LAB ASSIGNMENT # 08

Method

CSE110 | Programming Language I

	LAB TASKS	HOME TASKS
CODING	03	03
TRACING	01	02

NOTE: You need to submit only the Home Tasks. Submit all the Flowchart or Tracing tasks hand drawn or handwritten, respectively to your Lab Instructors before the next lab. Submit all the Homework Coding Tasks in the Google Form shared on buX.

Lab Tasks

(No Need to Submit)

Task 01

[A,B,C,D should be written in a single java file]

A. Write a method called **evenChecker** that takes an **integer** number as its argument and prints whether the number is even or odd **inside the method**.

Sample Method Call	Sample Output
`evenChecker(10);`	`Even!!`
`evenChecker(17);`	`Odd!!`

B. Write a method called **isEven** that takes an **integer** number as an argument and **returns** boolean true if the number is even otherwise **returns** boolean false.

Sample Method Call	Sample Output
`boolean result = isEven(10);` `System.out.println( result );`	`true`
`boolean result = isEven(17);` `System.out.println( result );`	`false`

C. Write a method called **isPos** that takes an **integer** number as an argument and **returns** boolean true if the number is positive otherwise **returns** boolean false.

Sample Method Call	Sample Output
`boolean result = isPos(-5);` `System.out.println( result );`	`false`
`boolean result = isPos(12);` `System.out.println( result );`	`true`

D. Write a method called **sequence()** that takes an **integer** in its parameter called n. Now, if n is **positive** then it prints all the **even** numbers from **0 to n**, otherwise if n is **negative** it prints all the **odd** numbers from **n to -1**.

Note: **You must call** the methods from **1B** and **1C**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output	Explanation
`sequence(10);`	`0 2 4 6 8 10`	Here, 10 is positive so 0,2,4,6,8,10 were printed.
`sequence(-7);`	`-7 -5 -3 -1`	Here, -7 is negative so -7,-5,-3,-1 were printed.
`sequence(7);`	`0 2 4 6`	Here, 7 is positive so 0,2,4,6 were printed
`sequence(-8);`	`-7 -5 -3 -1`	Here, -8 is negative so -7,-5,-3,-1 were printed.

Task02
[A,B,C should be written in a single java file]

A. Write a method called **circleArea** that takes an **integer** radius in its parameter and **returns** the **area** of the circle.

Note: area of a circle is πr^2

Sample Method Call	Sample Output
`double area = circleArea(5);` `System.out.println(area);`	`78.5398`

B. Write a method called **sphereVolume** that takes a **double** radius in its parameter and **returns** the **volume** of the sphere.

Note: volume of a sphere is $\frac{4}{3}\pi r^3$

Sample Method Call	Sample Output
`double volume = sphereVolume(5.0);` `System.out.println(volume);`	`523.5987`

C. Write a method called **findSpace** that takes two values in its parameters one is an **integer** diameter and another one is a String. Using the given diameter, this method should calculate the Area of a circle or the Volume of a sphere depending on the value of the second parameter. Finally, it should print the result **inside the method**.

Note: **You must call** the method written in task **2A & 2B**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output
`findSpace(10,"circle");`	`78.5398`
`findSpace(5,"sphere");`	`65.4498`
`findSpace(10,"square");`	`Wrong Parameter`

Task03
[A,B should be written in a single java file]

A. Write a method called **isTriangle** that takes 3 integer numbers as arguments. The method will **return** the boolean true if the 3 sides can form a valid triangle otherwise it'll **return** the boolean false.

Note: In a valid triangle, the sum of **any** two sides will be greater than the third side.

Sample Method Call	Sample Output	Explanation
`boolean res = isTriangle(7,5,10);` `System.out.println( res );`	`true`	Here, 7+5>10, 5+10>7 also, 10+7>5. Thus, these 3 sides can form a valid triangle.
`boolean res = isTtriangle(3,2,1);` `System.out.println( res );`	`false`	Here, 1+2<=3, thus, these 3 sides can NOT form a valid triangle.

B. Write a method called **triArea** that takes 3 sides of a triangle as 3 **integer** arguments. The method should calculate and print the area of the triangle only if it's a valid triangle otherwise print that it's not a valid triangle.
Area of triangle = √[s(s−a)(s−b)(s−c)], where 's' is the semi perimeter of the triangle. So, semi-perimeter = s = perimeter/2 = (a + b + c)/2.

Note: **You must call** the method written in task **3A**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output	Explanation
`triArea(3,2,1);`	`Can't form triangle`	Here, 1+2<=3, thus, these 3 sides can NOT form a valid triangle.
`triArea(7,5,10);`	`16.248`	Here, 7,5,10 is able to form a valid triangle so, using the formula we get the area as 16.248

Task04

Trace the following code to generate the outputs. Show the necessary trace table.

```
1  public class Trace1{
2      public static int f3(int a, int c){
3          System.out.println("F3 begins");
4          System.out.println(a+c);
5          System.out.println("F3 ends soon");
6          return a*5;
7      }
8      public static String f1(int n){
9          System.out.println("F1 begins");
10         f2(5,n);
11         System.out.println("F1 ends soon");
12         return n+1+" from F1";
13     }
14     public static void main(String[] args){
15         System.out.println("Starting soon...");
16         int c = 99;
17         String s = f1(c);
18         System.out.println(s);
19         System.out.println("The End");
20     }
21     public static void f2(int c, int d){
22         System.out.println("F2 begins");
23         System.out.println( f3(c+1,d-1) );
24         System.out.println("F2 ends");
25     }
26 }
```

Home Tasks

Task01

[A,B,C should be written in a single java file]

A. Write a method called **isPrime** which takes an integer in its parameter to check whether a number is prime or not. If the number is prime then the method returns boolean **true** otherwise it returns boolean **false**.

Sample Input	Sample Output
`boolean check = isPrime(7);` `System.out.println(check);`	`true`
`boolean check = isPrime(15);` `System.out.println(check);`	`false`

B. Write a method called **isPerfect** which takes an integer in its parameter to check whether a number is perfect or not. If the number is perfect then the method returns boolean **true** otherwise it returns boolean **false**.

Sample Input	Sample Output
`boolean check = isPerfect(6);` `System.out.println(check);`	`true`
`boolean check = isPerfect(33);` `System.out.println(check);`	`false`

C. Write a method called **special_sum** that calculates the sum of all numbers that are either prime numbers or perfect up till the integer value given in its parameter. This integer value must be taken as user input and passed into the method.

Note: **You must call** the methods written in task **1A & 1B**, otherwise this task will be **considered invalid**.

Sample Input	Sample Output	Output
8	`int result = special_sum(8);` `System.out.println(result);`	23
Explanation:	Between 1 to 8 the Prime numbers are 2,3,5,7 and 6 is a Perfect number. So, the summation is 2+3+5+7+6=23.	

Task02

[A,B,C should be written in a single java file]

A. Write a simple method called **showDots** that takes a number as an argument and then prints that amount of dots inside the method.
Note: You can use `System.out.print()` to avoid the next output being printed on the next line.

Sample Method Call	Sample Output
`showDots(5);`	.....
`showDots(3);`	...

B. Write a method called **show_palindrome** that takes a number as an argument and then prints a palindrome inside the method.
Note: You can use `System.out.print()` to avoid the next output being printed on the next line

Sample Method Call	Sample Output
`show_palindrome(5)`	123454321
`show_palindrome(3)`	12321

C. Write a method called **showDiamond** that takes an integer number as an argument and then prints a **palindromic diamond shape**. Moreover, the empty spaces surrounding the diamonds are filled with dots(**.**) .

Note: **You must call** the methods written in task **2A & 2B**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output

`showDiamond(5)`	`....1....`
	`...121...`
	`..12321..`
	`.1234321.`
	`123454321`
	`.1234321.`
	`..12321..`
	`...121...`
	`....1....`

`showDiamond(3)`	`..1..`
	`.121.`
	`12321`
	`.121.`
	`..1..`

Task03
[A,B should be written in a single java file]

A. Write a method called **calcTax** that takes 2 arguments which are **your age** then **your salary**. The method must calculate and **return** the tax as per the following conditions:

- No tax if you are less than 18 years old.
- No tax if you get paid less than 10,000
- 7% tax if you get paid between 10,000 and 20,000 (both inclusive)
- 14% tax if you get paid more than 20,000

Sample Method Call	Output	Explanation
`double t = calcTax(16,20000);` `System.out.println(t);`	`0.0`	Here, the age is less than 18 so 0 tax.
`double t = calcTax(20,18000);` `System.out.println(t);`	`1260.0`	Here, the age is greater than 18 and income is between 10K-20K so tax is 7% of 18000 = 1260.

B. Write a method called **calcYearlyTax** that takes no arguments. Inside the method it should take **first input as your age** and then **12 other inputs** as income of each month of the year. The method must calculate and print Tax for each month and finally print the total Tax of the whole year based on the **6A conditions**.

Note: **You must call** the method written in task **3A**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Input	Sample Output
calcYearlyTax()	22 8000 15000	Month1 tax: 0 Month2 tax: 1050.0 Month3 tax: 3080.0

	22000 2300 15300 21000 34000 9000 27000 88000 32000 7300	Month4 tax: 0 Month5 tax: 1071.0 Month6 tax: 2940.0 Month7 tax: 4760.0 Month8 tax: 0 Month9 tax: 3780.0 Month10 tax: 12320.0 Month11 tax: 4480.0 Month12 tax: 0 Total Yearly Tax: 33481.0

Task04

Trace the following code to generate the outputs. Show the necessary trace table.

```
1  public class Trace2{
2      public static void main (String[] args){
3          int a = 4, b = 7;
4          System.out.println(methodA(a,b));
5      }
6      public static double methodA(int m, int n){
7          int p = m+n-23, s = 0;
8          if (p<0){
9              System.out.println(p);
10             System.out.println(methodB(p+10));
11             s = methodB(p-10);
12         }
13         System.out.println(--s);
14         return p*m+s;
15     }
16     public static int methodB(int r){
17         int q = 6;
18         System.out.println(++r + q);
19         return q-r;
20     }
21 }
```

Task05

Trace the following code to generate the outputs. Show the necessary trace table.

```
1  public class Trace3{
2      public static boolean met1(int n1, int n2){
3          System.out.println("Method 1");
4          int n = n1+n2;
5          System.out.println(n);
6          return met3(n, n2)>n1;
7      }
8      public static int met2(int n, String s){
9          int p = 5;
10         System.out.println("Method 2");
11         System.out.println(met1(n,p));
12         return s.length();
13     }
14     public static double met3(int n, int p){
15         System.out.println("Method 3");
16         System.out.println(n/p);
17         return p;
18     }
19     public static void main (String[] args){
20         System.out.println("Main Method");
21         System.out.println(met2(6,"ABC"));
22     }
23 }
```