Experiment No: 1 Date:

Name of the Experiment: Write a java program to find the prime numbers in a given range

Code:

```java
package Rayhan;
import java.util.Scanner;
public class PrimeNumber {public static int primeNumber(int n)
{int c = 2, i, j;
for (c = 2; c <= n - 1; c++)
{if ((n % c) == 0){
break;}}
if (c == n){return n;}
else {return 0;}
}
public static void main(String[] args) {
int n1, n2, c = 2, i, j;
Scanner s = new Scanner( source: System.in);
System.out.print( s: "Enter the starting number : ");
n1 = s.nextInt();
System.out.print( s: "Enter the ending Number: ");
n2 = s.nextInt();
System.out.print("The prime numbers between the interval "
+ n1 + " and " + n2 + " are: ");
for (i = n1; i <= n2; i++)
{if (primeNumber( n: i) == 0){
continue;}else
{System.out.print(primeNumber( n: i) + ", ");}}}}
```

Output:

```
Output - Run (PrimeNumber)
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Enter the starting number : 10
Enter the ending Number: 50
The prime numbers between the interval 10 and 50 are: 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  12.636 s
Finished at: 2023-06-07T23:55:50+06:00
------------------------------------------------------------------------
```

Experiment No: 2 Date:

Name of the Experiment: Write a java program to find the nth number in a Fibonacci series.

Code:

```java
package Rayhan;
import java.util.Scanner;
public class FibonacciSeries {
public static void main(String args[]) {
int f_num= 0, s_num= 1, temp = 0;
Scanner scan= new Scanner( source: System.in);
System.out.print( s: "Enter a number of terms of series: ");
int num= scan.nextInt();
System.out.print( s: "Enter a number to find nth Fibonacci number: ");
int nth= scan.nextInt();
System.out.print("Fibonacci series till " + num + " terms: ");
for(int i=1; i<= num; i++){
System.out.print(f_num + ", ");
if(i == nth){
temp= f_num;
}
int n_num= f_num + s_num;
f_num= s_num;
s_num= n_num;
}
System.out.print("\nThe nth Fibonacci number of this series: " + temp);}}
```

Output:

```
Output - Run (FibonacciSeries)
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Enter a number of terms of series: 15
Enter a number to find nth Fibonacci number: 3
Fibonacci series till 15 terms: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,
The nth Fibonacci number of this series: 1
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  11.853 s
```

Experiment No: 3 Date:

Name of the Experiment: Write a program to give example for abstract class in java.

Code:

```java
package Rayhan;
abstract class Shape{
abstract void draw();}
class Rectangle extends Shape{
void draw(){System.out.println( x:"Draw a rectangle");}}
class Circle extends Shape{void draw(){
System.out.println( x:"Draw a circle");}}
class AbstractClass{
public static void main(String[] args){
System.out.println();
Shape C =new Circle();
C.draw();
Shape R =new Rectangle();
R.draw();}}
```

Output:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---

Draw circle
Draw a rectangle
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.012 s
Finished at: 2023-06-09T11:14:19+06:00
------------------------------------------------------------------------
```

Experiment No: 4 Date:

Experiment Name: Write a Java Program applying method overloading.

Code:

```java

package Rayhan;
class TestClass{
void test(int a){
System.out.println("a: " + a);
}
void test(int a, int b){
System.out.println("a and b: " + a + " " + b);
}
double test(double a){
System.out.println("double a: " + a);
return a*a;
}
}
public class Overlodding {
public static void main(String args[]) {
TestClass ob= new TestClass();
ob.test(a:10);
ob.test(a:10, b:20);
System.out.println("Return value of ob.test(10.10)" + ob.test(a:10.10));
}
}

```

Output:

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
a: 10
a and b: 10 20
double a: 10.1
Return value of ob.test(10.10)102.00999999999999
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.144 s
```

Experiment No: 5 Date:

Name of the Experiment: Write a java program to give the example for method overriding concepts.

Code:

```java
package Rayhan;
class Human {
public void think()
{
System.out.println( x: "Human is thinking.");
}
}
class Boy extends Human{
@Override
public void think(){
System.out.println( x: "He is overthinking.");
}

}
public class Overriding {
    public static void main( String args[]) {
Human a = new Human();
Human b = new Boy();
a.think();
b.think();
}}
```

Output:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Human is thinking.
He is overthinking.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.309 s
Finished at: 2023-06-08T23:26:18+06:00
------------------------------------------------------------------------
```

Experiment No: 6 Date:

Name of the Experiment: Write a program to give example for multilevel inheritance in Java.

Code:

```java
package Rayhan;
class Animal{
void eat(){
System.out.println(x:"Eating");}}
class Dog extends Animal{
void bark(){
System.out.println(x:"Barking");}}
class BabyDog extends Dog{
void weep(){
System.out.println(x:"Weeping");}}
public class MultilevelInheritance {
    public static void main(String[] args){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();}}
```

Output:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Weeping
Barking
Eating
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.728 s
Finished at: 2023-06-08T23:39:09+06:00
------------------------------------------------------------------------
```

Experiment No: 7 Date:

Name of the Experiment: Write a program to give example for multiple inheritance in Java.

Code:

```java
package Rayhan;
interface Printable{
void print();}
interface Readable{
void read();}
class A4 implements Printable,Readable{
@Override
public void print() {
System.out.println(x:"My");}
@Override
public void read(){
System.out.println(x:"Love");}}
public class MultipleInheritance {
public static void main(String[] args){
System.out.println();
A4 obj = new A4();
obj.print();
obj.read();}}
```

Output:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---

My
Love
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.951 s
```

Experiment No: 8 Date:

Name of the Experiment: Write a program to implement the concept of Encapsulation in java.

Code:

```java
package Rayhan;
class Student{
private String name;
private String id;
public String getName(){
return name;}
public String getId(){
return id;}
public void setName(String name){
this.name=name;}
public void setId(String id){
this.id=id;}}
class TestEncapsulation{
public static void main(String[] args){
System.out.println();
Student s=new Student();
s.setName( name: "Rayhan Hussain");
s.setId( id: "126/21|CSE-27");
System.out.println( x: s.getName());
System.out.println( x: s.getId());}}
```

Output:

```
Output - Run (TestEncapsulation)
Rayhan Hussain
126/21|CSE-27
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.183 s
Finished at: 2023-06-09T00:14:42+06:00
------------------------------------------------------------------------
```

Experiment No: 09 Date:

Name of the Experiment: Write a program to create a class named shape. In this class we have three sub classes circle, triangle and square each class has two members function named draw () and erase (). Create these using polymorphism concepts.

Code:

Polymorphism.java

```java
package Rayhan;
class Shape{public void draw() {System.out.println("Draw");}
public void erase(){System.out.println("Erase");}}
class Circle extends Shape{public void draw() {
System.out.println("Draw circle");}
public void erase(){System.out.println("Erase circle");}}
class Triangle extends Shape{
public void draw(){System.out.println("Draw triangle");}
public void erase(){System.out.println("Erase triangle");}}
class Square extends Shape{
public void draw(){System.out.println("Draw square");}
public void erase(){System.out.println("Erase square");}}
public class Polymorphism {
    public static void main(String[] args){
Shape C=new Circle();
C.draw();
C.erase();
Shape Tr=new Triangle();
Tr.draw();
Tr.erase();
Shape Sq=new Square();
Sq.draw();
Sq.erase();}}
```

Output:

Output - Run (Polymorphism)

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overloading ---
Draw circle
Erase circle
Draw triangle
Erase triangle
Draw square
Erase square
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.013 s
```

Experiment No: 10 Date:

Name of the Experiment: Write a java program for example of multiple catch statements occurring in a program.

Code:

```java
package Rayhan;
class MultipleCatchBlock {
public static void main(String[] args) {
try{
int ar[]=new int[5];
ar[5]=20/0;
}
catch(ArithmeticException e)
{
System.out.println( x: "Arithmetic Exception occurs");
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("ArrayIndexOutOfBounds Exception
occurs");
}
catch(Exception e)
{
System.out.println( x: "Parent Exception occurs");
}
System.out.println( x: "rest of the code");
}
}
```

Output:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Arithmetic Exception occurs
rest of the code
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.844 s
Finished at: 2023-06-09T11:02:20+06:00
------------------------------------------------------------------------
```

Experiment No: 11 Date:

Name of the Experiment: Write a program to create a package named mypack and import it in circle class.

Code:

Circle.java

```java
package Rayhan;
public class Circle
{double r=2;
public void area()
{System.out.println("Area of the circle = " + (3.14 * r *r));}}
```

Mypack.java

```java
import Rayhan.Circle;
public class Mypack{
public static void main(String[] args)
{Rayhan.Circle c = new Rayhan.Circle();
c.area();}}
```

Output:

Output - Run (Mypack)

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Overlodding ---
Area of the circle = 12.56
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.655 s
Finished at: 2023-06-09T10:43:42+06:00
------------------------------------------------------------------------
```