It 23018

1. Here's a java program that reads number from input.text, finds the highest number, calculates the sum of natural numbers up to that highest number, and writes the results to output.text.

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

public class NumberProcessor {
    public static void main (String[] args){
        String inputFile = "input.text";
        String outputfile = "output.text";

        try {
            File file = new File(inputFile);
            Scanner scanner = new Scanner (file);
            scanner.useDelimiter(",");
```

it 23018

```
List < Integer > numbers = new ArrayList <>();
        While (scanner.hasNext()) {
            if(scanner.hasNextInt()) {
                numbers.add(scanner.nextInt());
} else {
                    scanner.next();
                    }
                }
scanner.close();
    if (numbers.isEmpty()) {
        system.out.println("No valid number found
          in the input file");
        return; }
List<String> outputLines = new ArrayList <>
    for(int num : numbers) {
      int sum = (num* (num+1)) /2;
      outputLines.add(num+", "+sum);
    }
```

```
printwriter    writer = new    printwriter (outputFile);

    for (string line; outputLines) {

        writer. println(line);

    }

      writer. close();

system. out. println ("processing complete. Chec

    + outputFile);

} catch (FileNotFoundException e) {

  System. out. println("File not found" + e.get

      message());

}

}

}
```

2.

Difference between static and Final ~~racth~~ fields and methods

Feature	static	Final
Fields	shared across all instancey of the ~~cod~~ class	cannot be reassigned once initialized
methods	can be called without an instance. using the class name	cannot be ~~Over~~ overridden in subclasses.
Inheritance	can be inherited but not overridden if also final.	Methods cannot be overriden. and variable remain constant.

Example of accessing a static field using an object instead of the class Name.

```
class Example {
    static int staticVar = 10;
    int instanceVar = 20;
static void staticMethod () {
    System.out.println ("static method called");
    }
}
public class test {
    public static void main (String[] args) {
        Example obj = new Example ();
System.out.println (Example.staticVar);
        Example.staticMethod ();
System.out.println (obj.staticVar);
        obj.staticMethod ();
System.out.println (obj.instanceVar);
        obj.instanceMethod ();
```

3.

Here is a java program to find all factorion numbers within a given range

```
import java.util.Scanner;

public class FactorionFinder {

    private static final int[] factorial
        = new int[10];

    static {
        factorial[0] = 1;
        int fact = 1;
        for (int i = 1; i < 10; i++) {
            fact *= i;
            factorial[i] = fact;
        }
    }
}
```

it23018

```
pablic static boolean isfactorion(int num) {
int sum = 0, temp = num;
while (temp > 0) {
ind digit = temp % 10;
Sum += factorial[digit];
temp /= 10;
}
return sum == num;
}

public static void main(String[] args) {
Scanner scanner = new scanner(System.in
System.out.print("Enter the lower bound");
int lower = scanner.nextInt();
System.out.print("Enter the upper bound");
int upper = scannermextInt();
System.out.println("Factorion numbers");
boolean found = false;
for (int i = lower ; i <= upper; i++) {
if (isfactorion(i)) {
System.out.print(i+" ");
found = true; }
```

```
if (found) {
 System.out.println ("No factorial numbers foun
 }
    scanner.close();
 }
}
```

4. Differences ~~Ao~~ Among class, Local, and Instanee variable:

Variable type	Scope	Access metod	Life time
class variable	shared across all instance of the class	Accessed via className.VariableName	Exists as long as the program runs
instance variable	Belongs to an objects	Accessed via object.variable_Name	Exists as long as the instanee exists
Local variable	Limited to a method /block	Accessed only within the method	Destroyed after metod execution

6.

Access modifiers in java are keywords used to define the accessibility or scope of a class, method or variable. They help in encapsulating data and restricting unauthorized access.

Comparison of Access modifiers

Modifier	class	package	subclass	world
public	yes	yes	yes	yes
private	yes	NO	NO	NO
protected	yes	yes	yes	NO
default	yes	yes	NO	NO

Types of variabe in java,

1. Local variable
2. Instance variable
3. Static variable

Example code:

```
class variableExample {
    int instancevar = 10;
    Static int staticvar = 20;
void show() {
    int Localvar = 30;
    System.out.println(" Local variable" + Localvar);
System.out.println("Instance variable" + Instancevar);
System.out.println("static variable" + staticvar);
}
    public static void main(string[] args) {
variableExample obj = new variable
                                Example();
obj.show();
} }
```

8.

To write a program that can determine whether an input character is a letter whitespace or digite, ~~[illegible]~~ Here is am Example:

```
public class characterchecker {
  public static void checkcharacter(char[] arr){
    for(char ch : arr) {
      if(character.isLetter(ch)) {
        System.out.println(" "+ch" is a letter");
      }
    else if(character.iswhitespace(ch)) {
        System.out.println(" "+ch"is a whitespace");

    else if(character.is digit(ch)) {
      System.out.println(" "+ch"is a digit");

    else { system.out.println(" "+ch" is a
                special character");

  } } }
```

```
public static void main (string[] args) {
char[] inputArray = {'a', '1', ' ', '#',
                     'B', 3};
checkCharacter(inputArray);
}
}
```

33.

Java code:

```
import java.util.*;
  public class main{ {
   public static void main(String[] args) {
    Scanner scanner= new Scanner(System.in);
     String s = scanner.next();
List <Integer>V = new ArrayList <>();
 while(scanner.hasNextInt()) {
     V.add(Scanner.nextInt());
  }
    collection.sort(v);
if(s.equals("smallest")) {
  System.out.println(v.get(0));
} else {
    System.out.println(v.get(v.size()
  }
      scanner.close();
  }
```

36.

```
interface Alpha {
void method1();
void method2(); }
interface Beta {
 Void method3
    Void method4()
 }
abstract class AbstractBase implements
 Alpha {
    abstract void method5();
 }
class Finalclass extends AbstractBase
 implements Beta {
  @Override
 public void method1() {
 System.out.println("Method 1 from
                    Alpha");  }
```

```
@Override
public void method2 () {
System.out.println("method 2 from Alpha");
}

@Override
public void method3(){
System.out.println("method 3 from Beta");
}

@override
public void method4 () {
System.out.println("metho 4 from Beta");
}

@override
public void method5 () {
System.out.println("method 5 from Abstract
}
}
```

it23018

```
public class main {
public static void main(string[] args)
Finalclass finalclass = new Finalclass()
finalclass.method1();
finalclass.method2();
finalclass.method3();
finalclass.method4();
finalclass.method5();
}
}
```

it 23018

```
public class main {

public static void main(string[] args){

FinalClass finalclass = new FinalClass();

    finalclass.method1();
    finalclass.method2();
    finalclass.method3();
    finalclass.method4();
    finalclass.method5();
```

4

3