Final Examination Preperation
Handwritten Assignments

1. Write a Java program that reads a series of numbers from a file (input.txt), determing the highest number in the series, calculates the sum of natural numbers up to that highest number, and writes the result to another file (output.txt). Use (scanner) to read from the file and (Printwriter) to write to the file. Assume the number in the input file are seperated by commas.

Sample In: 10, 55, 1000, - - - 10
sample output: 55, 1540, 500500, 5050

Code:

```
import java.io.*;
import java.util.*;
public class Numberprocessor{
Public static void main(string[]args) throws
    FileNotFoundException {
    Scanner scanner= new.Scanner(new File("input
    scanner.useDelimiter(",\s");
PrintWriter writer=new PrintWriter("Output.txt
While(scanner.hasNextInt()){
```

```
int sum= num *(num+1)/2;
writer.println(num+","+sum);
}
Scanner.close();
writer.close();
system.out.println("Processint complete. Check
    output.txt");
}
}
```

Q:2: What are the differences you have ever found between static and final fields and methods? Expemplify what will happen if you try to access the static method/field with the objec instead of class name.

Feature	Static	Final
Definition	Belongs to the class rather than an instance	Prevents modification or overriding
Scope	Shared among all instances of a class	Ensures immutability or Prevents method overriding
Modification	can be changed unless also declared [Final]	Cannot be changed once assigned
Memory Allocation	Stored in the Methd Aread, shared across instances	Stored per object instance unless also static
Inheritance	Can be inherited and accessed via class name	can be inherited but not overridden
Method behavior	can be called using class name	cannot be overridden by subclasses.

Q:3: Write a java program to find all factorion numbers within a given range. ~~A number is called factorion if the sum~~. The program should take user input for the lower and upper bounds of range and print all. in the range.

Sample Input:
Lower range: 1
Upper range: 100000
Output: 1, 2, 145, 40585

Ans:

```
import java.util.Scanner;
public class FactorionFinder{
Private static int[] factorial=new int[10];
static{
 factorial [0]=1;
for(int i=1; i<10;i++){
 factorial [i] = factorial [i-1]*i;
  }
}
private static int sumOfFactorialofDigits(int num){
 int sum=0, temp= num;
while (temp>0){
sum+= factorial [temp%10];
 temp/=10;
}
```

Q#: private static boolean isFactorion(int num)
{return num==sumofFactorialofDigits(num);
}
public static void main(String[]args){
Scanner scanner=new Scanner(System.in);
System.out.print("Enter the lower bound of the range:");
int lower=scanner.nextInt();
System.out.print("Enter the upper bound of the range:");
int upper=scanner.nextInt();
scanner.close();
System.out.println("Factorion numbers in the ra
boolean found=false;
for(int i=lower;i<=upper;i++){
if(isFactorion()){
System.out.print(i+" ");
found=true;
}
}
if(!found){
System.out.println("None");
}
}

Q-4: Distinguish the differeces among class, local and instance variables. What is significance of this keyword?

class variable	Instance Variable	Local variable
① Inside a class, but outside any method, with static keyword	① Inside a class but outside any method, without static	① Inside a method or block
② Shared among all instances of the class	② Belongs to an instance of the class	② Only within that method/block
③ Exists as long as the class is loaded	③ Exists as long as the object exists	③ Created when the method is called and destroyed when it ends.
④ Can have private, public, protected or default	④ Can have private, public, protected or default	④ NO access modifier
⑤ Private static final, int MAX=100;	⑤ No Private int age;	⑤ int count = 0;

Class:

* A class in java is blueprint for creating objects. It defines the properties and behaviors that objects of the class will have.

*Significance:

① Classes define the structure and behavior objects.

② They are the foundation of object-oriented programming in Java, encapsulating both data and methods that operate on the data.

③ You can create multiple objects from the same class.

Instance:

An instance in Java refers to an individual object that is created from a class. When a class is instantiated using the new keyword an instance is created.

significance:

① Each instance has its own set of values for the fields defined in the class

② Instance allow interaction with methods and access to data that is specific to that object.

Local:

Local in Java refers to the scope of a variable. A local variable is a variable defined within a method or a block of a code, and it is only accessible within that method or block.

Significance:

① Local variables are temporary and only exist during the execution of the method in which they are

② They must be initialized before use.

Q5: Write a Java program that defines a method to calculate the sum of all elements in an integer array.

Ans:

```
import java.util.Scanner;
public class ArraySum{
public static int calculatesum(int[]arr){
 int sum=0;
 for(int num:arr){
  sum+=num;
  }
  return sum;
public static void main (String[]args){
Scanner scanner=new scanner(system.in);
System.out.print("Enter the number of elements in the array:");
int n=scanner.nextInt();
int[]numbers=new int[n];
System.out.print("Enter the elements of the array;
for(int i=0;i<n;i++){
numbers[i]=scanner.nextInt();
 }
 int result=caculatesum(numbers);
System.out.println("The sum of all element:"+rout
 scanner.close();
}
```

Q-6: What is called access Modifiers? compare the accessibility of Public, Private and Protected modifiers. Describe different types of variable in Java with example.

An access modifier is a keyword in Java. that specifies the visibility or accessibility of classes, methods, constructors, and variable, It controls the level of access that other classes or components can have to a particular class or its members. Using access modifiers is essential for implementing encapsulation which helps to protect data and restricts unauthorized access.

Public:
Class: Accessible from any other class.
Package: Accessible within the same package.
Subclass: Accessible.
World: Accessible.

Private:
Class: Accessible within the same class only.
Package: Not accessible
Subclass: Not accessible.
World: Not accessible.

Protected:
Class: Accessible within the same class.
Package: Accessible within the same Package.
Subclass: Accessible.
World: Not accessible.

Different types of variables:

1. Instance variable.
2. Class variables
3. Local variables
4. Parameter variables

Q.8: Write a Program that can determine the letter, whitespace and digit then we use an array to a function write on compo

```
public class characterTypeChecker{
Public static void checkCharacterType(char c){
if(character.isLetter(c)){
 System.out.println(c+" is a letter");
else if (character.isWhitespace(c)){
 System.out.println(""+c+" is a whitespace");
else if (character.isDigit(c)){
System.out.println(c+" is a digit");}
else{
System.out.println(c+" is neither a letter nor");
 }
 }
Public static void main(String[] args){
 char[]characters={'a','3','Z','1','\n'};
 for(char c:characters){
   checkCharacterType(c);
 }
}
```

Q-9: In Java, explain how method overriding works in the content of inheritance. What happens a subclass overrides a method from its superclass? How does the super keyword help in calling the superclass method, and what are the potential issues when overriding methods, especially when dealing with constructors?

Ans:

In Java, method overriding occurs when a subclass provides its specific implementation of a method that is already defined in its superclass. This allows the subclass to modify or extend the behavior of the superclass method. Method overriding is a core concept in inheritance and is fundamental to polymorphism.

How Method overriding work:

1. Same method signature: When a subclass overrides a method, it must have the same method signature as the method in the superclass. This includes.
   - Same method name
   - Same parameter types
   - Same return type.
2. The override Annotation: The override annotation is not required, but it is highly recommended.

Q.10: Differentiate between static and non-static members including necessary examples. Write a program that able to check either a number or string is Pallindrome or not.

Static members	Non-static members
Declared with the static keyword.	Declared without the static keyword
Static members are allocated memory once when the class is loaded	Non-static members are allocated memory for each instance
can be accessed without creating an instance of the class	Can only be accessed by creating an object of the cl
Static members exist as long as the class is loaded in memory	Non-static members exist as long as the object exists.
Accessed using the class name (e.g. ClassName.Static method()	Accessed using an object reference (object.method(

Ex- Static:

```
class counter{
 Static int count = 0;
 Static void incrementcount(){
 count++;
 System.out.println("count: "+count); } }
public class Main{
 publicstatic void main(String[] args){
  counter.incrementcount();
 counter. incrementcount();
 System.out.println ("Final count: "+counter.count);
 } }
```

Ex-Non-Static:

```
class Person{
 String name;
 void setName (String name){
    this. name= name; }
 {void displayName() {
 System.out.println ("Name:"+name); }
 }
public class main{
 public static void main (String[] args){
   Person p1 = new person();
    P1.setName ("Alice");
    P1.displayName(); } }
```

Plaindrom checker:

```
import Java util.Scanner;
public class Plaindromchecker{
public class static void boolean is plaindrome(String input)
{ int start = 0;
  int end = input.length() - 1;
 While(start < end) {
  if (input.charAt(start) != input.charAt(end)){
   return false; }
   start++;
   end--;
  }
  return true;
  }
public static boolean isPalindrome (int number){
  int original = number;
  int reversed = 0;
 While (number>0) {
  int digit = number % 10
  ~~reserv~~reversed = reversed * 10 + digit;
  number /= 10; }
  return original == reversed; }
Public static void main (String[]args){
  Scanner scanner = new scanner (System.in);
System.out.print ("Enter a string to check if it
                   is a palindrome:");
```

```
String inputstring = Scanner.nextLine();
if(isPalindrome(inputstring)){
System.out.println("The string is a palindrome
}
else{
 System.out.println("The string is not a plaindr
}
System.out.print("Enter a number to check if
                  it is a palindrome:");
 int inputNumber = scanner.nextInt();
IF (isPalindrome(inputNumber)){
System.out.println("The number is a palindrom
}else{
System.out.println("The number is not palindr
}
Scanner.close();

}
}
```

Q 11: What is called data abstraction and encapsulation? Describe with the example. What are the differences between Abstract class and Interface?

Abstraction:

Abstraction is the concept of hiding the internal details of an object and exposing only the essential features. This helps in reducing complexity by hiding unnecessary details and only showing the essential characteristics.

Ex:

```
abstract class Animal {
public abstract void sound();
 public void eat() {
System.out.println("This animal is eatin.");}
}
class Dog extends Animal {
 @Override
public void sound() {
 System.out.println("Dog barks");}
 }
public class Main{
Public static void main(String[]args){
Animal dog = new dog(); dog.sound() } }
```

Encapsulation:

Encapsulation refers to the practice of wrapping data and methods into a single unit called a class. It hides the internal state of an object from the outside world and only exposes what is necessary through getter and setter method.

```
class Person{
private String name;
Privet int age;
public String getName(){
return name; }
public void setName(String name){
    this.name = name; }
{public int getAge(){
return age;
}
public void setAge(int age){
if(age>0){
   this.age = age; }
   else{
   System.out.println("Age must be a positive
                       number.");
}
```

```
}
public class main{
 public static void main(String[] args){
   Person person = new Person();
   person.setName ("John");
   person.setAge (25);
  System.out.println ("Name:" + person.getNa
System.out.println("Age: " + person.getAge()
}
}
```