**HESHAN SANDARUWAN**
**200304512443**

1.
a. public static void myMethod() { } ;
- This is a legal method declaration. It's a public static void method named "myMethod" with no parameters.
b. public static void main() { }
- This is a legal method declaration for the "main" method used as the entry point in a Java application.
c. public void static subMethod( );
- This is an illegal declaration. The modifiers "void" and "static" should be reversed, and method name should be provided. It should be "public static void subMethod()".
d. public static void () { }
- This is an illegal declaration. It lacks a method name.
e. public static void_();
- This is a legal method declaration, but it has an unconventional method name that starts with an underscore. It's not a recommended naming convention.
f. public static void _(){}
- This is a legal method declaration with an unconventional method name starting with an underscore.
g. public static void myMethod(int x;){ }
- This is an illegal declaration. The parameter should be defined as "int x" without a semicolon.
h. public static void myMethod(x) { }
- This is an illegal declaration. It should specify the data type of the parameter, e.g., "public static void myMethod(int x)".
i. public static void myNewMethod(100) { }
- This is an illegal declaration. Method parameters should have a data type and a name, e.g., "public static void myNewMethod(int parameterName)".
j. public static void m(int a){return 0;}
- This is an illegal declaration. The return type of the method is declared as "void," which means it shouldn't return a value (return 0; is not allowed). It should be "public static int m(int a)" if you want to return an integer value.
k. public static void m1(){return;}
- This is a legal method declaration with a "void" return type.
l. public static int me(int a){return 0;}
- This is a legal method declaration. It's a public static method named "me" that returns an integer value and takes an integer parameter "a"Here

**HESHAN SANDARUWAN**
**200304512443**

are the correct method declarations and the reasons for the illegal ones:

```
2. class Example {
public static String printName(String name) {
return name;
}
public static void main(String args[]) {
printName(); // Line 1 (Illegal)
printName("CMJD"); // Line 2 (Legal)
Example.printName("IJSE"); // Line 3 (Legal)
MyClass.printName("IJSE"); // Line 4 (Legal)
MyClass.printName(); // Line 5 (Illegal)
String name1 = MyClass.printName("CMJD"); // Line 6 (Legal)
String name2 = Example.printName(" "); // Line 7 (Legal)
String name3 = printName(); // Line 8 (Illegal)
}
}class MyClass {
public static void printName(String name) {
System.out.println("My Name is : " + name);
}
public static String printName() {
return "Java";
}
}
3. class Example {
public static void myMethod() {
System.out.println("My Method()...");
}
public static void main(String args[]) {
int myMethod; // Line 1 (Legal)
myMethod; // Line 2 (Illegal - Unused variable declaration)
myMethod(); // Line 3 (Illegal - You can't call a variable as a method)
myMethod(){ } // Line 4 (Illegal - Incorrect syntax for method declaration)
myMethod(){ }; // Line 5 (Illegal - Incorrect syntax for method declaration)
Example.myMethod(); // Line 6 (Legal)
System.out.println("myMethod()");// Line 7 (Legal)
System.out.println(myMethod()); // Line 8 (Illegal - myMethod is an int
variable, not a method)
}
}
```

4. c. Prints: 1, 2, 3, 4, 5, 3

5.

# HESHAN SANDARUWAN
# 200304512443

b. return true;
c. return avg >= 50;
d. if (avg >= 50) { return true; } else { return false; }
e. if (avg >= 50) { return true; }
f. return avg >= 50 ? true : false;
6.
c. prints 1 3 5 3 4 6
7.
b. Prints: 1, 2, 3, 4, 12,
8. legal method declarations are a, d, and g.
9. To avoid a compilation error, you need to choose a data type for y that matches the parameter type of the myMethod method, which is int. So, options (a), (b), and (c) are valid choices.
10.Line 1: getNumbers(); - This line will cause a compilation error because you are calling the method getNumbers() without providing any arguments, but there are two versions of the getNumbers method (overloaded). The compiler won't be able to determine which one to call, and this ambiguity results in a compilation error.
Line 8: return x,y; - This line will also cause a compilation error. In Java, you cannot return multiple values using a comma (,) as you are attempting to do in this line. A method can only return a single value.
Line 11 and Line 12: These lines may also cause compilation errors, depending on how they are called. If the arguments provided during the method call do not match any of the method signatures, you will get a compilation error. For example, if you call getTotal(10.0, 100);, you will get a compilation error because there is no matching method with the provided arguments.
11.
x : 100
x : 101
x : 101
x : 102
12.import java.util.*;
class Example {
public static boolean isEven(int number) {
return number % 2 == 0;
}
public static void main(String args[]) {
Random r = new Random();
for (int i = 0; i < 10; i++) {
int rand = r.nextInt(100);

**HESHAN SANDARUWAN**
**200304512443**

```
System.out.println(isEven(rand) ? rand + " is an even number" : rand
+ " is an odd number");
}
}
}
13.import java.util.*;
class Example {
public static boolean isPass(double avg) {
return avg >= 50;
}
public static void main(String args[]) {
Scanner input = new Scanner(System.in);
System.out.print("Input average marks : ");
double avg = input.nextDouble();
System.out.println(isPass(avg) ? "Pass" : "Fail");
}
}
14.import java.util.*;
class Example {
public static int abs(int number) {
return Math.abs(number);
}
public static void main(String args[]) {
Random r = new Random();
for (int i = 0; i < 10; i++) {
int rand = r.nextInt();
System.out.println("Absolute value of " + rand + " : " + abs(rand));
}
}
}
15.class Example {
public static String toBinaryString(int value) {
return Integer.toBinaryString(value);
}
public static String toOctalString(int value) {
return Integer.toOctalString(value);
}
public static String toHexString(int value) {
return Integer.toHexString(value);
}
public static void main(String args[]) {
```

**HESHAN SANDARUWAN**
**200304512443**

```
System.out.println(toBinaryString(100)); // 1100100
System.out.println(toOctalString(100)); // 144
System.out.println(toHexString(100)); // 64
}
}
```

16.
c. A
AA
17.
a. public static int m(int x, int y) { return x + y; }
b. public static float m(double value) { return (float)value; }
c. public static String m(float a, double b) { return a + " " + b; }
d. public static boolean m(String... names) { return names.length > 0; }
e. public static void m(double d) { /* Method with no return value */ }
f. public static boolean m(String s, double d) { return s.equals("true") && d > 0; }
g. public static String m() { return "break"; }
18.The code segment cannot be compiled successfully because the Math.pow(a, b) and Math.pow(b, a) methods return double values, but the f method is declared to return an int. You cannot directly assign a double to an int without casting.
19.Modularity: Methods allow breaking down complex code into smaller, manageable, and reusable pieces. This promotes code organization and maintenance.
Reusability: Once defined, methods can be called multiple times, reducing code duplication. Developers can reuse a method with different inputs.
Abstraction: Methods allow developers to abstract away implementation details and expose a high-level interface, making code more readable and maintainable.
Testing: Methods can be tested in isolation, simplifying the debugging process and improving code reliability.
Encapsulation: Methods can encapsulate logic, protecting it from unauthorized access and making code more secure.
Collaboration: In team development, methods enable different team members to work on specific components independently.
20.The method calling stack, also known as the call stack or execution stack, is a data structure used to manage method calls and returns during program execution. It follows the Last-In-First-Out (LIFO) principle. Here's how it works:
When a method is called, its information, including parameters and return address, is pushed onto the stack.

# HESHAN SANDARUWAN
# 200304512443

The method is executed, and if it calls another method, that method's information is pushed onto the stack. This process continues for nested calls.
When a method completes its execution, its information is popped off the stack, and the program returns to the calling method.
The stack continues to grow and shrink as methods are called and return, maintaining a record of the program's execution flow.
The method calling stack is crucial for managing the order of method calls and ensuring that the program can return to the correct calling context. It plays a significant role in managing program flow, memory allocation, and execution order.
21.
Parameters: Parameters are variables declared in a method's signature, and they act as placeholders for values that need to be provided when the method is called. Parameters define what data the method expects and can use within its scope.
Arguments: Arguments are the actual values or expressions that are passed to a method when it is called. These values are assigned to the parameters defined in the method's signature.
22.public class Factorial {
public static int factorial(int n) {
if (n == 0) {
return 1;
} else {
return n * factorial(n - 1);
}
}
public static void main(String[] args) {
int n = 5;
int result = factorial(n);
System.out.println("Factorial of " + n + " is " + result);
}
}
23.public class RectangleAreaCalculator {
public static double calculateRectangleArea(double length, double width)
{
return length * width;
}
public static void main(String[] args) {
double length = 5.0;
double width = 3.0;

**HESHAN SANDARUWAN**
**200304512443**

```
double area = calculateRectangleArea(length, width);
System.out.println("Rectangle Area: " + area);
}
}
24.import java.util.Scanner;
public class MarksCalculator {
public static void main(String[] args) {
int totalSubjects = 10;
int totalMarks = 0;
Scanner scanner = new Scanner(System.in);
for (int i = 1; i <= totalSubjects; i++) {
System.out.print("Enter marks for subject " + i + ": ");
int marks = scanner.nextInt();
totalMarks += marks;
}
double average = (double) totalMarks / totalSubjects;
System.out.println("Total Marks: " + totalMarks);
System.out.println("Average Marks: " + average);
}
}
25.import java.util.Scanner;
public class MaxOfThreeNumbers {
public static int findMax(int num1, int num2, int num3) {
int max = num1;
if (num2 > max) {
max = num2;
}
if (num3 > max) {
max = num3;
}
return max;
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the first number: ");
int number1 = scanner.nextInt();
System.out.print("Enter the second number: ");
int number2 = scanner.nextInt();
System.out.print("Enter the third number: ");
int number3 = scanner.nextInt();
int maximum = findMax(number1, number2, number3);
```

**HESHAN SANDARUWAN**
**200304512443**

```
System.out.println("The maximum of the three numbers is: " +
maximum);
}
}
26.import java.util.Scanner;
public class CircleAreaCalculator {
public static double calculateCircleArea(double radius) {
return Math.PI * Math.pow(radius, 2);
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the radius of the circle: ");
double radius = scanner.nextDouble();
double area = calculateCircleArea(radius);
System.out.println("The area of the circle with radius " + radius + " is: "
+ area);
}
}
27.public class SumFromToCalculator {
public static int sumFromTo(int first, int last) {
int sum = 0;
for (int i = first; i <= last; i++) {
sum += i;
}
return sum;
}
public static void main(String[] args) {
int first = 4;
int last = 7;
int result = sumFromTo(first, last);
System.out.println("The sum from " + first + " to " + last + " is: " +
result);
}
}
28.import java.util.Scanner;
public class SumOfDigitsCalculator {
public static int sumOfDigits(int number) {
int sum = 0;
while (number > 0) {
int digit = number % 10;
sum += digit;
```

```
number /= 10;
}
return sum;
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a number: ");
int number = scanner.nextInt();
int digitSum = sumOfDigits(number);
System.out.println("The sum of digits of " + number + " is: " +
digitSum);
}
}
```

29.

```
public class SquareNumbersDisplay {
public static void displaySquareNumbers(int limit) {
for (int i = 1; i * i <= limit; i++) {
int square = i * i;
System.out.println(square);
}
}
public static void main(String[] args) {
int limit = 50; // You can change this value to specify the limit
displaySquareNumbers(limit);
}
}
```

30.

```
public class ReverseDigits {
public static int reverseNumber(int number) {
int reversed = 0;
while (number != 0) {
int digit = number % 10;
reversed = reversed * 10 + digit;
number /= 10;
}
return reversed;
}
public static void main(String[] args) {
int originalNumber = 7631;
int reversedNumber = reverseNumber(originalNumber);
System.out.println("Original Number: " + originalNumber);
System.out.println("Reversed Number: " + reversedNumber);
}
```

**HESHAN SANDARUWAN**
**200304512443**

```
}
31.public class ArmstrongNumberChecker {
public static boolean isArmstrong(int number) {
int originalNumber = number;
int sum = 0;
while (number > 0) {
int digit = number % 10;
sum += Math.pow(digit, 3);
number /= 10;
}
return sum == originalNumber;
}
public static void main(String[] args) {
int number = 371; // You can change this number
boolean isArmstrong = isArmstrong(number);
if (isArmstrong) {
System.out.println(number + " is an Armstrong number.");
} else {
System.out.println(number + " is not an Armstrong number.");
}
}
}
32.public class SmallestMultipleFinder {
public static long findSmallestMultiple(int range) {
long result = 1;
for (int i = 2; i <= range; i++) {
result = lcm(result, i);
}
return result;
}
public static long lcm(long a, long b) {
return (a * b) / gcd(a, b);
}
public static long gcd(long a, long b) {
if (b == 0) {
return a;
}
return gcd(b, a % b);
}
public static void main(String[] args) {
int range = 20; // You can change this value
```

**HESHAN SANDARUWAN**
**200304512443**

```
long smallestMultiple = findSmallestMultiple(range);
System.out.println("The smallest positive number divisible by all
numbers from 1 to " + range + " is: " + smallestMultiple);
}
}
33.public class OrderChecker {
public static boolean isOrdered(int a, int b, int c) {
return (a < b && b < c) || (a > b && b > c);
}
public static void main(String[] args) {
int num1 = 1;
int num2 = 2;
int num3 = 3;
boolean ordered = isOrdered(num1, num2, num3);
if (ordered) {
System.out.println("The numbers are in ascending or descending
order.");
} else {
System.out.println("The numbers are not in ascending or descending
order.");
}
}
}
34.import java.util.Scanner;
public class LeapYearChecker {
public static boolean isLeapYear(int year) {
return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a year: ");
int year = scanner.nextInt();
if (isLeapYear(year)) {
System.out.println(year + " is a leap year.");
} else {
System.out.println(year + " is not a leap year.");
}
}
}
35.import java.util.Scanner;
public class FibonacciSeries {
```

**HESHAN SANDARUWAN**
**200304512443**

```
public static void printFibonacciSeries(int n) {
int first = 0, second = 1;
System.out.print("Fibonacci Series up to " + n + ": ");
while (first <= n) {
System.out.print(first + " ");
int next = first + second;
first = second;
second = next;
}
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a number to limit the Fibonacci series: ");
int limit = scanner.nextInt();
printFibonacciSeries(limit);
}
}
36.public class TaxCalculator {
public static int calculateTax(int income) {
int tax = 0;
if (income > 500000) {
tax += (income - 500000) * 0.2; // 20% tax on the amount above
500,000
income = 500000;
}
if (income > 100000) {
tax += (income - 100000) * 0.1; // 10% tax on the amount above
100,000
}
return tax;
}
public static void main(String[] args) {
int income = 550000; // You can change this income
int tax = calculateTax(income);
System.out.println("Tax for an income of " + income + " CR is: " + tax +
" CR");
}
}
37.public class PalindromeChecker {
public static boolean isPalindrome(int number) {
int original = number;
```

**HESHAN SANDARUWAN**
**200304512443**

```
int reversed = 0;
while (number > 0) {
int digit = number % 10;
reversed = reversed * 10 + digit;
number /= 10;
}
return original == reversed;
}
public static void main(String[] args) {
int num = 121; // You can change this number
boolean isPalindromic = isPalindrome(num);
if (isPalindromic) {
System.out.println(num + " is a palindrome.");
} else {
System.out.println(num + " is not a palindrome.");
}
}
}
38.public class DecimalToBinaryConverter {
public static void decimalToBinary(int decimal) {
if (decimal == 0) {
System.out.println("Binary: 0");
return;
}
int[] binaryArray = new int[32];
int index = 0;
while (decimal > 0) {
binaryArray[index++] = decimal % 2;
decimal /= 2;
}
System.out.print("Binary: ");
for (int i = index - 1; i >= 0; i--) {
System.out.print(binaryArray[i]);
}
}
public static void main(String[] args) {
int decimalNumber = 10; // You can change this number
decimalToBinary(decimalNumber);
}
}
39.import java.util.Scanner;
```

**HESHAN SANDARUWAN**
**200304512443**

```
public class ShapeAreaCalculator {
public static double calculateRectangleArea(double length, double width)
{
return length * width;
}
public static double calculateCircleArea(double radius) {
return Math.PI * Math.pow(radius, 2);
}
public static double calculateTriangleArea(double base, double height) {
return 0.5 * base * height;
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int choice;
do {
System.out.println("Choose a shape:");
System.out.println("1. Rectangle");
System.out.println("2. Circle");
System.out.println("3. Triangle");
System.out.println("4. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
switch (choice) {
case 1:
System.out.print("Enter the length of the rectangle: ");
double length = scanner.nextDouble();
System.out.print("Enter the width of the rectangle: ");
double width = scanner.nextDouble();
double rectangleArea = calculateRectangleArea(length, width);
System.out.println("Area of the rectangle: " + rectangleArea);
break;
case 2:
System.out.print("Enter the radius of the circle: ");
double radius = scanner.nextDouble();
double circleArea = calculateCircleArea(radius);
System.out.println("Area of the circle: " + circleArea);
break;
case 3:
System.out.print("Enter the base of the triangle: ");
double base = scanner.nextDouble();
System.out.print("Enter the height of the triangle: ");
```

**HESHAN SANDARUWAN**
**200304512443**

```
double height = scanner.nextDouble();
double triangleArea = calculateTriangleArea(base, height);
System.out.println("Area of thetriangle: " + triangleArea);
break;
case 4:
System.out.println("Exiting the program.");
break;
default:
System.out.println("Invalid choice. Please choose again.");
}
} while (choice != 4);
}
}
```

40.

```
public class SmallestPositiveInteger {
public static int enough(int goal) {
int n = 1;
int sum = 0;
while (sum < goal) {
sum += n;
n++;
}
return n - 1;
}
public static void main(String[] args) {
int goal1 = 9;
System.out.println("The smallest positive integer for which 1+2+3+...+n
is at least 9: " + enough(goal1));
int goal2 = -7;
System.out.println("The smallest positive integer for which 1+2+3+...+n
is at least -7: " + enough(goal2));
}
}
```

41.

```
public class InvestmentCalculator {
public static double calculateFutureValue(double
initialAmount, double annualInterestRate, int numberOfYears)
{
double futureValue = initialAmount * Math.pow(1 +
annualInterestRate / 100, numberOfYears);
return futureValue;
}
public static void main(String[] args) {
```

**HESHAN SANDARUWAN**
**200304512443**

```
double initialAmount = 1000.0; // You can change the initial amount
double annualInterestRate = 5.0; // You can change the annual interest
rate
int numberOfYears = 5; // You can change the number of years
double futureValue = calculateFutureValue(initialAmount,
annualInterestRate, numberOfYears);
System.out.println("The future value of the investment is: " +
futureValue);
}
}
42.public class RecursiveValueFinder {
public static double m(double d) {
return 3 * d + 1;
}
public static double findValue(int n) {
double a = 0;
for (int i = 1; i <= n; i++) {
a = m(a);
}
return a;
}
public static void main(String[] args) {
int n = 4;
double result = findValue(n);
System.out.println("The value of a" + n + " is: " + result);
}
}
43.public class ConsecutiveNumbersChecker {
public static boolean areConsecutiveNumbers(int num1, int num2, int
num3) {
int[] numbers = {num1, num2, num3};
Arrays.sort(numbers);
return (numbers[1] - numbers[0] == 1 && numbers[2] - numbers[1] ==
1);
}
public static void main(String[] args) {
int a = 5, b = 4, c = 6; // You can change the values
boolean result = areConsecutiveNumbers(a, b, c);
if (result) {
System.out.println("The numbers are consecutive.");
} else {
```

```
System.out.println("The numbers are not consecutive.");
}
}
}
44.public class HexValueCalculator {
public static int hexValue(char c) {
if (Character.isDigit(c)) {
return Character.digit(c, 16);
} else {
char upperC = Character.toUpperCase(c);
if (upperC >= 'A' && upperC <= 'F') {
return 10 + (upperC - 'A');
}
}
return -1;
}
public static void main(String[] args) {
char character = 'F'; // You can change the character
int value = hexValue(character);
if (value != -1) {
System.out.println("The hexadecimal value of '" + character + "' is: "
+ value);
} else {
System.out.println("'" + character + "' is not a legal hexadecimal
digit.");
}
}
}
45.import java.util.Scanner;
public class HexToDecimalConverter {
public static int hexValue(char c) {
if (Character.isDigit(c)) {
return Character.digit(c, 16);
} else {
char upperC = Character.toUpperCase(c);
if (upperC >= 'A' && upperC <= 'F') {
return 10 + (upperC - 'A');
}
}
return -1;
}
```

**HESHAN SANDARUWAN**
**200304512443**

```
public static int hexToDecimal(String hexString) {
int value = 0;
for (int i = 0; i < hexString.length(); i++) {
int digitValue = hexValue(hexString.charAt(i));
if (digitValue == -1) {
return -1; // Not a valid hexadecimal string
}
value = value * 16 + digitValue;
}
return value;
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a hexadecimal string: ");
String hexString = scanner.next();
int decimalValue = hexToDecimal(hexString);
if (decimalValue != -1) {
System.out.println("Decimal value: " + decimalValue);
} else {
System.out.println("Invalid hexadecimal string.");
}
}
}
```