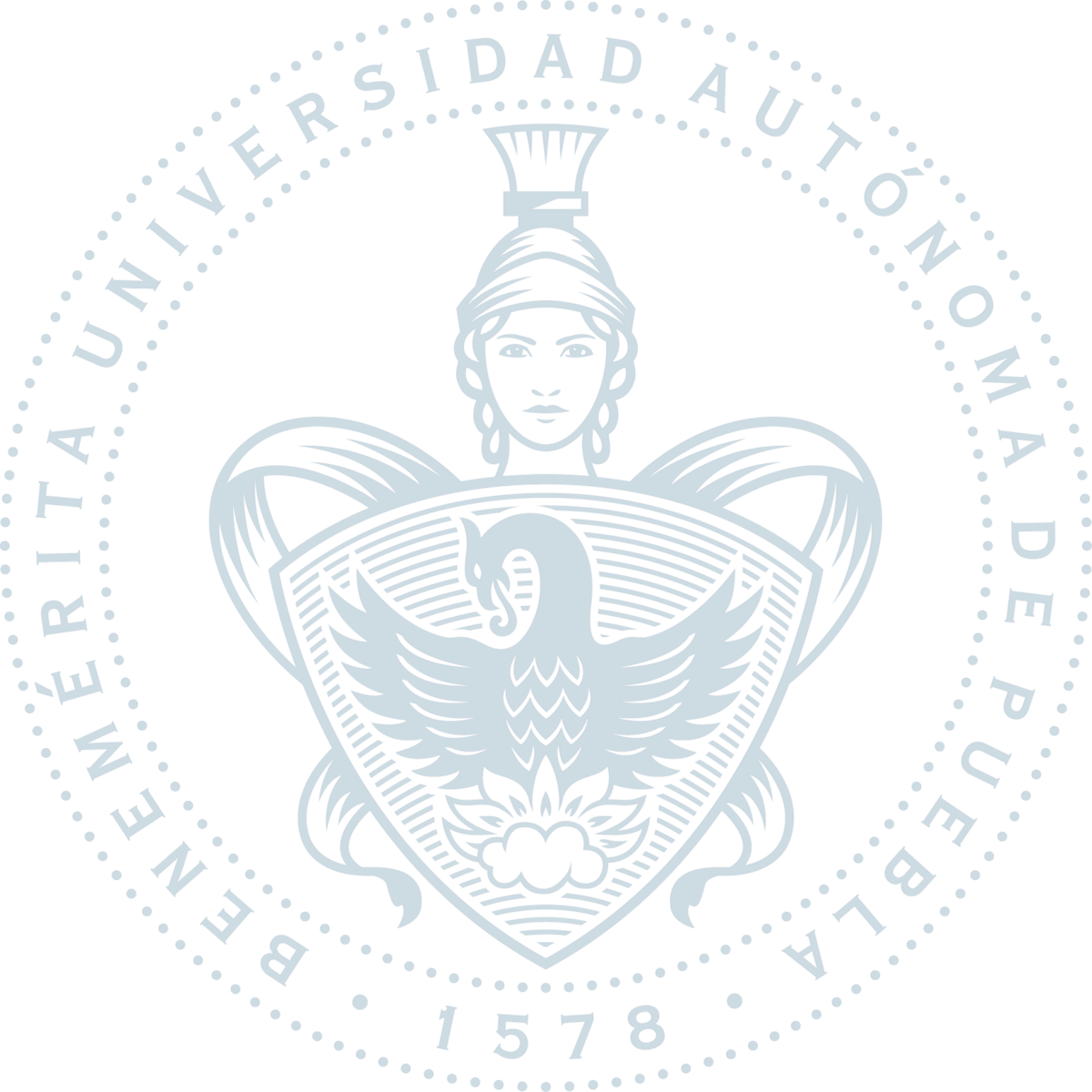
**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

BUAP

** FACULTAD DE CIENCIAS DE LA COMPUTACIÓN**

**INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN**

**PROGRAMACIÓN 2**

**“PRACTICA 17A. REPORTE RECURSIVIDAD”**

**PROFESOR:**

**DRA. CARMEN CERÓN GARNICA.**

**EQUIPO:  
BRENDA LIZETH COCA GARCIA**

**MATRICULA:201917154**

**BRENDA AILED RODRIGUEZ COLIS**

**MATRICULA:201910848**

**MIGUEL CARREON VAZQUEZ**

**MATRICULA: 201915389**

**IRVYN XICALE CARRERA**

**MATRICULA: 201963582**

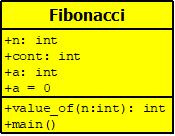
**INTRODUCCIÓN**La recursividad es una técnica potente de programación que puede utilizarse en lugar de la iteración para resolver determinados tipos de problemas.

Un método es recursivo cuando entre sus instrucciones se encuentra una llamada a sí mismo. La solución iterativa es fácil de entender.

En esta practica ponemos a prueba los conocimientos adquiridos en recursividad para crear un programa con métodos recursivos.

**PROGRAMA NÚMEROS DE FIBONACCI**

**DIAGRAMA UML**

****

**CLASE FIBONACCI**

//Fecha:3-Mayo-2021  
//Objetivo: Crear arreglo recursivo  
//Autor: Brenda Coca Garcia

import java.io.\*;  
import java.lang.\*;  
import java.util.\*;

class Fibonacci

{

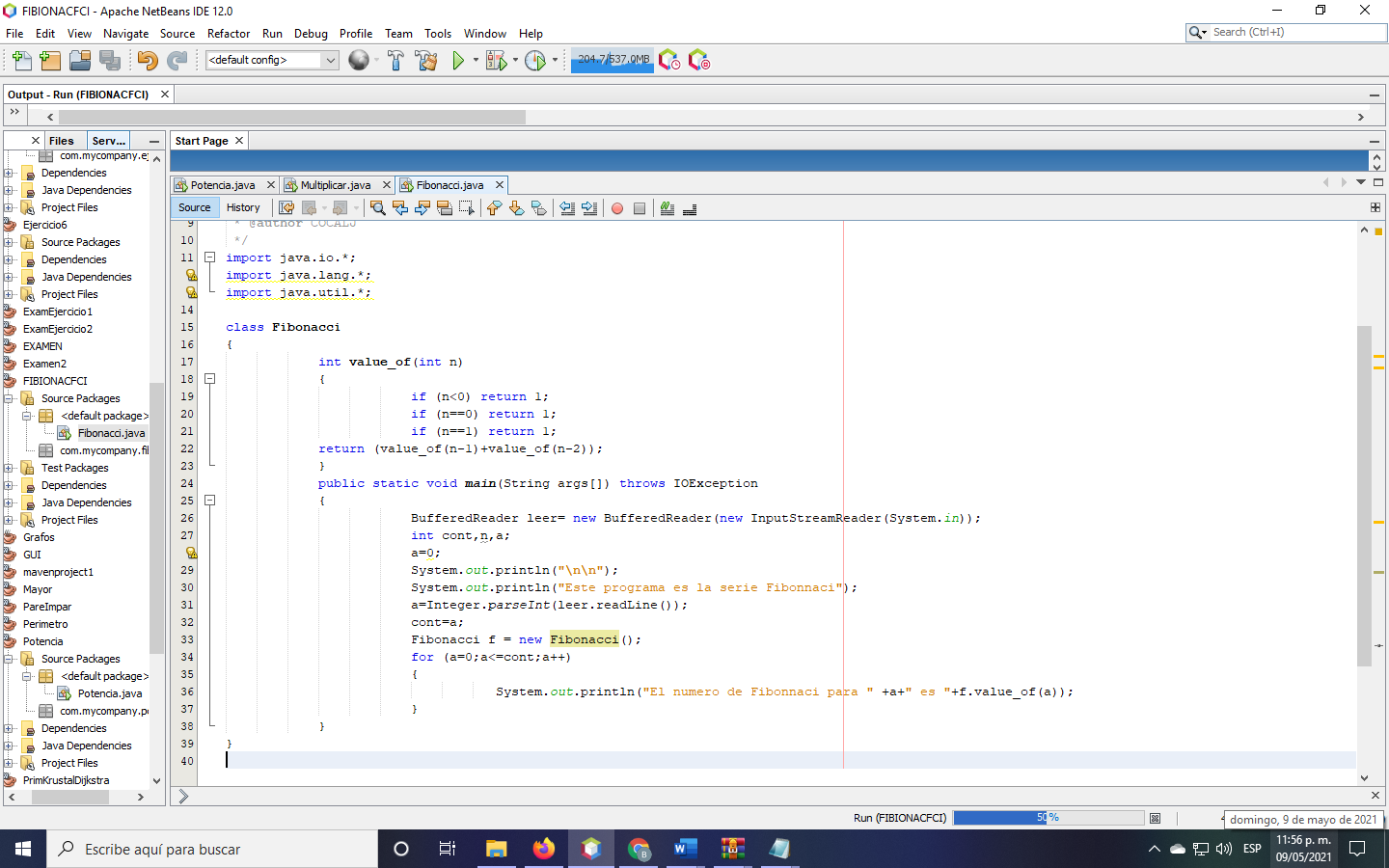
int value\_of(int n)

{  
 if (n<0) return 1;  
 if (n==0) return 1;  
 if (n==1) return 1;

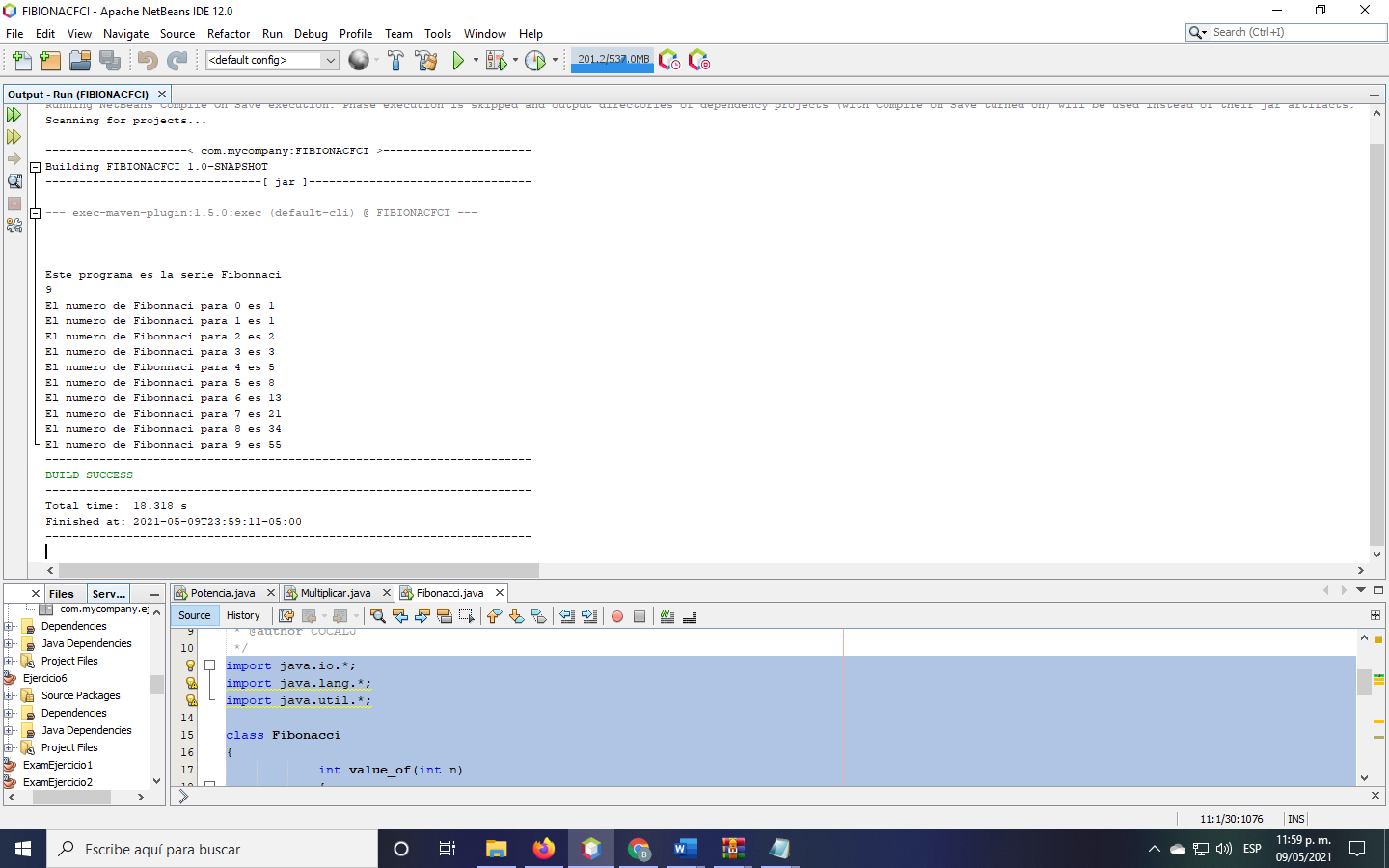
return (value\_of(n-1)+value\_of(n-2));

}  
 public static void main(String args[]) throws IOException

{  
 BufferedReader leer= new BufferedReader(new InputStreamReader(System.in));  
 int cont,n,a;  
 a=0;  
 System.out.println("\n\n");  
 System.out.println("Este programa es la serie Fibonnaci");  
 a=Integer.parseInt(leer.readLine());  
 cont=a;  
 Fibonacci f = new Fibonacci();  
 for (a=0;a<=cont;a++)  
 {  
 System.out.println("El numero de Fibonnaci para " +a+" es "+f.value\_of(a));  
 }  
 }  
}

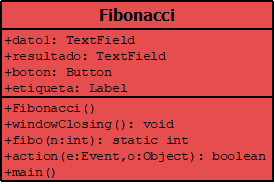


**CORRIDA PROGRAMA**



**PROGRAMA NÚMEROS DE FIBONACCI GUI**

**DIAGRAMA UML**

****

**CLASE FIBONACCI**

import java.awt.\*;  
import java.awt.Frame;  
import java.awt.event.\*;

public class Fibonacci extends Frame

{   
 //cajas de texto  
 TextField dato1=new TextField();  
 TextField resultado=new TextField();  
 Button boton=new Button("Resultado");  
 Label etiqueta=new Label();  
 //constructor  
 public Fibonacci()  
 {  
 //tamaños ylocalizaciones  
 this.setSize(400,400);  
 this.setLocation(20,20);  
 this.add(dato1);  
 dato1.setSize(60,20);  
 dato1.setLocation(30,50);  
 this.add(resultado);  
 resultado.setSize(60,20);  
 resultado.setLocation(250,50);  
 this.add(boton);  
 setTitle ("Serie de Fibonacci");  
 boton.setSize(100,30);  
 boton.setLocation(150,170);  
 add(etiqueta);  
 this.setVisible(true);  
 // Agregar evento de ventana para cerrar  
 addWindowListener (new WindowAdapter ()  
 {  
 public void windowClosing (WindowEvent evt)  
 {  
 hide ( );  
 dispose ( );  
 System.exit (0);  
 }  
 });  
 }  
 static int fibo(int n)  
 {  
 if (n<0) return 1;  
 if (n==0) return 1;  
 if (n==1) return 1;  
 return (fibo(n-1)+fibo(n-2));  
 }  
 public boolean action(Event e, Object o)  
 {  
 if(e.target==boton)  
 {  
 int a,temp;  
 a=Integer.parseInt(dato1.getText());

temp=fibo(a);

resultado.setText(temp+"");

}

return true;

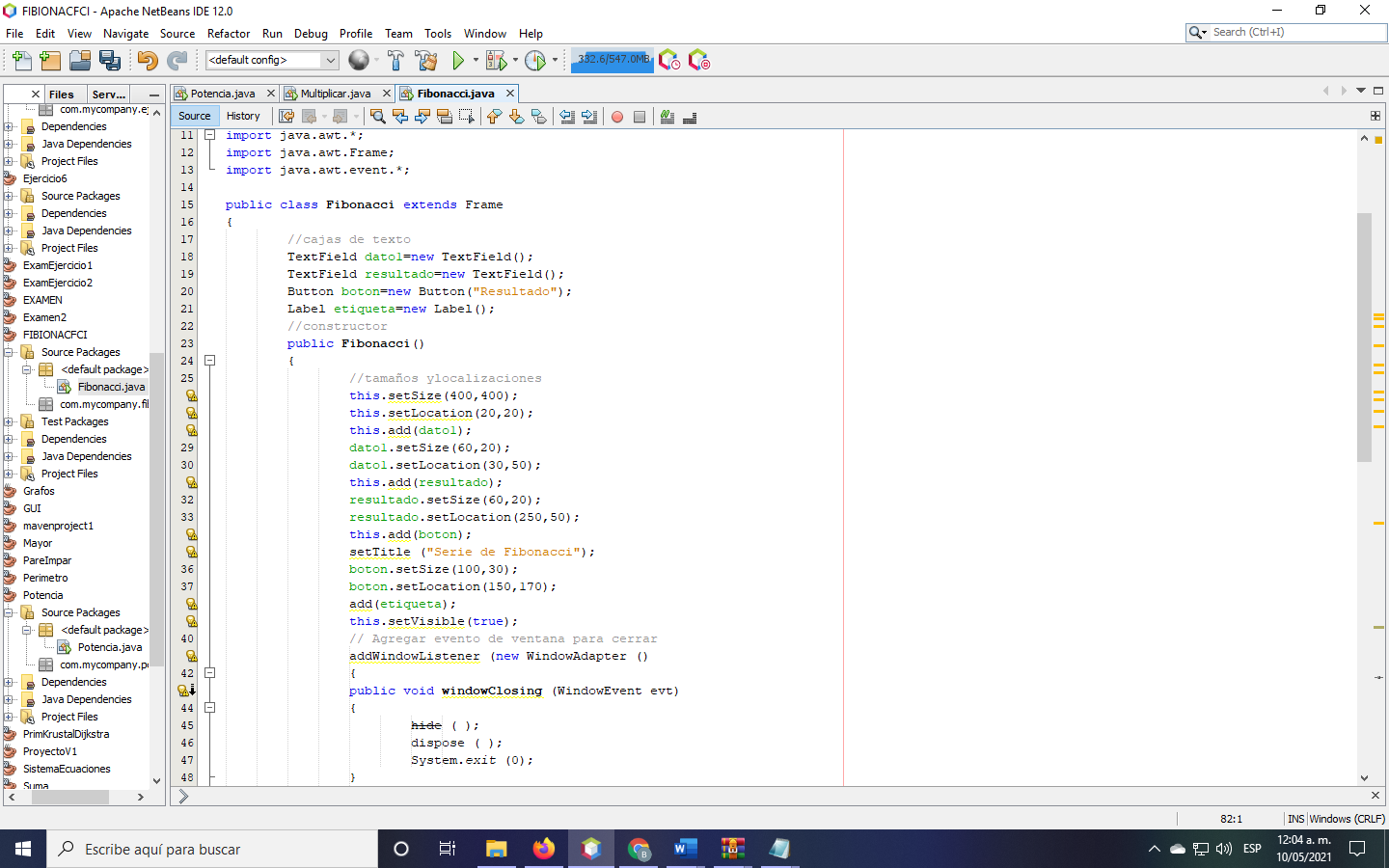
}  
 public static void main(String[] args)

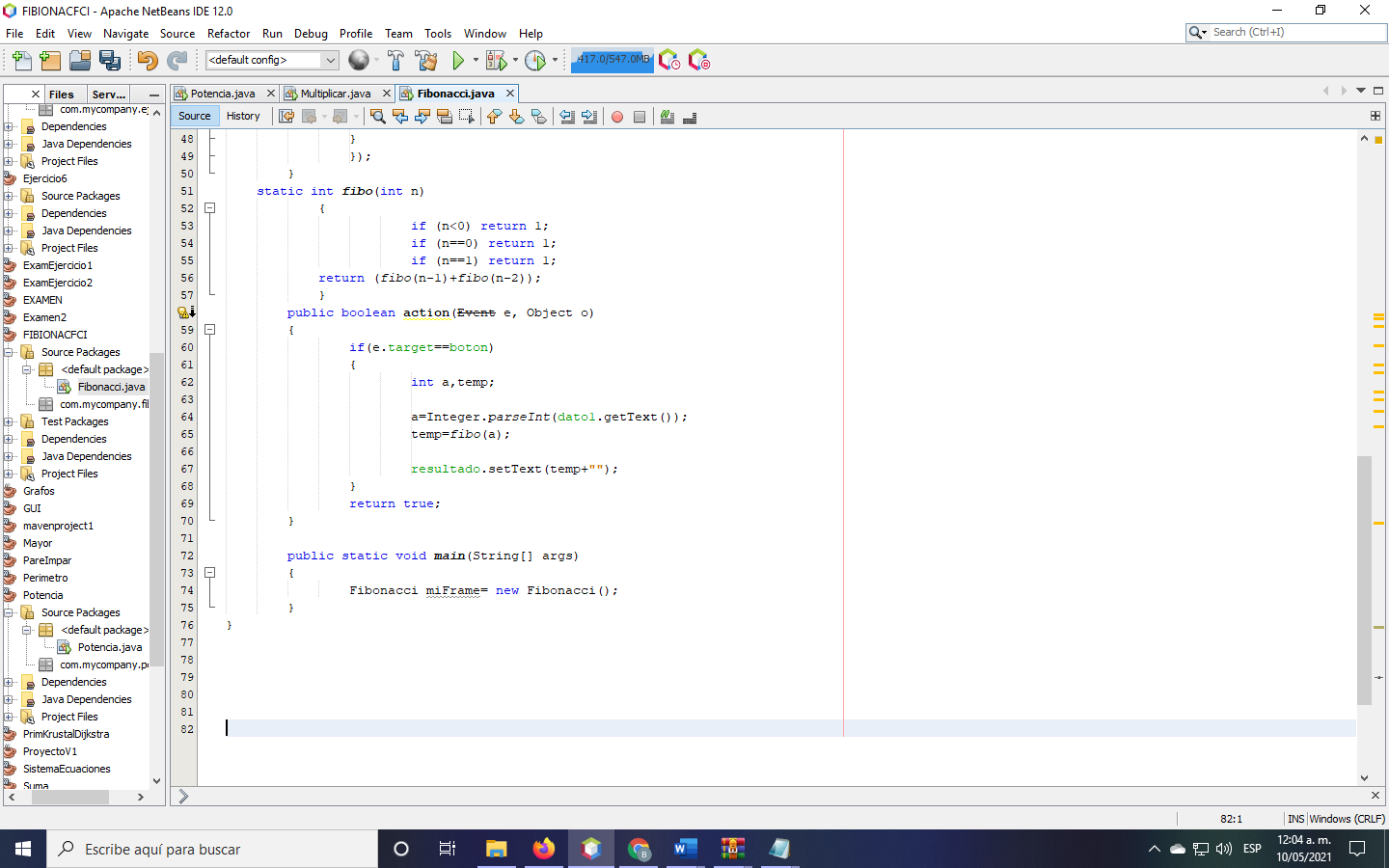
{

Fibonacci miFrame= new Fibonacci();

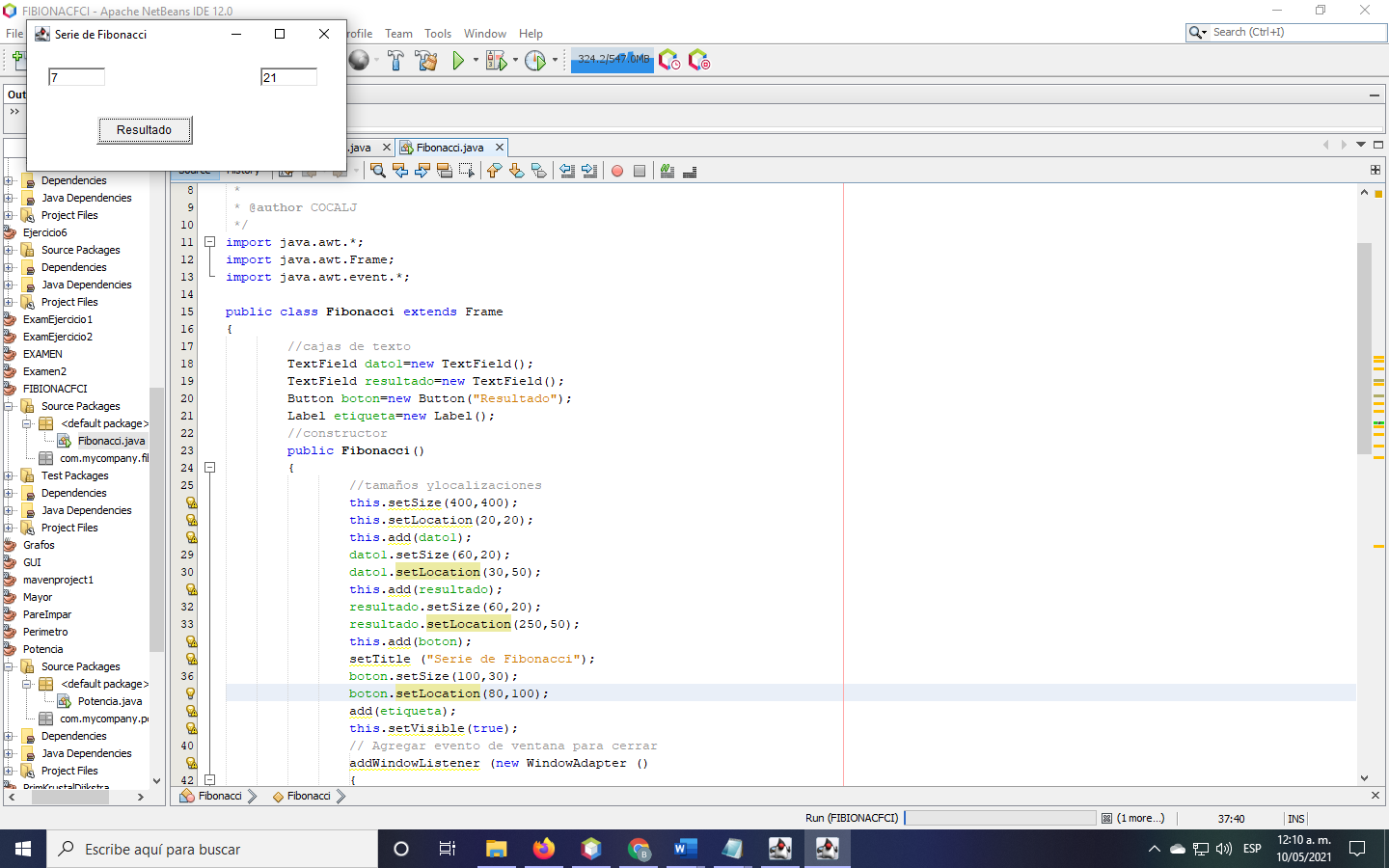
}

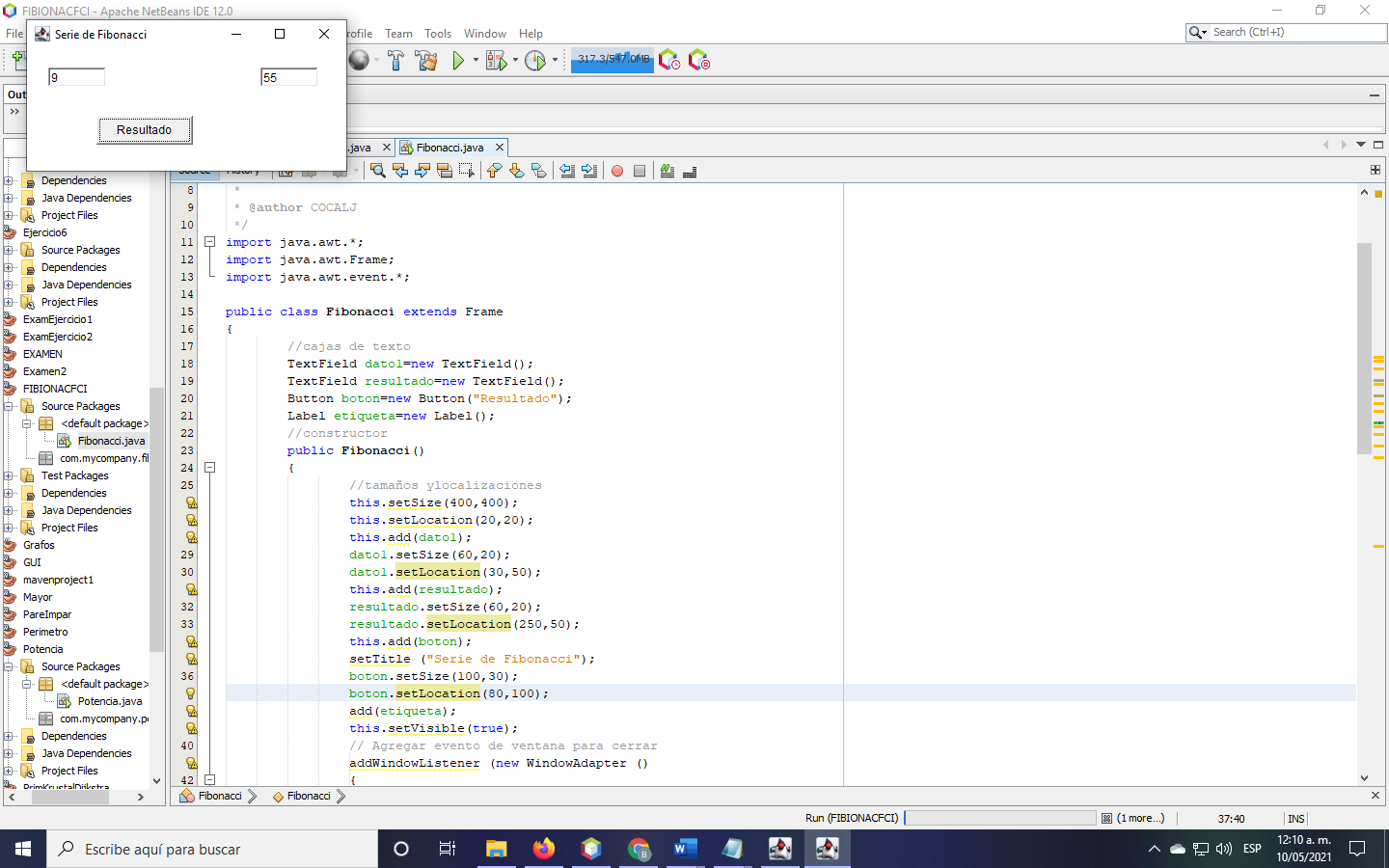
}

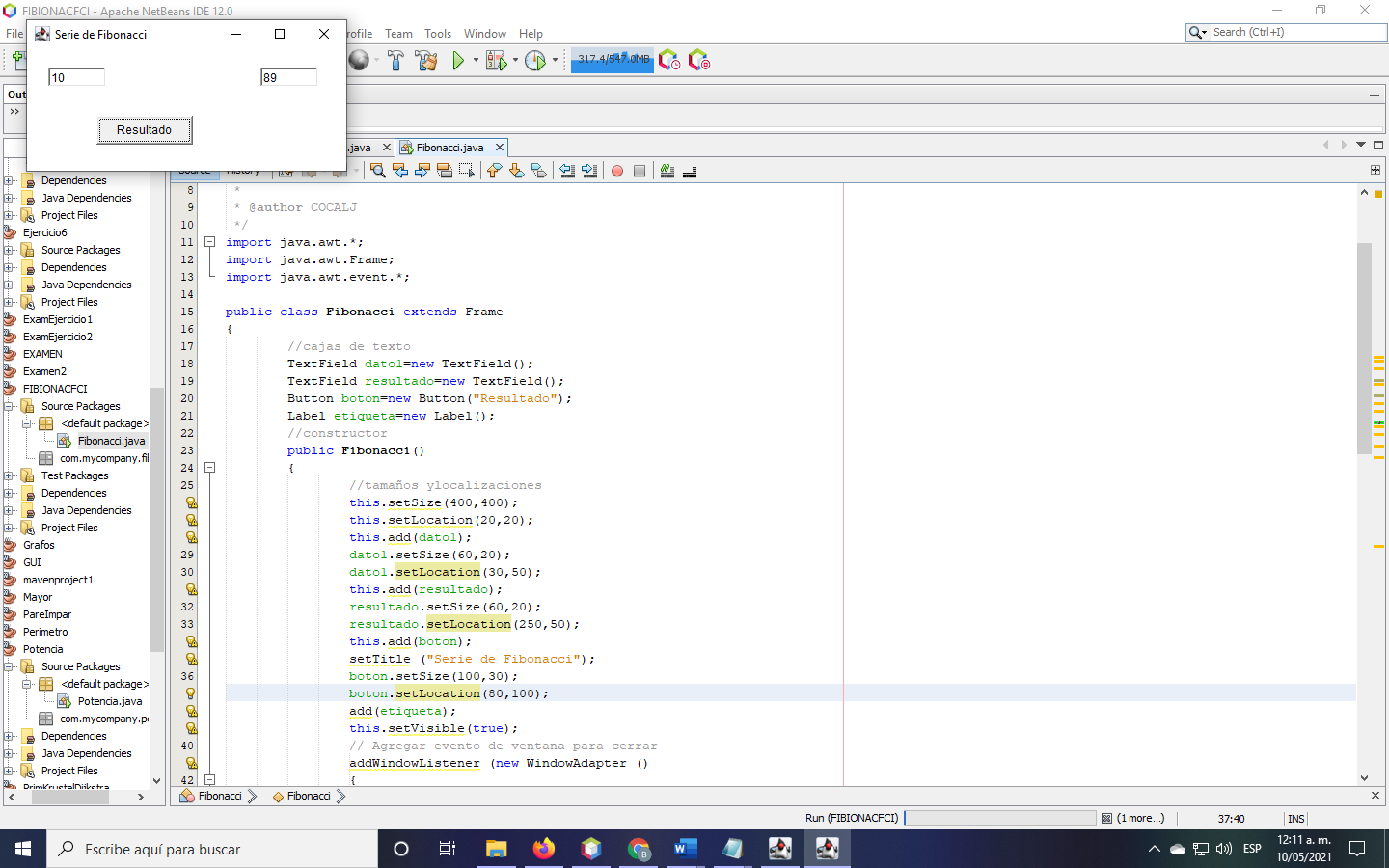




**CORRIDA PROGRAMA**

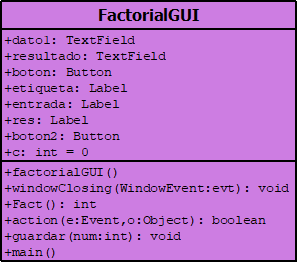






**FACTORIAL GUI**

**DIAGRAMA UML**

****

**CLASE FACTORIAL**

import java.awt.\*;  
import java.awt.Frame;  
import java.awt.event.\*;  
import java.io.\*;

public class factorialGUI extends Frame{

//crear los componentes

TextField dato1=new TextField();  
TextField resultado=new TextField();  
Button boton=new Button("Calcular");  
Label etiqueta=new Label();  
Label entrada=new Label("Dame el numero:");  
Label res=new Label("Resultado:");  
Button boton2=new Button("Guardar");  
int c=0;

public factorialGUI()  
{  
setLayout(null);  
this.setSize(400,400);  
this.setLocation(20,20);  
//caja texto  
//posicionar mis elementos  
//eqtiqueta  
res.setBounds(250,50,120,20);  
//entrada.setLocation(20,50);  
this.add(res);  
entrada.setBounds(30,50,120,20);  
//entrada.setLocation(20,50);  
this.add(entrada);

//testfield  
dato1.setSize(60,20); //dimensión  
dato1.setLocation(30,80);//posicionar lugar  
this.add(dato1);// añade al contenedor

//caja de resultado  
resultado.setSize(60,20);  
resultado.setLocation(250,80);  
this.add(resultado);

//boton1  
boton.setSize(100,30);  
boton.setLocation(80,170);  
this.add(boton);  
add(etiqueta);

//boton2  
boton2.setSize(100,30);  
boton2.setLocation(200,170);  
this.add(boton2);  
add(etiqueta);

//titulo de la ventana  
setTitle ("Factorial de un numero");  
this.setVisible(true);

// Agregar evento de ventana para cerrar

addWindowListener (new WindowAdapter ()

{  
 public void windowClosing (WindowEvent evt)  
 {  
 hide ( );  
 dispose ( );  
 System.exit (0);  
 }  
 });  
}  
 static public int Fact (int x)

{  
 if( x == 0 )  
 {return 1;}  
 else  
 {return x \* Fact(x-1);}  
 }  
public boolean action(Event e, Object o)  
{  
 int a;  
 if(e.target==boton)  
{  
 a=Integer.parseInt(dato1.getText());  
 c=Fact(a);  
 //guardar(c);  
 resultado.setText(c+"");  
}  
 if(e.target==boton2)  
{   
 guardar(c);  
 }

return true;

}

public void guardar (int num)

{

//Guradar de modo texto

FileWriter fichero = null;

PrintWriter inicio = null;

try

{

fichero = new FileWriter("prueba.txt");  
 inicio = new PrintWriter(fichero);  
 inicio.println("resultado = " + num);

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

// Nuevamente aprovechamos el finally para

// asegurarnos que se cierra el fichero.

if (null != fichero)

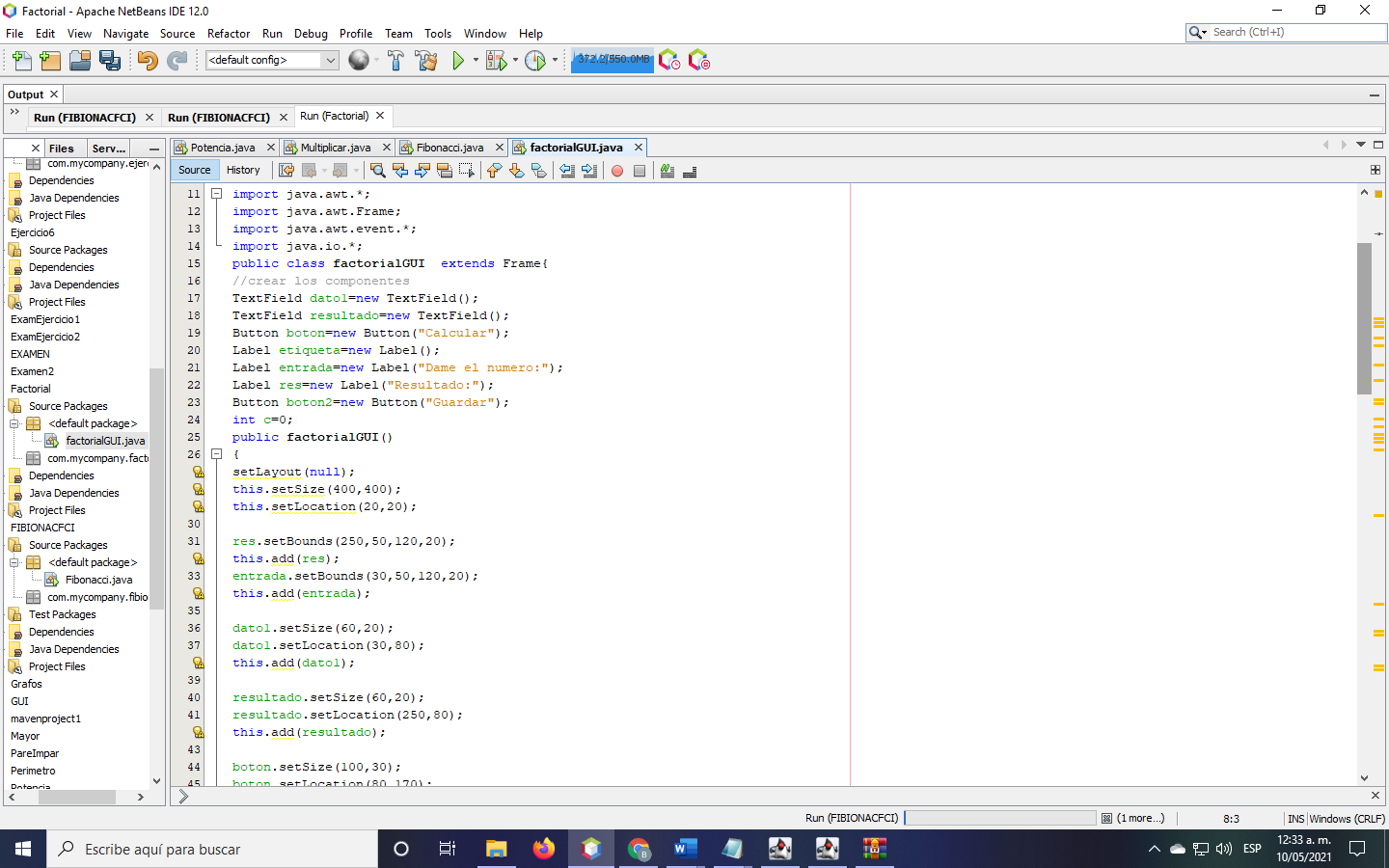
fichero.close();

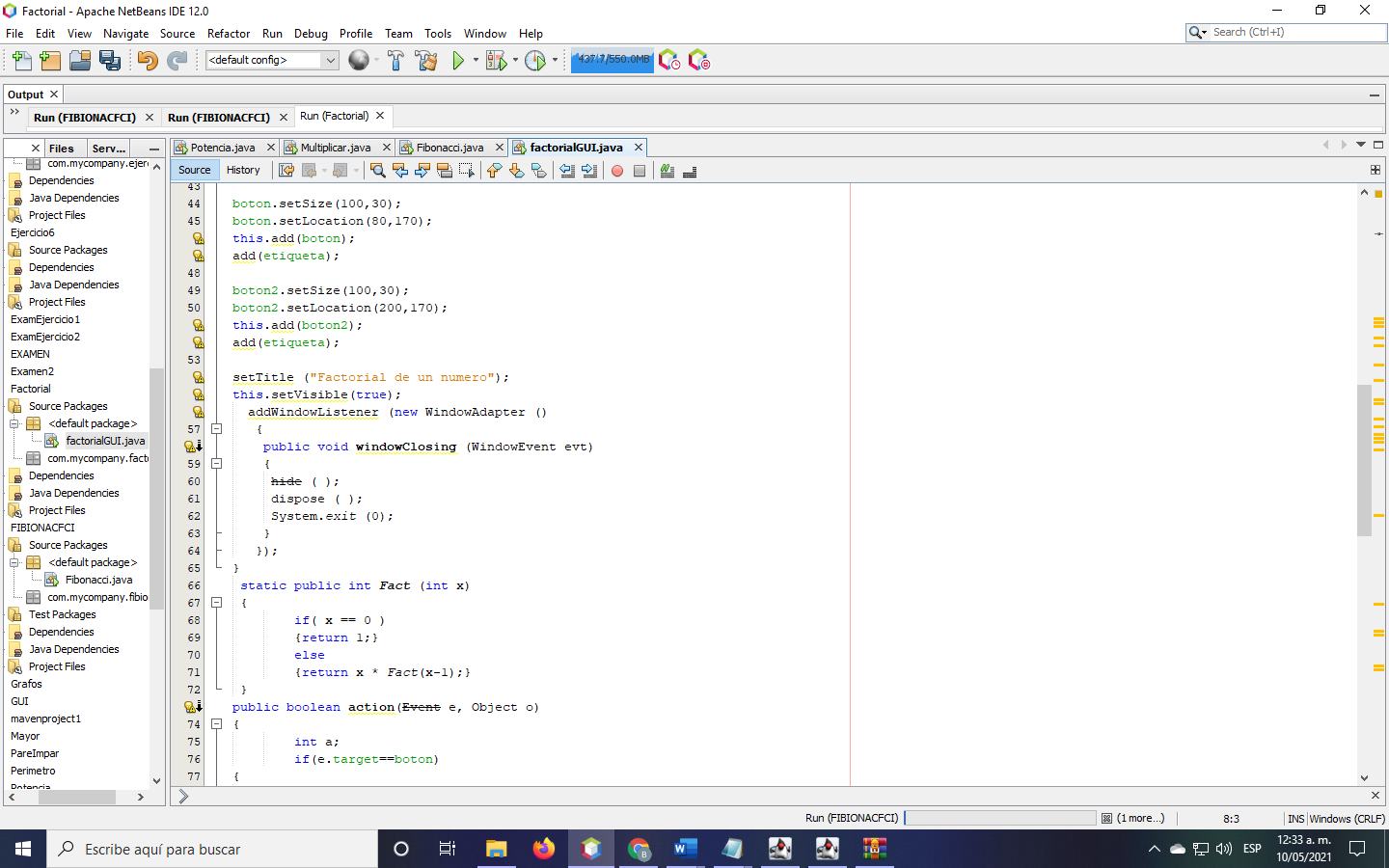
} catch (Exception e2) {

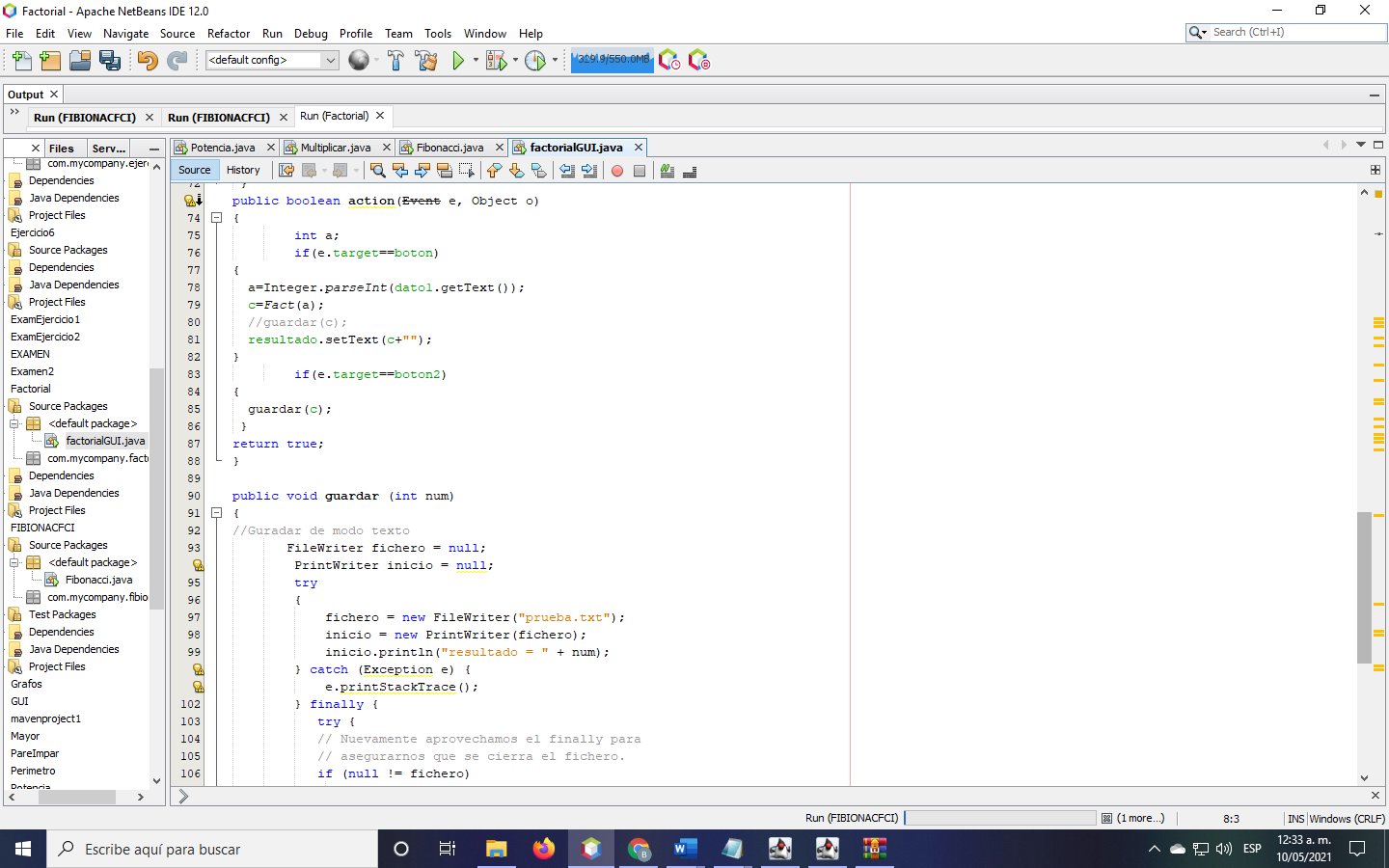
e2.printStackTrace();  
 }  
 }  
 }  
public static void main(String[] args) {  
factorialGUI miFrame= new factorialGUI();

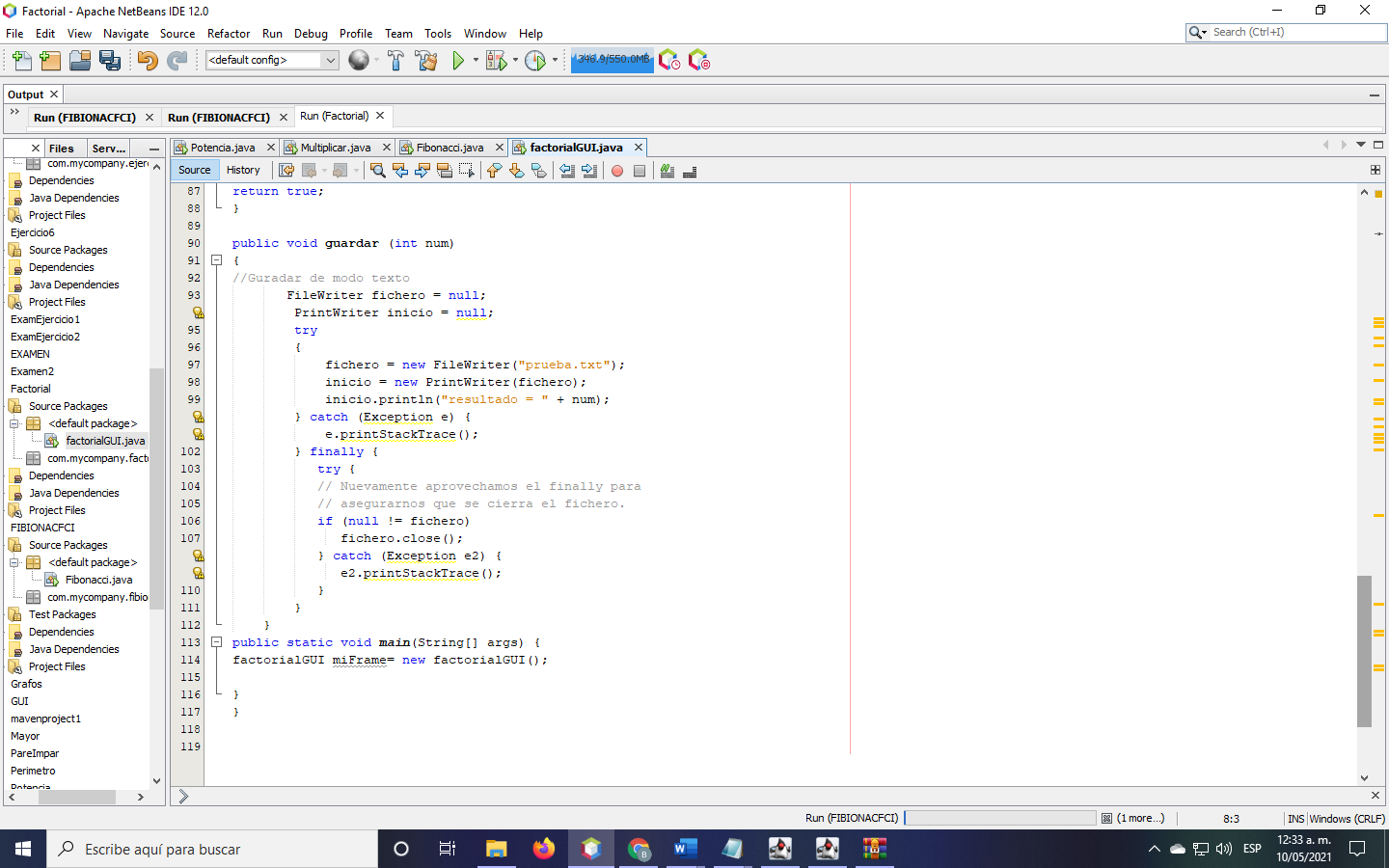
}

}

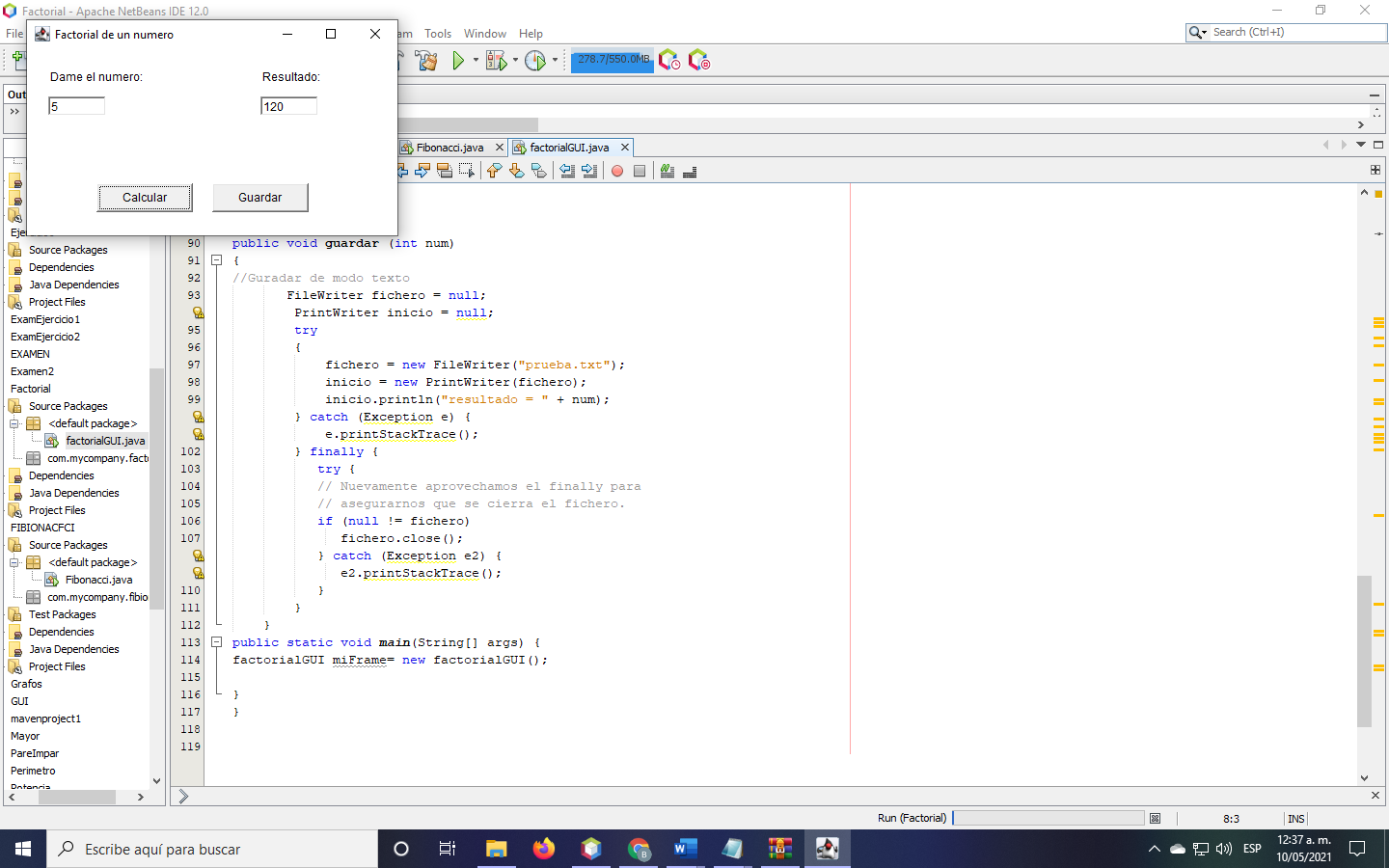


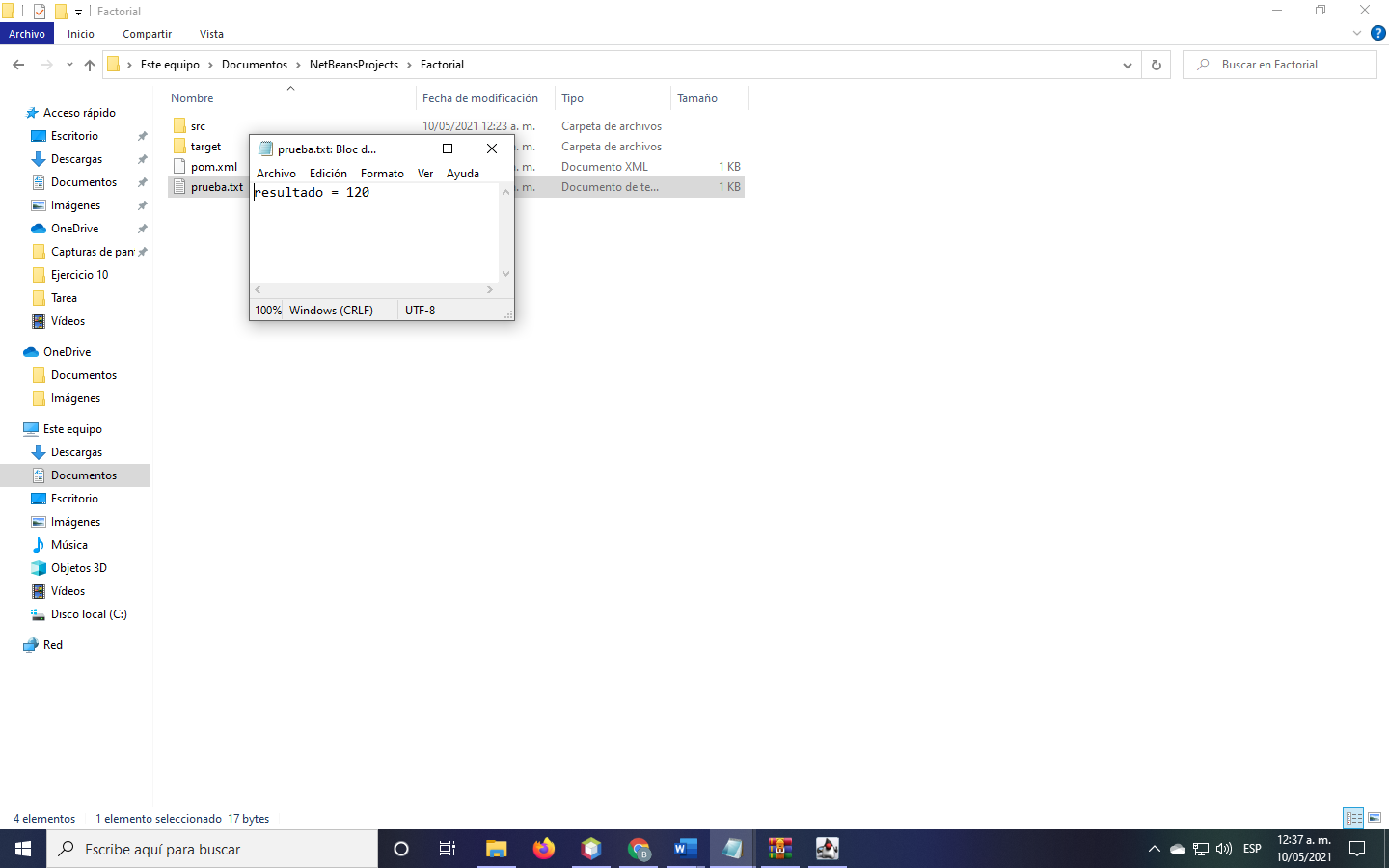






**CORRIDA PROGRAMA**





**CONCLUSIÓN**

Las soluciones recursivas suelen ser más lentas que las iterativas por el tiempo empleado en la gestión de las sucesivas llamadas a los métodos. Además consumen más memoria ya que se deben guardar los contextos de ejecución de cada método que se llama.

A pesar de estos inconvenientes, en ciertos problemas, la recursividad conduce a soluciones que son mucho más fáciles de leer y comprender que su correspondiente solución iterativa. En estos casos una mayor claridad del algoritmo puede compensar el coste en tiempo y en ocupación de memoria.  
De todas maneras, numerosos problemas son difíciles de resolver con soluciones iterativas, y sólo la solución recursiva conduce a la resolución del problema (por ejemplo, Torres de Hanoi o recorrido de Árboles).

**REFERENCIAS BIBLIOGRÁFICAS**

* Fagg, A.H.(2017). GUI Programming: Components. Consultado el día 12 de agosto de 2020 en: <https://www.cs.ou.edu/~fagg/classes/cs2334/lecture/GUI-Components.pdf>
* Horstmann, C. S., & Cornell, G. (2013). Core Java Volumen I Fundamentals. New Jersey: Prentice Hall.
* Joyanes Aguilar, L., & Zahonero Martínez, I. (2011). Programación en Java 6. Algoritmos y programación orientada a objetos. Cd. de México: Mc Graw Hill.
* Poo, D., Kiong, D., & Ashok, S. (2008). Object-Oriented Programming and Java. Singapore: Springer.
* Sznajdleder, P. A. (2016). Java a fondo. Alfaomega.
* Liang, Y. D. ( 2014 ). Introduction to Java Programming, Comprehensive Version. Pearson
* Malik D. S. (2012 ). Java Programming: From Problem Analysis to Program Design, Fifth Edition. Cengage Learning
* GeeksforGeeks. (Última edición: 20 de Julio de 2020) , GeeksforGeeks. https://www.geeksforgeeks.org/types-of-recursions .Consultado de 30 de julio del 2020

**PROGRAMACION II**

**RUBRICA HOJA DE EVALUACIÓN DE PRACTICAS**

**MATRICULA: 201963582 FECHA: 9-MAYO-2021**

**NOMBRE: Irvyn Xicale Cabrera NO. PRACTICA: 17A INDIVIDUAL ( X) COLABORATIVA( )**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Deficiente** | **Suficiente** | **Bueno** | **Excelente** | **Calificación Obtenida** | |
| **CRITERIOS** | **1-5.9** | **6‐7.9** | **8‐9** | **9.1‐10** |
| **%** | **Puntos** |
| **CONOCIMIENTO TEORICO**  **20%** | Conocimiento deficiente de los fundamentos teóricos de POO y no puede aplicarlos en el diseño de clases. | Conocimiento confuso de los fundamentos teóricos de POO y el diseño de las clases y relaciones es incompleto. | Conocimiento claro de los fundamentos teóricos POO, pero requiere mejorar el modelado de las clases, sus métodos  y sus relaciones entre clases. | Dominio del Conocimiento de los fundamentos teóricos POO y puede aplicarlos de forma completa en el modelado de las clases, métodos y todas las relaciones entre clase. |  |  |
| **EJECUCCIÓN DE LA PRACTICA**  **30%** | No puede realizar la práctica ya que desconoce el entorno de trabajo y desarrollo de la práctica en lenguaje UML y Java. | Realiza la práctica de forma incompleta ya que desconoce el entorno de trabajo del lenguaje UML y Java | Realiza la práctica, pero requiere mejorar en el manejo del entorno de trabajo del lenguaje de programación (sintaxis y semántica) | Realiza la práctica de forma correcta y completa, demuestra dominio del entorno de trabajo del lenguaje de programación (sintaxis y semántica). |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOLUCIÓN DE LA PRACTICA**  **40%** | No puede generar las soluciones o programas a los problemas planteados ya que no posee el dominio teórico y práctico del modelado y el lenguaje de programación. | Propone soluciones confusas o programas incompletos a los problemas planteados, ya que carece del dominio del modelado y lenguaje de programación | Genera soluciones con poca profundidad y los programas no están orientados de acuerdo a los problemas solicitados, por lo cual no tiene un dominio profundo de la temática y del Lenguaje. | Genera soluciones con profundidad y los programas son correctos de acuerdo a los problemas planteados, por lo cual demuestra un dominio de la temática y del Lenguaje de Programación. |  |  |
| **ACTITUD DE APRENDER Y COLABORAR EN EQUIPO DE TRABAJO**  **10%** | No posee una actitud proactiva para un aprendizaje autónomo y no le gusta participar y trabajar en equipo. | Posee una actitud propositiva para un aprendizaje autónomo, participa, pero no le gusta trabajar en equipo. | Posee una actitud propositiva logrando un aprendizaje autónomo, colaborativo, le gusta trabajar en equipo, pero requiere mejorar su participación y portaciones de forma profunda. | Posee una actitud proactiva logrando un aprendizaje autónomo, participa con propuestas concretas y profundas, le gusta trabajar en equipo y asume su responsabilidad dentro para lograr éxito del equipo |  |  |
| **Total** | | | | |  |  |  |

