

A diagram featuring a vertical grey axis on the left and a horizontal grey axis at the bottom. A dark grey arrow points right from the vertical axis, and a light grey arrow points right from the horizontal axis. In the top right corner, a dark grey triangle is outlined with dashed lines, containing the word 'SOLUCIONARIO' in white, slanted, uppercase letters.

SOLUCIONARIO

Conceptos de desarrollo y IDE's

Desarrollo de aplicaciones

Multiplataforma &

Desarrollo de aplicaciones Web

Entornos de desarrollo



Actividad

Realizar las actividades completando la información relativa al análisis de los diferentes lenguajes de programación.

Practica con eclipse.

Objetivos

- Analizar las características de algunos de los principales lenguajes de programación.
- Distinguir los distintos procesos de desarrollo de un software.
- Conocer qué es un IDE y en qué nos tenemos que fijar para elegir un IDE.
- Instalar IDE Eclipse.
- Crear y depurar un proyecto Java con Eclipse.

Actividad 1

Características de los lenguajes de programación



Solución Actividad 1:

En esta actividad deberás:

1. Define los siguientes conceptos relacionados con la clasificación de los lenguajes de programación.

Conceptos de los lenguajes de programación

Conceptos de los lenguajes de programación	Definición
Nivel de abstracción	Llamamos nivel de abstracción al modo en que los lenguajes de programación se alejan del código máquina y se acercan cada vez más a un lenguaje similar al que nos comunicamos. Cuando más alejado esté del código máquina, de mayor nivel será el lenguaje. Dicho de otra manera, se podría ver el nivel de abstracción como la cantidad de “capas” de ocultación de código máquina que hay entre el código que escribimos y el código que la máquina ejecutará en último término.
Paradigma de programación	Es un enfoque particular para la construcción de software, un estilo de programación que facilita la tarea de programación o añade mayor funcionalidad al programa dependiendo del problema que haya que abordar. Todos los paradigmas de programación pertenecen a lenguajes de alto nivel y es común que un lenguaje pueda usar más de un paradigma de programación.
Forma de ejecución	Depende de cómo un programa se ejecute dentro de un sistema, podríamos definir tres categorías de lenguajes.

- 
- 
2. Explica de forma breve y clara cada una de las **fases del proceso de compilación** de un programa:

Fase	Descripción	¿Qué se obtiene?
Análisis Lexicográfico	Se leen de manera secuencial todos los caracteres de nuestro código fuente, buscando palabras reservadas, operaciones, caracteres de puntuación y los agrupa en cadenas de caracteres que se llaman lexemas	Cadenas de caracteres que se llaman lexemas , agrupados formando tokens .
Análisis Sintáctico-Semántico	Agrupar los componentes léxicos estudiados en el análisis anterior en forma de frases gramaticales. Con el resultado del proceso del análisis sintáctico, se revisa la coherencia de las frases gramaticales	Árbol sintáctico
Generación de código intermedio	Una vez finalizado el análisis, se genera una representación intermedia a modo de pseudoensamblador con el objetivo de facilitar la tarea de traducir al código objeto	Representación intermedia a modo de pseudoensamblador
Optimización de código	Revisa el código pseudoensamblador generado en el paso anterior optimizándolo para que el código resultante sea más fácil y rápido de interpretar por la máquina.	Código optimizado
Generación de código	Genera el código objeto de nuestro programa en un código de lenguaje máquina relocalizable, con diversas posiciones de memoria sin establecer, ya que no sabemos en qué parte de la memoria volátil se va a ejecutar nuestro programa.	Código objeto
Enlazador de librerías	Como se ha comentado anteriormente, se enlaza el código objeto con las librerías necesarias, produciendo en último término el código final ejecutable	Código ejecutable

*Debéis explicarlo de forma breve y claro. Este ejercicio bien resuelto os resultará muy útil para el examen.

3- Rellenar la siguiente tabla

Tipo de lenguaje	Ejemplos de lenguajes
De primera generación	Lenguaje maquina
De segunda generación	Lenguaje ensamblador
De tercera generación*	C, Pascal
De cuarta generación*	Natural, SQL
De quinta generación	Lisp, Prolog
Compilado*	C, C++
Interpretado*	Javascript, PHP
Imperativo*	Basic, Pascal, C

*mínimo dos ejemplos de lenguajes

Actividad 2

El Proceso de desarrollo

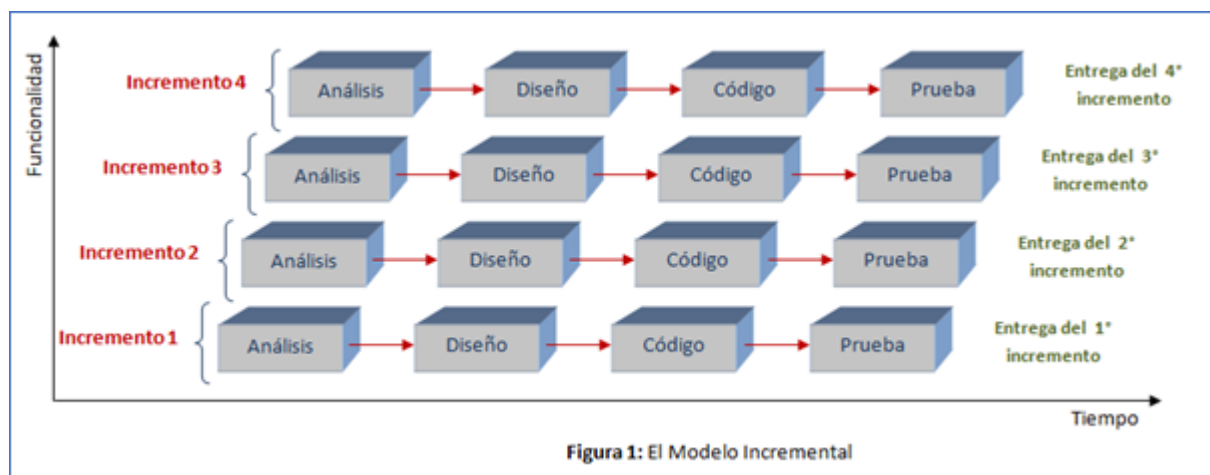
El proceso de desarrollo de un software se modela usando diferentes estrategias o métodos. El modelo en cascada visto en clase es un ejemplo de unos de estos modelos. Otros modelos usados normalmente son: el Modelo iterativo, el Modelo en espiral, el Modelo a V, y el Modelo incremental.

En este ejercicio, has de explicar el **modelo en espiral**

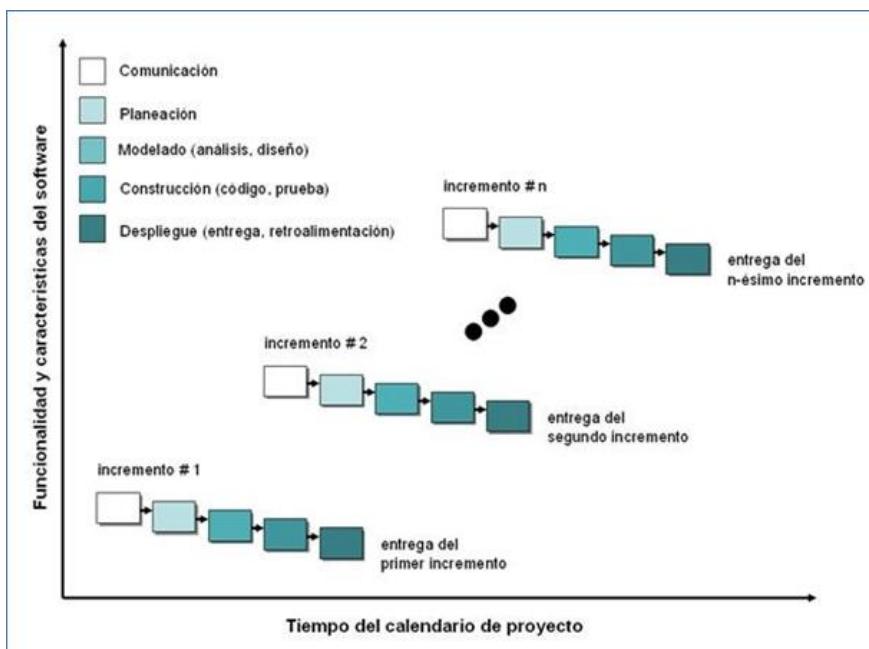
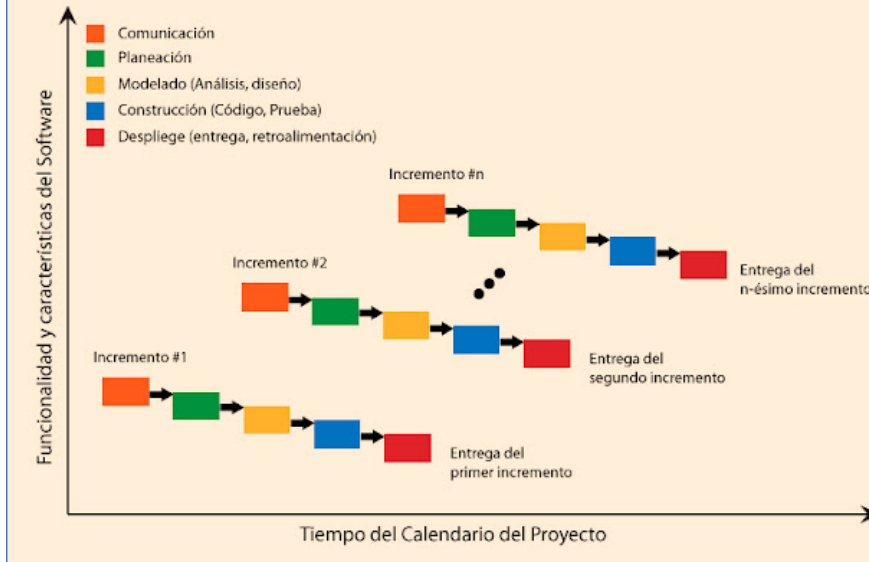
- Creando un diagrama que muestre las diferentes etapas y su orden
- Explica detalladamente los diferentes pasos
- Comenta según tu criterio, si este modelo te parece mejor o peor en comparación con el modelo en cascada estudiado en clase y argumenta la respuesta

Encontramos varios diagramas en Internet que explican el modelo (os los mostramos a continuación) y en todos ellos podemos ver como representan las iteraciones que se van sucediendo en el tiempo, de cada una de las fases.

Las salidas de una fase sirven de entradas para la misma fase en la siguiente iteración, proporcionando una evolución progresiva de las entregas que se hacen del proyecto.

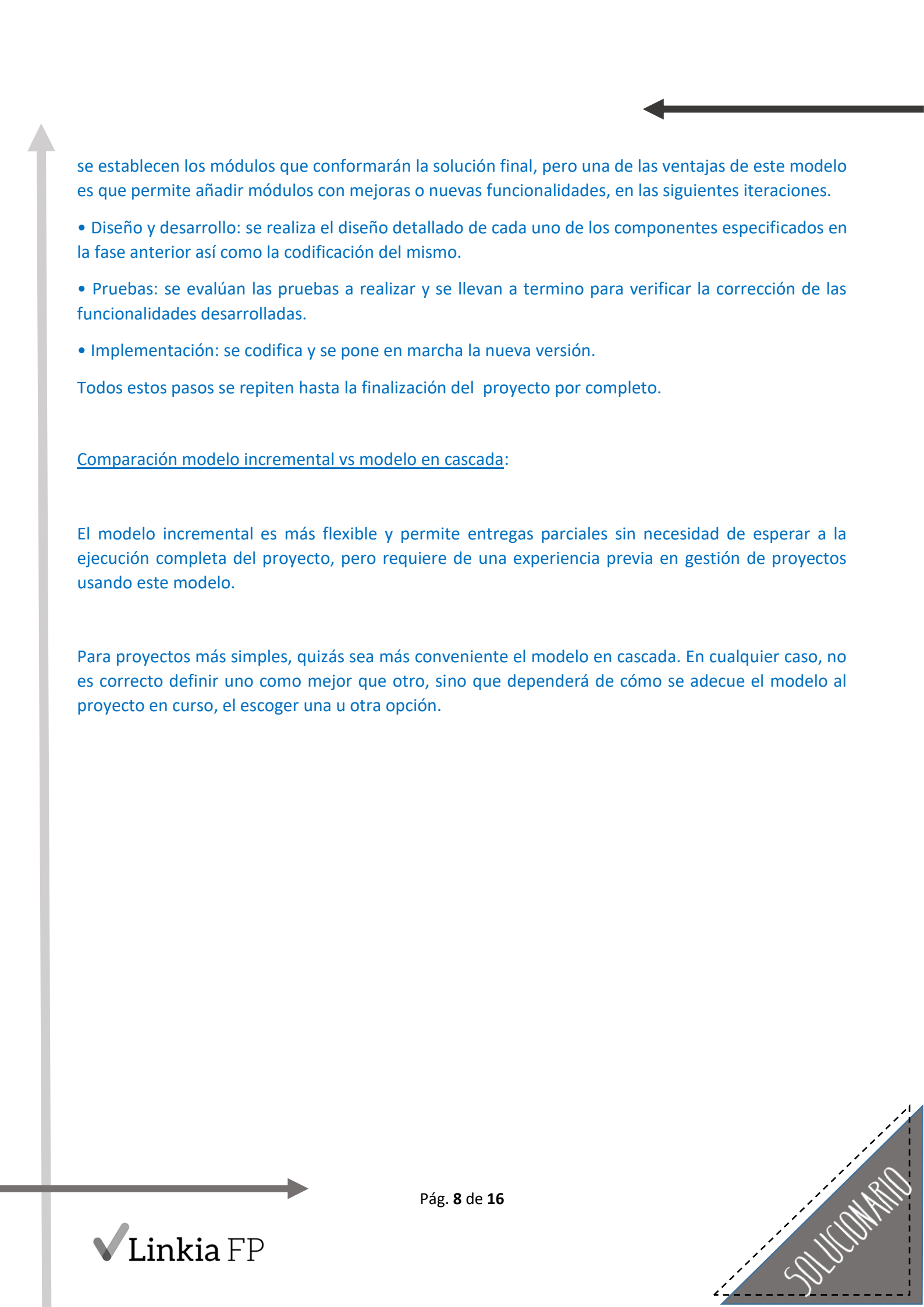


Modelo Incremental



Fases del modelo incremental:

- **Análisis preliminar:** el equipo de analistas de sistemas, revisa los requisitos. Es muy importante en el desarrollo de cualquier solución de software el análisis previo. Cuando se usa el modelo incremental,



se establecen los módulos que conformarán la solución final, pero una de las ventajas de este modelo es que permite añadir módulos con mejoras o nuevas funcionalidades, en las siguientes iteraciones.

- Diseño y desarrollo: se realiza el diseño detallado de cada uno de los componentes especificados en la fase anterior así como la codificación del mismo.
- Pruebas: se evalúan las pruebas a realizar y se llevan a termino para verificar la corrección de las funcionalidades desarrolladas.
- Implementación: se codifica y se pone en marcha la nueva versión.

Todos estos pasos se repiten hasta la finalización del proyecto por completo.

Comparación modelo incremental vs modelo en cascada:

El modelo incremental es más flexible y permite entregas parciales sin necesidad de esperar a la ejecución completa del proyecto, pero requiere de una experiencia previa en gestión de proyectos usando este modelo.

Para proyectos más simples, quizás sea más conveniente el modelo en cascada. En cualquier caso, no es correcto definir uno como mejor que otro, sino que dependerá de cómo se adecue el modelo al proyecto en curso, el escoger una u otra opción.

Actividad 3

Patrones de diseño

Los patrones de diseño son una solución a problemas comunes en el diseño y desarrollo de software y son un recurso muy útil para los programadores. En esta actividad deberás:

1. Enumera los diferentes tipos de patrones de desarrollo y explica cuál es el objetivo de cada uno.

Solución Actividad 3:

Tipos de patrones de desarrollo

- Patrones creacionales: ofrecen soluciones respecto a cómo se construyen los objetos_
 - Fábrica abstracta, Constructor virtual, método de fabricación
- Patrones estructurales: ofrecen soluciones respecto a cómo los objetos se componen / agregan,
 - Decorador, Objeto compuesto, Fachada, Puente, Adaptador, Peso Ligero
- Patrones de comportamiento: ofrecen soluciones respecto a la interacción y responsabilidades entre clases y objetos,
 - Estado, Visitante, Iterador, Cadena de responsabilidad, Orden, Intérprete, Mediador, etc.

2- A continuación, verás una serie de **casos de aplicación de patrones de diseño**. Estos casos son del tipo “el antes y el después” de la aplicación. El objetivo de este ejercicio es que explicar los cambios se han hecho en cada caso.

Patrón Fachada

En primer lugar, vemos los prototipos de las clases (no os fijéis en la implementación, ya que no es relevante) que intervienen en el relativo proceso de retirada de efectivo de un cajero automático

```
public class Autenticacion{
    /* ... */
    public boolean leerTarjeta(){}
    public String introducirClave(){}
    public boolean comprobarClave(String clave){}
    public Cuenta obtenerCuenta(){}
    public void alFallar(){}
}

public class Cajero{
    /* ... */
    public int introducirCantidad(){}
    public boolean tieneSaldo(int cantidad){}
    public int expedirDinero{}
    public String imprimirTicket(){}
}

public class Cuenta{
    /* ... */
    public double comprobarSaldoDisponible(){}
    public boolean bloquearCuenta(){}
    public boolean desbloquearCuenta{}
    public void retirarSaldo(int cantidad){}
    public boolean actualizarCuenta(){}
    public void alFallar(){}
}
```

```

public class FachadaCajero{
    private Autenticacion autenticacion = new Autenticacion();
    private Cajero cajero = new Cajero();
    private Cuenta cuenta = null;
    public void introducirCredenciales(){
        boolean tarjeta_correcta = autenticacion.leerTarjeta();
        if(tarjeta_correcta){
            String clave = autenticacion.introducirClave();
            boolean clave_correcta = autenticacion.comprobarClave(clave);
            if(clave_correcta){
                cuenta = autenticacion.obtenerCuenta();
                return;
            }
        }
        autenticacion.alFallar();
    }

    public void sacarDinero(){
        if(cuenta != null){
            int cantidad = cajero.introducirCantidad();
            int tiene_dinero = cajero.tieneSaldo(cantidad);
            if(tiene_dinero){
                boolean hay_saldo_suficiente = ((int)cuenta.comprobarSaldoDisponible()) >= cantidad;
                if(hay_saldo_suficiente){
                    cuenta.bloquearCuenta();
                    cuenta.retirarSaldo(cantidad);
                    cuenta.actualizarCuenta();
                    cuenta.desbloquearCuenta();
                    cajero.expedirDinero();
                    cajero.imprimirTicket();
                }
            }
            else{
                cuenta.alFallar();
            }
        }
    }
}

```

Interfaz:

```

public static void main(String[] args){
    FachadaCajero cajero_automatico = new FachadaCajero();
    cajero_automatico.introducirCredenciales();
    cajero_automatico.sacarDinero();
}

```

Explica que está consiguiendo la aplicación del patrón fachada en este caso (comportamiento, llamadas que hace, etc):

Mediante estas operaciones, el cliente tendría que acceder a 3 subsistemas y realizar una inmensa cantidad de operaciones. Sin embargo, crearemos una fachada para ofrecer una interfaz mucho más amigable:

De esta manera, hemos conseguido que para que un cliente use el cajero, no tenga que realizar todas las operaciones de sus subsistemas. En lugar de ello proporcionamos una interfaz mucho más simple que facilita enormemente su uso

Actividad 4

Practicar con Eclipse.

Utilizando el material “Eclipse: Instalación, creación de un proyecto Java y depurador”, instalar el entorno de desarrollo Eclipse.

1. Crea un nuevo proyecto y crear una nueva clase llamada **Factorial** con el siguiente código:

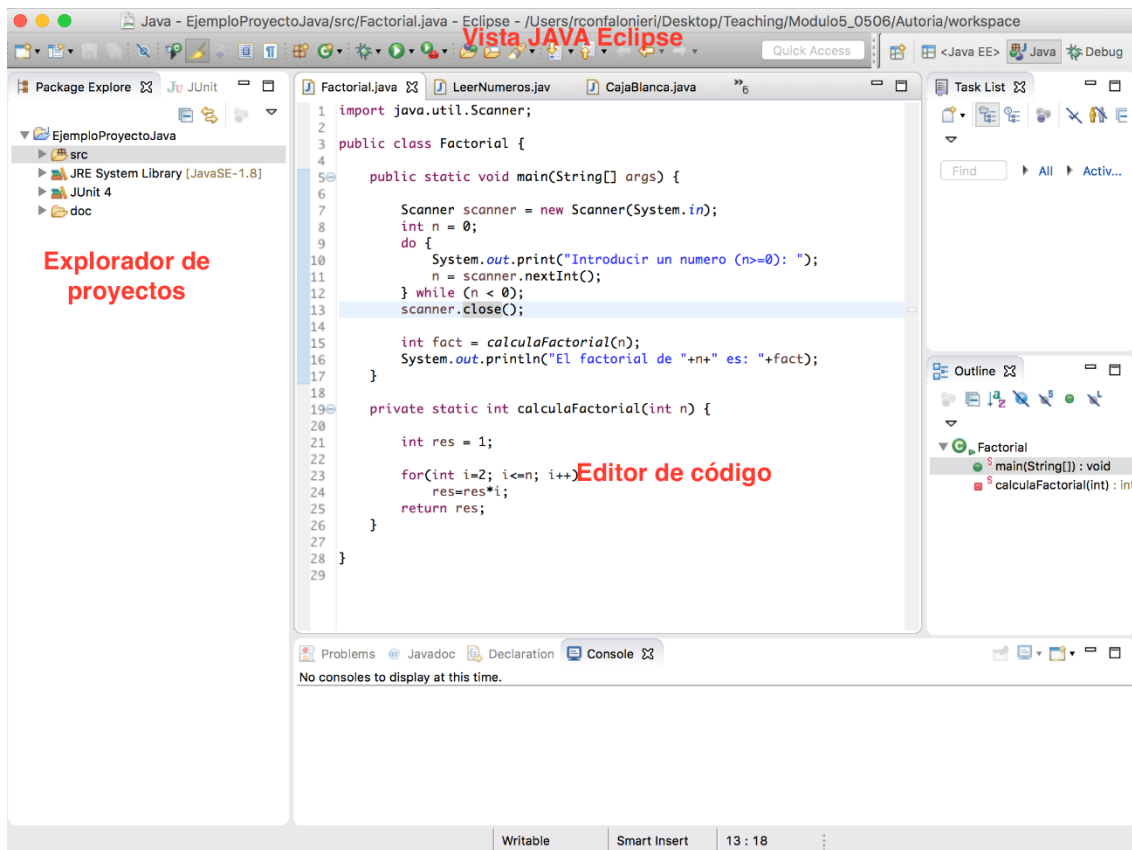
```
public static void main(String[] args) {  
  
    Scanner scanner = new Scanner(System.in);  
    int n = 0;  
    do {  
        System.out.print("Introducir un numero (n>=0): ");  
        n = scanner.nextInt();  
    } while (n < 0);  
    scanner.close();  
  
    int fact = calculaFactorial(n);  
    System.out.println("El factorial de "+n+" es: "+fact);  
}  
  
private static int calculaFactorial(int n) {  
  
    int res = 1;  
  
    for(int i=2; i<=n; i++)  
        res=res*i;  
    return res;  
}
```

2. Importar la librería necesaria para el correcto funcionamiento del proyecto y a continuación realizar un documento final con una captura de pantalla que muestre las siguientes funcionalidades:
 - Explorador de proyecto de Eclipse y editor de código de Eclipse.
 - Vista Java de Eclipse.
 - Vista de depuración de Eclipse.
3. Ejecutar el código en modo depuración, introducir el valor 5 y añadir las capturas de pantalla que muestren los valores de la variable `res` del método `calculaFactorial`.

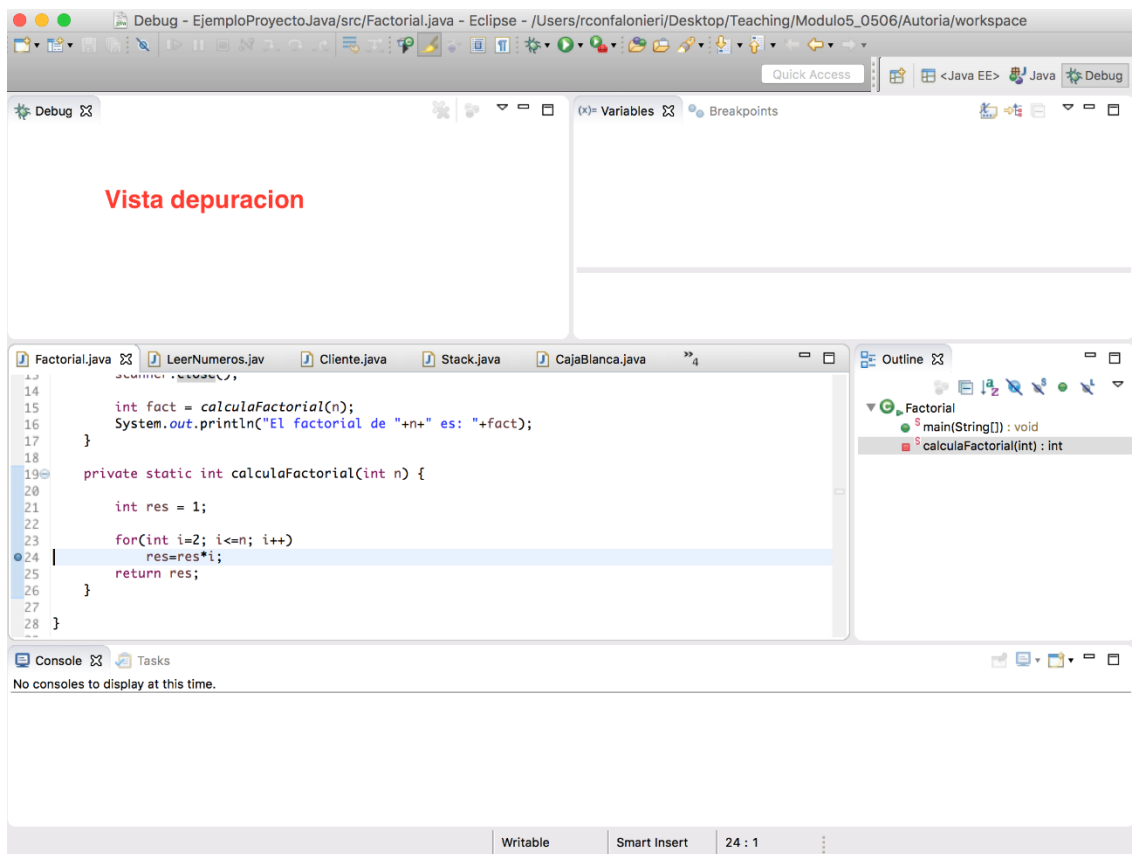
Solución Actividad 5:

Captura de pantalla que muestre las siguientes funcionalidades:

- Explorador de proyecto de Eclipse y editor de código de Eclipse.
- Vista Java de Eclipse.



- Vista de depuración de Eclipse.



- Ejecutar el código en modo depuración, introducir el valor 5 y añadir las capturas de pantalla que muestren los valores de la variable res del método calculaFactorial.

