

PART-I

Data Types, Variables, Arrays, Operators, Control Statements, String

1. Introduction to OOP concepts. Comparison of java with other OOP programming languages.
Intro. To JRE , JVM , JDK , JAVADOC , Command Line Argument.

ANS.

List of OOP Concepts in Java :

There are four main OOP concepts in Java. These are:

- **Abstraction.** Abstraction means using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works in order to enjoy it. In Java, abstraction means simple things like **objects**, **classes**, and **variables** represent more complex underlying code and data. This is important because it lets avoid repeating the same work multiple times.
- **Encapsulation.** This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself. This way, we can re-use objects like code components or variables without allowing open access to the data system-wide.
- **Inheritance.** This is a special feature of Object Oriented Programming in Java. It lets programmers create new classes that share some of the attributes of existing classes. This lets us build on previous work without reinventing the wheel.
- **Polymorphism.** This Java OOP concept lets programmers use the same word to mean different things in different contexts. One form of polymorphism in Java is **method overloading**. That's when different meanings are implied by the code itself. The other form is **method overriding**. That's when the different meanings are implied by the values of the supplied variables. See more on this below.

Comparison of java with other OOP programming languages:

A Comparison Table:								
	Eiffel	Smalltalk	Ruby	Java	C++	Python	Perl	Visual Basic
Encapsulation/Information Hiding	Yes	Yes	Yes	Yes	Yes	No	Yes?	Yes?
Inheritance	Yes	Yes	Yes	Yes	Yes	Yes	Yes?	No
Polymorphism/Dynamic Binding	Yes	Yes	Yes	Yes	Yes	Yes	Yes?	Yes (through delegation)
All pre-defined types are Objects	Yes	Yes	Yes	No	No	Yes	No	No
All operations are messages to Objects	Yes	Yes	Yes	No	No	No	No	No
All user-defined types are Objects	Yes	Yes	Yes	Yes	No	Yes	No	No

JAVA PROGRAMMING (Java Programs)

Command line argument in Java :

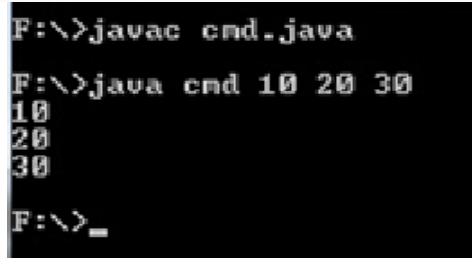
The command line argument is the argument passed to a program at the time when you run it. To access the command-line argument inside a java program is quite easy, they are stored as string in **String** array passed to the args parameter of **main()** method.

Example

```
class cmd
{
    public static void main(String[] args)
    {
        for(int i=0;i< args.length;i++)
        {
            System.out.println(args[i]);
        }
    }
}
```

Execute this program as **java cmd 10 20 30**

10
20
30



```
F:\>javac cmd.java
F:\>java cmd 10 20 30
10
20
30
F:\>_
```

Java Virtual machine (JVM) : is the virtual machine that runs the Java bytecodes. You get this bytecode by compiling the **.java** files into **.class**files. **.class** files contain the bytecodes understood by the JVM.

The **Java Runtime Environment (JRE)** : is a software package which bundles the libraries (jars) and the Java Virtual Machine, and other components to run applications written in the Java. JVM is just a part of JRE distributions.

JDK is a superset of JRE. : JDK contains everything that JRE has along with development tools for developing, debugging, and monitoring Java applications. You need JDK when you need **to develop Java applications**.

JAVADOC : **Javadoc** (originally cased **JavaDoc**) is a documentation generator created by Sun Microsystems for the **Java** language (now owned by Oracle Corporation) for generating API

JAVA PROGRAMMING (Java Programs)

documentation in HTML format from **Java** source code. ... **Javadoc** does not affect performance in **Java** as all comments are removed at compilation time.

2. Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz".

startOz("ozymandias") → "oz"

startOz("bzoo") → "z"

startOz("oxx") → "o"

CODE:

```
package pra.pkg2;
import java.util.*;
public class Pra2
{
    public static void main(String[] args)
    {
        String s;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter String : ");
        s = in.nextLine();
        String ans = startOz(s);
        System.out.println(ans);
    }
    static public String startOz(String s)
    {
        if((s.charAt(0)=='o' || s.charAt(0)=='O') && (s.charAt(1)=='z' || s.charAt(1)=='Z'))
        {
            return s.substring(0,2);
        }
        else if((s.charAt(0)=='o' || s.charAt(0)=='O'))
        {
            return s.substring(0,1);
        }
        else if((s.charAt(1)=='z' || s.charAt(1)=='Z'))
        {
            return s.substring(1,2);
        }
        else
        {
            return "String is not valid";
        }
    }
}
```

OUTPUT:

```
run:
Enter String :
ozymandias
oz
BUILD SUCCESSFUL (total time: 5 seconds)
```

JAVA PROGRAMMING (Java Programs)

3. Given two non-negative int values, return true if they have the same last digit, such as with 27 and 57. Note that the % "mod" operator computes remainders, so 17 % 10 is 7.

lastDigit(7, 17) → true

lastDigit(6, 17) → false

lastDigit(3, 113) → true

CODE:

```
package pra3;
import java.util.Scanner;
public class Pra3
{
    public static void main(String[] args)
    {
        int a,b;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter 2 integer values : ");
        a = in.nextInt();
        b = in.nextInt();
        boolean ans = lastdigit(a,b);
        System.out.println(ans);
    }
    public static boolean lastdigit(int a,int b)
    {
        int t1,t2;
        t1 = a%10;
        t2 = b%10;
        if(t1==t2)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

OUTPUT:

```
Enter 2 integer values :
34
54
true
BUILD SUCCESSFUL (total time: 3 seconds)
```

4. Given an array of ints , return true if the sequence of numbers 1, 2, 3 appears in the array somewhere.

array123([1, 1, 2, 3, 1]) → true

array123([1, 1, 2, 4, 1]) → false

JAVA PROGRAMMING (Java Programs)

array123([1, 1, 2, 1, 2, 3]) → true

CODE:

```
package pra4;
import java.util.*;
public class Pra4
{
    public static void main(String[] args)
    {
        int length;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter length of array : ");
        length = in.nextInt();
        int []a = new int[length];
        System.out.println("Enter values for int array ");
        for(int i=0;i<length;i++)
        {
            int temp = in.nextInt();
            a[i] = temp;
        }
        boolean ans = array123(a);
        System.out.println(ans);
    }
    public static boolean array123(int a[])
    {
        for(int i=0;i<a.length;i++)
        {
            if(a[i]==1 && a[i+1]==2 && a[i+2]==3)
            {
                return true;
            }
        }
        Return false;
    }
}
```

OUTPUT:

```
Enter length of array :
6
Enter values for int array
1 2 3 4 5 6
true|
```

5. Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings.

stringMatch("xxcaazz", "xxbaaz") → 3

stringMatch("abc", "abc") → 2

stringMatch("abc", "axc") → 0

JAVA PROGRAMMING (Java Programs)

CODE:

```
package pra5;
import java.util.*;
public class Pra5
{
    public static void main(String[] args)
    {
        String a,b;
        Scanner in = new Scanner (System.in);
        a = in.nextLine();
        b = in.nextLine();
        int ans = stringMatch(a,b);
        System.out.println(ans);
    }
    public static int stringMatch(String a , String b)
    {
        int temp=0;
        int l1=a.length();
        int l2=b.length();
        if(l1<l2)
        {
            for(int i=0;i<l1-1;i++)
            {
                if((a.charAt(i)==b.charAt(i)) && (a.charAt(i+1)==b.charAt(i+1)))
                {
                    temp++;
                }
            }
        }
        else
        {
            for(int i=0;i<l2-1;i++)
            {
                if((a.charAt(i)==b.charAt(i)) && (a.charAt(i+1)==b.charAt(i+1)))
                {
                    temp++;
                }
            }
        }
        return temp;
    }
}
```

OUTPUT:

```
ABCD
ABCD
3
```

6. Given an array of strings, return a new array without the strings that are equal to the target string. One approach is to count the occurrences of the target string, make a new array of the correct length, and then copy over the correct strings.
wordsWithout(["a", "b", "c", "a"], "a") → ["b", "c"]

JAVA PROGRAMMING (Java Programs)

wordsWithout(["a", "b", "c", "a"], "b") → ["a", "c", "a"]

wordsWithout(["a", "b", "c", "a"], "c") → ["a", "b", "a"]

CODE:

```
package pra6;
import java.util.Arrays;
import java.util.*;
public class Pra6
{
    public static void main(String []args)
    {
        int length;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter length of string array : ");
        length = in.nextInt();
        String []s = new String[length+1];
        System.out.println("Enter values for String array : ");
        for(int i=0;i<s.length;i++)
        {
            String temp = in.nextLine();
            s[i] = temp;
        }
        System.out.println("Original Array :"+Arrays.toString(s));
        String remove;
        System.out.println("Enter value of string which you want to remove : ");
        remove = in.nextLine();
        String arr[] = wordsWithout(s,remove);
        System.out.print("Final Array : ");
        for(int i=0;i<arr.length;i++)
        {
            System.out.println(arr[i]);
        }
    }
    static public String[] wordsWithout(String s[],String remove)
    {
        int nullcounter=0;
        int temp=0;
        for(int i=0;i<s.length;i++)
        {
            if(s[i].equals(remove))
            {
                s[i]=null;
                nullcounter++;
            }
        }
        String []newarray = new String[s.length-nullcounter];
        for(int i=0;i<s.length;i++)
        {
            if(s[i] != null)
            {
```

JAVA PROGRAMMING (Java Programs)

```
        newarray[temp] = s[i];
        temp++;
    }
}
return newarray;
}
}
```

OUTPUT:

```
Enter length of string array :
3
Enter values for String array :
PRATIK
POP
LOL
Original Array :[, PRATIK, POP, LOL]
Enter value of string which you want to remove :
LOL
Final Array :
PRATIK
POP
```

7. Display numbers in a pyramid pattern.

```

      1
    1 2 1
  1 2 4 2 1
1 2 4 8 4 2 1
  1 2 4 8 16 8 4 2 1
1 2 4 8 16 32 16 8 4 2 1
  1 2 4 8 16 32 64 32 16 8 4 2 1
    1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
```

CODE:

```
package pra7;
import java.util.*;
import java.text.DecimalFormat;
public class Pra7
{
    public static void main(String[] args)
    {
        int row;
        DecimalFormat f = new DecimalFormat("#");
        Scanner in = new Scanner(System.in);
        System.out.print("Enter no of row : ");
        row = in.nextInt();
        int odd=1;
        int noOfSpaces=row-1;
        for(int i=1;i<=row;i++)
        {
            for(int l=1;l<=noOfSpaces;l++) //this loop is printing spaces
            {
                System.out.print("  ");
            }

```


JAVA PROGRAMMING (Java Programs)

```
int k=1; //this variabe is for printing numbers in pattern
for(int j=1;j<=odd;j++)
{
    if(j==1 || j==odd)
    {
        System.out.format("  1");
        continue;
    }
    else if(j<=i)          //for printing no in given pattern
    {
        k=k*2;
    }
    else
    {
        k=k/2;
    }
    System.out.format("%4d",k);

}
odd = odd + 2;
System.out.println();
noOfSpaces = noOfSpaces - 1;
}
}
```

OUTPUT:

Enter no of row : 10

```

              1
            1 2 1
          1 2 4 2 1
        1 2 4 8 4 2 1
      1 2 4 8 16 8 4 2 1
    1 2 4 8 16 32 16 8 4 2 1
  1 2 4 8 16 32 64 32 16 8 4 2 1
1 2 4 8 16 32 64 128 64 32 16 8 4 2 1
```

8. The problem is to write a program that will grade multiple-choice tests. Assume there are eight students and ten questions, and the answers are stored in a twodimensional array. Each row records a student's answers to the questions, as shown in the following array.

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

JAVA PROGRAMMING (Java Programs)

The key is stored in a one-dimensional array:

Key to the Questions:

0 1 2 3 4 5 6 7 8 9

Key D B D C C D A E A D

Your program grades the test and displays the result. It compares each student's answers with the key, counts the number of correct answers, and displays it.

CODE:

```
package pra8;

import java.util.*;

public class Pra8
{
    public static void main(String[] args)
    {
        char [][] ans = new char[8][10];
        Scanner in = new Scanner (System.in);
        for(int i=0;i<ans.length;i++)
        {
            System.out.println("Enter ans for 10 question : STUDENT "+i);
            for(int j=0;j<ans[0].length;j++)
            {
                ans[i][j] = in.next().charAt(0);
            }
        }
        char []key = {'d','b','d','c','c','d','a','e','a','d'};
        System.out.println("Original Key is : "+Arrays.toString(key));
        int []grade = new int[8];
        grade = calculate(ans,key);
        System.out.println("Final grade for all students is : "+Arrays.toString(grade));
    }

    public static int[] calculate(char [][]ans , char []key)
    {
        int i=0;
        int []marks = new int[ans.length];
        while(i<8)
```

JAVA PROGRAMMING (Java Programs)

```
{
    for(int j=0;j<ans[0].length;j++)
    {
        if(ans[i][j]==key[j])
        {
            marks[i]++;
        }
    }
    i++;
}
return marks;
}
```

OUTPUT:

```
Enter ans for 10 question : STUDENT 0
a b c d d c b a a b
Enter ans for 10 question : STUDENT 1
a a a b c a c b c d
Enter ans for 10 question : STUDENT 2
a b c d d c b a a b
Enter ans for 10 question : STUDENT 3
a b c d d c b a a b
Enter ans for 10 question : STUDENT 4
a b c d d c b a a b
Enter ans for 10 question : STUDENT 5
a b c d d c b a a b
Enter ans for 10 question : STUDENT 6
a b c d d c b a a b
Enter ans for 10 question : STUDENT 7
d b d c c d a e a d
Original Key is : [d, b, d, c, c, d, a, e, a, d]
Final grade for all students is : [2, 2, 2, 2, 2, 2, 2, 10]
```

9. The problem is to check whether a given Sudoku solution is correct.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

(a) Puzzle

Solution →

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b) Solution

CODE:

```
import java.util.*;
```

```
class Sudoku {
```

Pratik Dhoriyani

GitHub : <https://github.com/pratikdhoriyani>

JAVA PROGRAMMING (Java Programs)

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int arr[][] = {{5, 3, 0, 0, 7, 0, 0, 0, 0},
                   {6, 0, 0, 1, 9, 5, 0, 0, 0},
                   {0, 9, 8, 0, 0, 0, 0, 6, 0},
                   {8, 0, 0, 0, 6, 0, 0, 0, 3},
                   {4, 0, 0, 8, 0, 3, 0, 0, 1},
                   {7, 0, 0, 0, 2, 0, 0, 0, 6},
                   {0, 6, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 4, 1, 9, 0, 0, 5},
                   {0, 0, 0, 0, 8, 0, 0, 7, 9}};

    System.out.println("Solve the given sudoku : ");
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            System.out.print(arr[i][j] + " ");
        }
        System.out.println();
    }

    int solution_arr[][] = {{5, 3, 4, 6, 7, 8, 9, 1, 2},
                           {6, 7, 2, 1, 9, 5, 3, 4, 8},
                           {1, 9, 8, 3, 4, 2, 5, 6, 7},
                           {8, 5, 9, 7, 6, 1, 4, 2, 3},
                           {4, 2, 6, 8, 5, 3, 7, 9, 1},
                           {7, 1, 3, 9, 2, 4, 8, 5, 6},
                           {9, 6, 1, 5, 3, 7, 2, 8, 4},
                           {2, 8, 7, 4, 1, 9, 6, 3, 5},
                           {3, 4, 5, 2, 8, 6, 1, 7, 9}};

    System.out.println("Solution of sudoku : ");
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            System.out.print(solution_arr[i][j] + " ");
        }
        System.out.println();
    }

    int row_sum[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
    int col_sum[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
    int box_sum[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            row_sum[i] += solution_arr[i][j];
            col_sum[j] += solution_arr[j][i];
        }
    }

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            box_sum[0] += solution_arr[i][j];
        }

        for (int j = 3; j < 6; j++) {
            box_sum[1] += solution_arr[i][j];
        }
    }
```

```
        for (int j = 6; j < 9; j++) {  
            box_sum[2] += solution_arr[i][j];  
        }  
    }
```

```
for (int i = 3; i < 6; i++) {  
    for (int j = 0; j < 3; j++) {  
        box_sum[3] += solution_arr[i][j];  
    }  
}
```

```
for (int j = 3; j < 6; j++) {  
    box_sum[4] += solution_arr[i][j];  
}
```

```
for (int j = 6; j < 9; j++) {  
    box_sum[5] += solution_arr[i][j];  
}  
}
```

```
for (int i = 6; i < 9; i++) {  
    for (int j = 0; j < 3; j++) {  
        box_sum[6] += solution_arr[i][j];  
    }  
}
```

```
for (int j = 3; j < 6; j++) {  
    box_sum[7] += solution_arr[i][j];  
}
```

```
for (int j = 6; j < 9; j++) {  
    box_sum[8] += solution_arr[i][j];  
}  
}
```

```
int flag = 0;  
for (int i = 0; i < 9; i++) {  
    if (row_sum[i] == 45) {  
        flag = 1;  
    } else {  
        flag = 0;  
        break;  
    }  
}
```

```
if (col_sum[i] == 45) {  
    flag = 1;  
} else {  
    flag = 0;  
    break;  
}
```

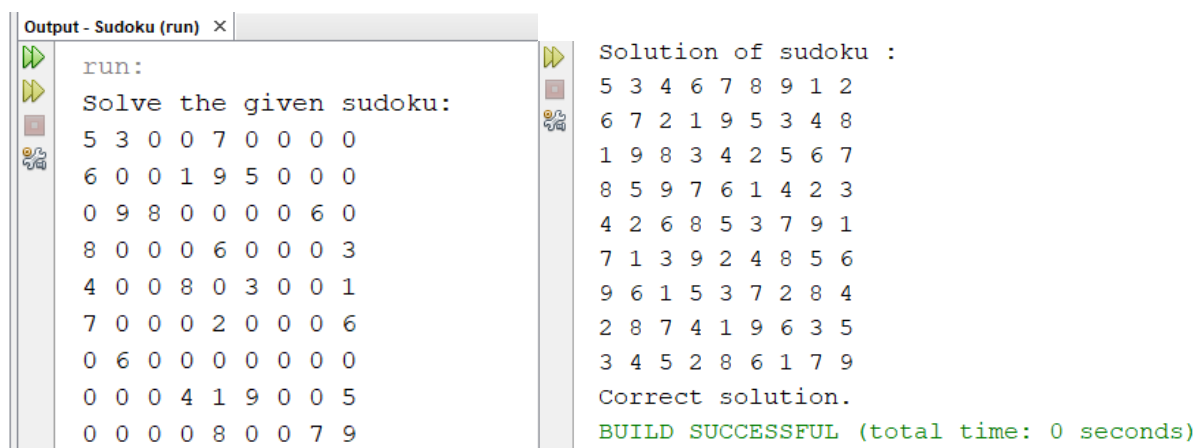
```
if (box_sum[i] == 45) {
```

JAVA PROGRAMMING (Java Programs)

```
        flag = 1;
    } else {
        flag = 0;
        break;
    }
}

if (flag == 1) {
    System.out.println("Correct solution.");
} else {
    System.out.println("Incorrect solution.");
}
}
```

OUTPUT:



```
Output - Sudoku (run) ×
run:
Solve the given sudoku:
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 0 0 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9

Solution of sudoku :
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
Correct solution.
BUILD SUCCESSFUL (total time: 0 seconds)
```

10. Implement Caesar Cipher.

CODE:

```
package pra.pkg10;

import java.util.Scanner;

public class Pra10
{
    public static void main(String[] args)
    {
        String s;
        String incrept="";
        String decrept="";
        Scanner in = new Scanner (System.in);
        System.out.println("Enter String : ");
```

JAVA PROGRAMMING (Java Programs)

```
s = in.nextLine();
int key;
System.out.println("Enter Key : ");
key = in.nextInt();
for(int i=0;i<s.length();i++)
{
    int a1 = s.charAt(i);
    int n1 = (a1+key);
    if(n1>122)
    {
        n1 = n1-26;
    }
    char c1 = (char)n1;
    incrept += Character.toString(c1);
}
System.out.print("Increpted String : " + incrept);
for(int i=0;i<s.length();i++)
{
    int a2 = incrept.charAt(i);
    int n2 = (a2-key);
    if(n2<97)
    {
        n2 = n2+26;
    }
    char c = (char)n2;
    decrept += Character.toString(c);
}
System.out.print("\n");
System.out.print("Decrepted String : " + decrept);
}
}
```

OUTOUT:

JAVA PROGRAMMING (Java Programs)

```
Enter String :  
pratik  
Enter Key :  
4  
Increpted String : tvexmo  
Deccrepted String : pratik
```


PART-II

Object Oriented Programming : Classes, Methods, Inheritance

1. Design a class named Circle containing following attributes and behavior.

- One double data field named radius. The default value is 1.
- A no-argument constructor that creates a default circle.
- A Single argument constructor that creates a Circle with the specified radius.
- A method named getArea() that returns area of the Circle.
- A method named getPerimeter() that returns perimeter of it.

Code:

```
package part.pkg2.pra.pkg1;
class Circle
{
    public double radius;
    public Circle()
    {
        radius = 1d;
    }
    public Circle(double r)
    {
        radius = r;
    }
    public double getArea()
    {
        double area = 3.14d*radius*radius;
        return area;
    }
    public double getParimeter()
    {
        double parimeter = 2*3.14d*radius;
        return parimeter;
    }
}
public class Part2Pra1
{
    public static void main(String[] args)
    {
        Circle c1 = new Circle();
        Circle c2 = new Circle(8d);
        System.out.println("RADIUS OF CIRCLE : "+c1.radius + "\nThe Area of Circle : "+c1.getArea()+"\nThe Parimeter of the Circle : "+c1.getParimeter());
        System.out.println("\nRADIUS OF CIRCLE : "+c2.radius + "\nThe Area of Circle : "+c2.getArea()+"\nThe Parimeter of the Circle : "+c2.getParimeter());
    }
}
```

Output:

JAVA PROGRAMMING (Java Programs)

```
RADIUS OF CIRCLE : 1.0
The Area of Circle : 3.14
The Parimeter of the Circle : 6.28

RADIUS OF CIRCLE : 8.0
The Area of Circle : 200.96
The Parimeter of the Circle : 50.24
```

2. Design a class named Account that contains:

- A private int data field named id for the account (default 0).
- A private double data field named balance for the account (default 500₹).
- A private double data field named annualInterestRate that stores the current interest rate (default 7%). Assume all accounts have the same interest rate.
- A private Date data field named dateCreated that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id and initial balance.
- The accessor and mutator methods for id, balance, and annualInterestRate.
- The accessor method for dateCreated.
- A method named getMonthlyInterestRate() that returns the monthly interest rate.
- A method named getMonthlyInterest() that returns the monthly interest.
- A method named withdraw that withdraws a specified amount from the account.
- A method named deposit that deposits a specified amount to the account.

Code:

```
import java.util.Scanner;
import java.util.Date;
class Account
{
    private int id;
    private double balance;
    private double annualInterestRate;
    private Date dateCreated ;
    Account(int id,double balance)
    {
        this.id = id;
        this.balance = balance;
    }

    Account()
    {
        this(0,500);
        annualInterestRate = 0.07;
    }

    int getId(){return id;}
    double getBalance(){return balance;}
    double getAnnualInterestRate(){return annualInterestRate;}
    Date getDateCreated(){return dateCreated;}

    void setId(int id){this.id = id;}
    void setBalance(double balance){this.balance = balance;}
```

JAVA PROGRAMMING (Java Programs)

```
void setAnnualInterestRate(double AnnualInterestRate)
{
    This. annualInterestRate = annualInterestRate;
}

double getMonthlyInterestRate(){return (annualInterestRate/12);}
double getMonthlyInterest(){return (balance*annualInterestRate);}

void withdrawAmount(double amount)
{
    if(balance > amount)
        balance -= amount;
    else
        System.out.println("Insufficient Balance");
}

void depositAmount(double amount)
{
    balance += amount;
}

void display()
{
    System.out.println("ID : " + id);
    System.out.println("Balance : " + balance);
    System.out.println("Annual Interest Rate : " + annualInterestRate);
}

}

class Practical12
{
    public static void main(String args[])
    {
        Account ac1 = new Account();
        Account ac2 = new Account(132,50000);
        ac1.display();
        System.out.println();
        ac2.display();
    }
}
```

Output:

```
ID : 0
Balance : 500.0
Annual Interest Rate : 0.07

ID : 132
Balance : 50000.0
Annual Interest Rate : 0.0
```

3. Use the Account class created as above to simulate an ATM machine. Create 10 accounts with id AC001.....AC010 with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices.

1. Balance inquiry
2. Withdraw money [Maintain minimum balance 300₹]
3. Deposit money
4. Money Transfer
5. Create Account
6. Deactivate Account
7. Exit

Code:

```
import java.util.Scanner;
import java.util.Date;
class Account
{
    private int id;
    private double balance;
    private double annualInterestRate;
    private Date dateCreated ;

    Account(int id,double balance)
    {
        this.id = id;
        this.balance = balance;
    }

    Account()
    {
        this(0,500);
        annualInterestRate = 0.07;
    }

    int getId(){return id;}
    double getBalance(){return balance;}
    double getAnnualInterestRate(){return annualInterestRate;}
    Date getDateCreated(){return dateCreated;}

    void setId(int id){this.id = id;}
    void setBalance(double balance){this.balance = balance;}
    void setAnnualInterestRate(double annualInterestRate){this.annualInterestRate =
annualInterestRate;}

    double getMonthlyInterestRate(){return (annualInterestRate/12);}
    double getMonthlyInterest(){return (balance*annualInterestRate);}

    void withdrawAmount(double amount)
    {
```

JAVA PROGRAMMING (Java Programs)

```
        if(balance > amount + 300)
        {
            balance -= amount;
            System.out.println("Withdraw Sucessfull");
        }
        else
            System.out.println("Insufficient Balance");
    }

    void depositAmount(double amount)
    {
        balance += amount;
    }

    void display()
    {
        System.out.println("ID : " + id);
        System.out.println("Balance : " + balance);
        System.out.println("Annual Interest Rate : " + annualInterestRate);
    }
}

class Practical13
{
    public static void main(String args[])
    {
        Account[] acc = new Account[10];

        for(int i=0;i<10;i++)
        {
            acc[i] = new Account(i+1,300);
        }

        System.out.println("Please enter the account number:");
        Scanner scnr = new Scanner(System.in);

        int acc_number;
        acc_number = scnr.nextInt();
        int choice;

        do{
            System.out.println("Welcome to the ATM Machine. Please select the proper
choice:\n");

            System.out.println("Enter 1 for Balance Inquiry");
            System.out.println("Enter 2 for Withdraw money");
            System.out.println("Enter 3 for Deposit money");
            System.out.println("Enter 4 for Money Transfer");
            System.out.println("Enter 5 for Exiting");
            System.out.println("\nEnter your choice:");
            choice = scnr.nextInt();
        }
    }
}
```

```
switch(choice)
{
    case 1:
    {
        System.out.println("Your Account Balance is: " +
acc[acc_number].getBalance());
        break;
    }
    case 2:
    {
        System.out.println("Enter the money you want to withdraw:");
        double money = scnr.nextDouble();

        acc[acc_number].withdrawAmount(money);
        break;
    }
    case 3:
    {
        System.out.println("Enter the money you want to deposit:");
        double depo = scnr.nextDouble();

        acc[acc_number].depositAmount(depo);
        System.out.println("Money Deposited");
        break;
    }
    case 4:
    {
        System.out.println("Enter the account in which you want to transfer
the amount: ");

        int transfer = scnr.nextInt();

        System.out.println("Enter the money you want to transfer:");
        double money_transfer = scnr.nextDouble();

        if(acc[acc_number].getBalance() > money_transfer + 300)
        {
            acc[transfer].depositAmount(money_transfer);
            acc[acc_number].withdrawAmount(money_transfer);
            System.out.println("Money transfer Successful");
        }
        else
        {
            System.out.println("The Account has insufficient balance");
        }
        break;
    }
    case 5:
    {
        break;
    }
}
```

JAVA PROGRAMMING (Java Programs)

```
                default:
                {
                    System.out.println("Invalid Choice");
                    break;
                }
            }
        }
    while(choice !=5);
    }
}
```

Output:

```
Enter 1 for Balance Inquiry
Enter 2 for Withdraw money
Enter 3 for Deposit money
Enter 4 for Money Transfer
Enter 5 for Exiting

Enter your choice:
5
```

4. (Subclasses of Account) In Programming Exercise 2, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Draw the UML diagram for the classes and then implement them. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount and invokes their toString() methods.

Code:

```
import java.util.Scanner;
class Office
{
    private int employee_Number;
    private String employee_Name;
    private double salary;

    Office()
    {
        this(1,"",0);
    }
    Office(int employee_Number,String employee_Name,double salary)
    {
        this.employee_Number = employee_Number;
        this.employee_Name = employee_Name;
        this.salary = salary;
    }

    void setEmployeeNumber(int employee_Number){this.employee_Number = employee_Number;}
    void setEmployeeName(String employee_Name){this.employee_Name = employee_Name;}
    void setSalary(double salary){this.salary = salary;}
```

JAVA PROGRAMMING (Java Programs)

```
int getEmployeeNumber(){return employee_Number;}
String getEmployeeName(){return employee_Name;}
double getSalary(){return salary;}
void Display()
{
    System.out.println("Employee ID : " + employee_Number);
    System.out.println("Employee Name : " + employee_Name);
    System.out.println("Employee Salary : " + salary);
}
}

class Teaching extends Office
{
    private String designation;
    Teaching()
    {
        super();
        designation = "";
    }
    Teaching(int id,String name,double pay,String designation)
    {
        super(id,name,pay);
        this.designation = designation;
    }
    void setDesignation(String designation){this.designation = designation;}
    String getDesignation(){return designation;}
    void Display()
    {
        super.Display();
        System.out.println("Employee Designation : " + designation);
    }
}

class Non_Teaching extends Office
{
    private String designation;
    Non_Teaching()
    {
        super();
        designation = "";
    }
    Non_Teaching(int id,String name,double pay,String designation)
    {
        super(id,name,pay);
        this.designation = designation;
    }
    void setDesignation(String designation){this.designation = designation;}
    String getDesignation(){return designation;}
    void Display()
```


JAVA PROGRAMMING (Java Programs)

```
{
    super.Display();
    System.out.println("Employee Designation : " + designation);
}
}

class Practical14
{
    public static void main(String args[])
    {
        Teaching t1 = new Teaching(1,"John",10000,"Pune");
        Non_Teaching t2 = new Non_Teaching(2,"Sam",5000,"Assistant Manager");

        t1.Display();
        System.out.println();
        t2.Display();
        System.out.println("\nMurtaza Vasi");
        System.out.println("18DCS132");
        System.out.println("CSE II");
    }
}
```

Output:

```
Employee ID : 1
Employee Name : John
Employee Salary : 10000.0
Employee Designation : Pune

Employee ID : 2
Employee Name : Sam
Employee Salary : 5000.0
Employee Designation : Assistant Manager
```

5. Develop a Program that illustrate method overloading concept.

Code:

```
package test2;
import java.util.*;

public class Test2 {
    public static void main(String[] args) {

        int l,b,sa,ra;
        float r,ca;
        int a;
        Scanner in=new Scanner(System.in);
        System.out.println("enter parameter for find area of rectangle");
        System.out.println("enter the length :");
        l=in.nextInt();
        System.out.println("enter the breadth :");
        b=in.nextInt();
```

JAVA PROGRAMMING (Java Programs)

```
System.out.println("enter parameter for find the area of circle");
System.out.println("enter the radius :");
r=in.nextFloat();
System.out.println("enter parameter for find area of Square: ");
System.out.println("enter the length :");
a=in.nextInt();
ra=area(l,b);
System.out.println("area of rectange: "+ra);
ca=area(r);
System.out.println("area of circle :"+ca);
sa=area(a);
System.out.println("area of square:"+sa);
}
static int area(int l,int b)
{
    return l*b;
}
static float area(float r)
{
    return r*r*3.14f;
}
static int area(int a)
{
    return a*a;
}
}
```

Output:

```
enter parameter for find area of rectangle
enter the length :
12
enter the breadth :
3
enter parameter for find the area of circle
enter the radius :
4
enter parameter for find area of Square:
enter the length :
4
area of rectange: 36
area of circle :50.24
area of square:16
```

Part-3

Package and interface

- 1. WAP that illustrate the use of interface reference. Interface Luminious Object has two method lightOn() and lightOff(). There is one class Solid extended by 2 classes Cube and Cone. There is one class LuminiousCone extends Cone and implements Luminious Interface. LumminuiousCube extends Cube and implements Luminious Interface. Create a object of LuminiousCone and LuminousCube and use the concept of interface reference to invoke the methods of interface.**

Code:

```
package interface1;
interface Luminious
{
    public void lightOn();
    public void lightOff();
}
class solid
{
    solid()
    {
        System.out.print("solid :");
    }
}
class cone extends solid
{
    cone()
    {
        System.out.println("cube");
    }
}
class cube extends solid
{
    cube()
    {
        System.out.println("cone");
    }
}
class LuminiousCube extends cube implements Luminious
{
    public void lightOn()
    {
        System.out.println("LuminiousCube : light On");
    }
    public void lightOff()
    {
        System.out.println("LuminiousCube : light Off");
    }
}
class LuminiousCone extends cone implements Luminious
{
    public void lightOn()
```

JAVA PROGRAMMING (Java Programs)

```
{
    System.out.println("LuminiousCone : light On");
}
public void lightOff()
{
    System.out.println("LuminiousCone : light Off");
}
}
public class Interface1
{
    public static void main(String[] args) {
        LuminiousCube lcube=new LuminiousCube();
        LuminiousCone lcone=new LuminiousCone();
        lcube.lightOn();
        lcube.lightOff();
        lcone.lightOn();
        lcone.lightOff();
    }
}
```

Output:

```
solid :cone
solid :cube
LuminiousCube : light On
LuminiousCube : light Off
LuminiousCone : light On
LuminiousCone : light Off
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 2. WAP that illustrate the interface inheritance. Interface P is extended by P1 and P2 interfaces. Interface P12 extends both P1 and P2. Each interface declares one method and one constant. Create one class that implemetns P12. By using the object of the class invokes each of its method and displays constant.**

Code:

```
package interface2;
interface P
{
    final int p=1;
    public void pDisplay();
}
interface P1 extends P
{
    final int p1=2;
    public void p1Display();
}
interface P2 extends P
{
    final int p2=3;
    public void p2Display();
}
interface P12 extends P1,P2
```

JAVA PROGRAMMING (Java Programs)

```
{
    final int p12=4;
    public void p12Display();
}
class Display implements P12
{
    public void pDisplay()
    {
        System.out.println("p :"+p);
    }
    public void p1Display()
    {
        System.out.println("p1 :"+p1);
    }
    public void p2Display()
    {
        System.out.println("p2 :"+p2);
    }
    public void p12Display()
    {
        System.out.println("p12 :"+p12);
    }
}
public class Interface2 {

    public static void main(String[] args) {
        Display d=new Display();
        d.pDisplay();
        d.p1Display();
        d.p2Display();
        d.p12Display();
    }
}
```

Output:

```
run:
p :1
p1 :2
p2 :3
p12 :4
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 3. Create an abstract class Robot that has the concrete subclasses , RobotA, RobotB, RobotC. Class RobotA1 extends RobotA, RobotB1 extends RobotB and RobotC1 extends RobotC. There is interface Motion that declares 3 methods forward(), reverse() and stop(), implemented by RobotB and RobotC. Sound interface declare method beep() implemented by RobotA1, RobotB1 and RobotC1. Create an instance method of each class and invoke beep() and stop() method by all objects.**

Code:

```
package interface3;
interface Motion
{
```

JAVA PROGRAMMING (Java Programs)

```
    public void forward();
    public void reverse();
    public void stop();
}
interface Sound
{
    public void beep();
}
abstract class Robot
{
    abstract void rDisplay();
}
class RobotA extends Robot
{
    void rDisplay()
    {
        System.out.println("Robot in A");
    }
    void raDisplay()
    {
        System.out.println("RobotA ");
    }
}
class RobotB extends Robot implements Motion
{
    void rDisplay()
    {
        System.out.println("Robot in B");
    }
    public void forward()
    {
        System.out.println("forwardB ");
    }
    public void reverse()
    {
        System.out.println("reverseB ");
    }
    public void stop()
    {
        System.out.println("stopB ");
    }
    void rbDisplay()
    {
        System.out.println("RobotB ");
    }
}
class RobotC extends Robot implements Motion
{
    void rDisplay()
    {
        System.out.println("Robot in C");
    }
    public void forward()
    {
```

JAVA PROGRAMMING (Java Programs)

```
        System.out.println("forwardC ");
    }
    public void reverse()
    {
        System.out.println("reverseC ");
    }
    public void stop()
    {
        System.out.println("stopC ");
    }
    void rcDisplay()
    {
        System.out.println("RobotC ");
    }
}
class RobotA1 extends RobotA implements Sound
{
    void ra1Display()
    {
        System.out.println("RbotA1 ");
    }
    public void beep()
    {
        System.out.println("beepA1 ");
    }
}
}
class RobotB1 extends RobotB implements Sound
{
    void rb1Display()
    {
        System.out.println("RobotB1");
    }
    public void beep()
    {
        System.out.println("beepB1 ");
    }
}
}
class RobotC1 extends RobotC implements Sound
{
    void rc1Display()
    {
        System.out.println("RobotC1 ");
    }
    public void beep()
    {
        System.out.println("beepC1 ");
    }
}
}
public class Interface3 {

    public static void main(String[] args) {
        RobotB b=new RobotB();
        RobotC c=new RobotC();
    }
}
```

JAVA PROGRAMMING (Java Programs)

```
        RobotA1 a1=new RobotA1();
        RobotB1 b1=new RobotB1();
        RobotC1 c1=new RobotC1();
        b.stop();
        c.stop();
        a1.beep();
        b1.beep();
        c1.beep();
    }
}
```

Output:

```
stopB
stopC
beepA1
beepB1
beepC1
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Develop a Program that illustrate method overriding concept.

Code:

```
package interface4;
abstract class Area
{
    abstract void area();
}
class Rectangle extends Area
{
    int l=12;
    int b=15;
    void area()
    {
        System.out.println("rectangle area :"+(l*b));
    }
}
class Circle extends Area
{
    void area()
    {
        float r=5;
        System.out.println("circle area :"+(3.14*r*r));
    }
}
class Square extends Area
{
    int a=10;
    void area()
    {
        System.out.println("square area :"+(a*a));
    }
}
```


JAVA PROGRAMMING (Java Programs)

```
public class Interface4 {  
  
    public static void main(String[] args) {  
        Rectangle r=new Rectangle();  
        Circle c=new Circle();  
        Square s=new Square();  
        r.area();  
        c.area();  
        s.area();  
    }  
}
```

Output:

```
run:  
rectangle area :180  
circle area :78.5  
square area :100  
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Write a java program which shows importing of classes from other user define packages.

Code:

```
package interface5;  
import subinterface5.*;  
import java.util.*;  
class teacherinfo  
{  
    String name;  
    String subject;  
    void getteacherinfo()  
    {  
        System.out.println("TEACHER info:");  
        Scanner in=new Scanner(System.in);  
        System.out.println("name :");  
        name=in.nextLine();  
        System.out.println("subject name:");  
        subject=in.nextLine();  
    }  
}  
public class Interface5  
{  
    public static void main(String[] args)  
    {  
        studentinfo s=new studentinfo();  
        teacherinfo t=new teacherinfo();  
        s.getstudentinfo();  
        t.getteacherinfo();  
    }  
}
```

JAVA PROGRAMMING (Java Programs)

```
package subinterface5;
import java.util.*;

public class studentinfo {
    int stu_id;
    String name1;
    public void getstudentinfo()
    {
        System.out.println("STUDENT info ");
        Scanner in=new Scanner(System.in);
        System.out.println("student id :");
        stu_id=in.nextInt();
        System.out.println("name :");
        name1=in.next();
    }
}
```

Output:

```
STUDENT info
student id :
18
name :
dhruv
TEACHER info:
name :
saurabhkumar
subject name:
maths
BUILD SUCCESSFUL (total time: 35 seconds)
|
```

6. Write a program that demonstrates use of packages & import statements.

Code:

```
package interface6;
import circle.*;
import rectangle.*;

public class Interface6 {

    public static void main(String[] args) {
        Circleinfo c=new Circleinfo();
        Rectangleinfo r=new Rectangleinfo();
        c.area();
        c.volume();
        r.area();
        r.volume();
    }
}
```

JAVA PROGRAMMING (Java Programs)

```
package circle;

public class Circleinfo {

    int a=10;
    public void area()
    {
        System.out.println("rectangle area: "+(a*a));
    }
    public void volume()
    {
        System.out.println("rectangle volume: "+(a*a*a));
    }
}
```

```
package rectangle;
```

```
public class Rectangleinfo {
    int l=10;
    int b=20;
    int h=30;
    public void area()
    {
        System.out.println("rectangle area: "+(l*b));
    }
    public void volume()
    {
        System.out.println("rectangle volume: "+(l*b*h));
    }
}
```

Output:

```
run:
rectangle area: 100
rectangle volume: 1000
rectangle area: 200
rectangle volume: 6000
BUILD SUCCESSFUL (total time: 1 second)
```

7. Write a program that illustrates the significance of interface default method.

Code:

```
package interface7;
interface TestInterface
{
    public void square(int a);
}
```

JAVA PROGRAMMING (Java Programs)

```
        default void show()
        {
            System.out.println("Default Method Executed");
        }
    }

class Interface7 implements TestInterface
{
    public void square(int a)
    {
        System.out.println(a*a);
    }

    public static void main(String args[])
    {
        Interface7 d = new Interface7();
        d.square(4);

        d.show();
    }
}
```

Output:

```
run:
16
Default Method Executed
BUILD SUCCESSFUL (total time: 1 second)
|
```

PART-IV

Exception Handling

1. WAP to show the try - catch block to catch the different types of exception.

Code:

```
package part.pkg4.pra.pkg1;
import java.util.*;
class InvalidAge extends Exception
{
    int age;
    public InvalidAge(int a)
    {
        age = a;
    }
    @Override
    public String toString()
    {
        return "The error has been occurred due to value of AGE as : "+age;
    }
    void setAge() throws InvalidAge
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter value for AGE(above or equal to 18) : ");
        int a = in.nextInt();
        if(a<18)
            throw new InvalidAge(a);
        else
            age = a;
    }
}
class InvalidTemp extends Exception
{
    int temp;
    public InvalidTemp(int t)
    {
        temp = t;
    }
    @Override
    public String toString()
    {
        return "The error has been occurred due to value of TEMPRETURE as : "+temp;
    }
    void setTemp() throws InvalidTemp
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter value for TEMPRETURE(between 16-32): ");
```

JAVA PROGRAMMING (Java Programs)

```
int temp = in.nextInt();
int t=temp;
if(temp<16 || temp>32)
    throw new InvalidTemp(t);
else
    temp = t;
}
}
public class Part4Pra1
{
    public static void main(String[] args)
    {
        InvalidAge o1 = new InvalidAge(0); //by default value is given 0
        try
        {
            o1.setAge();
        }
        catch(InvalidAge o)
        {
            System.out.println(o);
        }

        InvalidTemp o2 = new InvalidTemp(0); //by default value is given 0
        try
        {
            o2.setTemp();
        }
        catch(InvalidTemp o)
        {
            System.out.println(o);
        }

    }
}
```

Output:

```
Enter value for AGE(above or equal to 18) :
17
The error has been occurred due to value of AGE as : 17
Enter value for TEMPRETURE(between 16-32):
11
The error has been occurred due to value of TEMPRETURE as : 11
```

2. WAP to generate user defined exception using “throw” and “throws” keyword.

Code:

```
package part.pkg4.pra.pkg2;
import java.util.*;
public class Part4Pra2
```

JAVA PROGRAMMING (Java Programs)

```
{
    public static void main(String[] args)
    {
        int a,b;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter value for NUMERATOR & DENOMENATOR(except 0) : ");
        a = in.nextInt();
        b = in.nextInt();
        try
        {
            Part4Pra2.divide(a, b);
        }
        catch(IllegalArgumentException o)
        {
            System.out.println(o.getMessage());
        }

    }
    public static void divide(int a,int b) throws IllegalArgumentException
    {
        if(b==0)
        {
            throw new IllegalArgumentException("Error occured due to value '0' as DENOMINATOR...");
        }
        else
        {
            System.out.println("Division : "+(a/b));
        }
    }
}
```

Output:

```
Enter value for NUMERATOR & DENOMENATOR(except 0) :
5
0
Error occured due to value '0' as DENOMINATOR...
```

- 3. Write a program that raises two exceptions. Specify two ‘catch’ clauses for the two exceptions. Each ‘catch’ block handles a different type of exception. For example the exception could be ‘ArithmeticException’ and ‘ArrayIndexOutOfBoundsException’. Display a message in the ‘finally’ block.**

Code:

```
package part.pkg4.pra.pkg3;
import java.util.*;
public class Part4Pra3
{
    public static void main(String[] args)
```

JAVA PROGRAMMING (Java Programs)

```
{
    int a=5,b=0;
    int arr[] = new int[5];
    Scanner in = new Scanner(System.in);
    System.out.println("Enter value for NUMERATOR & DENOMENATOR(except 0) : ");
    a = in.nextInt();
    b = in.nextInt();
    try
    {
        int devision = a/b;
    }
    catch(ArithmeticException o)
    {
        System.out.println(o.getMessage());
    }

    try
    {
        arr[6] = 55;
    }
    catch(ArrayIndexOutOfBoundsException o)
    {
        System.out.println(o.getMessage());
    }

    finally
    {
        System.out.println("THIS IS FINALLY BLOCK...");
    }
}
```

Output:

```
Enter value for NUMERATOR & DENOMENATOR(except 0) :
5
0
/ by zero
6
THIS IS FINALLY BLOCK...
```

PART-V

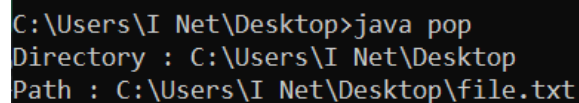
File Handling & Streams

1. WAP to show how to create a file with different mode and methods of File class to find path, directory etc.

CODE :

```
import java.io.File;
public class pop
{
    public static void main(String[] args)
    {
        File f1 = new File("C:\\Users\\I Net\\Desktop\\file.txt");
        System.out.println("Directory : "+f1.getParent());
        System.out.println("Path : "+f1.getPath());
    }
}
```

OUTPUT :



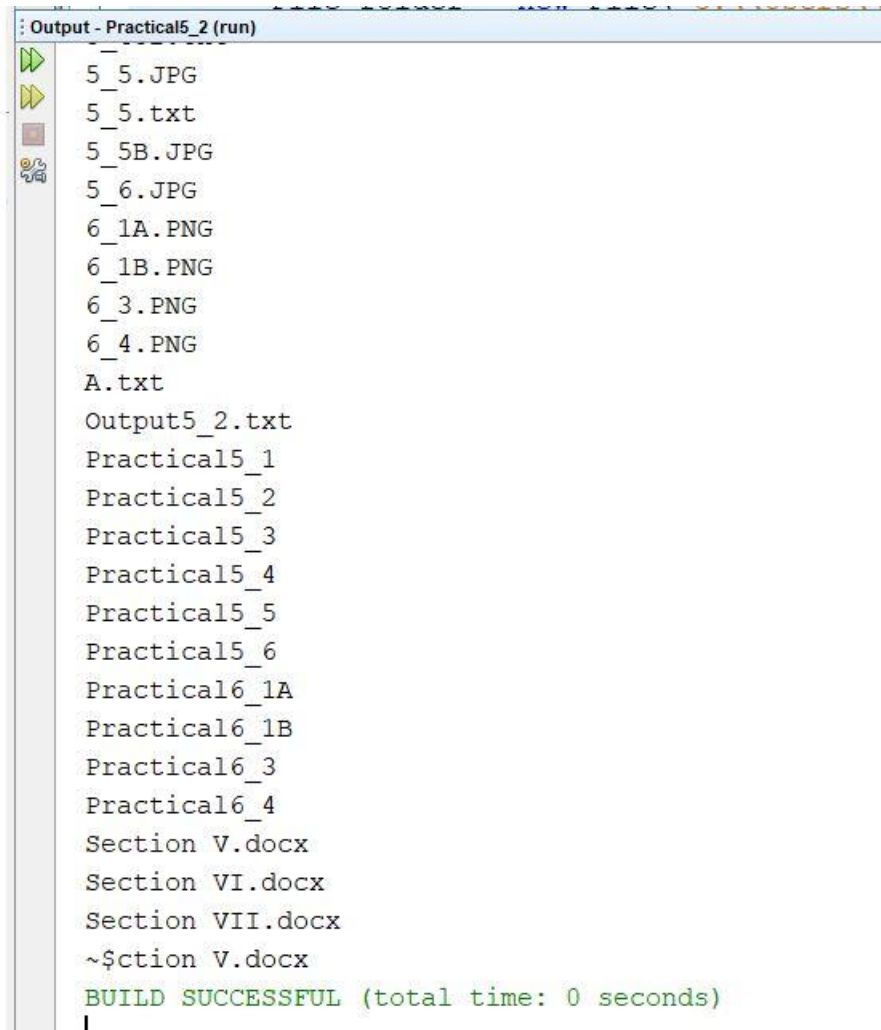
```
C:\Users\I Net\Desktop>java pop
Directory : C:\Users\I Net\Desktop
Path : C:\Users\I Net\Desktop\file.txt
```

2. Write a program to show a tree view of files and directories under a specified drive/volume.

CODE :

```
package practical5_2;
import java.io.File;
public class Practical5_2
{
    public static void main(String[] args)
    {
        File folder = new File("C:\\Users\\Ankur\\Desktop\\practical");
        String[] files = folder.list();
        for (String file : files)
        {
            System.out.println(file);
        }
    }
}
```

OUTPUT :



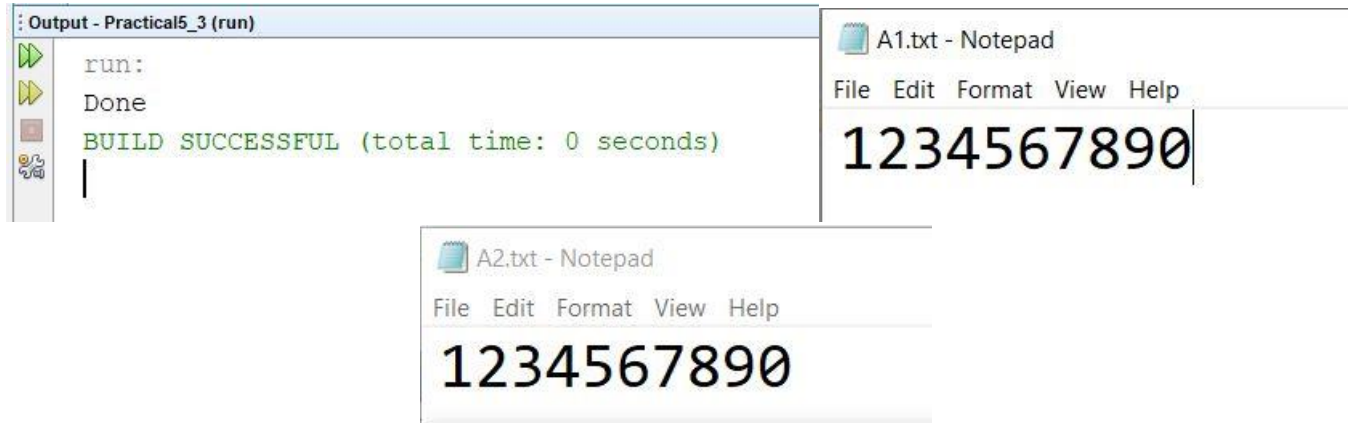
3. Write a program to transfer data from one file to another file so that if the destination file does not exist, it is created.

CODE :

```
package practical5_3;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class Practical5_3
{
    public static void main(String[] args) throws IOException
    {
        FileInputStream fis = new FileInputStream("C:\\\\Users\\I Net\\A1.txt");
        FileOutputStream fos = new FileOutputStream("C:\\\\Users\\I Net\\A2.txt");
        int b;
        while ((b=fis.read()) != -1)
            fos.write(b);
        fis.close();
        fos.close();
    }
}
```

```
}
```

OUTPUT :



4. WAP to show use of character and byte stream.

CODE :

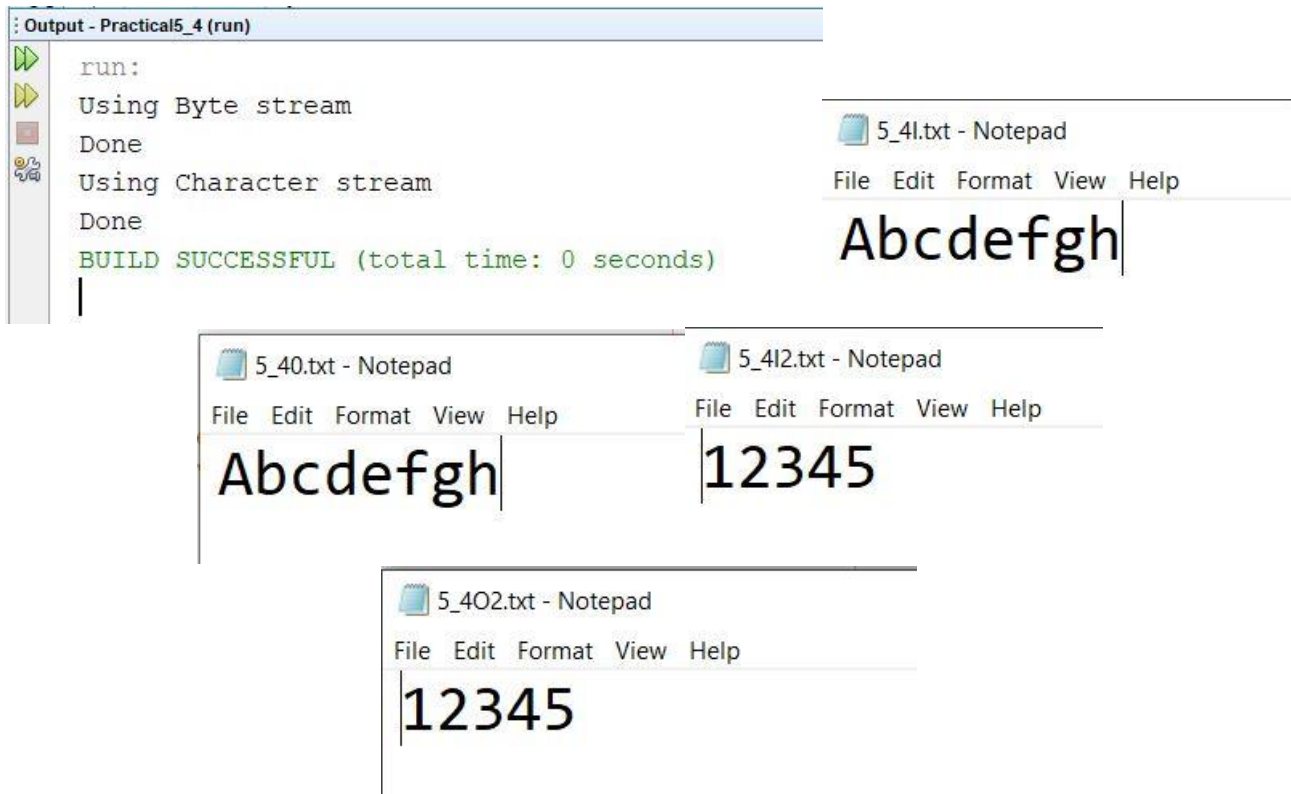
```
package practical5_4;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class Practical5_4
{
    public static void main(String args[]) throws IOException {
        System.out.println("Using Byte stream");
        FileInputStream in = null;
        FileOutputStream out = null;
        try
        {
            in = new FileInputStream("C:\\Users\\I Net\\5_4l.txt");
            out = new FileOutputStream("C:\\Users\\I Net\\5_40.txt");
            int c;
            while ((c = in.read()) != -1)
            {
                out.write(c);
            }
        }
        finally
        {
            if (in != null)
            {
                in.close();
            }
        }
    }
}
```

JAVA PROGRAMMING (Java Programs)

```
    }
    if (out != null)
    {
        out.close();
    }
    System.out.println("Done");
}
System.out.println("Using Character stream");
FileReader input = null;
FileWriter output = null;
try {
    input = new FileReader("C:\\Users\\Ankur\\Desktop\\practical\\5_4I2.txt");
    output = new FileWriter("C:\\Users\\Ankur\\Desktop\\practical\\5_4O2.txt");
    int d;
    while ((d = input.read()) != -1) {
        output.write(d);
    }
}finally {
    if (input != null) {
        input.close();
    }
    if (output != null) {
        output.close();
    }
    System.out.println("Done");
}
}
```

OUTPUT :

JAVA PROGRAMMING (Java Programs)



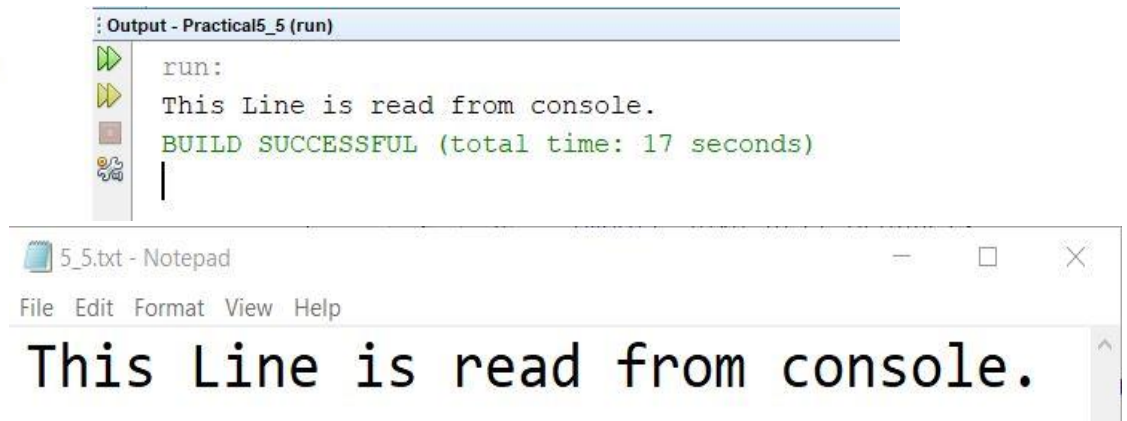
5. WAP to read console input and write them into a file. (BufferedReader /BufferedWriter).

CODE :

```
package practical5_5;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Practical5_5
{
    public static void main(String[] args) throws IOException
    {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        FileWriter fw = new FileWriter("C:\\\\Users\\I Net\\5_5.txt");
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(str);
        bw.close();
    }
}
```

OUTPUT :



6. WAP to demonstrate methods of wrapper class.

CODE :

```
package practical5_6;
public class Practical5_6
{
    public static void main(String[] args)
    {
        System.out.println("For Character Wrapper Class");
        Character ch = 'a';
        Character cha = 'b';
        System.out.println("Character value : "+ch.charValue());
        System.out.println("Compare with other object : "+ch.compareTo(cha));
        System.out.println("Class name : "+ch.getClass());
        System.out.println("Compare with other value"+ch.equals(cha));
        System.out.println("HashCode value"+ch.hashCode());
        System.out.println("");
        System.out.println("For Integer Wrapper Class");
        Integer i1 = 55;
        Integer i2 = 85;
        System.out.println("Byte value : "+i1.byteValue());
        System.out.println("Float value : "+i1.floatValue());
        System.out.println("Double value : "+i1.doubleValue());
        System.out.println("Integer value : "+i1.intValue());
        System.out.println("Long value : "+i1.longValue());
        System.out.println("Short value : "+i1.shortValue());
        System.out.println("Compare with other object : "+i1.compareTo(i2));
        System.out.println("Compare with other value : "+i1.equals(i2));
        System.out.println("Class name : "+i1.getClass());
        System.out.println("HashCode value : "+i1.hashCode());
```

JAVA PROGRAMMING (Java Programs)

```
}  
}
```

OUTPUT :

```
Output - Practical5_6 (run)  
run:  
For Character Wrapper Class  
Character value : a  
Compare with other object : -1  
Class name : class java.lang.Character  
Compare with other valuefalse  
Hashcode value97  
  
For Integer Wrapper Class  
Byte value : 55  
Float value : 55.0  
Double value : 55.0  
Integer value : 55  
Long value : 55  
Short value : 55  
Compare with other object : -1  
Compare with other value : false  
Class name : class java.lang.Integer  
Hashcode value : 55  
BUILD SUCCESSFUL (total time: 0 seconds)  
|
```