p2.showdata();
p1.getdata();
p1.showdata();
Student s1 = new Student();
Student s2 = new Student("Jeff Bezos", "4/10/1975", 5);
s1.showdata();
s2.showdata();
s1.getdata();
s1.showdata();
Faculty f1 = new Faculty();
Faculty f2 = new Faculty("Jeff Bezos", "4/10/1975", 63984);
f1.showdata();
 f2.showdata();
 f1.getdata();
 f1.showdata();
 }
}
```

OUTPUT:-

```
Name: Sudham
DOB: 26/06/2002
Name: atharv
DOB: 4/10/1975
Enter name:
sudham
Enter DOB:
22/06/2002
Name: sudham
DOB: 22/06/2002
Name: Sudham
DOB: 26/06/2002
Roll No.: 20
Name: atharv
DOB: 4/10/1975
Roll No.: 5
Enter name:
sudam
Enter DOB:
22/06/2002
Enter Roll No.:
50
Name: sudam
DOB: 22/06/2002
Roll No.: 50
Name: Sudham
DOB: 26/06/2002
Faculty ID: 64
Name: atharv
DOB: 4/10/1975
Faculty ID: 63984
Enter name:
```

Experiment no. 10

AIM :-Program on exception handling.

1. Write a simple division program using try and catch block to study different types of exceptions (Accept number input from user) Use finally block in same program.

THEORY :-

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

1. Write a simple division program using try and catch block to study different types of exceptions (Accept number input from user) Use finally block in same program.

Program :-

```
import java.util.*;
import java.lang.*;
public class division
{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
try
{
System.out.println("Enter 1st number: ");
int a = in.nextInt();
System.out.println("Enter 2st number: ");
int b = in.nextInt();
float c = 0;
c = (float)a / b;
System.out.println("Answer: " + c);
}
catch (ArithmeticException e)
{
System.out.println("Number cannot divide by zero\nTry some other value for Denominator.");
}
catch (InputMismatchException e)
{
System.out.println("Please enter only Integer values");
}
```

```
finally
{
System.out.println("Finally block");
}
}
}
```

OUTPUT:-

```
Enter 1st number:
1
Enter 2st number:
4
Answer: 0.25
Finally block
```

EXPERIMENT NO.11

AIM :- Program on user defined exception.

1. Throwing a user defined exception.

THEORY :-

In Java, we can create our own exception class and throw that Exception using ‘throw’ keyword. Those exceptions are known as user defined exceptions.

some reasons to use user-defined exception so catch and provide specific treatment to a subset of existing Java exceptions.

Business logic exceptions. These are the exceptions related to business logic and work flow. It is useful for the application uses of the developers to understand to exact problem.

1. Throwing a user defined exception.

Program :-

```
import java.util.*;
import java.lang.*;
public class exceptionh {
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
try
{
System.out.println("Enter 1st number: ");
int a = in.nextInt();
System.out.println("Enter 2st number: ");
int b = in.nextInt();
float c = 0;
c = (float) a / b;
if (b == 0)
{
throw new Exception("Denominator is zero");
}
else
{
c = (float) a / b;
System.out.println("the result is " + c);
}
```

```
}
catch (Exception e)
{
System.out.println(e);
}
}
}
```

OUTPUT:-

```
Enter 1st number:
4
Enter 2st number:
5
the result is 0.8
```

EXPERIMENT NO. 12

AIM :- Program on Multithreading.

THEORY :-

Multi threading in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. A thread is a lightweight subprocess, the smallest unit of processing. Multiprocessing and multi threading, both are used to achieve multitasking.

Program :-

```
class counter {
int count;
public synchronized void increment() {
count++;
}
}
public class multithreading {
public static void main(String[] args) throws InterruptedException {
```

```
counter c = new counter();
Thread t1 = new Thread(new Runnable() {
public void run() {
for (int i = 1; i <= 100000; i++) {
c.increment();
}
System.out.println(" Hello world ");
}
});
Thread t2 = new Thread(new Runnable() {
public void run() {
for (int i = 1; i <= 100000; i++) {
c.increment();
}
System.out.println(" Hello world ");
}
});
t1.start();
t2.start();
t1.join();
t2.join();
System.out.println(c.count + " is the count ");
}
}
```

OUTPUT:-

```
 Hello world
 Hello world
200000 is the count
```

EXPERIMENT NO. 13

AIM :- Program on applet class.

THEORY :-

Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Advantage of Applet

There are many advantages of applet. They are as follows:

o It works at client side so less response time.

o Secured

o It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

Plugin is required at client browser to execute applet java.awt.Component class

The Component class provides 1 life cycle method of applet.

public void paint(Graphics g):- is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.

java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

1.public void init():- is used to initialized the Applet. It is invoked only once.

2.public void start():- is invoked after the init() method or browser is maximized. It is used to start the Applet.

3.public void stop():- is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

4.public void destroy():- is used to destroy the Applet. It is invoked only once.

CODE :-

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class appp extends Applet implements ActionListener {
Button b;
TextField tf;
public void init() {
tf = new TextField();
tf.setBounds(30, 40, 150, 20);
b = new Button("Click");
b.setBounds(80, 150, 60, 50);
add(b);
add(tf);
b.addActionListener(this);
setLayout(null);
}
public void actionPerformed(ActionEvent e) {
tf.setText("Welcome");
}
} // EventAppletEx.html <html> <body> <applet code="EventApplet.class" width="300"
// height="300"> </applet> </body> </html>
```

OUTPUT:

Name!- Singh Sudham Dharmendra

Roll no!- AIML D-50

Branch :- CSE (AI & ML)

Subject :- OOPs (Skill Based lab Carse)
Object Oriented Programming with Java.

Topic !- Assignment 1

Date of Submission !- 20/10/2021

Signature :-

LO1 . To apply fundamental programming constructs.

(Q.1) What is object oriented programming? Elaborate different OOPs concept with suitable examples.

→ Object - oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of ~~both~~ such modules on demand. This means that an object is considered to be a partitioned area of computer memory that stores data and a set of operation that can access data.

Basic Concepts of OOPs :-

- Objects & Classes :-

① Objects are the basic runtime entities in an object oriented system. They may represent a person, a place, a bank account, or any item that the program may handle.
Any programming problem is analyzed in terms of objects & the nature of communication between them.

② A class may be thought of as a 'data type' and an object as a 'variable' of that data type.
A class is thus a collection of objects of similar type.

For example:- If fruit has been defined as a class then statement
fruit mango*
will create an object mango belonging to the class fruit.

• Data Abstraction & Encapsulation :-

① The wrapping up of data and method into a single unit (class) is known as encapsulation.

② The data is not accessible to the outside world and only those methods, which are wrapped in the class, can access it.

③ These methods provide the interface between the object's data and the program.

④ Abstraction refers to the act of a representing essential features without including the background details or explanation.

For example :-

An electronic device mobile phone used to send text message and voice communication. The user is not concerned about the internal circuitry of the mobile phone the only that he is concerned about is the fact that by pressing the green 'Answer key', he can talk with remote person.

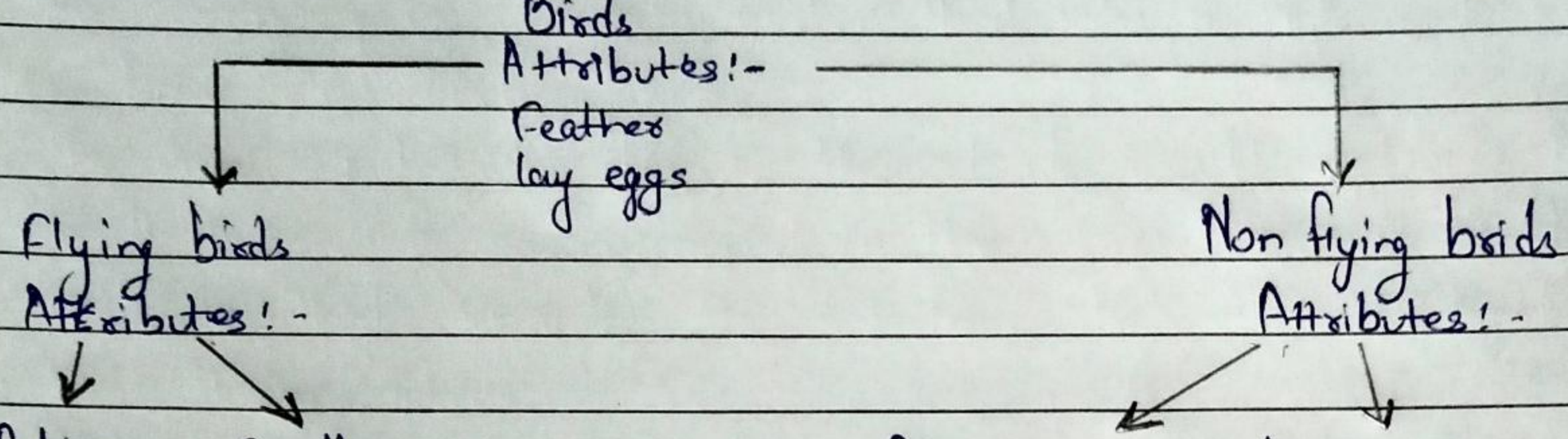
Standard operation that can be performed on mobile phone - dialing a call, receiving a call, sending and receiving the text messages, etc.

⑤ The common data elements that are found in all types of mobile phone ie; address book, inbox, etc,. We can refer to these set of data and operations as an abstraction of a generic mobile phone.

• Inheritance & Polymorphism :-

① Inheritance is the process by which object of one class acquire the properties of objects of another class.

For example!-



In OOPs, the concept of inheritance provides the idea of reusability.

② Polymorphism is another important OOP concept.
It means the ability to take more than one form.
For example!-
An operation may exhibit different behaviours in different instances. Consider, the operation of addition.
For two numbers, the operation will generate a sum.
If the operands are strings, then the operation would produce a third by concatenation.
Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface.

(Q.2) What is the Java Virtual Machine? Explain its significance.

→ Java Virtual Machine :-

Java Virtual Machine (JVM) is a program that interprets the intermediate Java byte code and generate the desired output. It is because of byte code & JVM concepts that programs written in Java are highly portable.

Significance :-

① JVM allows developers to write Java programs that are highly secure with help of its built-in security features

② A program that is cross platform has the capability to run successfully on different types of hardware.

③ Java is also a cross platform language ie, it is possible to run a single piece of code written on a particular hardware that has JVM installed on it.
Therefore, JVM makes Java a cross platform languages.

④ The Java Virtual Machine comes with a Just-in-time compiler which converts the Java code into a low level machine language than can run as fast as the regular applications.

(Q.3) Explain Various features of Java.

→ Java is not only reliable, portable and distributed but also simple, compact and interactive. The most striking feature of the language is that it is a platform-neutral language. Programs developed in Java can be executed anywhere on any system. Sun Microsystem officially describes Java with the following attributes:-

(1) Compiled & Interpreted.

(a) Java combines both compile and interpretation approaches thus making Java a two-stage system. Java compiler translates source code into what is known as byte code instructions

(b) Bytecodes are not machine instructions and therefore, in the second stage, Java interpreter generate machine code that can be directly executed by the machine that is running in the Java program.

(2) Platform - Independent & Portable:-

Java programs can be easily moved from one computer system to another, anywhere and anytime. Changes and upgrades in operating systems, processors and system resource will not force any changes in Java programs.

(3) Robust & Secure:-

(a) Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types.

(b) Security becomes an important issue for a language that is used for programming on Internet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

(4) Multithreaded and Interactive:-

(a) Multithreading means handling multiple tasks simultaneously. Java supports multithreading programs.

(b) Java runtime comes with tools that support multiprocess synchronization and construct smoothly running interactive systems.

⑤ Dynamic & Extensible :-

Java is a dyramic language. It is capable of linking in new class libraries, methods, and objects.
Java programs supports function written in other language such as C & C++. These functions are known as native methods. They are linked dynamically by runtime.

⑥ Object-oriented :-

Java is a true object-oriented language. Almost everything in Java is an object. All program code and data reside within objects & classes.
Java comes with an extensive sets of classes, arrangeed in packages, that we can use in our programs by inheritance.

⑦ Distributed :-

Java is designed as a distributed language for creating applications on network. It has the ability to share both data & programs.

LO2 To illustrate the concept of packages, classes & objects.

Q.1) Explain the concept of package in Java with suitable example.

→ Packages are Java's way of grouping a variety of classes and/or interface together. The grouping is usually done according to functionality. In fact, packages act as "containers" for classes.

Benefits of package organized classes :-

① Classes contained in the packages of other programs can be easily reused.

② In packages, classes can be unique compared with classes in other packages. That is, two classes in two different package can have the same name.

③ Packages provide a way to "hide" classes thus preventing other programs or packages from accessing classes that are meant for internal use only..

④ Packages also provide a way for separating "design" from "coding".

* Some Java packages and their classes :-

Package name		contents
a) java.lang	→	Support classes. Include classes for primitive types, strings, math functions, threads and exceptions.

Package name		content
b) java.util	→	language utility classes such as vectors, hash tables, random no., date, etc.
c) java.io	→	input/output support classes.
d) java.net	→	classes for networking. They include classes for communication with local computer as well as with internet access.
e) java.applet	→	classes for creating and implementing applets.
f) java.awt	→	set of classes for implementing graphical user interface.

(Q.2) What are different types of Java Input/Output functions?

→ Java brings various streams with its I/O package that helps the user to perform all the input output operations. These streams supports all the types of objects, data-types, characters, files, etc to fully execute the I/O operations.

Three standard or default streams are :-

(i) System.in :- This is the standard input stream that is used to read characters from the keyboard or any other standard input device.

(ii) System.out :- This is the standard output stream that is used to produce the result of a program on an output device like computer screen.

Various print functions that we use to output the statements :-

(a) print() ; This method in Java is used to display a text on the console.

(b) println() ; This method in Java is also used to display a text on the console.
It prints the text on the console and the cursor move to the start of the next line at the console.

(iii) System.err :- This is standered error stream that is used to output all the error data that a program might throw, on a computer screen or any standard output device.

This stream also uses all the mentioned functions to output the error data : print(), println(), etc

Q.3) What is a constructor? Explain constructor overloading with a suitable example.

→ ① Java supports a special type of method, called a constructor, that enables to intialize itself when it is created.

② Constructors have the same name as the class itself. Secondly, they do not specify a return type, not even void. This is because they return the instance of the class itself.

③ In Java, it is possible to create methods that have same name, but different parameter lists and different definitions. This is called method overloading.

(i) ~~Method~~ For example :-

```
Class Room
{
  float length;
  float breadth;
  Room (float x, float y)          // constructor 1
  {
    length = x;
    breadth = y;
  }
  Room (float x)                   // constructor 2
  {
    length = breadth = x;
  }
  int area()
  {
    return (length * breadth);
  }
}
```

Here, we are overloading the constructor method Room().
An object representing a rectangular room will be created as

```
Room room1 = new Room (25.0, 15.0);
                          // constructor 1
```

on the other hand, if the room is square, then;

```
Room room2 = new Room (20.0);
                          // constructor 2
```

Q.4) Compare the static & public methods in Java.

→

Static Method	Public Method
① Static members are associated with the class itself rather than individual objects	① A variable or method declared as public has the ~~weight~~ widest possible visibility and accessible everywhere.
② static variable are used when we want to have a variable common to all instances of a class.	② Public variable are used when we want to make any variable or method visible to all the classes outside the class.
static int count; static int max (int x, int y);	public int numbers; public void sum () (.....)
The members that are declared static are shown above are called static members.	The members shown above that are declared public are public members.

LO3 To elaborate the concept of Strings, arrays & Vectors.

Q.1) Explain various functions used to manipulate Strings.

→ String manipulation is the most common part of many Java programs. String represent a sequence of character. The string class defines a number of methods that allows us to accomplish a varity of string manipulation tasks.

(i) S2 = S1. toLowerCase; → Converts the string S1 to all lowercase.
(ii) S2 = S1. toUpperCase; → Converts the string S1 to all uppercase.
(iii) S2 = S1. replace ('x', 'y'); → Replace all appearance of x with y.
(iv) S2 = S1. trim(); → Remove white spaces at the beginning and end of the string S1.
(v) S1. equals (S2) → Return 'true' if S1 is equal to S2.
(vi) S1. equals Ignore Case (S2) → Returns 'true' if S1 = S2, ignoring the case of characters.
(vii) S1. length() → Gives the length of S1.
(viii) S1. charAt (n) → Given nth character of S1.
(ix) S1. compareTo (S2) → Returns negative if S1 < S2, positive if S1 > S2, and zero if S1 is equal S2.
(x) S1. concat (S2) → Concatenates S1 & S2
(xi) S1. substring (n) → Gives substring starting from n^{th} character
(xii) S1. sbstring (n,m) → Gives substring starting from n^{th} characters upto m^{th} (not including m^{th}).
(xiii) String. value of (P) → Creates a string object of the parameter P.
(xiv) P. to string() → Creates a string representation of the objectP
(xv) S1. index of ('x') → Gives the position of the first occurence of 'x'
(xvi) S1. index of ('x', n) → Gives the position of 'x' that occurs after n^{th} position in S1.

Q.2) Write note on the following :-

→ (i) Arrays :-

- A java array is a group of similar typed variables that use a shared name.
- When an array is created, the size of the array (or length) is also fixed.
- An array index always begins with 0. Each compartment has a numerical index that we use to access values.
- In java, there are a few different types of array that we can work with.
- A single dimensional array is a normal array that is used most often. This type of array contains sequential elements that are of the same type, such as a list of integers.
- A multidimensional array is an array of arrays. A two dimensional array is an array made up of multiple one dimensional arrays. A three dimensional array is an array made up of multiple two dimensional arrays.

Array declarations :-

type arrayname [ ];

for example :-

int numbers [ ];

we do not enter the size of array in declaration.

(ii) Vectors :-

- Vector class is contained in the java.util package
- It is used to create a generic dynamic array that can hold objects of any type and any number.
- The objects do not have to be homogenas.

Vectors are created like arrays as follows;

```
Vector intVect = new Vector ();
                    // declaring without size.

Vector list = new Vector (3);
                    // declaring with size.
```

Vector possess a number of advantages over arrays :-

① It is convenient to use vectors to store objects.
② A vector can be used to store a list of objects that may vary in size.
③ We can add and delete objects from the list as and when required.

.Important vector Methods :-

(a) list size () -> gives the number of the objects present.
(b) list.addElement (item) -> Adds the item specified to list at end
(c) list.elementAt (10) -> Gives the number of the 10th object.
(d) list.removeElement (item) -> Remove the specific item from the list.
(e) list.insertElementAt (item, n) -> Insert the element at n^{th} position.

Name :- Singh Sudham Dharmendra

Roll no :- AIML D - 50

Branch :- CSE (AI & ML)

Subject :- OOPs (Skill Based Lab Course)
Object Oriented Programming with Java.

Topic :- Assignment 2

Date of Submission :- 30/11/2021

Signature :-

LO4 - To implement the concept of inheritance and interfaces

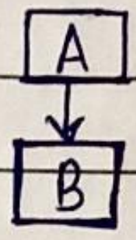
Q.1) Explain the various types of inheritance in Java.

→ On the basics of class, there can be three types of inheritance in Java : Single, Multilevel & hierarchical.

SINGLE INHERITANCE :-

a) When a class inherts another class, it is known as a Single inheritance.

b) There is only one super class

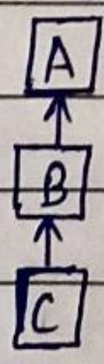


Single Inheritance

c) The diagram shows that class B extends only one class which is A. Here A is a parent class of B and B would be a child class of A.

MULTILEVEL INHERITANCE :-

a) In the Multilevel inheritance, a derived class will inherit a base class and as well as the derived class also act as the base class to other class.



Multilevel Inheritance

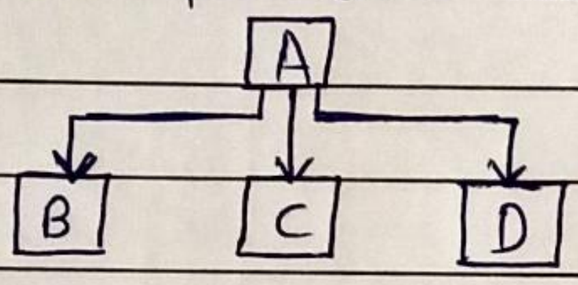
b) Here, in the diagram the class c inherits class B & class B inherits class A which means B is a parent class of c & A is a parent class of B.

c) Class C is implicitly inheriting the properties and methods of class A along with class B that's why it is called multi-level inheritance.

HIERARCHICAL INHERITANCE :-

a) When more than one classes inherit a same class then this is called hierarchical inheritance.

b) There is one super class and many subclass.



Hierarchical Inheritance

c) Class B, C & D extends a same class A, in the following dicg.

Q.2) Explain the super & final keywords in Java.

→ ① The subclass constructor uses the keyword super to invoke the constructor method of the Superclass. The keyword super is used subject to the following conditions.

- Super may only be used within a subclass constructor method.
- The call to superclass constructor must appear as the first statement within the subclass constructor.
- The parameter in the super call must match the order and type of the instance variable declared in the superclass.

② If we wish to prevent the subclasses from over-riding the members of the superclass, we can declare them as final using the keyword final as a modifier.

for example :-

```
final int SIZE = 100;
final void showstatus () {.....}
```

(3) Making the method final ensures that the functionality defined in this method will never be altered in any way.
(4) Similarly, the value of a final variable can never be changed.
(5) Final variables, behave like class variables and they ~~the~~ do not take any space on individual object of the class.
(6) Sometimes we may like to prevent a class being ~~fro~~ ~~subclassed~~ is called a final class. This is achieved in Java using the keyword final as follows :-

```
final class A class (.....)
final class B class extends someclass (.....)
```

(7) Any attempt to inherit ~~these~~ ~~place~~ classes will cause an error and the compiler will not allow it.
Declare a class final, prevents any unwanted ~~extend~~ extensions to the class.

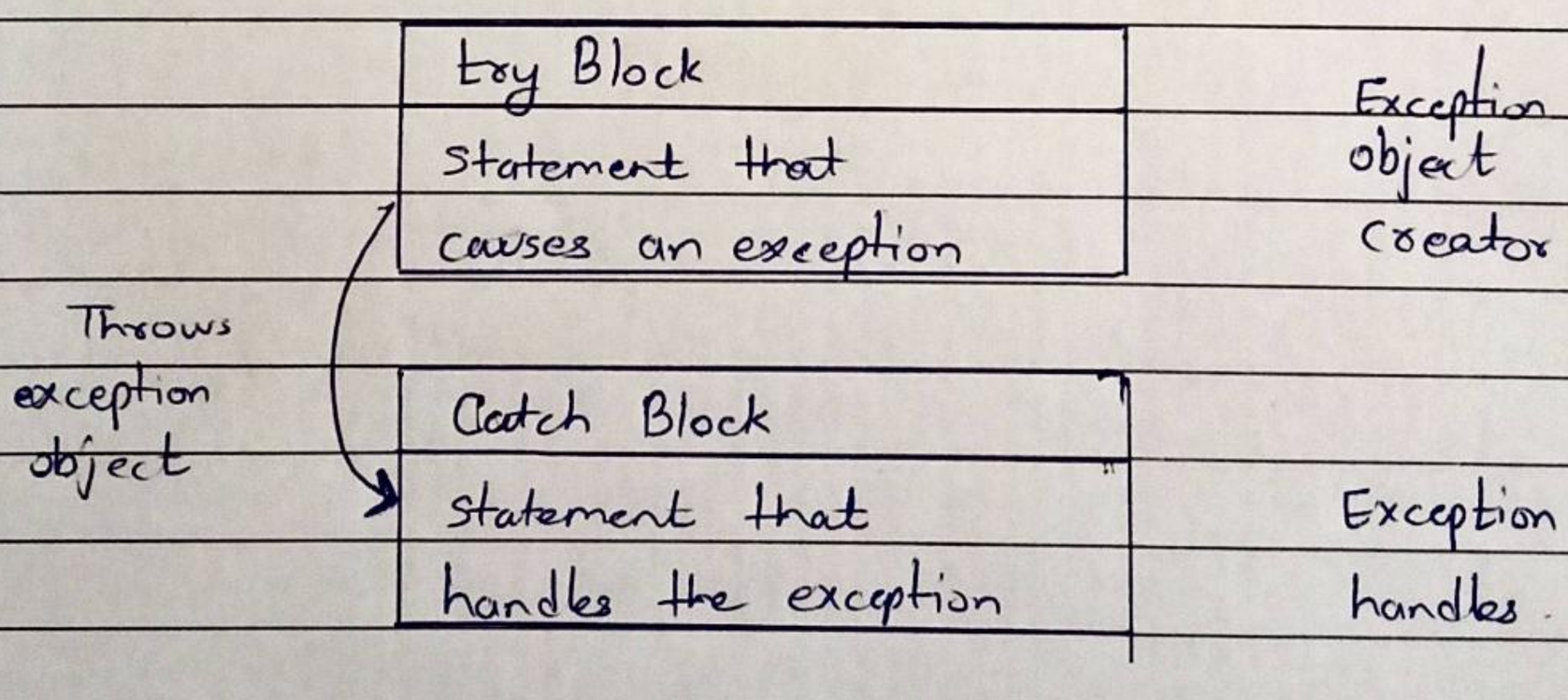
LO5 - To implement the concept of exception handling and multithreading.

Q.1) What do you mean by exception handling? Explain the use of try catch and finally in handling the exceptions.

→ ① An exception is a condition that is caused by a run-time error in the program. When the Java interpreter encounters an error such as diving an integer by zero, it creates an exception object and throws it (i.e. informs us that an error has occurred).

② If the exception objection is not caught and handled properly, the interpreter will display an error message and will terminated the program.

③ If we want the program to continue with the execution of the remaining code, then we should try to catch the exception object thrown by the error condition and then display an appropriate message for taking corrective actions. This task is known as exception handling.



Exception handling mechanism.

(4) Java uses a keyword try to preface a block of code likely to cause an error condition and "throw" an exception. A catch block defined by the keyword catch "catches" the exception "thrown" by the try block and handles it appropriately. The catch block is added immediately after the try block.

```
.....
.....
try
{
  statement ; // generates an exception
}
catch (Exception - type e)
{
  statement; // process the exception.
}
.....
.....
```

(5) Java supports another statement known as finally statement that can be used to handle an exception that is not caught by any of the previous catch statements, finally block can be used to handle any exception generated with in a try block.

(6) It may be added immediately after the try block or after the last catch block, as follows:

```
try
{
.....
.....
}
finally
{
.....
.....
}
```

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

Q.2) Differentiate between throw & throws.

→

throw	throws
① Java throw keyword is used to throw an exception in the code, inside the function or the block of code.	① Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
② Using throw keyword, we can only propagate unchecked exception ie, the exception cannot be propagated using throw only.	② Using throws keyword, we can declare both checked & unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.
③ The throw keyword is followed by an instance of exception to be thrown	③ The throws keyword is followed by class names of exceptions to be thrown.
④ throw is used within the method.	④ throws is used with the method signature.
⑤ We are allowed to throw only one exception at a time, ie, we cannot throw multiple exceptions	⑤ We can declare multiple exceptions using throws keyword that by the method. for eg, main() throws IO Exception, SQL Exception.

Singh Sudham Dharmendra
Roll no!- AIML050

Q.3) What is the significance of multithreading. Explain the thread lifecycle.

→ Multithreading in Java is a process of executing multiple threads simultaneously

Significance :-

a) It doesn't block the user because threads are independent and you can perform multiple operation at the same time.
b) You can perform many operations together, so it saves time.
c) Threads are independent, so it doesn't affect other threads if an exception occurs in a single threads.

A thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution.

LIFE CYCLE :

In Java, a thread always exists in any one of the following states. These states are :-

1. New
2. Active
3. Blocked / Waiting
4. Timed Waiting
5. Terminated

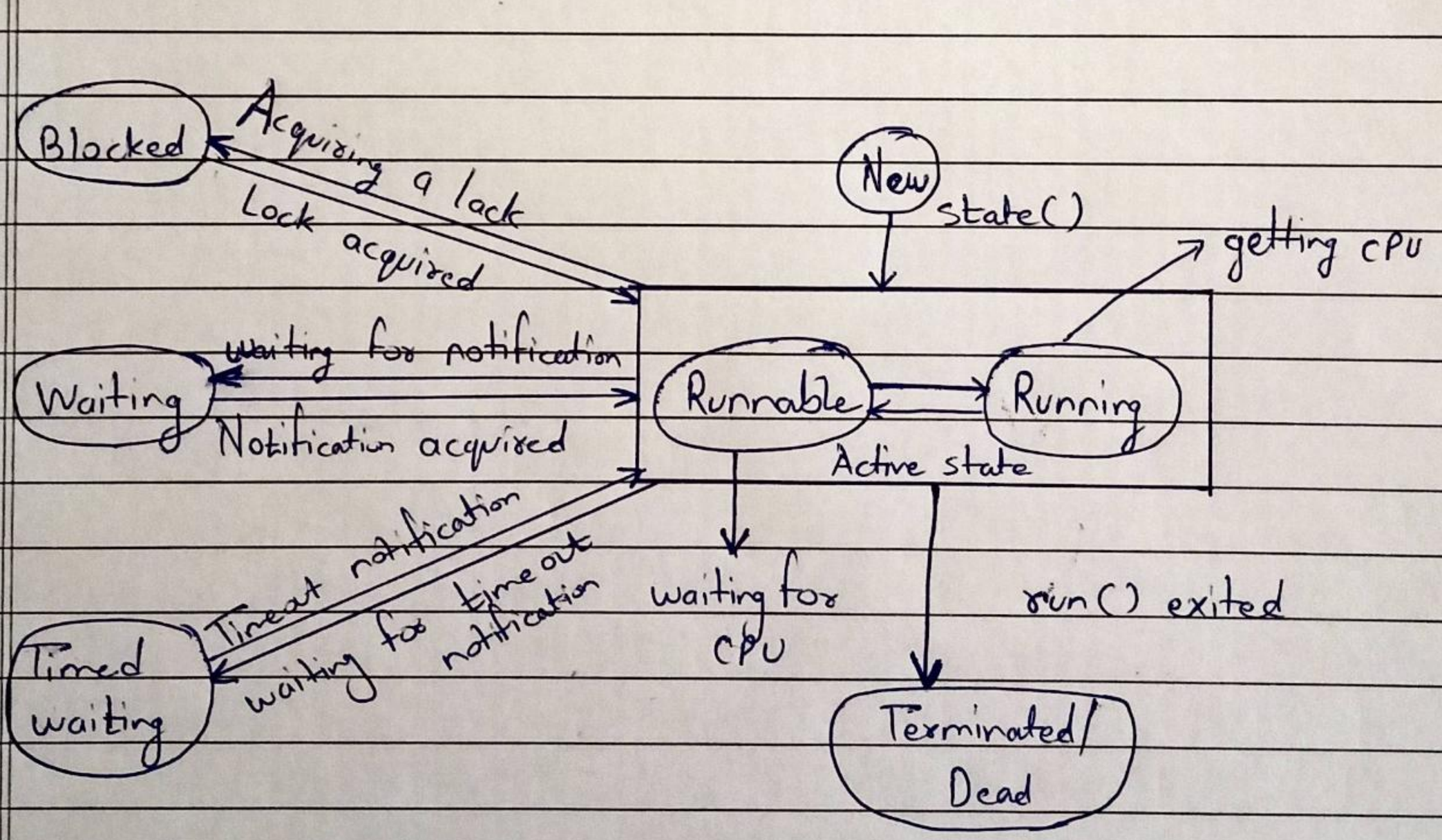
1. New :- Whenever a new thread is created, it is always in the new state. For a thread in the new state, the code has not been run yet and thus has not begun its execution.
2. Active :- When a thread invokes the start() method, it moves from the new state to the active state. It contains two state :
one is runnable, and the other is running.

3. Blocked or Waiting :- Whenever a thread is inactive for a span of time (not permanently) then, either the thread is in the blocked state or is in the waiting state.

4. Time waiting :- Time waiting is used when we invoke the sleep () method on a specific thread. The sleep () method puts the thread in the timed wait state. After the time runs out, the thread wakes up to start its execution from when it has left earlier.

5. Terminated :- A thread reaches the termination state because of the following reasons :-

- When a thread has finished its job, then it exists or terminate normally.
- Abnormal termination : It occurs when some unusual events such as an unhandled exception or segmentation fault.



Life Cycle of a Thread

Lo6 - To develop GUI based application

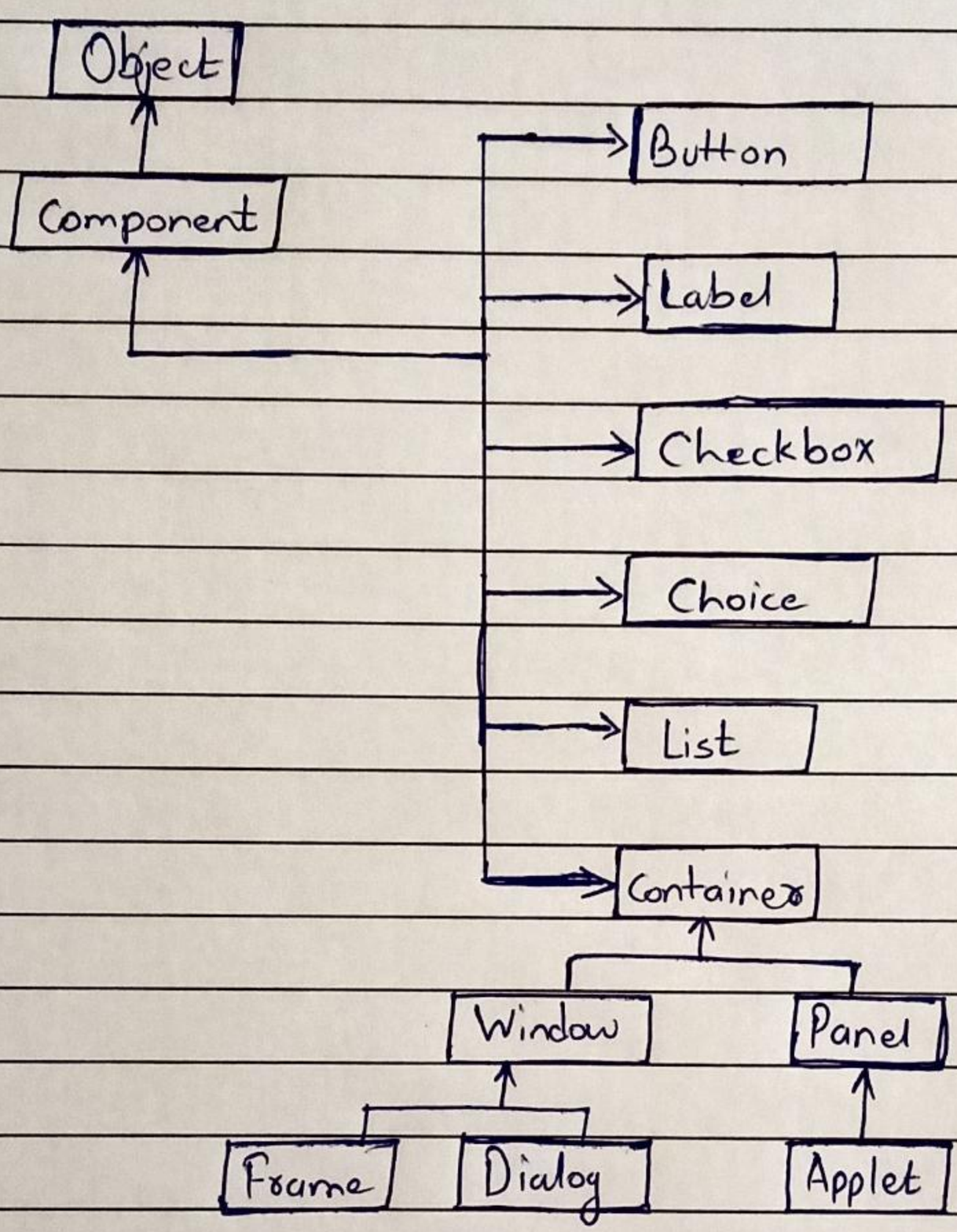
Q.1) What is Swing class in JAVA ?

→ ① Java Swing is part of JAVA foundation classes.
It is used to create window-based application which makes it suitable for developing light-weight desktop applications.
② Java swing is built on top of an abstract windowing toolkit API purely written in JAVA programming language.
③ Java swing Provide lightweight and platform-independent components, making it suitable and efficient in designing and developing desktop based applications (system)
④ Swing components are contained in the javax.swing package.

Q.2) Explain AWT. Draw Java AWT Hierarchy.

→ ① Java AWT (Abstract window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based application in Java.
② Java AWT components are platform-development ie; components are displayed according to the view of operation system.
③ AWT is heavy weight ie; its component are using the resources of underlying operating system (OS).
④ The java-awt package provides classes for AWT API such as Textfield, label, TextArea, Radiobutton, Checkbox, Choice, List etc.
⑤ The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.
⑥ Java AWT calls the native platform calls the native platform (operating System) subroutine for creating API components like Textfield, checkbox, button, etc.

* JAWA AWT HIERARCHY :-

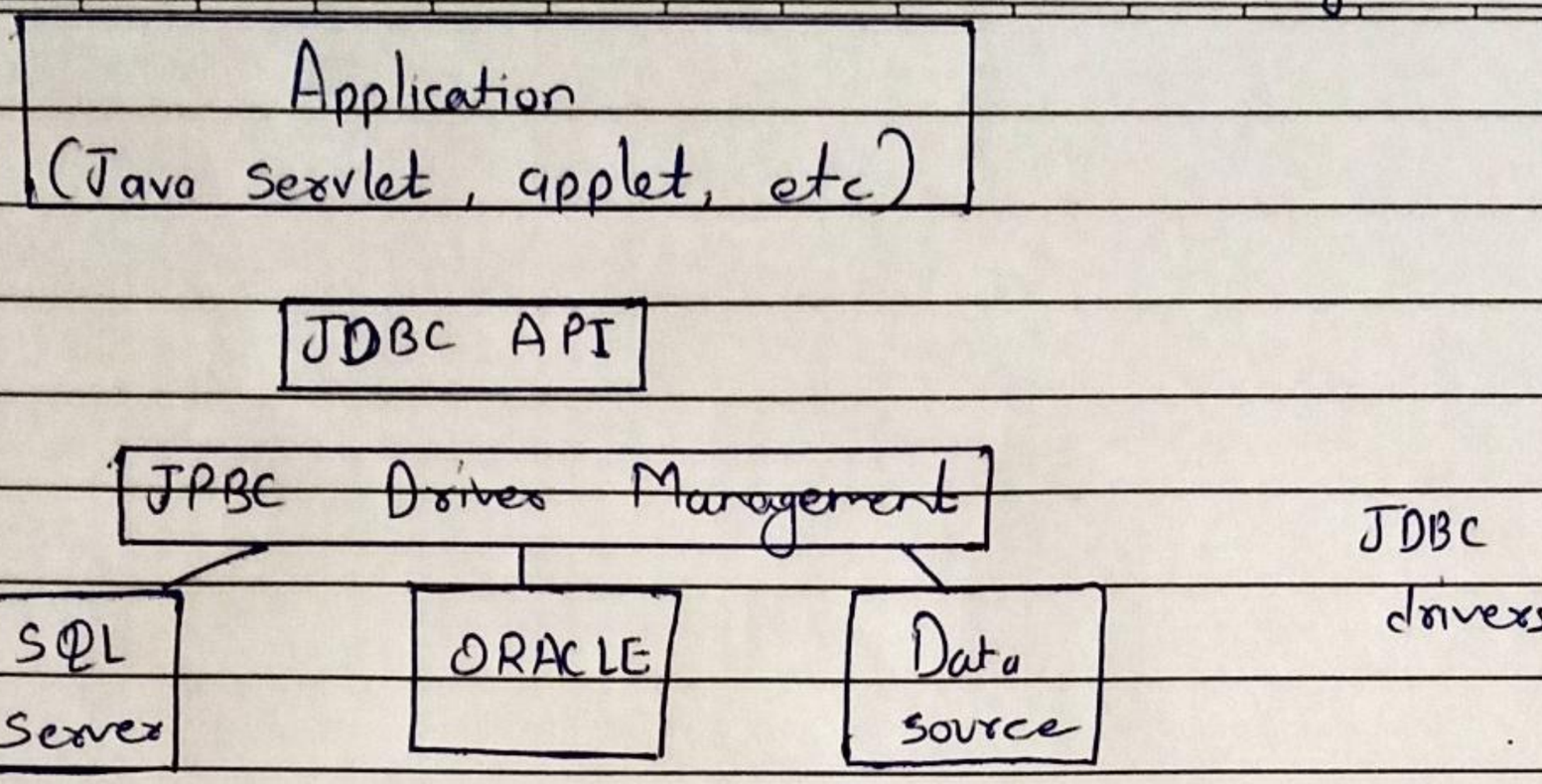


Q.3) Explain JDBC with architectural diagram.

→ JDBC (Java Database Connectivity) is an API (Application progamming interface) which is used in java programming to interact with databases.

The classes and interfaces of JDBC allows application to send request made by users to the specified database.

Singh Sudham Dharmendra
Roll no:- AIMLD50



Architecture of JDBC.

1. Application :-

It is a java applet or a servlet which communicates with a data source.

2. The JDBC API :-

The JDBC API allows Java programs to excute SQL statements and retrieve results. Some of the important classes & interfaces defined in JDBC API are as follows:-
Driver manager, Drivers, Connection, statement, Prepared statement, callable statement, Resultset, SQL data.

3. Drivers Manager :-

It plays an important role in the JDBC architecture. It uses some database - specific drivers to effectively connect enterprise application to database.

4. JDBC drivers :-

To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

CURRENCY CONVERTOR

Submitted in partial fulfillment of the requirements of the degree

**BACHELOR OF ENGINEERING **

in

Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

Sem - III

by

AIMLD50_SINGH SUDHAM DHARMENDRA

AIMLD41_RUPARELIYA AKSHAR JAYANTI

Supervisor

Prof. Dr. Monika Mangla

Lokmanya Tilak College of Engineering

Koparkhairne, Navi Mumbai - 400 709

University of Mumbai

(AY 2021-22)

ABSTRACT:

- An easily accessible online currency converter is very useful to show travelers how their own currencies will fare when exchanged with other foreign currency.
- Moreover, currency converters help international import and export businesses by helping them determine the selling and buying profits of different products.
- Conversion from one currency to another is a very important endeavor especially when it comes to marketing and travel.
- Currency conversion system is implemented to reduce human power to automatically recognize the amount monetary value of currency and convert it into the other currencies without human supervision.
- The software interface that we are proposing here could be used for various currencies (we are using four in our project).

CODE:

Main.java

Run

Output

Clear

```java
import java.util.*;
import java.text.DecimalFormat;

class CurrencyConverter {
    public static void main(String[] args) {
        int i = 1;
        while (i == 1) {
        double rupee, dollar, pound, euro, yen;
        int code;
        DecimalFormat f = new DecimalFormat("##.###");
        Scanner sc = new Scanner(System.in);
        System.out.println("\nEnter the currency code:\n1:Rupees\n2:Dollar\n3:Pound\n4:Euro\n5:Yen"
            );
        code = sc.nextInt();
        switch (code) {
            case 1: {
                System.out.println("Enter amount in rupees");
                rupee = sc.nextFloat();
                dollar = rupee / 75.432;
                System.out.println("Dollar : " + f.format(dollar));
                pound = rupee / 99.49;
                System.out.println("Pound : " + f.format(pound));
                euro = rupee / 85.22;
                System.out.println("Euro : " + f.format(euro));
                yen = rupee / 0.66;
                System.out.println("Yen : " + f.format(yen));

                break;
            }
            case 2: {
                System.out.println("Enter amount in dollar");
                dollar = sc.nextFloat();
                rupee = dollar * 75.432;
                System.out.println("Rupees : " + f.format(rupee));
                pound = dollar * 0.76;
                System.out.println("Pound : " + f.format(pound));
                euro = dollar * 0.88;
```

Main.java

Run

Output

Clear

```java
                euro = dollar * 0.88;
                System.out.println("Euro : " + f.format(euro));
                yen = dollar * 113.88;
                System.out.println("Yen : " + f.format(yen));
                break;
            }
            case 3: {
                System.out.println("Enter amount in Pound");
                pound = sc.nextFloat();
                rupee = pound * 99.55;
                System.out.println("Rupees : " + f.format(rupee));
                dollar = pound * 1.32;
                System.out.println("Dollar : " + f.format(dollar));
                euro = pound * 1.17;
                System.out.println("Euro : " + f.format(euro));
                yen = pound * 150.24;
                System.out.println("Yen : " + f.format(yen));
                break;
            }
            case 4: {
                System.out.println("Enter amount in Euro");
                euro = sc.nextFloat();
                rupee = euro * 85.22;
                System.out.println("Rupees : " + f.format(rupee));
                dollar = euro * 1.13;
                System.out.println("Dollar : " + f.format(dollar));
                pound = euro * 0.86;
                System.out.println("Pound : " + f.format(pound));
                yen = euro * 128.69;
                System.out.println("Yen : " + f.format(yen));
                break;
            }
            case 5: {
                System.out.println("Enter amount in Yen");
            yen = sc.nextFloat();
            rupee = yen * 0.66;
            System.out.println("Rupees : "+f.format(rupee));
```

Main.java

Run

Output

Clear

```
}
case 4: {
    System.out.println("Enter amount in Euro");
    euro = sc.nextFloat();
    rupee = euro * 85.22;
    System.out.println("Rupees : " + f.format(rupee));
    dollar = euro * 1.13;
    System.out.println("Dollar : " + f.format(dollar));
    pound = euro * 0.86;
    System.out.println("Pound : " + f.format(pound));
    yen = euro * 128.69;
    System.out.println("Yen : " + f.format(yen));
    break;
}
case 5: {
    System.out.println("Enter amount in Yen");
yen = sc.nextFloat();
    rupee = yen * 0.66;
    System.out.println("Rupees : "+f.format(rupee));
    dollar = yen * 0.0088;
    System.out.println("Dollar : "+f.format(dollar));
    pound = yen * 0.0067;
    System.out.println("Pound : "+f.format(pound));
    euro = yen * 0.0078;
    System.out.println("Euro : "+f.format(euro));
    break;
    }

    default:
        System.out.println("you have Entered invalid choice:");
        System.out.println("please enter valid choice!!!");
        break;
    }
  }

}
}
```

OUTPUT:

Output Clear

```
java -cp /tmp/oxJThIeQrH CurrencyConverter
Enter the currency code:
1:Rupees
2:Dollar
3:Pound
4:Euro
5:Yen
1
Enter amount in rupees
5555
Dollar : 73.642
Pound : 55.835
Euro : 65.184Yen : 8416.667Enter the currency code:
1:Rupees
2:Dollar
3:Pound
4:Euro
5:Yen2
Enter amount in dollar75.1
Rupees : 5664.943
Pound : 57.076
Euro : 66.088
Yen : 8552.388

Enter the currency code:
1:Rupees
2:Dollar
3:Pound
4:Euro
5:Yen
```