Entornos de desarrollo

Desarrollo de aplicaciones Multiplataforma &

Desarrollo de aplicaciones Web

Conceptos de desarrollo y IDE’s

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.

|  |
| --- |
| **¿Cómo lo hago?** |
| 1. Rellena los datos que se piden en la tabla “Antes de empezar”. 2. Haz uso de fuentes comunes como Arial, Calibri, Times New Roman etc. 3. Utiliza el color negro para desarrollar tus respuestas y usa otros colores para destacar contenidos o palabras que creas necesario resaltar. 4. Recuerda entregar la actividad en formato PDF a no ser que el profesor o profesora indique lo contrario. 5. Recuerda nombrar el archivo siguiendo estas indicaciones:  * Ciclo\_Módulo o crédito\_Tema\_ACT\_número actividad\_Nombre y apellido   + Ejemplo: AF\_M01\_T01\_ACT\_01\_Maria Garcia |

|  |  |
| --- | --- |
| **Antes de empezar…** | |
| Nombre |  |
| Apellidos |  |
| Módulo/Crédito |  |
| UF (solo ciclos LOE) |  |
| Título de la actividad |  |

Actividad 1

***Características de los lenguajes de programación***

|  |  |
| --- | --- |
| **Conceptos de los lenguajes de programación** | **Definición** |
| Nivel de abstracción | El nivel de abstracción de un lenguaje implica lo alejado que esta del código máquina.  Cuanto mas parecido sea a nuestro lenguaje y menos al código máquina, mayor será el nivel de abstracción del lenguaje. |
| Paradigma de programación | El paradigma de programación se basa en el método para llevar a cabo los cálculos del proceso, y en la forma en la que deben estructurarse las tareas que debe realizar el programa. Se diferencian unos de otros en la forma de abstraer los elementos del lenguaje de programación y los pasos a seguir para la solución del problema. |
| Forma de ejecución | Los lenguajes, según su forma de ejecución pueden ser de 3 tipos:  -Compilados, antes de ejecutarse, es decir, bajar el nivel hasta el código máquina.  -Interpretados, se ejecutan línea a línea. Un intérprete ejecuta el código.  -Virtuales, se compilan pero no generan ejecutable, usan una maquina virtual capaz de entender el código intermedio bytecode y lo ejecuta en la maquina física. |

En esta actividad deberás:

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

**Conceptos de los lenguajes de programación**

1. 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 | Consiste en un programa que recibe como entrada el código fuente de otro programa. | Una salida compuesta de tokens. |
| Análisis Sintáctico-Semántico | Se comprueba el texto de entrada en base a una gramática dada, la del lenguaje de programación | Un código correcto. |
| Generación de código intermedio | Transforma el código al lenguaje más próximo a la plataforma de ejecución. | Un código intermedio. |
| Optimización de código | Realiza una serie de mejoras del código. | Un código optimizado. |
| Generación de código | El compilador convierte el programa sintácticamente correcto en una serie de instrucciones que deben ser interpretadas por una máquina. | A partir del generador de código se obtiene el código objeto. |
| Enlazador de librerías | A partir del código generado y los recursos necesarios ( bibliotecas)  quita los recursos que no necesita y enlaza los que necesite al código objeto. | Genera un código ejecutable. |

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

1. Rellenar la siguiente tabla:

|  |  |
| --- | --- |
| **Tipo de lenguaje** | **Ejemplos de lenguajes** |
| De primera generación | Lenguaje màquina. |
| De segunda generación | Lenguaje ensamblador, Fortran. |
| De tercera generación\* | C, C#, Java |
| De cuarta generación\* | PHP, .NET |
| De quinta generación | Mercury, OPS5 |
| Compilado\* | C++, Go, Rust |
| Interpretado\* | Ruby, Phyton |
| Imperativo\* | BASIC, Perl |

\*mínimo dos ejemplos de lenguajes

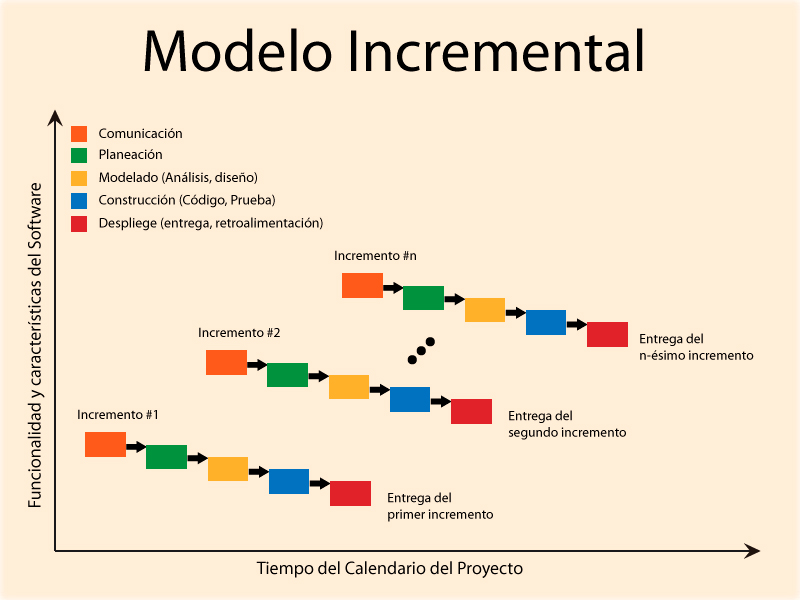
Actividad 2

***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 incremental**

* 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



http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

El modelo de desarrollo incremental es el ciclo de vida de desarrollo software en el cual un proyecto es descompuesto en una serie de incrementos, cada uno de los cuales suministra una porción de la funcionalidad respecto de la totalidad de los requisitos del proyecto.

Los requisitos tienen asignada una prioridad y son entregados según el orden de prioridad en el incremento correspondiente. En algunas (pero no en todas) versiones de este modelo de ciclo de vida, cada subproyecto sigue un “mini-modelo V” con sus propias fases de diseño, codificación y pruebas.

***Pasos***

* Comunicación de los objetivos y especificaciones que busca el proyecto.
* Planeación de las tareas e iteraciones, cada una debe conseguir objetivos específicos.
* Modelado (análisis, diseño), se define la evolución de las iteraciones y si supera a la que precede. Se establecen los incrementos.
* Construcción (Código, prueba),se construye el código y se prueba.
* Despliegue (Entrega, retroalimentación)

Se genera software operativo de forma rápida, es apropiado para proyectos grandes. Me parece mejor que el método en espiral o en cascada, por su rapidez, por su financiación por partes del proyecto y por su baja probabilidad de fallar en el objetivo final.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:

|  |  |  |
| --- | --- | --- |
| **Definición de patrón de diseño** | | |
| Solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de **plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales**a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error. | | |
| **Tipo de patrón de diseño** | **Definición** | **Ejemplos de patrones** |
| Creacionales | Proporcionan diversos mecanismos de creación de objetos, que aumentan la flexibilidad y la reutilización del código existente de una manera adecuada a la situación. | Abstract Factory,  Builder Patterns,  Factory Method. |
| Estructurales | Facilitan soluciones y estándares eficientes con respecto a las composiciones de clase y las estructuras de objetos. | Adapter,  Bridge,  Decorator. |
| Comportamiento | Se ocupan de la **comunicación entre objetos de clase.** Se utilizan para detectar la presencia de patrones de comunicación ya presentes y pueden manipular estos patrones. | Command,  Interpreter,  Iterator. |

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

2- A continuación, verás un **caso de aplicación de un patrón de diseño**. Este caso es 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 Autentificacion{  
        /\* ... \*/  
        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(){}  
}

Aplicación del patrón fachada a continuación:

public class FachadaCajero{  
        private Autentificacion = new Autentificacion();  
        private Cajero = new Cajero();  
        private Cuenta = null;  
        public void introducirCredenciales(){  
                boolean tarjeta\_correcta = autentificacion.leerTarjeta();   
                if(tarjeta\_correcta){  
                        String clave = autentificacion.introducirClave();  
                        boolean clave\_correcta = autentificacion.comprobarClave(clave);  
                        if(clave\_correcta){  
                                cuenta = autentificacion.obtenerCuenta();  
                                return;  
                       }  
                }  
                autentificacion.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):**

|  |
| --- |
| public class FachadaCajero{         private Autentificacion = new Autentificacion();         private Cajero = new Cajero(); Declaracion de variables         private Cuenta = null;         public void introducirCredenciales(){                 boolean tarjeta\_correcta = autentificacion.leerTarjeta();                  if(tarjeta\_correcta){                         String clave = autentificacion.introducirClave(); **Bucle para comprobar**                         boolean clave\_correcta = autentificacion.comprobarClave(clave); **si la clave de tarjeta es**                          if(clave\_correcta){ **correcta**                                 cuenta = autentificacion.obtenerCuenta();                                 return;                        }                 } **Si la autentificacion fallase**         } **llamaria al metodo Fallar()**                 autentificacion.alFallar();          public void sacarDinero(){                 if(cuenta != null){ **Si el numero de cuenta es correto**                         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); **Acciones que se haran si**                                          cuenta.actualizarCuenta(); **hay saldo disponible**                                         cuenta.desbloquearCuenta();                                        cajero.expedirDinero();                                        cajero.imprimirTicket();                                 }                                 else{                                         cuenta.alFallar(); **Metodo si falla el proceso**                                }                         }                 }         } }  Interfaz:  public static void main(String[] args){         FachadaCajero cajero\_automatico = new FachadaCajero();         cajero\_automatico.introducirCredenciales(); **Métodos**         cajero\_automatico.sacarDinero(); } |
|  |
|  |

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** resultado = 1;

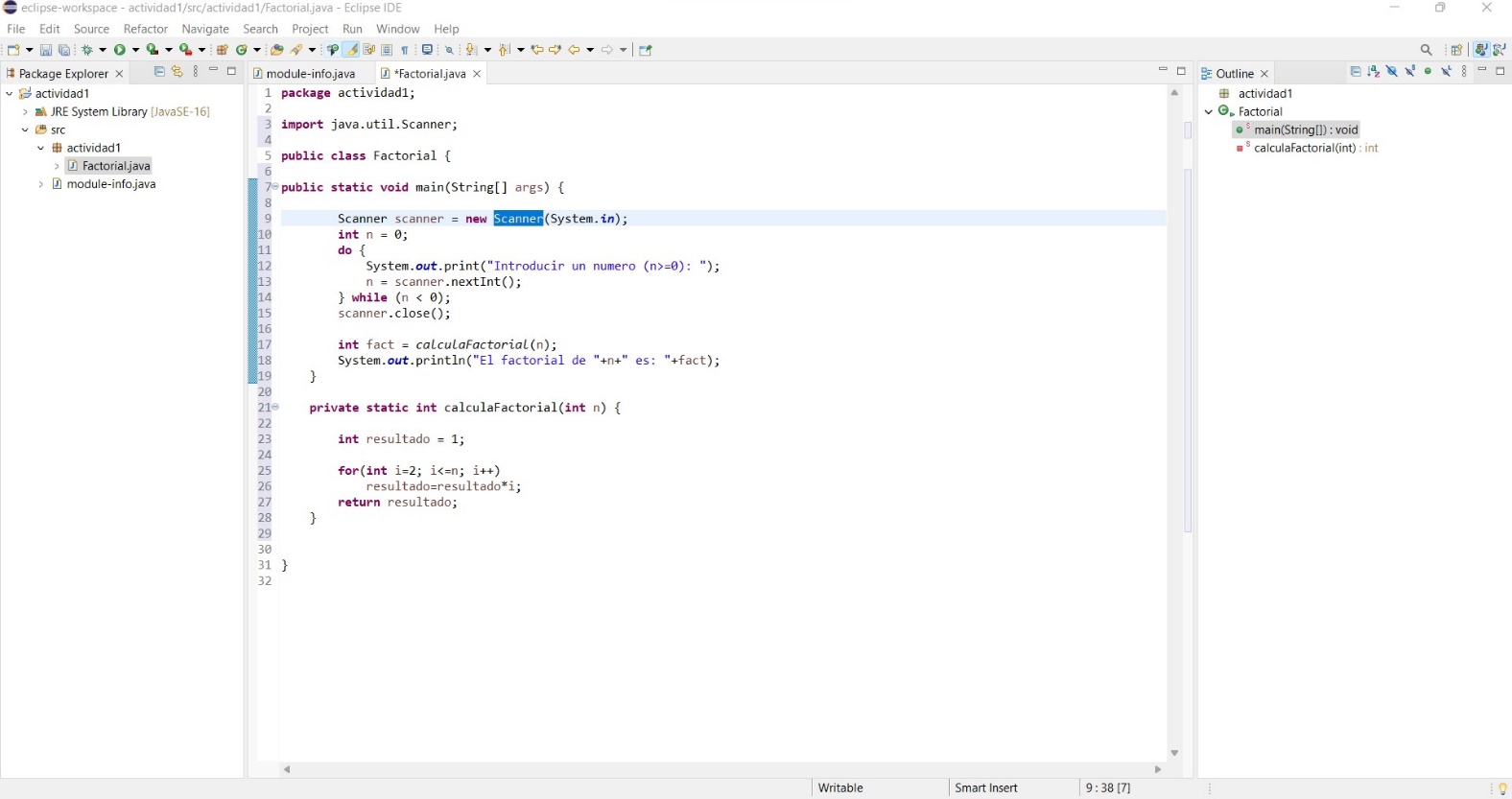
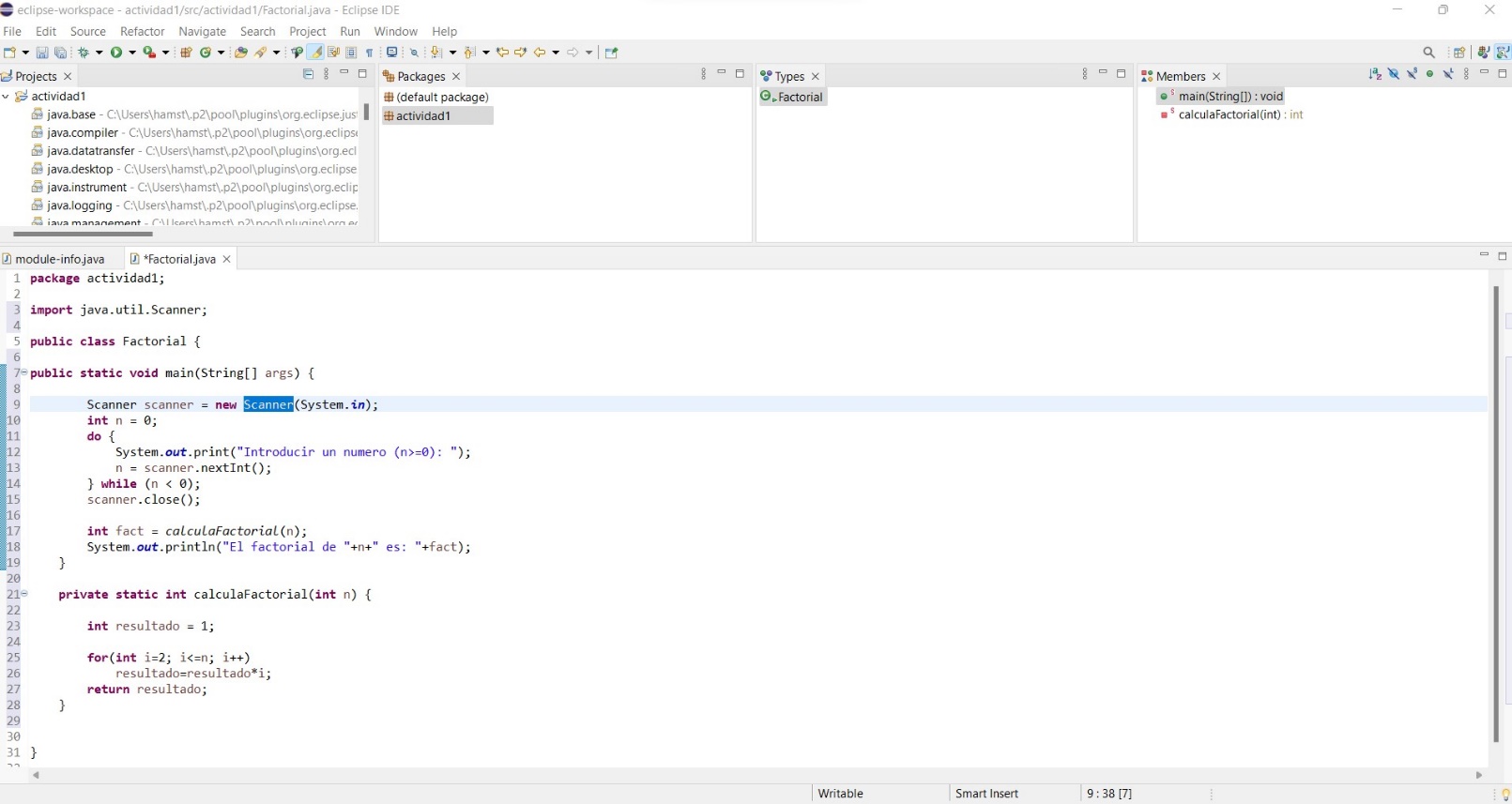
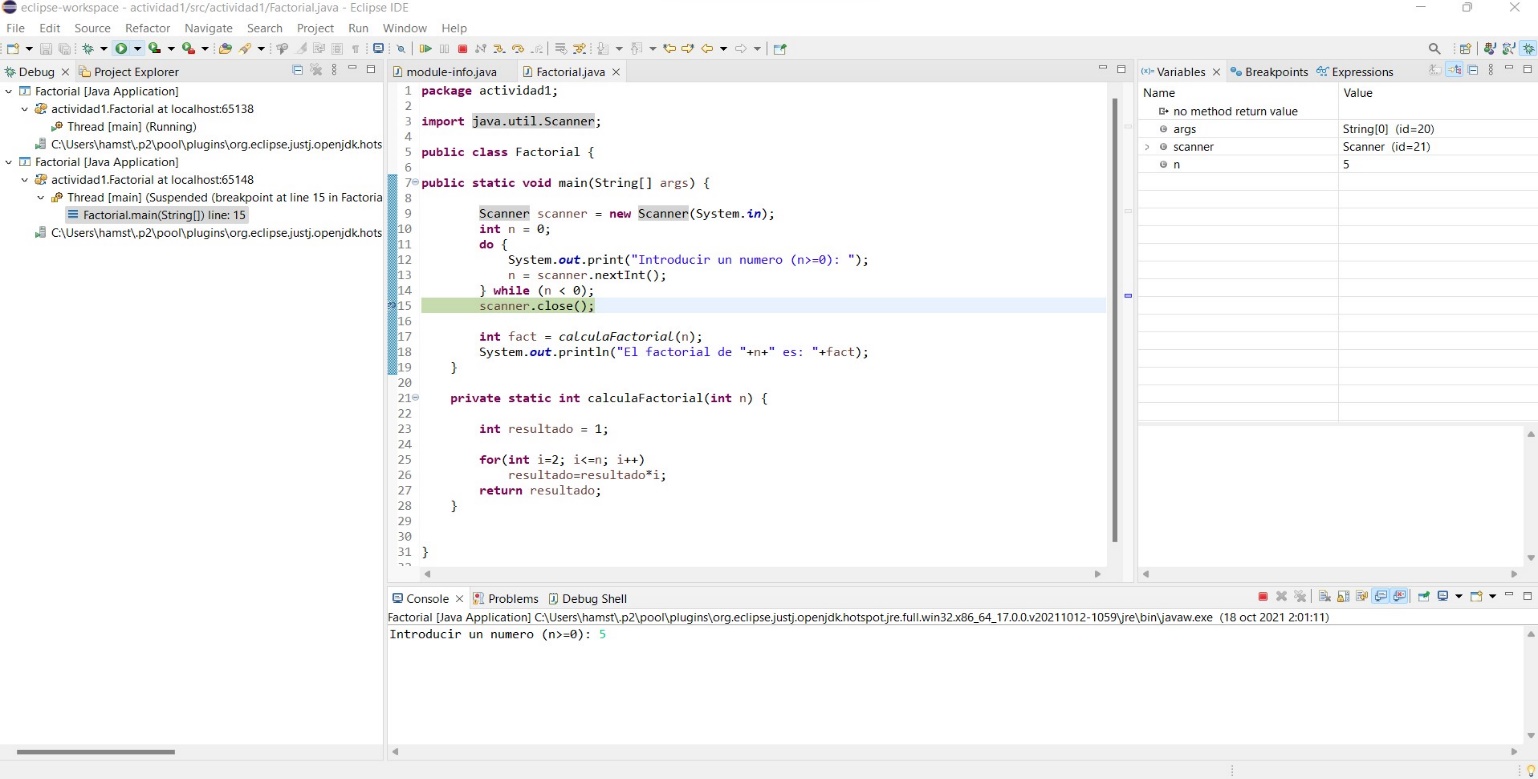
**for**(**int** i=2; i<=n; i++)

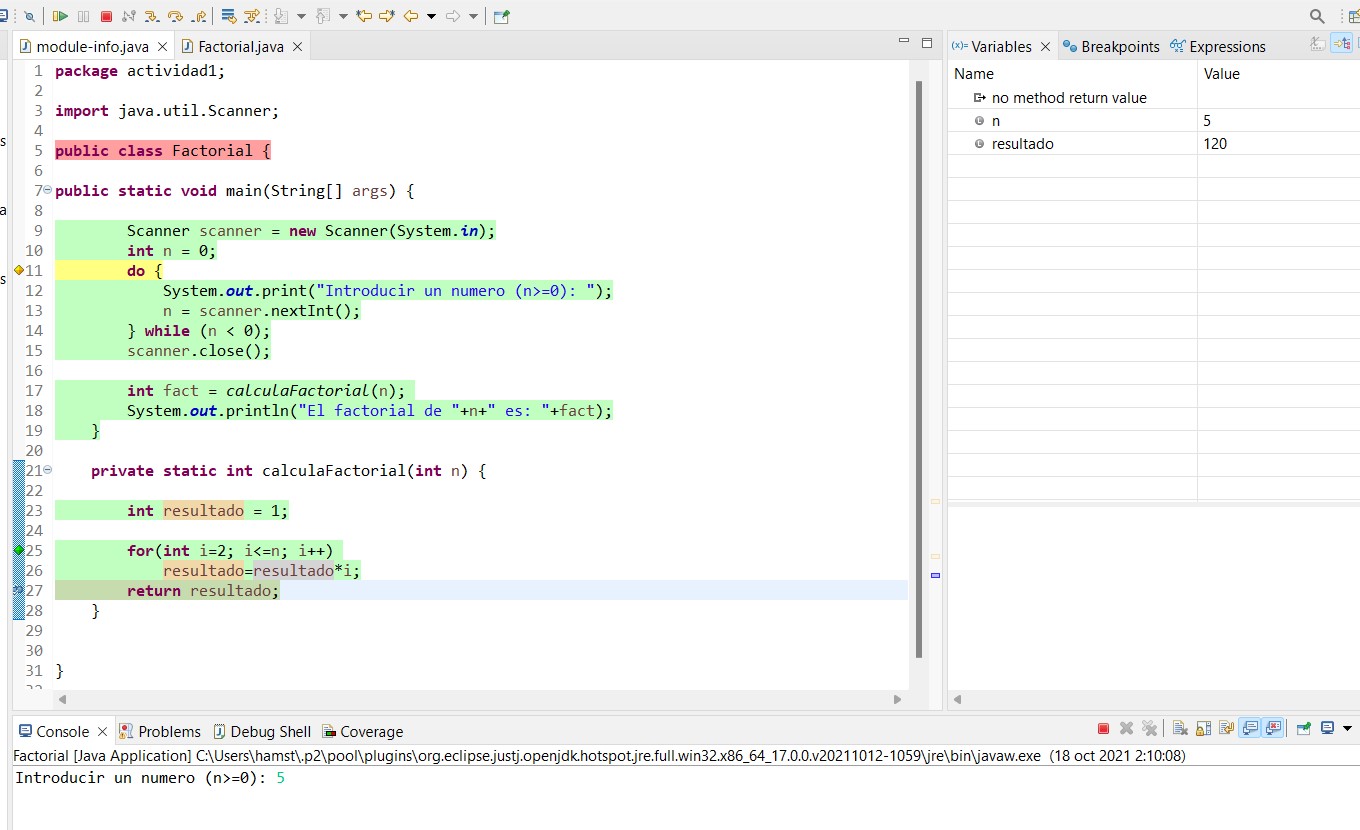
resultado=resultado\*i;

**return** res;

}

1. 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.

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