

PRACTICA TEMA 2

Antonio Muñoz Cubero

12 de Noviembre de 2020

Índice

| | |
|---|-----------|
| 1. Enunciado | 1 |
| 2. Primeros pasos | 2 |
| 2.1. Creacion del Workspace | 2 |
| 2.2. Personalización del IDE | 3 |
| 3. Proyecto | 5 |
| 3.1. Creación del Proyecto | 5 |
| 3.2. Creación de la Clase | 7 |
| 3.3. Primera compilación | 9 |
| 3.4. Depuración y corrección de errores | 10 |
| 3.5. Ejecución final | 11 |
| 4. Desinstalar Plugins | 13 |
| 5. Actualizar complementos | 14 |
| 6. Índice de Figuras | 15 |

1. Enunciado

Tenemos que copiar el código del archivo adjunto, que este caso, muestro en la imagen inferior.

- Compilar el programa y corregir los errores sintácticos. Incluyendo las librerías que sean necesarias.
- Realizar la depuración y posterior ejecución del programa.
- Personalizar la configuración del IDE, creando una configuración nueva, llamada **miconfiguración** para ese proyecto de la siguiente manera: El directorio de trabajo será una carpeta llamada *DirectorioTrabajoED*, que estará situada en el escritorio.
- Tras ello, desinstalar los plugins instalados anteriormente.
- Actualizar todos los complementos del IDE (si no hay nada que actualizar pon la captura de la pantalla donde se indica).

```
/* Aplicación que convierte un número en decimal a su correspondiente en binario */
package ed_tarea2;

public class ED_Tarea2 {

    public static void main(String[] args) {
        String texto=JOptionPane.showInputDialog("Introduce un numero");
        int numero=Integer.parseInt(texto);
        String binario=decimalBinario(numero);
        System.out.println("El numero "+numero+ " en binario es "+binario);
    }
    public static String decimalBinario (int numero){
        String binario="";
        String digito;
        for(int i=numero;i>0;i/=2){
            if(i%2==1){
                digito="1"
            }else{
                digito="0";
            }
            binario=digito+binario;
        }
        return binario;
    }
}
```

2. Primeros pasos

2.1. Creacion del Workspace

Lo primero que hacemos al empezar a trabajar en un proyecto utilizando, en este caso, el IDE *Eclipse*, es crear nuestro lugar de trabajo o como en el IDE es llamado "**Workspace**". realizandose de la siguiente manera: Una vez tenemos

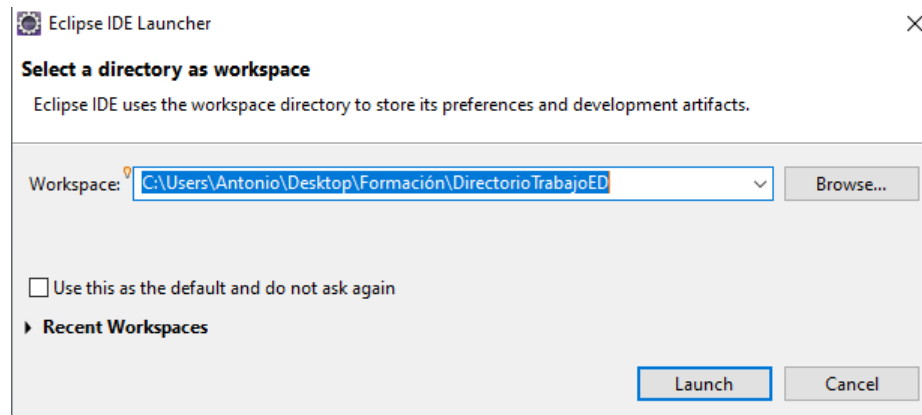


Figura 1: Ventana al iniciar Eclipse donde elegimos nuestro workspace.

nuestro espacio de trabajo, ahora procedemos a personalizar nuestro IDE para comenzar a escribir nuestro código.

2.2. Personalización del IDE

En mi caso lo primero que haré será modificar las preferencias para que al "tabular", Eclipse interprete que ponga **2 espacios** y no lo que venga por defecto. Esto es algo de gusto personal y no influye en nada más. Para cambiar dichas preferencias tenemos que irnos a *Window* → *Java* → *Code Style* → *Formatter*, una vez en *Formatter* le damos a *new*.

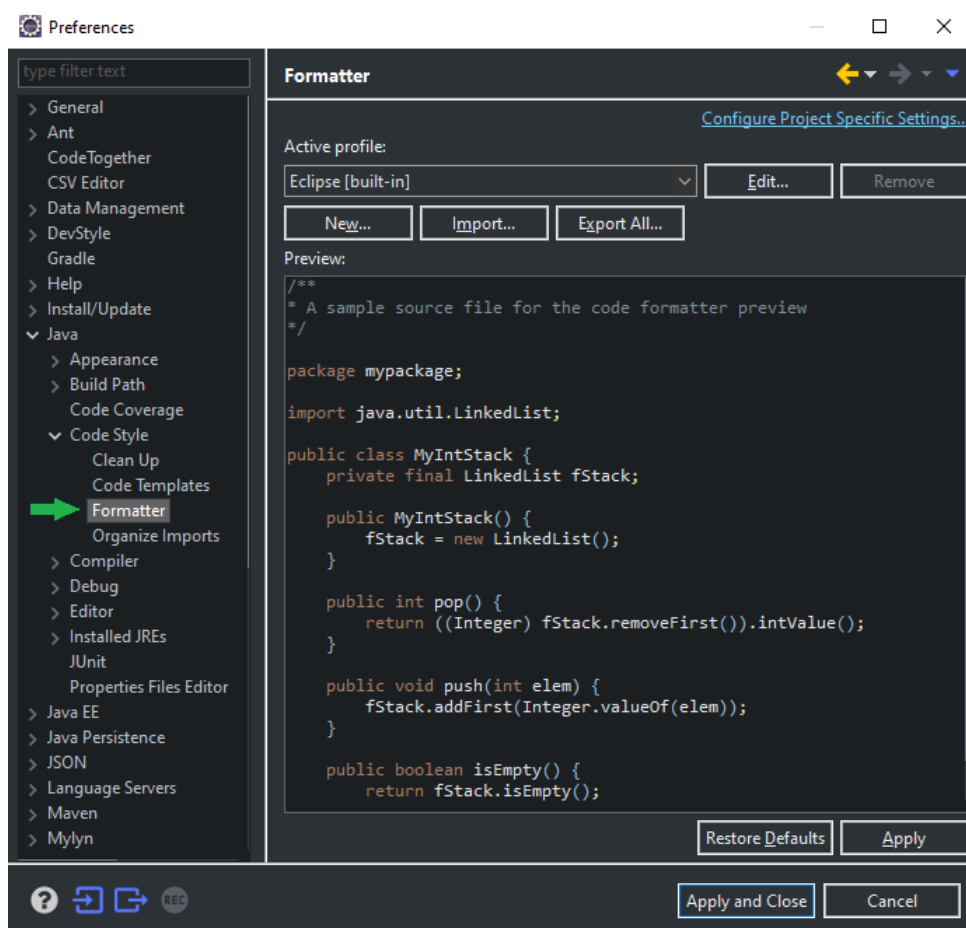


Figura 2: Ventana para añadir o editar formatos

Una vez dentro, ponemos nombre a nuestra nueva configuración, en este caso, tal y como dicta esta práctica se llamará **miconfiguracion**. Además he añadido la siguiente configuración.

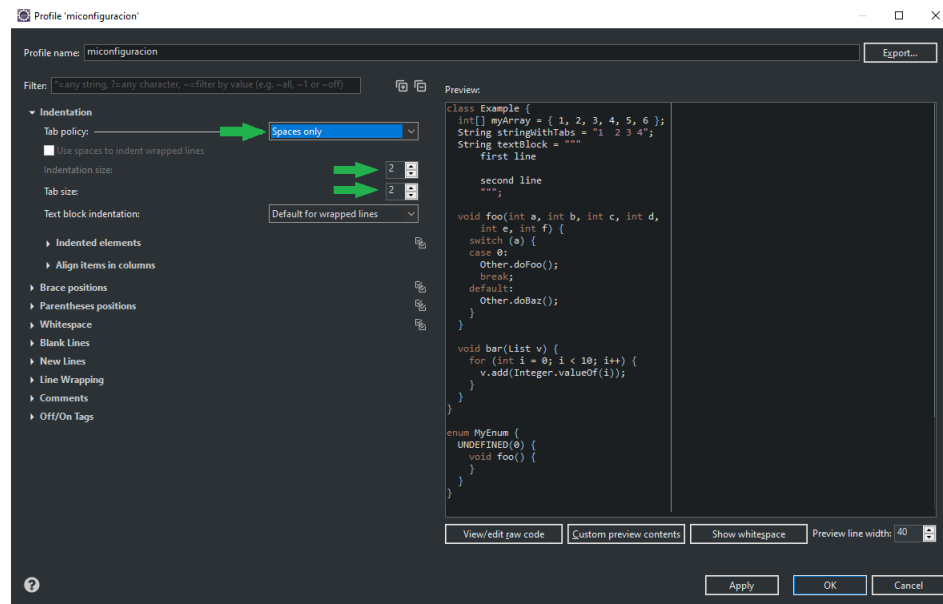


Figura 3: Preferencias de *miconfiguracion*

Y hecho esto, ya tendríamos el formato que queremos, además de eso, yo quiero instalar un "Plugin" para que Eclipse se vea más bonito. Por tanto para ello nos iremos a *Help* → *Marketplace* → *Populars* e instalamos **Darkest Dark Theme** como vemos en la imagen.

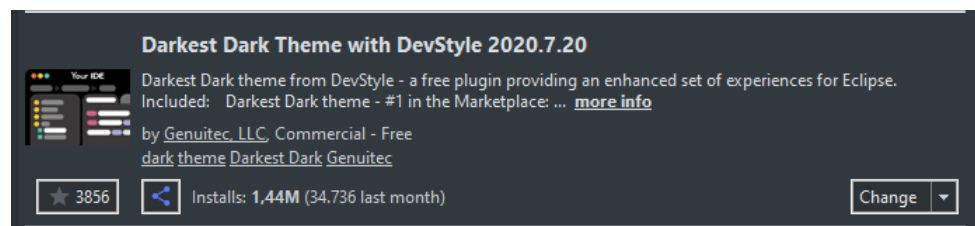


Figura 4: Plugin para un nuevo Theme de Eclipse

3. Proyecto

3.1. Creación del Proyecto

Una vez tenemos nuestro entorno de trabajo listo a nuestro gusto y con todas las herramientas funcionando a la perfección, pasamos a la creación del proyecto. Para ello, seguimos la siguiente ruta una vez tengamos abierto nuestro Workspace como hemos hecho anteriormente, *File* → *New* → *Project*.

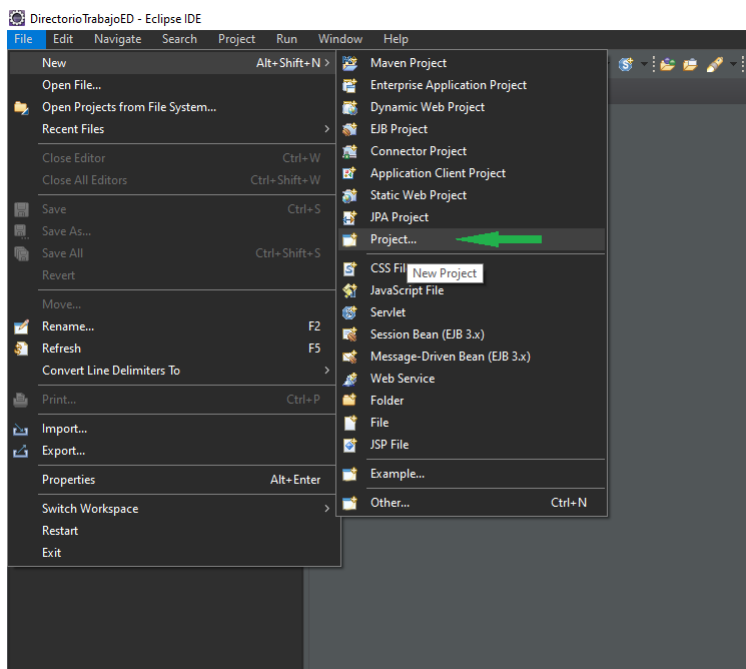


Figura 5: Creación de un nuevo proyecto

Una vez seleccionado *Project* nos aparecerá una lista donde deberemos seleccionar *Java* → *Java Project*.

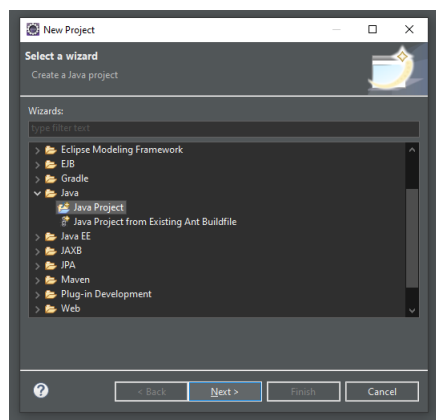


Figura 6: Distintos formatos de proyectos en Eclipse

Una vez seleccionado *Java Project* nos aparecerá una ventana de Personalización para nuestro proyecto donde debemos incluir el nombre que le vamos a dar en mi caso **ED-Tarea2** seleccionar la ubicación por defecto y nuestro *JDK* por defecto y pulsar *Finish*.

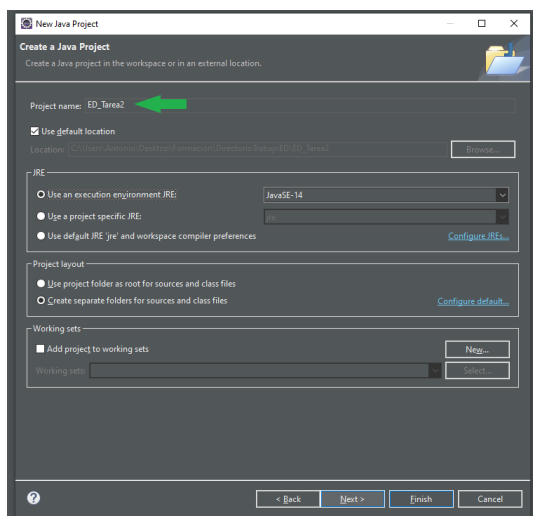


Figura 7: Configuración del proyecto, primera vista.

3.2. Creación de la Clase

Después de haber creado nuestro proyecto, el siguiente paso en la tarea es crear nuestra *Clase* para poder pasar a copiar nuestro código. Para ello debemos seguir la siguiente ruta *File* → *New* → *Class*.

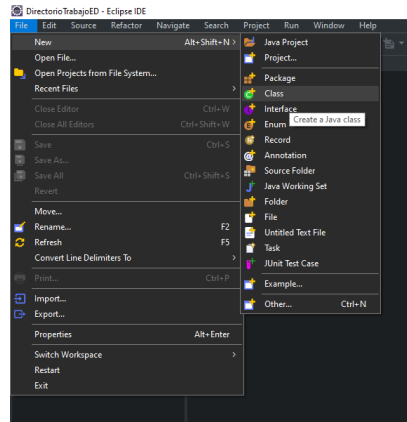


Figura 8: Creación de la clase

Una vez pulsado el botón de *Class* se nos abrirá una nueva ventana para configurar nuestra Clase Java, donde debemos ingresar el nombre de la clase, si queremos que se incluya en algún *Package*, y los métodos que queramos que cree por defecto, en nuestro caso le diremos que cree un método *main*.

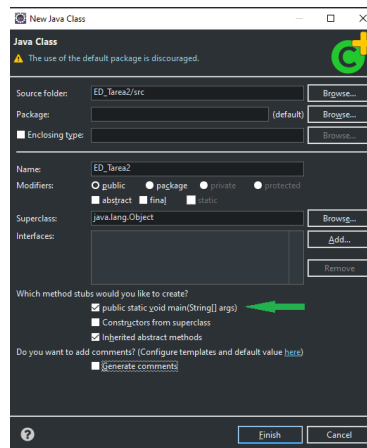


Figura 9: Configuración de la clase.

Una vez creada la clase ahora copiamos el código y seguimos los siguientes pasos.

1. Compilamos para ver errores en el código.
2. Ejecutamos para ver funcionamiento
3. Debugueamos para detectar el posible mal funcionamiento
4. Corregimos el funcionamiento
5. Compilamos y ejecutamos.

El código copiado quedaría tal que así

```
//Aplicacion que convierte un numero en decimal a su correspondiente en binario
public class ED_Tarea2 {

    public static void main(String[] args) {

        String texto = JOptionPane.showInputDialog("Introduce un numero");
        int numero = Integer.parseInt(texto);
        String binario = decimalBinario(numero);
        System.out.println("El numero "+numero+" en binario es "+binario);

    }

    public static String decimalBinario(int numero) {

        String binario="";
        String digito;

        for(int i = numero ; i>0 ; i/=2) {

            if(i%2==0) {
                digito="1";
            }else {
                digito="0";
            }

            binario = digito + binario;
        }

        return binario;
    }

}
```

Figura 10: Código fuente.

3.3. Primera compilación

Para compilar el programa por primera vez, abrimos nuestra terminal, en mi caso el **CMD** de Windows, en el directorio donde se encuentre nuestro fichero. Después de eso, debemos ejecutar la siguiente línea de comandos.

```
\$ javac ED_Tarea2.java
```

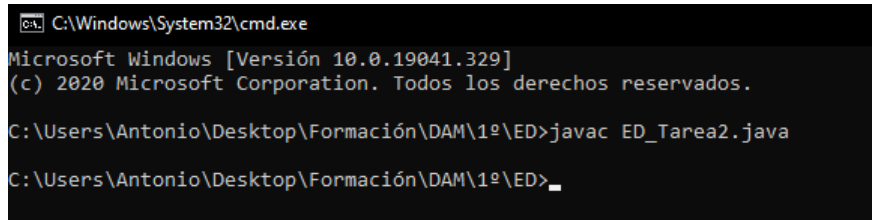


Figura 11: Primera compilación

Como observamos en la imagen, **no** se han producido errores, por tanto, procedemos a su ejecución.

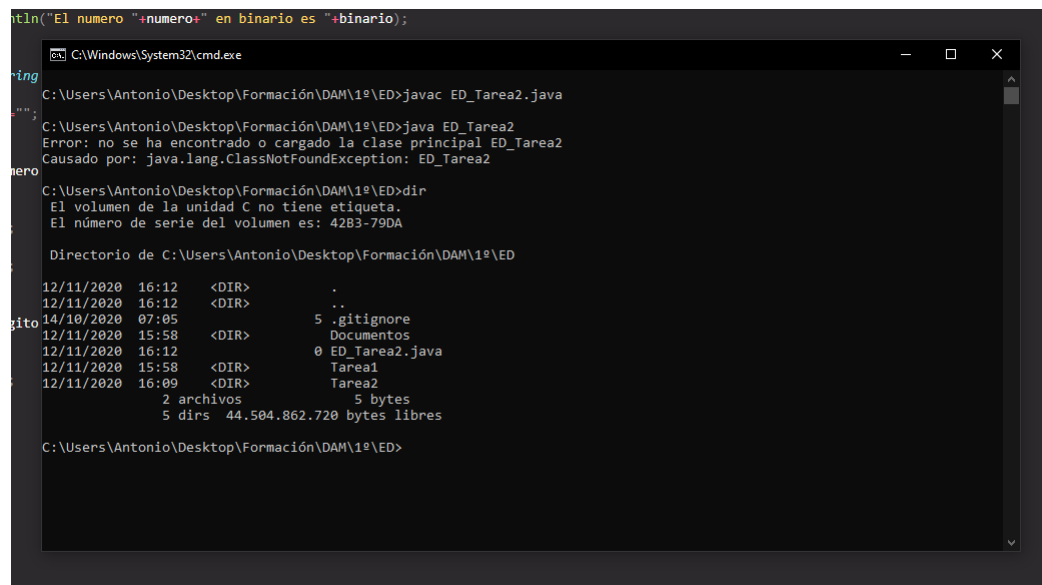


Figura 12: La compilación no crea el .class

3.4. Depuración y corrección de errores

Como hemos visto en la *Figura 10*, cuando hemos compilado, nuestro JDK no crea el .class necesario para la ejecución del programa. Por tanto a la hora de su ejecución no podemos llevarla a cabo, pasamos al siguiente paso, la depuración y corrección de errores.

Lo primero que logro identificar en el código de la *Figura 8*, es que no hemos importado las librerías para la ejecución del método **JOptionPane** y por tanto, esa puede ser una de las causas de dicho error. procedemos a la su implementación añadiendo en las primeras líneas de nuestro código lo siguiente:

```
1 import javax.swing.JOptionPane;
```

Esto hará que importe las librerías necesarias para su uso y no tengamos problemas, después de dicha importación. Procedamos al debugger.

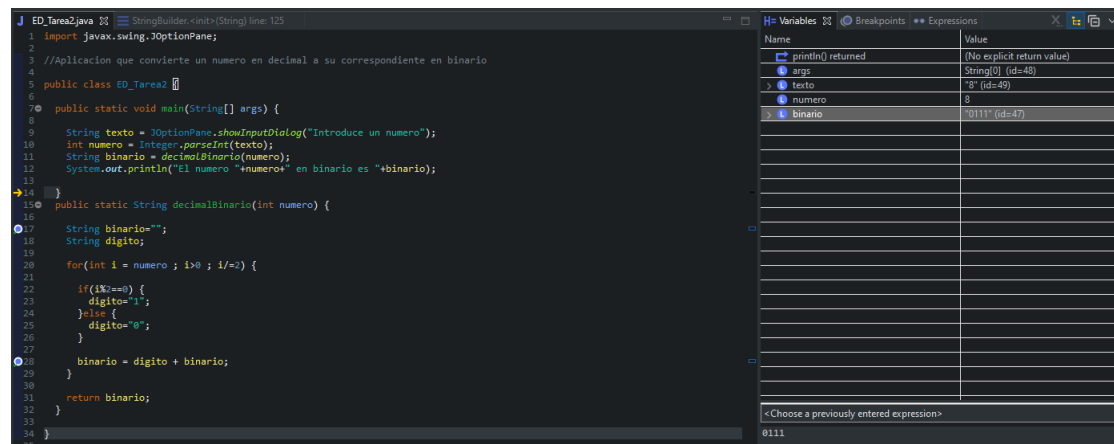


Figura 13: Ejecución con el debug

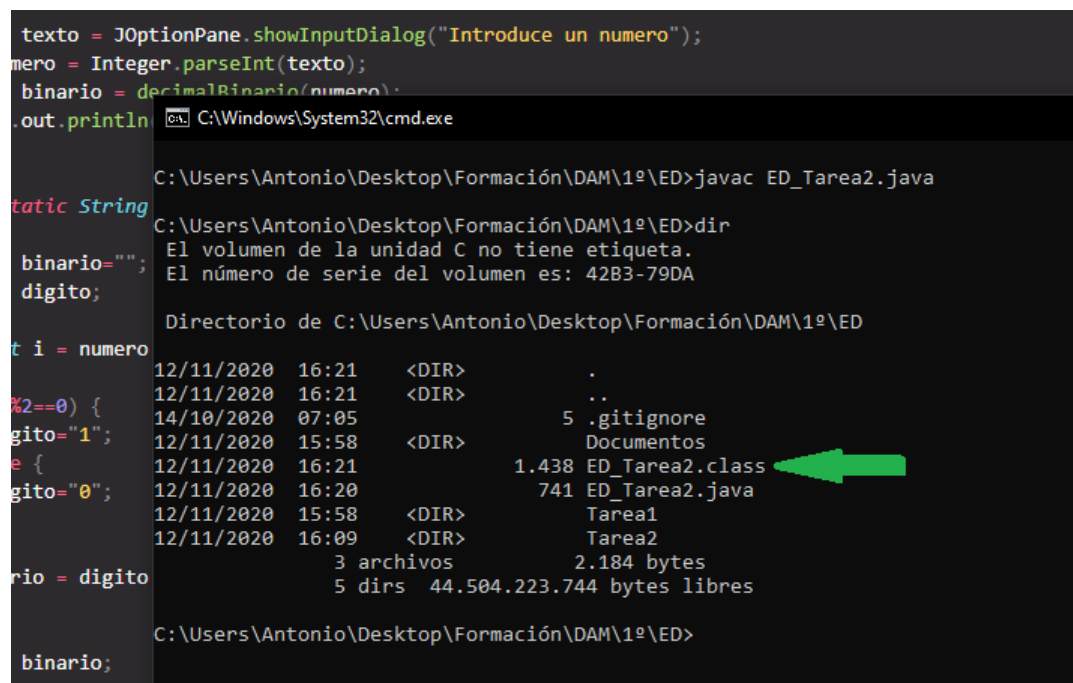
Observamos que el binario sale invertido, por tanto, su solución es sencilla, intercambiar las líneas 23 y 25.

3.5. Ejecución final

Una vez corregido los errores del código, procedemos a compilar el programa para asegurarnos de su buen funcionamiento. Ejecutando el mismo comando que anteriormente.

```
\$ javac ED_Tarea2.java
```

Ahora podemos observar como si nos ha creado nuestro **.class** correspondiente al archivo *ED-Tarea2.java*



```

texto = JOptionPane.showInputDialog("Introduce un numero");
numero = Integer.parseInt(texto);
binario = decimalBinario(numero);
System.out.println(binario);
}

static String decimalBinario(int numero) {
    String binario = "";
    while (numero > 0) {
        int digito = numero % 2;
        binario = digito + binario;
        numero = numero / 2;
    }
    return binario;
}

public static void main(String[] args) {
    int numero = 428379;
    String binario = decimalBinario(numero);
    System.out.println(binario);
}

```

```

C:\Windows\System32\cmd.exe
C:\Users\Antonio\Desktop\Formación\DAM\1º\ED>javac ED_Tarea2.java
C:\Users\Antonio\Desktop\Formación\DAM\1º\ED>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 42B3-79DA

Directorio de C:\Users\Antonio\Desktop\Formación\DAM\1º\ED
12/11/2020  16:21    <DIR>          .
12/11/2020  16:21    <DIR>          ..
14/10/2020  07:05           5 .gitignore
12/11/2020  15:58    <DIR>      Documentos
12/11/2020  16:21       1.438 ED_Tarea2.class
12/11/2020  16:20       741 ED_Tarea2.java
12/11/2020  15:58    <DIR>      Tarea1
12/11/2020  16:09    <DIR>      Tarea2
               3 archivos        2.184 bytes
               5 dirs      44.504.223.744 bytes libres

C:\Users\Antonio\Desktop\Formación\DAM\1º\ED>

```

Figura 14: Creación del .class

Ahora si podemos proceder a su ejecución, ejecutando el siguiente comando:

```
\$ java ED_Tarea2
```

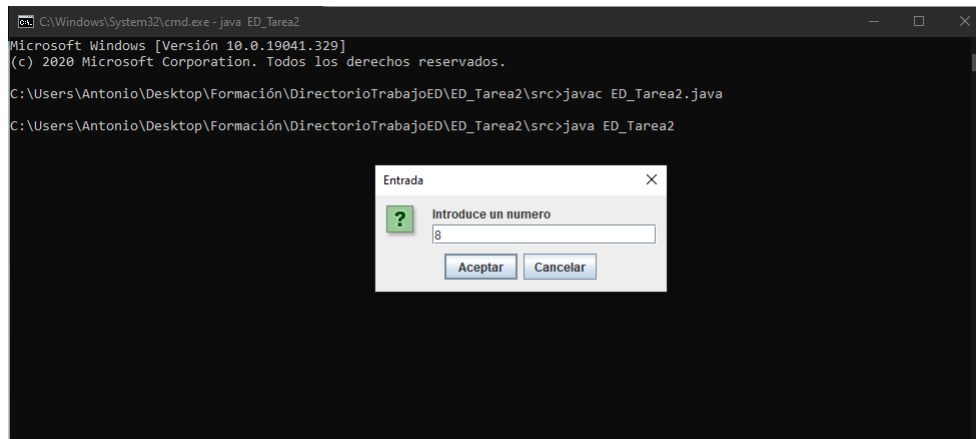


Figura 15: Ejecucion ED-Tarea2

Y obtenemos el siguiente resultado

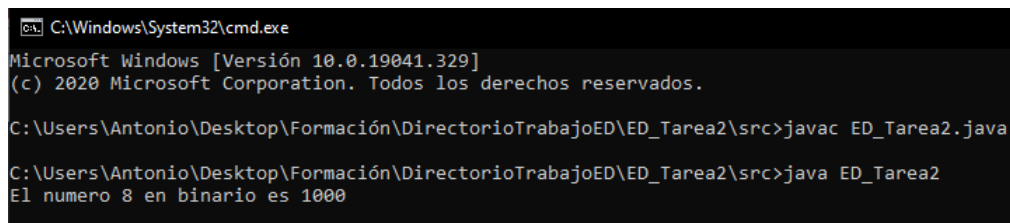


Figura 16: Resultado ED-Tarea2

Ahora nuestro programa funciona correctamente habiendo hecho buen uso del **Debug** y otras funcionalidades de nuestro *IDE Eclipse*.

4. Desinstalar Plugins

Para desinstalar los *plugins* instalados, debemos ir a la siguiente ruta *Help* → *Eclipse Marketplace* Una vez dentro nos vamos a la pestaña *Installed* y pulsamos

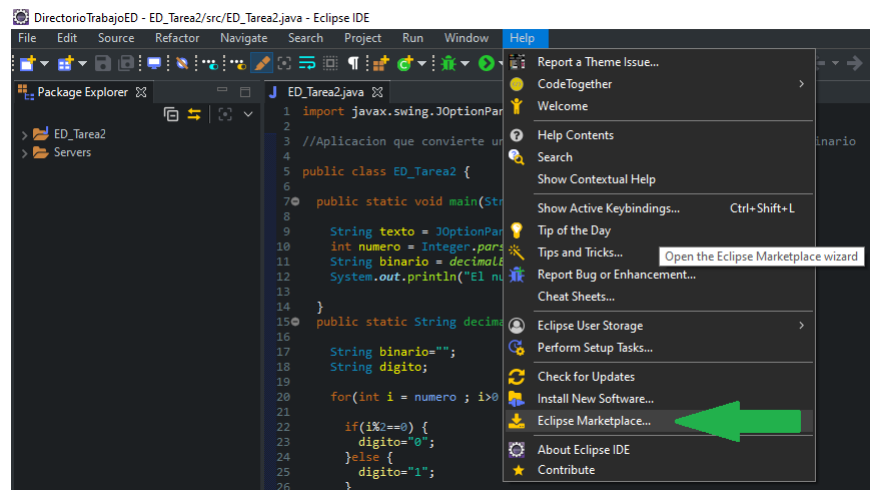


Figura 17: Ruta a Eclipse Marketplace.

en desinstalar en el plugin que deseemos desinstalar de nuestro IDE.

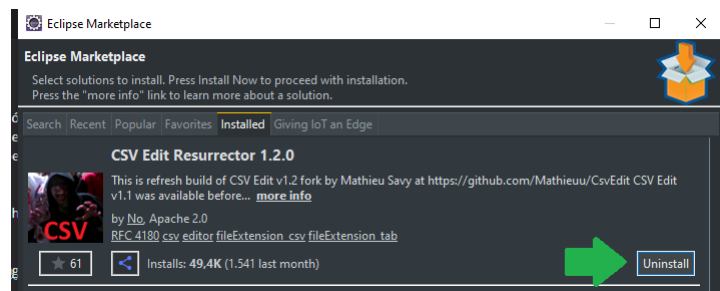


Figura 18: Desinstalar un plugin.

5. Actualizar complementos

Para actualizar los componentes de nuestro IDE, debemos acceder a la siguiente ruta *Help* → *Check for Updates* Acto seguido tendremos que esperar un

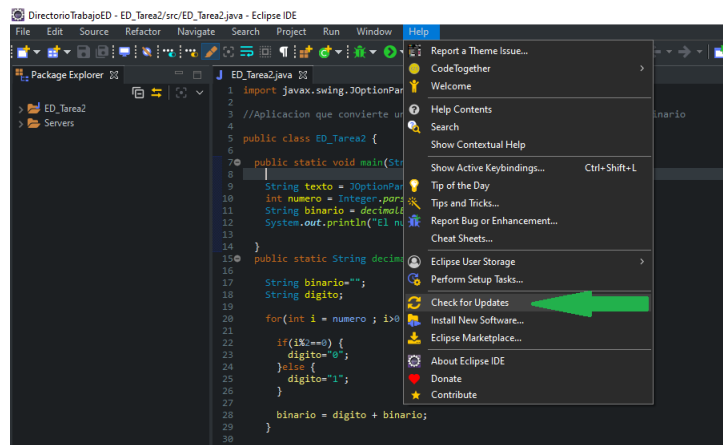


Figura 19: Ruta para actualizar componentes

poco mientras comprueba los paquetes que disponen de actualización. Una vez el IDE lo compruebe nos aparecerá una ventana donde muestra todos los paquetes que podemos actualizar, debemos pulsar en *Next* Después de esa ventana,

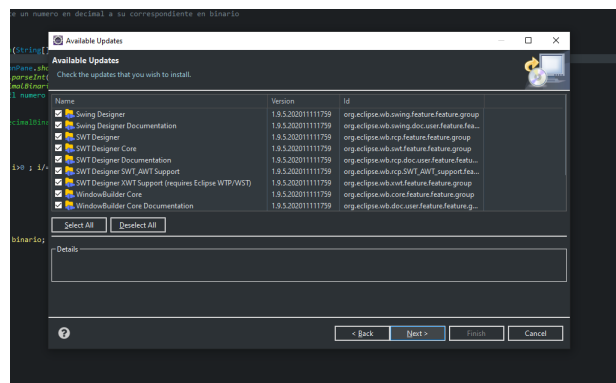


Figura 20: Actualizar componentes.

aceptaríamos los términos de uso y una vez el programa termine de actualizar todos los paquetes nos pedirá que reiniciemos *Eclipse*. Una vez reiniciado, ya tendríamos nuestros paquetes actualizados.

6. Índice de Figuras

Índice de figuras

| | | |
|-----|--|----|
| 1. | Ventana al iniciar Eclipse donde elegimos nuestro workspace. . . . | 2 |
| 2. | Ventana para añadir o editar formatos | 3 |
| 3. | Preferencias de <i>miconfiguracion</i> | 4 |
| 4. | Plugin para un nuevo Theme de Eclipse | 4 |
| 5. | Creación de un nuevo proyecto | 5 |
| 6. | Distintos formatos de proyectos en Eclipse | 6 |
| 7. | Configuración del proyecto, primera vista. | 6 |
| 8. | Creación de la clase | 7 |
| 9. | Configuración de la clase. | 7 |
| 10. | Código fuente. | 8 |
| 11. | Primera compilación | 9 |
| 12. | La compilación no crea el .class | 9 |
| 13. | Ejecución con el debug | 10 |
| 14. | Creación del .class | 11 |
| 15. | Ejecucion ED-Tarea2 | 12 |
| 16. | Resultado ED-Tarea2 | 12 |
| 17. | Ruta a Eclipse Marketplace. | 13 |
| 18. | Desinstalar un plugin. | 13 |
| 19. | Ruta para actualizar componentes | 14 |
| 20. | Actualizar componentes. | 14 |