1. Write a Java program that reads a series of numbers from a file input.txt determines the higest number in the series, calculates the sum of natural numbers up to the highest number, and write the result anothe file output.txt. Use scanner to read from the file and PrintWriter to write to the file Assume the numer in the input file separately.

Sample Input.txt:

   10, 55, 1000, --------------100

Output.txt:

   55, 1540, 500500------ 5050

The code is given below:

Code:

```java
import java.io.File;
import java.io.PrintWriter;
import java.util.Scanner;

public class ReadFile_sumseries {
    public static void main(String[] args) {
        try {
            File inFile = new File("input.txt");
            Scanner in = new Scanner(inFile);
            PrintWriter out = new PrintWriter(new File("output
            if (in.hasNextLine()) {
            String line = in.nextLine();
            String[] nums = line.split(",");
            for (int i=0; i < nums.length; i++) {
                int n = Integer.parseInt(nums[i]);
                int sum = (n*(n+1))/2);
                out.print(sum);
                if (i! = nums.length -1) {
                    out.print(",");
                }
            }
        }
    }
    in.close();
    out.close();
```

```
System.out.println ("File written successfully.");
} catch (Exception e) {
System.out.println ("File not found.");
}

}

}
```

2. What are the difference you have even found between static and final fields and methods? Exemplify what will happen if you try to access the static method/field with the object instead of class name.

Ans:

The difference between static and final field is given below:

Feature	satatic	final
Definition	Belong the class, not instances	Once, assigned, cannot be change
Fields	Shared among all intances	Value cannot be modified once intialized
Methods	Belongs to the class, can be called with out an instance	Cannot be overridden by subclasses
Usage	Uses for constants, utility methods, and share data	Used to enforce immutability and prevent method overriding

Example a of static and final field:

```java
class Example {
    static int staticVar = 10;
    final int finalVar = 20;

    static void staticMethod() {
        System.out.println("static Method called");
    }

    final void finalMethod() {
        System.out.println("Final Method called");
    }
}

public class Test {
    public static void main(String[] args) {
        Example obj = new Example();

        obj.staticMethod();
        System.out.println(Example.staticVan);
    }
}
```

If I Access static Members Using an Object

1. It complie and run successfully but not recomanded. Java allows calling static member
x using and object reference, but it gives a warning because static member belong to the class not the object.

2. Final Field behave, normally finalVar must be intialized either during declaration or in the constructor.

3. Final method can be accessed normally, but cannot be overrride in subclass.

3. Write a java program to find all factorion numbers within a given range. A number is called a factorian if the sum of the factorials of its digits equals the number itself. The program should take user input for the lower and upper bound

Code :

```java
import java.utill.Scanner;

public class FactorionFinder {
    public static void main (String[] args) {
        Scanner scanner = new Scanner (System.in);
        System.out.print ("Enter lower bound; ");
        int lower = scanner.nextInt();
        System.out.print("Enter upper bound: ");
        int upper = scanner.nextInt();
        scanner.close();

        for ( int i= lower; i <= upper; i++) {
            if (isFactorion(i)) System.out.print (i + " ");
        }
    }
}
```

```java
private static boolean isFactonion (int num) {
    int sum = 0, temp = num;

    while (temp > 0) {
        sum += factorial (temp%10);
        temp /= 10;

    }

    return   sum == num;

}
private static int factorial (int n) {
    int[] fact = { 1, 2, 6, 24, 120, 720, 5040,
        40320, 362880};

    return fact[n];

}

}
```

4. Distinguish the difference among class, local, and instance variables. What is significance of this keyword?

Feature	Class variable	Instance variable	local variable
Definition	A variable defined with the static keyword inside a class, shared by all instance of the class	A variable inside a class but outside any method unique to each object instance	A variable defined inside a method, constructor, on block.
Scope	Class-wide, accessible to all instances of the class	object-specific, exists as long as the object exists.	method on block-specific, exists only during the method execution
Lifetime	As long As class is loaded in memory	As long as the object instance exists	Limited to the execution time of the method block
memory	stored in the static memory	stored in the heap pas part of the object instance	stored in the stack memory during method execution.
Initilization	Intitilize when the class is loaded	Intialize when the object is created	must be intilized before use.

The significace of the this keyword:

Definition: The this keyword is a keyword is a reffernce variable referring to the current instance of the class. It is often used inside instance method on constructors to nefer to the current object.

Uses:

1. Distinguishing Instance variable: It helps distinguish instance varlable for parameters on local variable that have the same name.

2. Accessing Instance Methods/ Variables: It allows access to instance variables and methods.

3. Passing current object: It can be used to pass the current object as a parameter to another method or construeton.

Example:

```
class myClass {
    int instanceVar;
    Myclass (int instanceVan) {
        this.instanceVar = instanceVan;
    }
}
```

5. Write a java program that defines a method to calculate the sum of all elements in an integer anny. The method should take on integer anny as parameter and return the sum. Demonstrate this method by passing an anny of integers from the main method:

code:

```
public class SumAnnay {
    public static int calculate Sum (int[] ann){
        int sum = 0;
        for( int sum=
        for ( int num : ann) {
            Sum += sum;
        }
        return sum;
    }
    public static void main ( string[] args){
        int [] numbens = { 1, 2, 3, 9, 5, 6};
        int sum = calculate sum (numbers);
        system.out. println ("The sum of the anny elements
              is: " + sum);
    }
}
```

6. What is called access modifier? compare the accessibity of Public, Private and procted mat modifier describe the different type of variable in java with example.

An access modifier in Java is a keyword used to specify the visibility on accessibity of classes, methods, variables, on constructors.

The main access modifier are:

1. public
2. private
3. procted
4. default

Modifier	Accessibility	Description
public	Accessable from anywhere	Member declaned as public and can be access from anywhere
private	Accessable only within the same class	Member declared as private not accessable from any where. They car only be accessed with in the class they are defin
Protected	Accessable within the same packadge and by subclass	member declare as protected

Variable are:

    1. Instance variable

    2. Class variable

    3. local variable

    4. Parameter

1. Instance variable : A variable defined in a class but outside methods. Each object of the class has it own copy.

Code:

```
class can {
    String model;
    void displaymodel() {
        System.out. println ("Model: " + model);
    }
}

public class Test {
    public class Test static void main(String[] args) {
        Can can1 = new Can();
        can1.model = "Toyota";
        can1.displayModel();
    }
}
```

2. class variable:

A variable declared with the static keyword.

Shared by all instances of the class.

Example:

```
class car {
    static int count = 0;

    car()
        count++;

    }

}

public class Test {
    public class static void main ( strings [] args) {

    new car ();
    new car ();
    System. out. println ( " Total cans: " + Can. count);

    }

}
```

3. local variable: A variable decleed inside a method or block. Only accesible within that method or block.

Example:

```
class Example {
    void addNumbers (){
        int num1 = 5, num2 = 10;
        int sum = num1 + num2;
        System.out.println ( " Sum: " + sum);
    }
}

public class Test{
    public static void main ( String[] args){
        Example obj = new Example ();
        obj.addNumbers ();
    }
}
```

4. Parameter: A variable passes to a method.
Only accessble within the method

```java
class Example {
    void printMessage (String message)
    System.out.println (message);
    }
}
public class Test {
    public static void main (String [] args)
    {
        Eoxample obj = new Example ();
        obj.printMessage ("Hello, Java!");
    }
}
```

7. Write a Java program to find the smallest positive root of a quadratic equation of the form.

$$ax^2 + bx + c = 0$$

using the quadratic formula,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The program should:

1. Take integer input for coefficient a, b and c.
2. Compute the root using Math.sqrt (double a).
3. Determine the smallest positive root using Math.min (double a, double b)
4. Print the smallest positive root, if real root exist. othewise print " No real roots".

Sample Input: a, b and c : 1, -3, 2

Sample output:

The smallest positive root is : 1.0;

Code :

```java
import java.util.Scanner;
public class QuadraticEquation {
    public static void main (String [] args) {
        Scanner scanner = new Scanner (System.in);
        System.out.print ("Enter the co-efficient a, b and c: ");
        int a = scanner.nextInt (), b = scanner.nextInt (),
        c = scanner.nextInt ();
        double discriminant = b*b - 4*a*c;
        if ( discriminant >= 0) {
            double root1 = (-b + math.sqrt (discriminant)) / (2*a);
            double root2 = (-b - math.sqrt (discriminant)) / (2*a);
            if ( root1 > 0 && root2 > 0) {
                System.out.println ("The smallest positive root is:"
                            + Math.min (root1, root2)); }
            
            else if ( root1 > 0) {
                System.out.println ("The smallest positive root
                        is: " + root1); }
```

```
else if ( roota > 0) {
  System.out.println("The smallest positive root
                      + root2); }


  else {
    System.out.println(" No real roots.");

  }

  scanner.close();

  }

}
```

8. Write a program that can determine the letters, witespace and digit. How do we pass an array to a fonction? write an example.

Code:

```java
public class CharacterType {
    public static void determin CharacterType
        ( char[] chars) {
    for( char c: chars) {
    if ( Character, is Letter (c)) {
        System. out. println ( c + " is    a letter.");

    }
    else if ( Character, isDigit (c)) {
        System.out. println ( c + " is  a digit,");
    } else if ( Character. isWhitespace ((c))) {
        System.out. println ( "White space  detected.");

    }

    }

    }

    public static void main ( string [] args) {
        char[] characters = { 'A', ' ', '5', 'b', '9',
        determine CharacterType (characters);

    }

}
```

9. In Java, explain how method over overriding works in the context of inheritance, what happens when a subclass override a method from it superclass. How does the super class help calling superclass method, what are the potential issus when overriding methods., especially when dealing with constructor.

Method overriding allows a subclass to provide a specific implementation of a method already defined in it superclass. This is a core feature of inheritance in object orianted programming and is essential for achieving runtime polymorphison.

How super Helps:

The super keyword is used to call a method on constructo from the superclass.

super. methodName() is used to call the superclass's overridden method from subclass.

This is useful when the subclass wants to add extra behaviour to superclass method instead of completely replacing it.

10. Differentiate between static and non-static members including necessary example. Write a program that able to check eitam a number or string is palingdom or not.

Aspect	static members	Non-static members
memory allocation	Allocate once for the class	Allocate separately for each object
Access	Accessed using class name	Accessed using an object
Scope	shared across all instance	unique to all instance
Usage	class-level functionally constants	Intance level functionality

Cheak a number for palingdom or no

11. what is called class abstraction and encapsulation? describe with example. what are the difference between abstract a class and Interface?

Ans:

Abstraction :

Abstraction is the concept of hiding the internal implementation details of a class and only exposing essential features. It allows the user to focus on what object does rather than how it does it.

Example of Abstraction using an abstract class:

```
abstract class Vehicle {
    abstract void start();
}
class Can extends Vehicle {
    void start() {
        System.out. println (" Can starts with a
        key");
    }
}
```

```
public class main {
    public static void main ( String[] args) {
        Vechicle myCar = new car();
        myCar. start();
    }
}
```

Encapsulation:

Encapsulation is the practice of wrapping data variables and code (methods) into a single unit cla and restricting direct access to some details.

Example:

```
class person {
    private string name;
    public void setName
        this.name = name;
    }
    public String getName(){
        return name;
    }
}
```

```
public class Emp {
    public static void main ( string[] args) {

        Person p = new person();
        p.setName ("John");

        System.out. println ( p. getName());

    }

}
```

Difference between Abstract a class and Interface.

Points	Abstract Class	Interface
Definition	A class that contain abstract methods and may have concrete method	A collection of abstract methods that most be implemented by any class that uses it.
Method	Can have both abstract and non-abstract methods	Can only have abstract method.
Variable	Can have intance variable	Only public static and final variable
Access modifier	Can have any access modifier such as private, proteted	methods are always public by default.

12. Create a Java program using inheritance to perform multiple numerical operations. Implement a 'Base Class' with common functionalities and extend it into four specilized classs. to handle different tasks. Use a 'Main class' to execute all method.

Answer:

```java
class Base Class {
        void print ( string msg, Object res)
        }
        System.out. point ( msg + res);

    }

}

class Sum Class extends Baseeluss {
        double compute sum () {
            double sum = 0;
            for ( double i=1; i2 = 0.1; i -= 0.1)
                sum += i;
                return sum;
        }
    }
}
```

```
Class    DivisonMultipleclass     extends    Base Class {

    int ged ( int a, int b)
    {
    return  b == 0 ? a: ged (b, a7.b);
    }

    int lem ( int a, int b)
    {

    return (a* b) / ged (a,b);

    }

    }

Class    Number Conversion Class    extends  Base Class {

    Stringo to Binary (int num)
    {
    return Integen. for Binany String (num);


    }

    string toltox (int num)

    {

    return Integen. to Hex Stniang (num);

    }

    string toOct ( int num)
    {
    return Integer to Octal Stning (num);

    {

    {
```

```
Class    CoustomPoint Class    extends    BaseClass {

        void  pn( string msg) {
            System. out. println( " ** **" + msg + "# # #");
        }
    }


public class  Inheritance To Perform Numerical OP {

    public static void  main ( string [] args) {

        sumclass    sumobj = new  SumClass ();

        Sum.Obj. point (" Sum:",  sum Obj. computed conc ());

    DivisonMultipleClass divMolobj = new  Divison Multiple class ();

    int  a = 36,  b = 60;


div Mul Obj. print (" GCD",  divMul Obj. ged (a, b) );
div Mul Obj. print ( "LcM",  divMul Obj. lem (a, h));


NumberConvension class  numCon Obj = new. Number Con
                                          version class ();
```

```
int     num = 42;
num Conv Obj . print ("Binary:    ", num Conv Obj . to Bin
                                 ary (num));

num Conv Obj . print ("Hex:   ", num Conv Obj  toBinary(num));

num Con Obj . print(" Octal:  ", num Conv Obj .  to Octal
                                        - (num));

new   Custom Print  class() . pr( "Execution Completed");

    }

  }
```