Introducción de la materia

Este curso continuamos con el aprendizaje de la programación y nos introducimos en una nueva metodología que es la programación orientada a objetos.
En la materia anterior trabajamos los conceptos básicos de programación y la metodología de programación estructurada (organizar los programas en distintas funciones), ahora veremos en esta materia la programación orientada a objetos (POO)
Si bien el lenguaje Python permite trabajar con la metodología de POO en esta materia veremos esta metodología empleando el lenguaje Java, con el objetivo de prepararnos en las próximas materias a programar en la plataforma movil de Android.

Instalación de la máquina virtual de Java

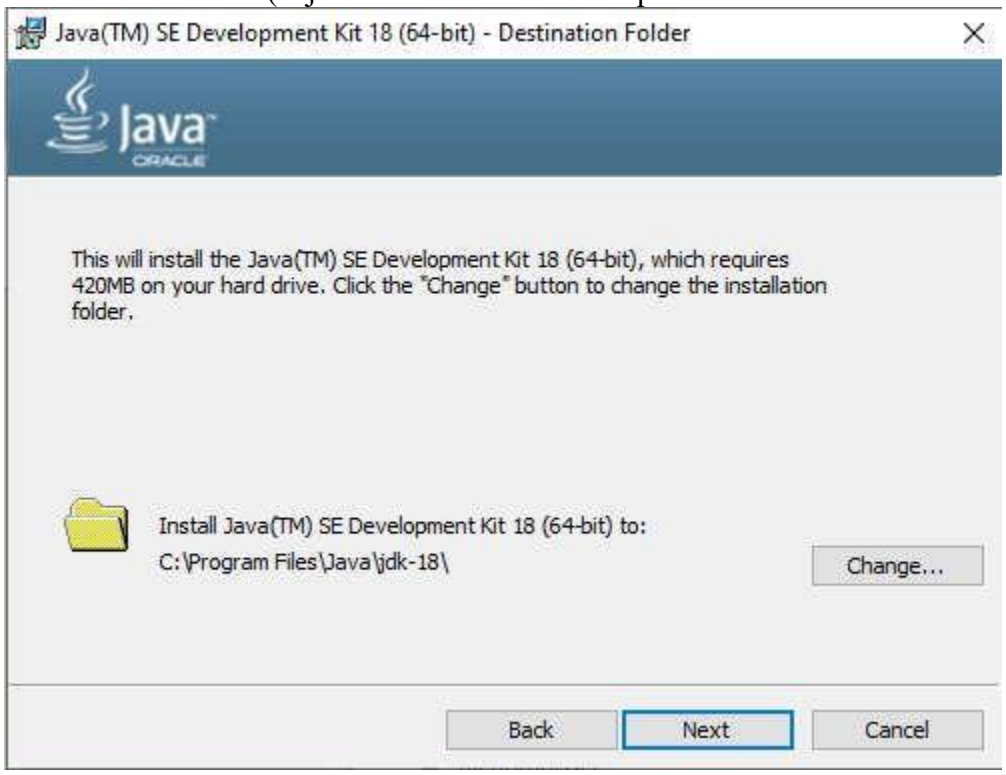
Para poder hacer este curso debemos instalar el compilador de Java y la máquina virtual de Java. Estas herramientas las podemos descargar de:
Java SE Development Kit 18.

Si disponemos como sistema operativo Windows debemos descargar:

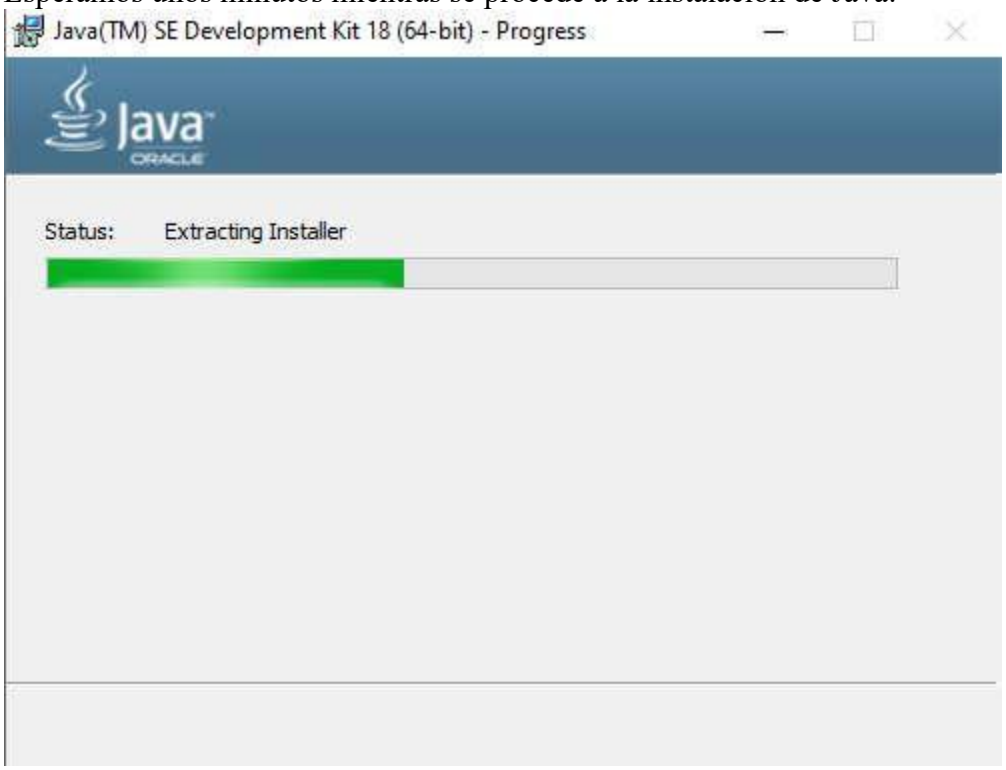
La versión a instalar conviene que sea la última (en el momento de desarrollar este curso disponemos la versión 18)

Una vez que tenemos el JDK (Java Development Kit) procedemos a instalarlo:

Presionamos el botón "next". Haremos la instalación por defecto por lo que presionamos el botón next nuevamente (dejaremos el directorio por defecto donde se almacenará Java):



Esperamos unos minutos mientras se procede a la instalación de Java:



Una vez finalizado el proceso de instalación debe aparecer un diálogo similar a este y presionamos el botón Close (por el momento no vamos a instalar "Tutorials, Api documentation

etc.):

Con esto ya tenemos instalado en nuestra computadora el compilador de Java y la máquina virtual que nos permitirá ejecutar nuestro programas. En el próximo concepto instalaremos un editor de texto para poder codificar nuestros programas en Java.

Instalación del editor Eclipse

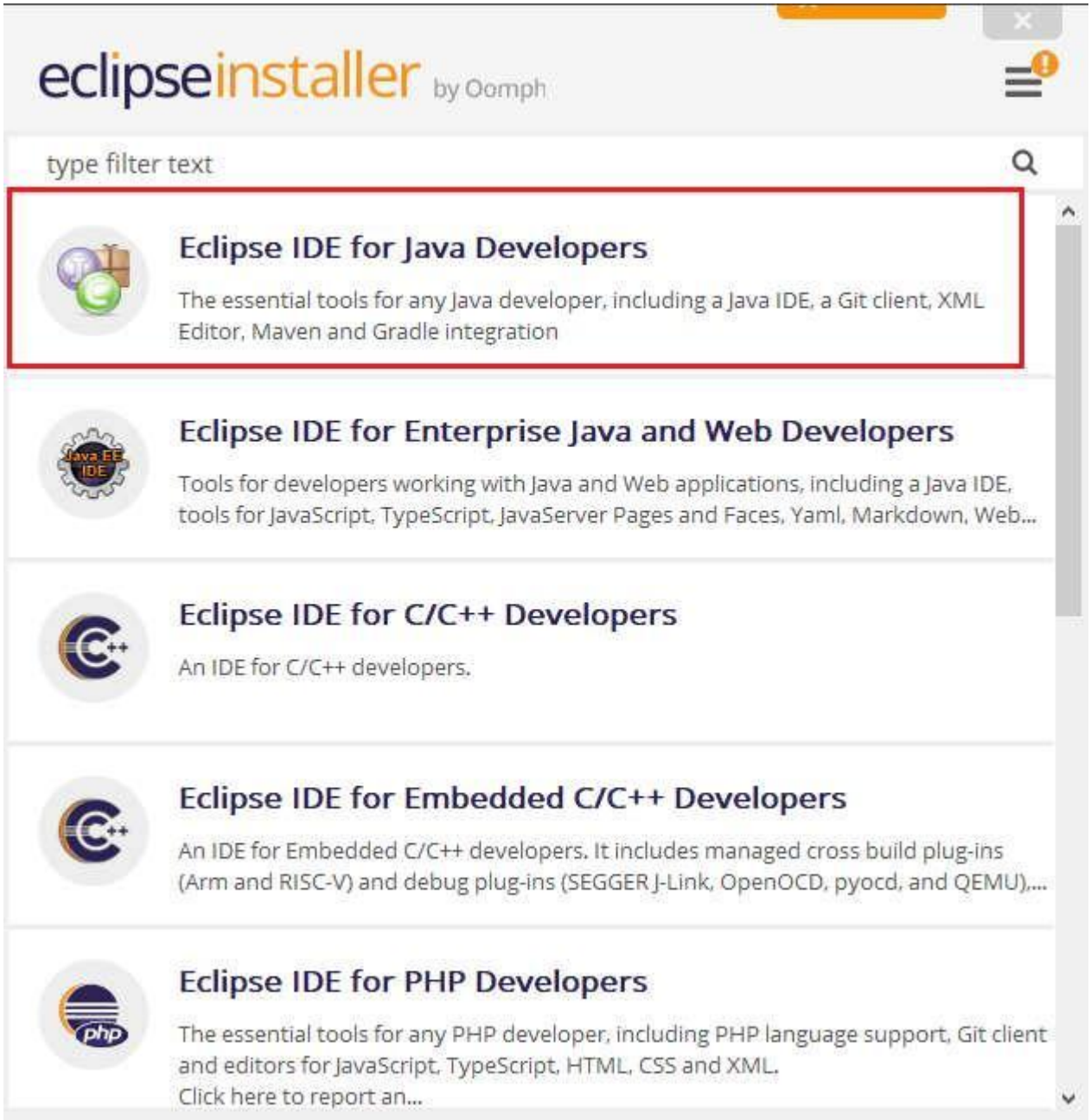
Ya tenemos instalado el lenguaje Java (con dichas herramientas podemos ejecutar y compilar programas codificados en java), pero ahora necesitamos instalar el Eclipse que es un editor para codificar los programas, de forma en Python disponemos el editor IDLE (si bien podemos utilizar otros editores a parte de Eclipse, nosotros utilizaremos este para seguir el curso)

Descarga

Para la descarga del editor Eclipse lo hacemos del sitio:
Eclipse

La versión a instalar en el momento de desarrollar este curso es la 2022 (puede instalar una más nueva si lo desea):

Una vez que descargamos el instalador de Eclipse procedemos a ejecutarlo y debemos elegir el entorno "Eclipse IDE for Java Developers":

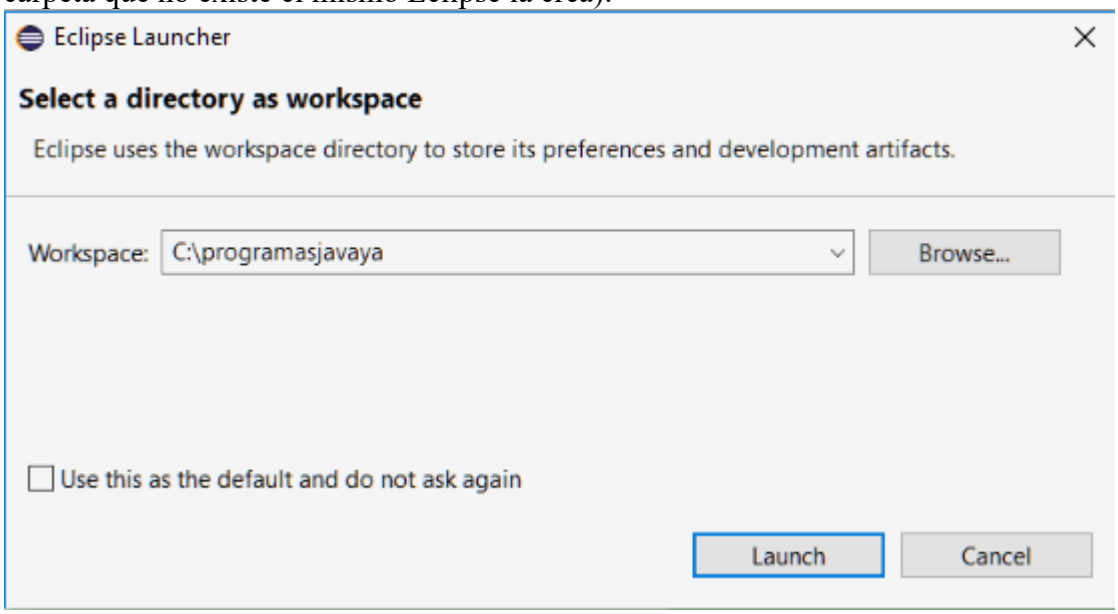


Seleccionamos la carpeta donde se instalará el Eclipse (podemos dejar la carpeta por defecto o elegir otra):

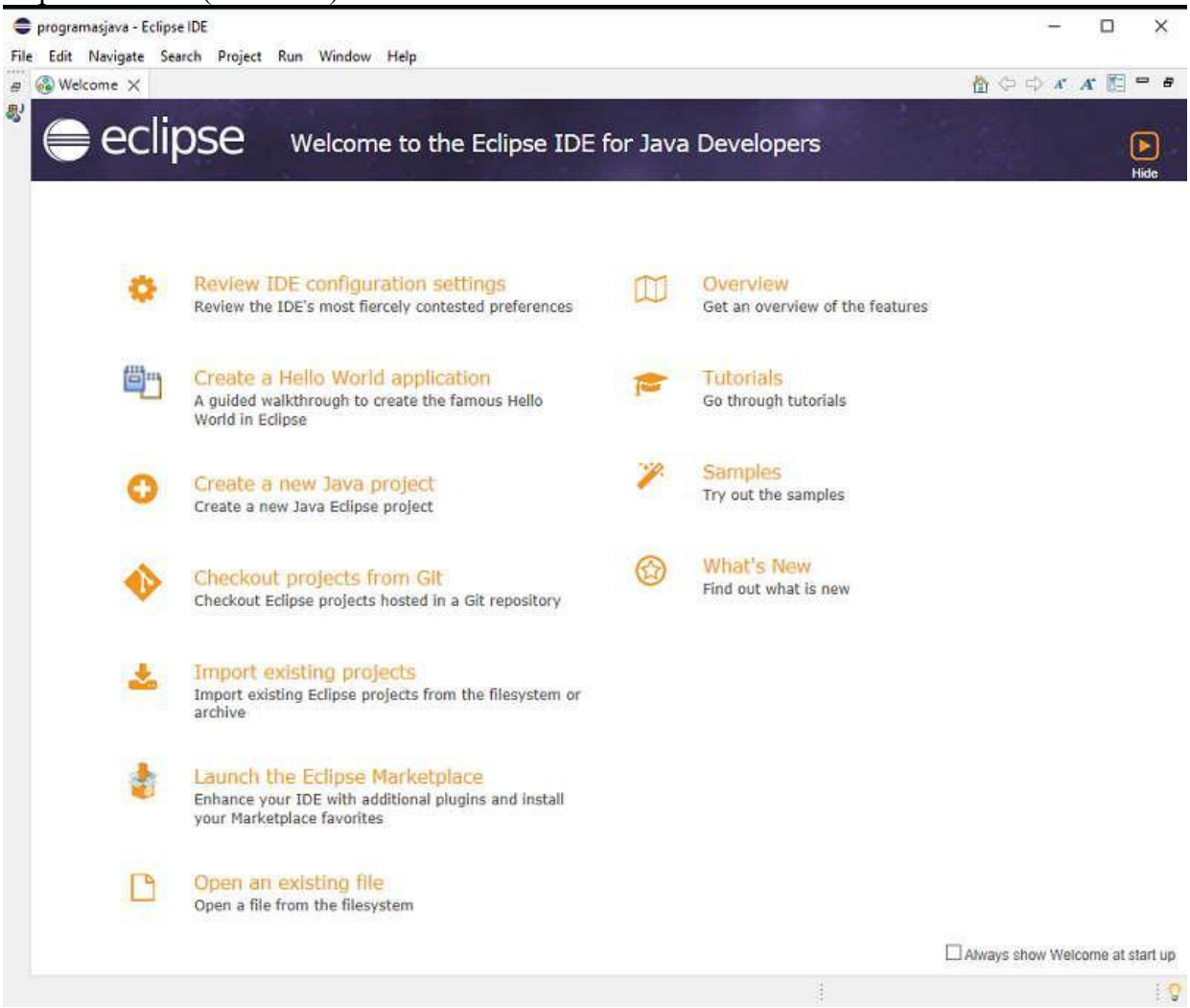
Una vez instalado en forma completo el "Eclipse IDE for Java Developers" nos dirigimos a la carpeta donde se instalaron los archivos y procedemos a ejecutar el programa eclipse.exe (podemos ingresar desde el acceso directo que se crea en el escritorio)

Primero aparece un mensaje de inicio del Eclipse:

Luego la primera vez que ejecutemos el editor Eclipse aparece un diálogo para seleccionar la carpeta donde se almacenarán los programas que desarrollaremos (podemos crear una carpeta donde almacenaremos todos los proyectos que desarrollaremos en el curso, si indicamos una carpeta que no existe el mismo Eclipse la crea):

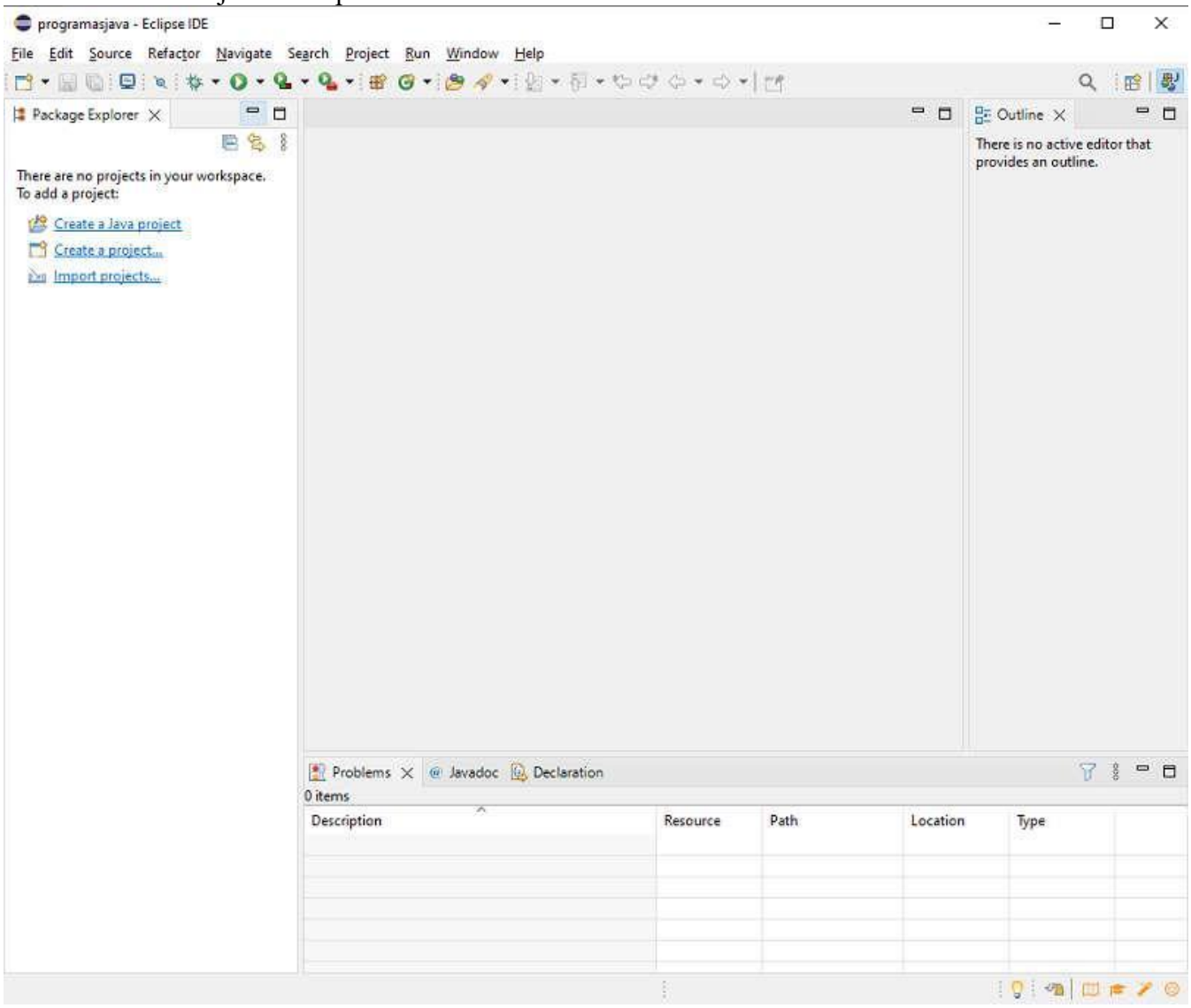


Luego de configurar la carpeta donde se crearán los proyecto aparece el editor con una pantalla de presentación (Welcome):



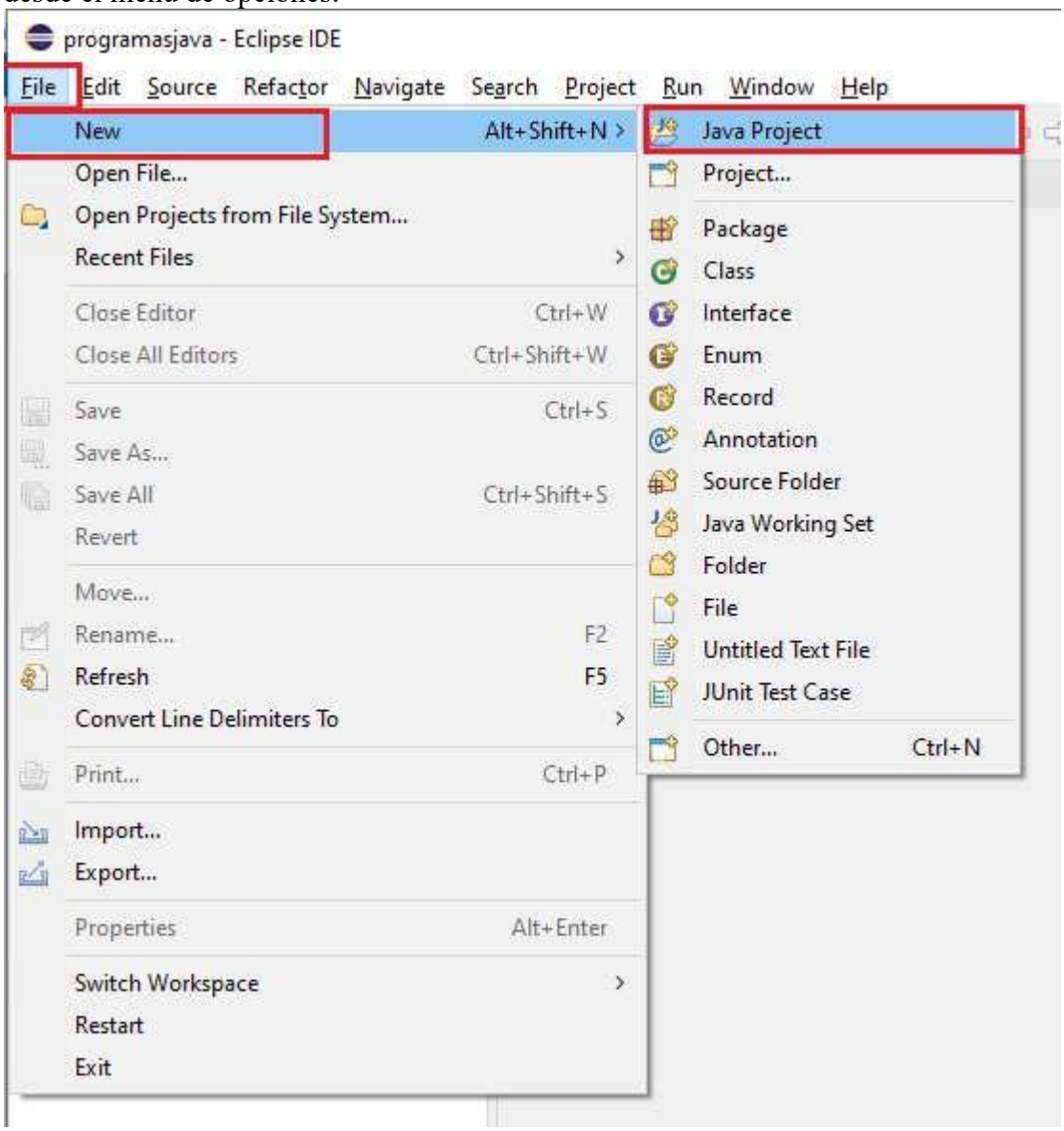
Esta ventana de bienvenida la podemos cerrar seleccionando el ícono: "Hide", con lo que aparece el entorno de trabajo del Eclipse (si queremos nuevamente ver la ventana de bienvenida podemos activarla desde el menú de opciones: Help -> Welcome)

El entorno de trabajo del Eclipse es:

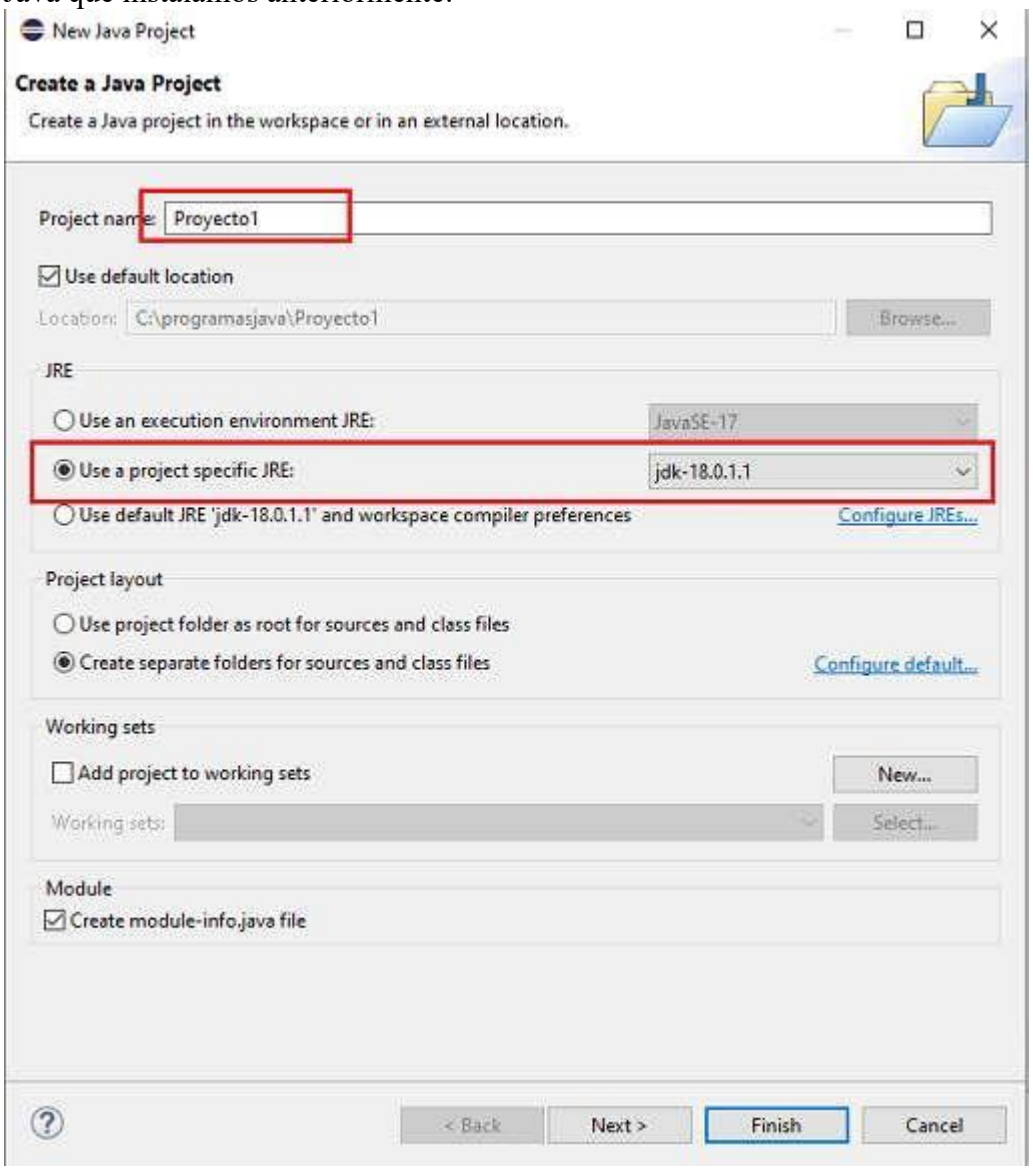


Pasos para crear un programa con Eclipse

El Eclipse es un entorno de trabajo profesional, por lo que en un principio puede parecer complejo el desarrollo de nuestros primeros programas.
Todo programa en Eclipse requiere la creación de un "Proyecto", para esto debemos seleccionar desde el menú de opciones:

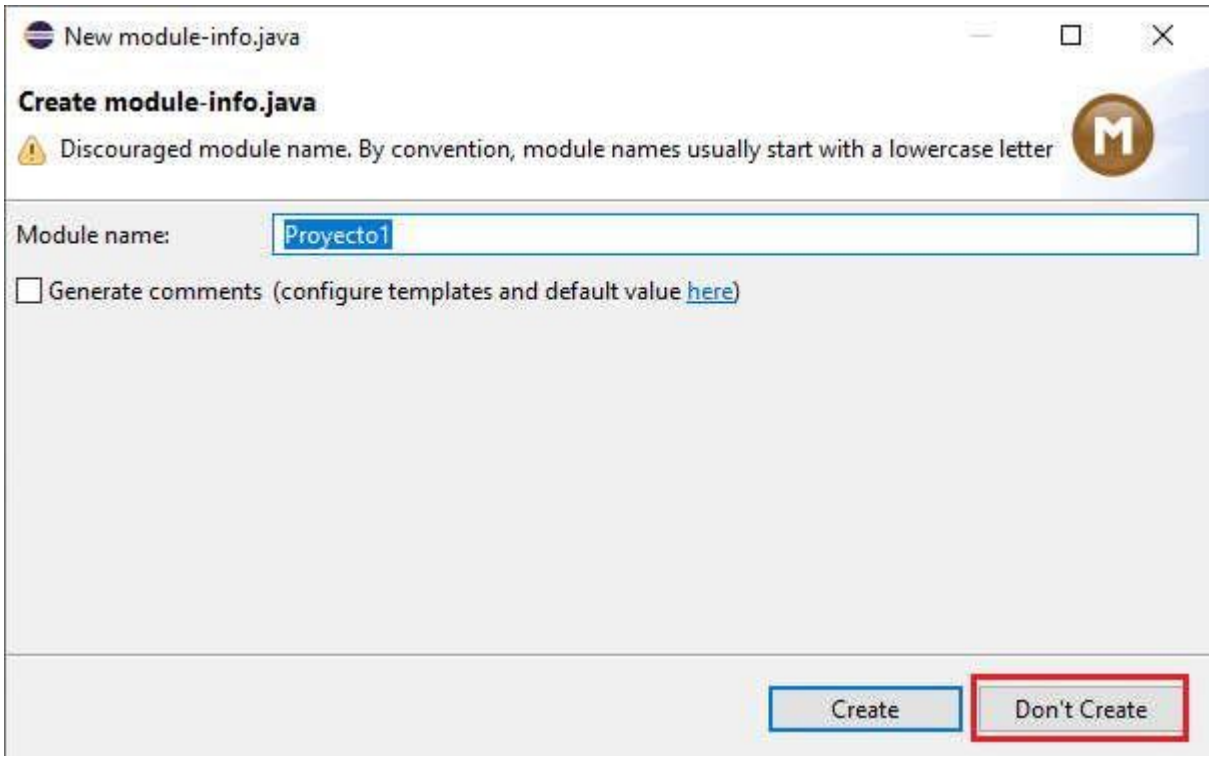


Ahora aparece el diálogo donde debemos definir el nombre de nuestro proyecto y la versión de Java que instalamos anteriormente:

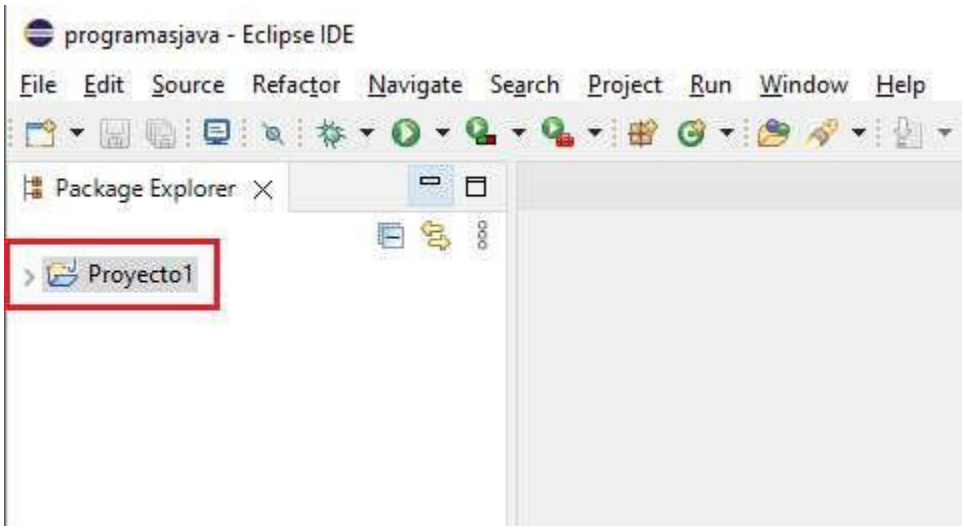


En el campo de texto "Project Name" ingresamos como nombre: Proyecto1 y dejamos todas las otras opciones del diálogo con los valores por defecto. Presionamos el botón "Finish".

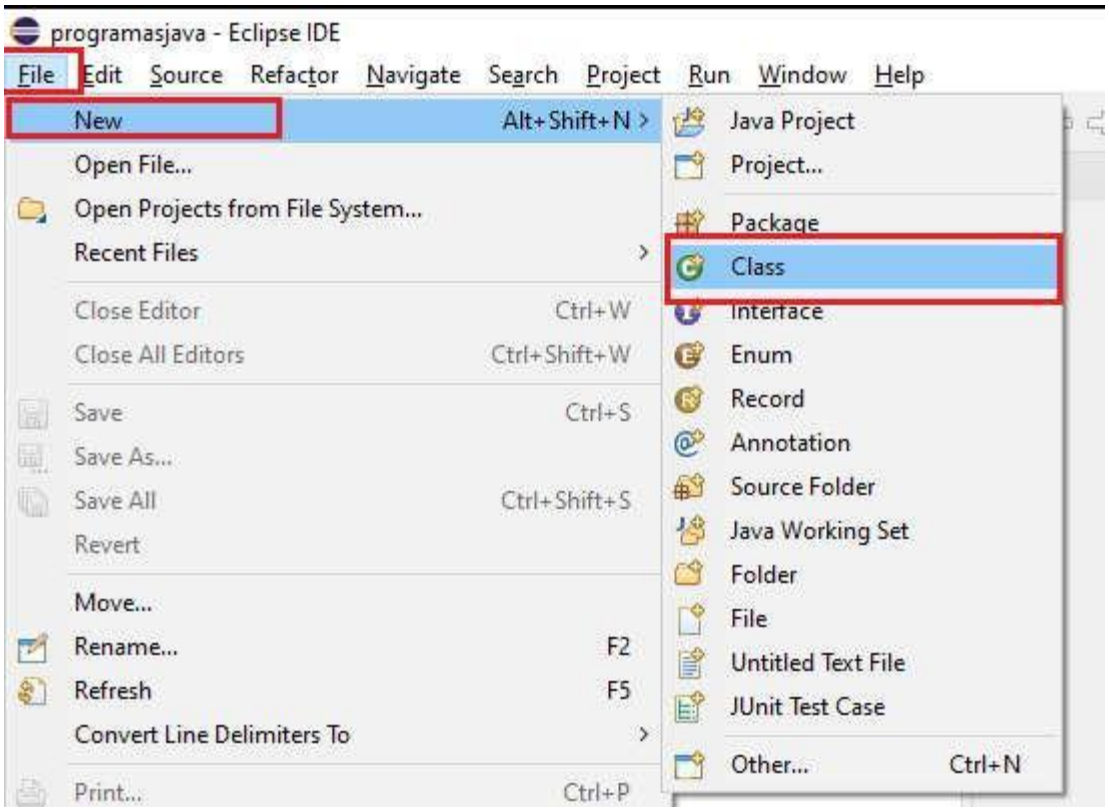
Aparece un nuevo diálogo que nos pide si queremos crear un módulo para nuestro proyecto, elegimos la opción para que no lo crea:



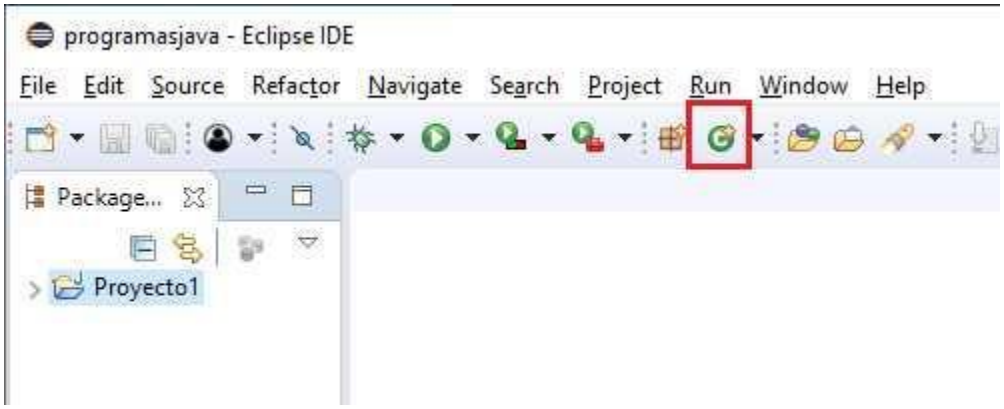
Ahora en la ventana de "Package Explorer" aparece el proyecto que acabamos de crear:



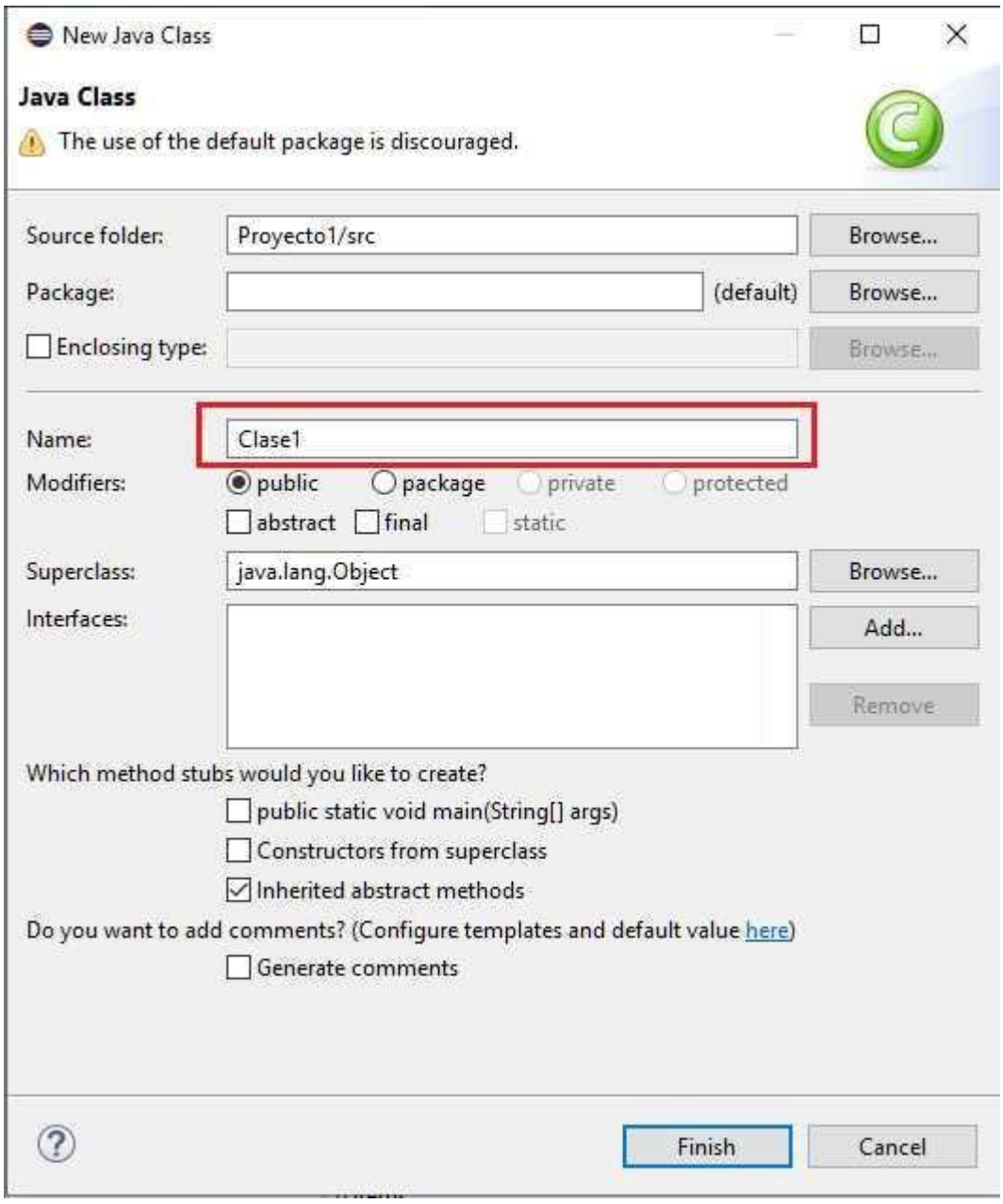
Como segundo paso veremos que todo programa en Java requiere como mínimo una clase. Para crear una clase debemos seleccionar desde el menú de opciones:



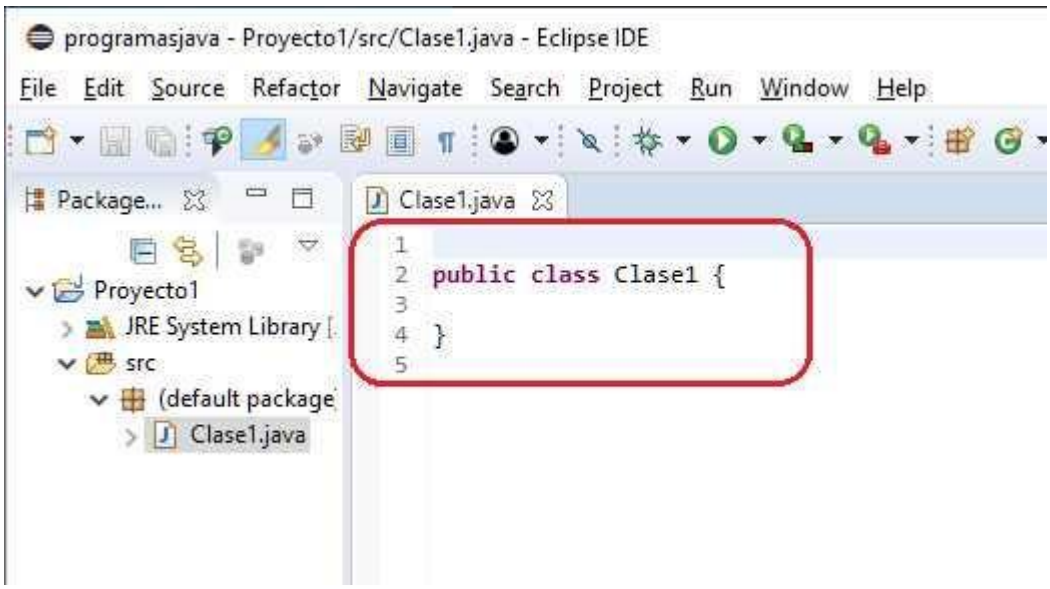
O desde la barra de íconos del Eclipse:



En el diálogo que aparece debemos definir el nombre de la clase (en nuestro primer ejemplo la llamaremos Clase1 (con mayúscula la letra C), luego veremos que es importante definir un nombre que represente al objetivo de la misma), los otros datos del diálogo los dejamos con los valores por defecto:



Luego de presionar el botón "Finish" tenemos el archivo donde podemos codificar nuestro primer programa:



Más adelante veremos los archivos que se crean en un proyecto, ahora nos dedicaremos a codificar nuestro primer programa. En la ventana de edición ya tenemos el esqueleto de una clase de Java que el entorno Eclipse nos creó automáticamente.

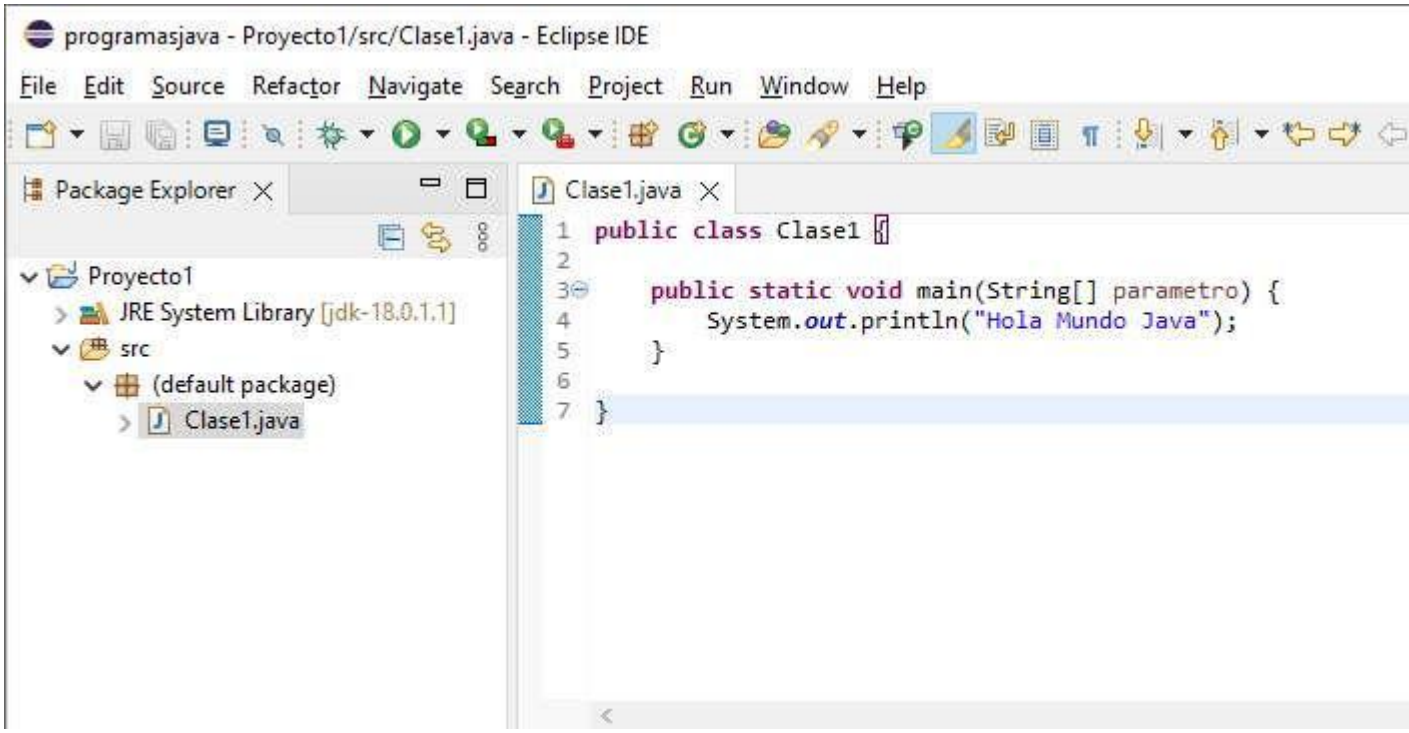
```
public class Clase1 {

}
```

Todo programa en Java debe definir la función main. Esta función la debemos codificar dentro de la clase: "Clase1".

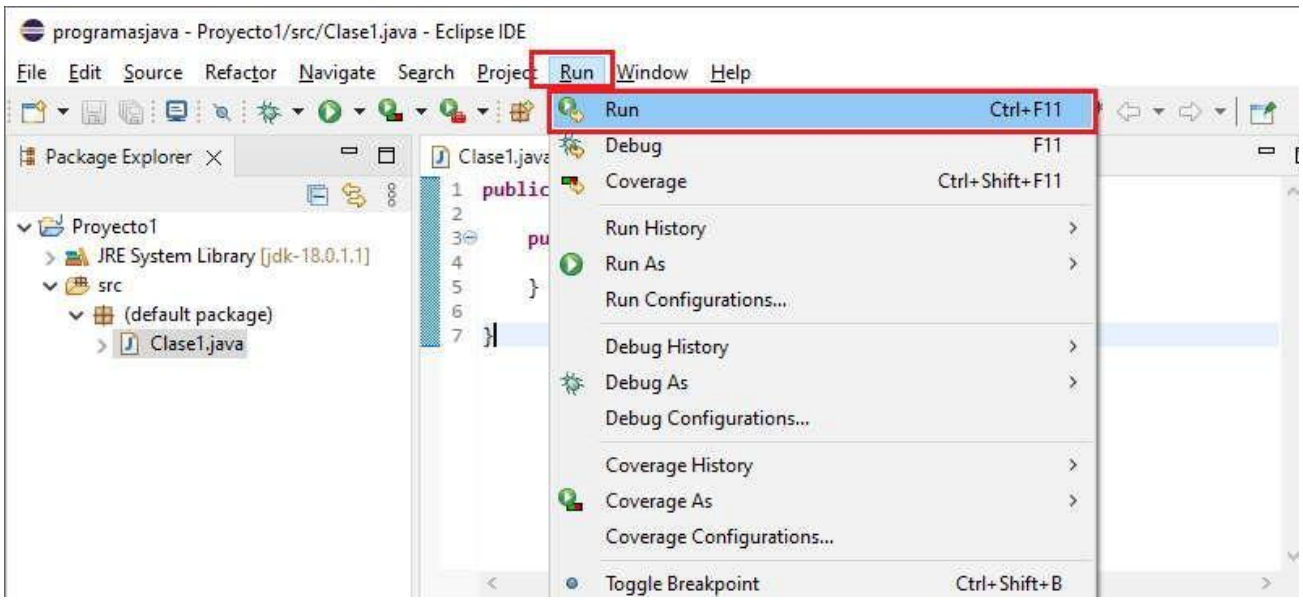
Procedemos a tipear lo siguiente:

```
public class Clase1 {

    public static void main(String[] parametro) {
        System.out.println("Hola Mundo Java");
    }

}
```

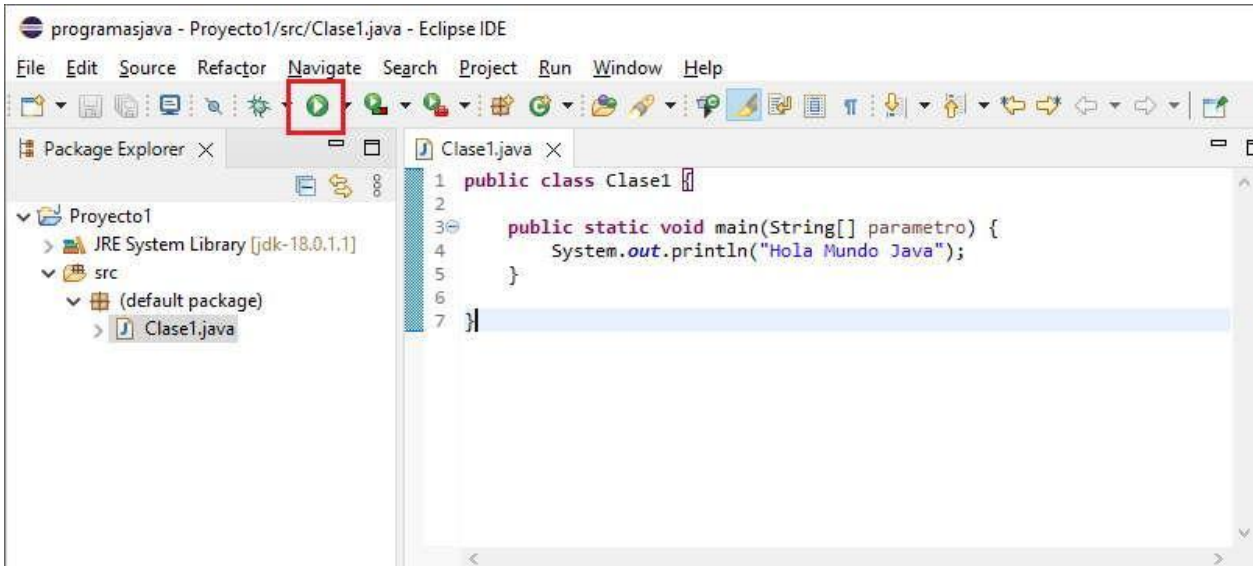
Es decir tenemos codificado en el entorno del Eclipse nuestro primer programa:



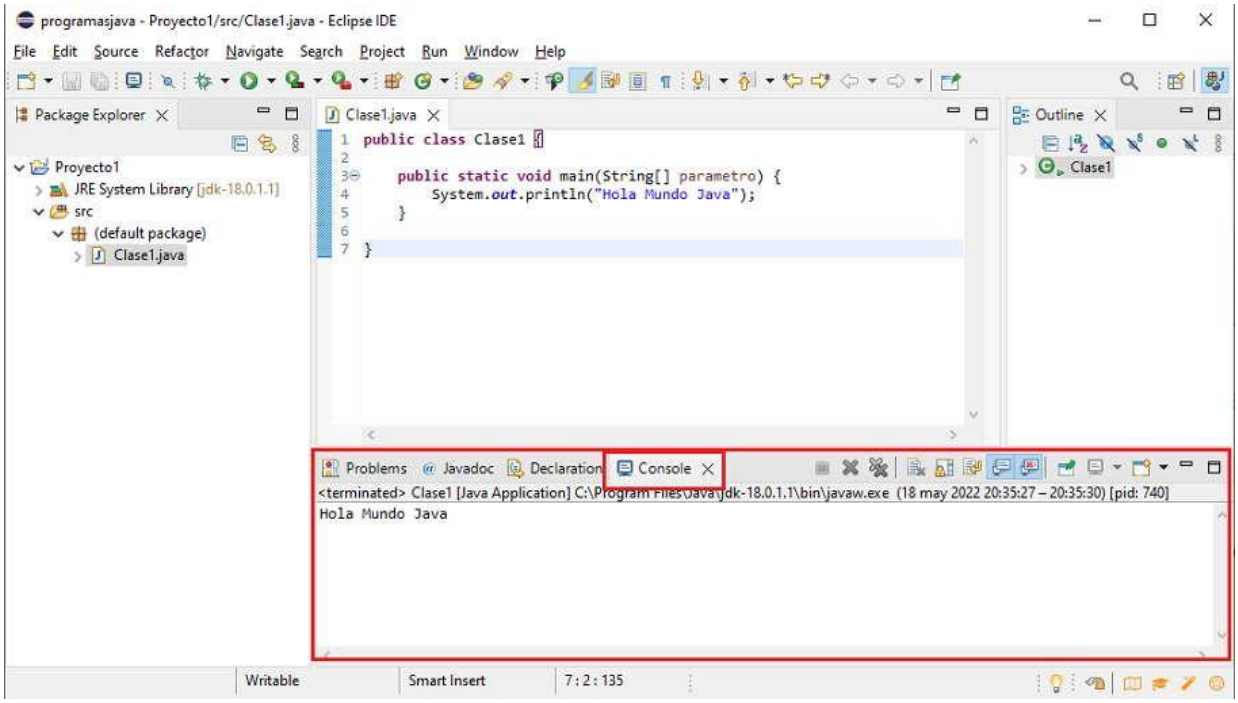
Como último paso debemos compilar y ejecutar el programa, esto lo podemos hacer desde el menú de opciones:



O desde la barra de íconos del Eclipse:



Si no hay errores de codificación debemos ver el resultado de la ejecución en una ventana del Eclipse llamada "Console" que aparece en la parte inferior (puede aparecer un diálogo pidiendo que grabemos el archivo, el cual confirmamos):



Lo más importante es que quede claro los pasos que debemos dar para crear un proyecto en Eclipse. El objetivo de una clase, la función main etc. los veremos a lo largo de este curso.

Recordemos en Python tenemos la función “print” para hacer salidas por la consola, en Java debemos utilizar la sintaxis:

System.out.println(“Hola Mundo Java”);

En Java es obligatorio el punto y coma al final de una instrucción y también es sensible a mayúsculas y minúsculas como Python.

Java es un lenguaje que utiliza la Programación Orientado a Objetos (POO), a lo largo de este curso aprenderemos esta nueva metodología de programación.

Cuando trabajamos con POO debemos disponer obligatoriamente una clase mediante las palabras claves (el nombre de nuestra clase es Clase1):

public class Clase1 {

}

public: Los miembros (clases, interfaces, métodos y variables) marcados como public son accesibles desde cualquier otra clase o paquete.

Luego en Java todo programa empeza en la función main que obligatoriamente debe estar contenida en una clase:

```
public static void main(String[] parametro) {

    System.out.println("Hola Mundo Java");
}
```

La main es como una función de Python con un parámetro. Luego veremos como se crean las funciones en Java (en realidad en Java le llamaremos métodos en lugar de funciones)

Es necesario en Java encerrar entre llaves todos los bloques a diferencia de Python que requiere la indentación para representar los bloques.

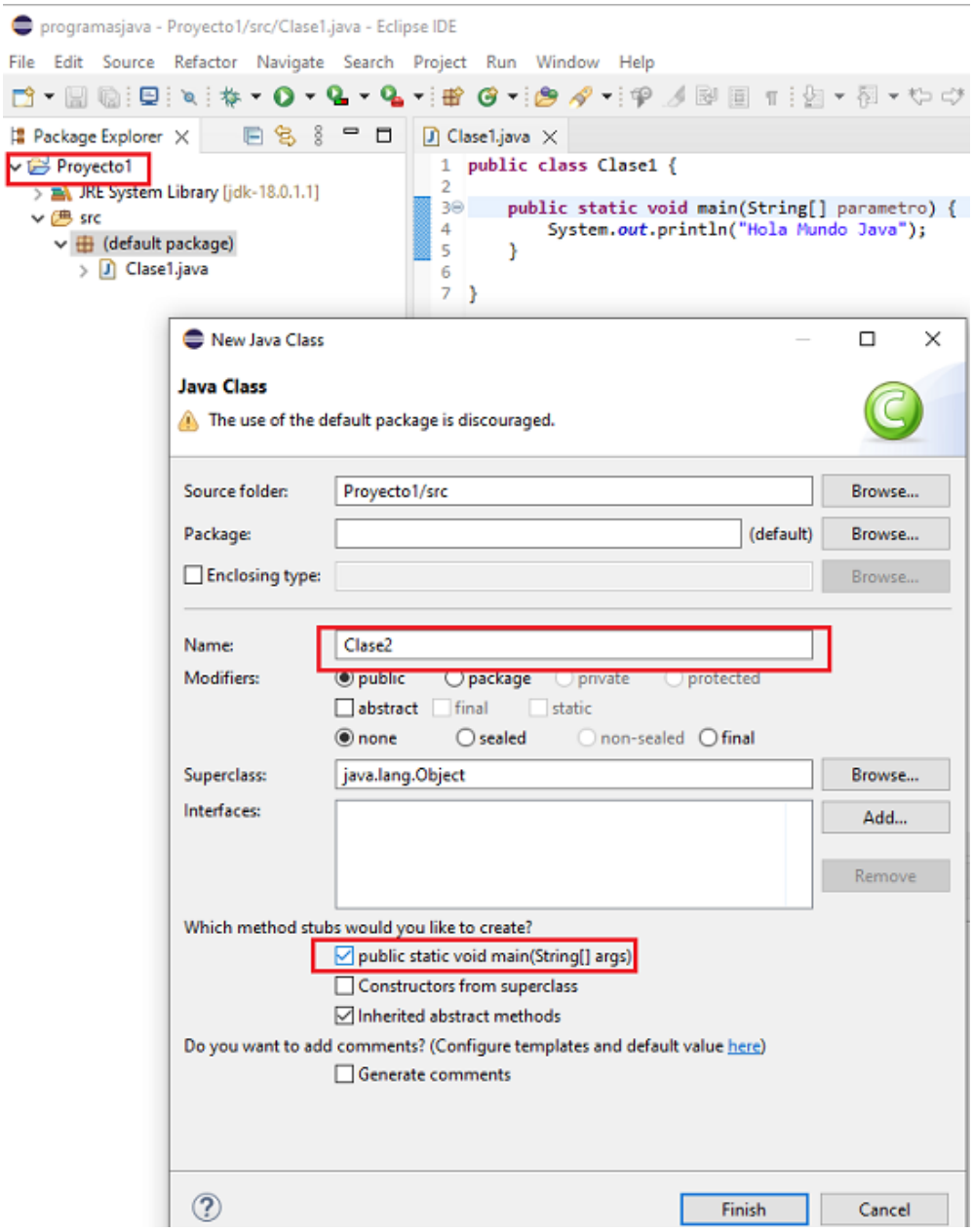
La indentación en Java no es obligatoria como Python, pero es una buena costumbre para que nuestro código sea más legible.

Tipos de variables

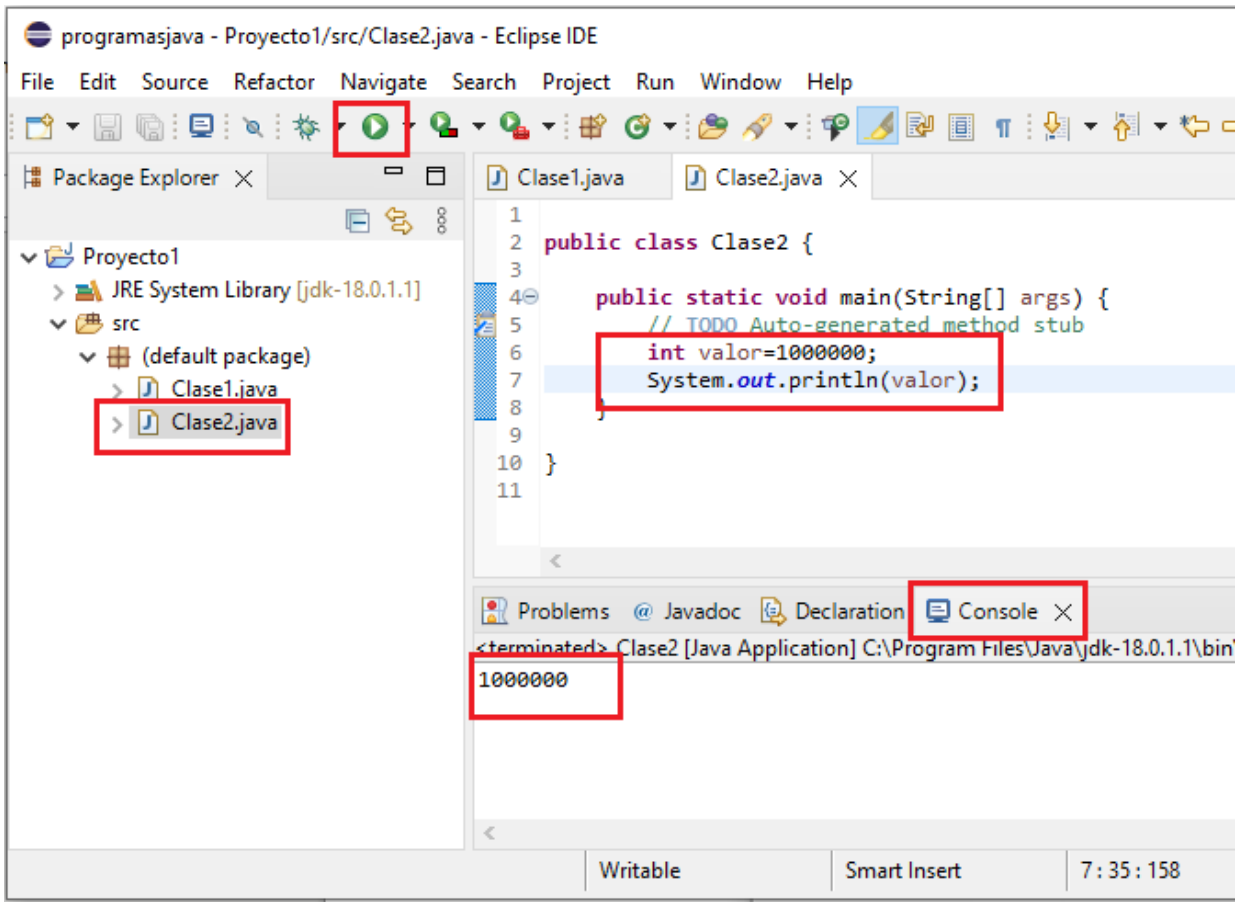
Recordemos que en Python una variable se define cuando le asignamos un valor. Java es un lenguaje fuertemente tipado y requiere que definamos el tipo de variable previo a utilizarla.

Java dispone un gran conjunto de variables primitivas para tipo de datos enteros y flotantes.

Veamos la sintaxis para definir una variable entera en Java, para no tener que crear tantos proyectos con pequeños programas vamos a crear en el mismo proyecto una segunda clase como vimos anteriormente y vamos a pedir que genere automaticamente la main:



Ahora definamos una variable entera que almacene el valor 1 millon y luego lo mostremos por la consola:



Los comentarios de una línea en Java son las dos barras:

```
// Esto es un comentario de línea.
```

Y los comentarios de bloque:

```
/*
            Esto es un comentario de
            bloque.
*/
```

Entonces para definir una variable entera en Java disponemos la plabra clave int seguida del nombre de la variable, podemos inmediatamente inicializar dicha variable con el operador de asignación.

Toda instrucción en Java finaliza con un punto y coma. Además como Python es Java un lenguaje sensible a mayúsculas y minúsculas.

En Java según el tamaño de valor entero a almacenar podemos definir variables enteras de tipo:

byte [-128, 127]

short [-32.768, 32.767]

int [-2147483648, 2147483647]

long [-9223372036854775808, 9223372036854775807]

Luego según el rango de valor a almacenar en la variable entera elegiremos el que sea más chico, esto luego repercute en la velocidas y requerimientos de memoria de la aplicación.

Para almacenar valores con parte decimal disponemos dos tipos de variables:

float: conocido como tipo de precisión simple, emplea un total de 32 bits. Con este tipo de datos es posible representar números en el rango de 1.4x10-45 a 3.4028235x1038.

double: sigue un esquema de almacenamiento similar al anterior, pero usando 64 bits en lugar de 32. Esto le permite representar valores en el rango de 4.9x10-324 a 1.7976931348623157x10308.

Otro tipo de dato primitivo en Java son los valores lógicos (booleanos)

boolean: Los dos valores posibles de este tipo son true y false (a diferencia de Python deben ir en minúsculas dichos valores)

char: Podemos almacenar un único carácter.

Modifiquemos la Clase2 y definamos una variable de cada tipo primitivo visto anteriormente:

```
public class Clase2 {

	public static void main(String[] args) {
		byte edad=12;
		short largo=10000;
		int    ancho=2000000000;
		long distancia=700000000000000000L;
		float peso1=50.25F;
		double peso2=67.45;
		boolean existe=true;
		System.out.println("Contenido de la variable byte:"+edad);
		System.out.println("Contenido de la variable short:"+largo);
		System.out.println("Contenido de la variable int:"+ancho);
		System.out.println("Contenido de la variable long:"+distancia);
		System.out.println("Contenido de la variable float:"+peso1);
		System.out.println("Contenido de la variable double:"+peso2);
		System.out.println("Contenido de la variable boolean:"+existe);
	}

}
```

Para cargar una variable float con cierto valor debemos finalizar con la letra L, lo mismo para una variable float agregamos la letra F.

El método println muestra por pantalla un mensaje o el contenido de una variable. Podemos utilizar el operador + para concatenar el mensaje y una variable de cualquier tipo (luego se transforma en una cadena en forma automática las variables enteras, float y boolean)

La declaración de una clase el obligatoria en Java y se hace con la palabra 'class', la palabra clave 'public' hace pública la clase para que pueda ser accedida por otras clases que se encuentran en otros paquetes (un paquete agrupa un conjunto de clases que se refieren a un mismo tema, luego veremos como se crean los paquetes)

Otro tipo de variable de uso muy común en Java son las cadenas de caracteres (String), su empleo veremos que difiere bastante con Python ya que no se pueden utilizar con los operadores relacionales (==, >,>=,<,<=,!=)

Para definir e inicializar un String en Java utilizamos el operador de asignación y la cadena debe ir obligatoriamente entre comillas dobles (no se pueden utilizar las comillas simples como Python):

```
String nombre="Rodriguez Pablo";
System.out.println("Su nombre es "+nombre);
```

Estructuras condicionales (if/else)

De forma similar a Python en Java disponemos de la estructura condicional if. Creemos una nueva clave y definamos en el método main un algoritmo con la sintaxis de la instrucción if/else

Problema:

Generar dos valores aleatorios comprendidos entre 0 y 9. Mostrar el mayor de ellos o un mensaje que indique cual de los mismos es el mayor.

```
public class Clase3 {


	public static void main(String[] args) {
		int valor1=(int)(Math.random()*10);
		int valor2=(int)(Math.random()*10);
		System.out.println("El primer valor aleatorio generado es "+valor1);
		System.out.println("El segundo valor aleatorio generado es "+valor2);
		if (valor1==valor2) {
			System.out.println("Los valores generados son iguales");
		} else {
			if (valor1>valor2) {
				System.out.println("El mayor es "+valor1);
```

```
                } else {
                        System.out.println("El mayor es "+valor2);
                }
        }
    }
}
```

En Java el primer método que se ejecuta es el main, en este caso mediante el método random() de la clase Math procedemos a generar un valor aleatorio comprendido entre 0 y 1 (el método random retorna un valor de tipo double, por ejemplo 0.0001 , 0.9999 , 0.34233 etc. nunca llaga a uno y nunca a cero) luego multiplicamos dicho valor por 10 y mediante el operador cast (int) transformamos el valor double a tipo int y procede a almacenarse en la variable valor1:

```
        int valor1=(int)(Math.random()*10);
        int valor2=(int)(Math.random()*10);
```

Como vemos es un poco más engorroso generar un valor aleatorio comprendido entre 0 y 9 que en Python.

Ahora procedemos a verificar con una serie de if anidados cual de los dos variables tiene un valor igual, como se pueden generar dos valores aleatorios iguales el primer if verifica dicha situación:

```
    if (valor1==valor2) {
                System.out.println("Los valores generados son iguales");
        }
```

Las condiciones en Java son obligatorias que se dispongan entre paréntesis y para indicar el bloque del verdadero en Java lo encerramos entre llaves (recordemos que en Python el intérprete reconoce cual es el bloque gracias a la indentación. Java no es obligatorio la indentación del código, pero disponemos las llaves para hacer notar al compilador donde comienza el bloque del verdadero del if)

En el caso de tratarse de una estructura condicional compuesta disponemos como en este ejemplo el bloque del else:

```
     else {
                if (valor1>valor2) {
                        System.out.println("El mayor es "+valor1);
                } else {
                        System.out.println("El mayor es "+valor2);
                }
        }
```

Es muy importante hacer notar que las llaves son la sintaxis en Java para indicar los bloques tanto del verdadero como el falso de un if.

Los operadores relacionales son los mísmos que en Python:

Operadores Relacionales:

```
== Igualdad
!= Desigualdad
```

```
< menor
<= menor o igual
> mayor
```
>= mayor o igual

Operadores lógicos:

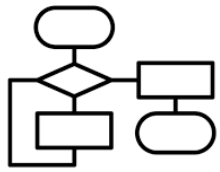
```
Python: and  Java: &&
```

Python: or Java: ||

Python: not Java: !

Operadores matemáticos:

```
+
-
*
/
%  (resto de una división)
```



Problemas propuestos

1 - Generar un valor aleatorio entre 0 y 1000. Mostrar la cantidad de dígitos que tiene dicho número.

2 – Generar 3 valores aleatorios por teclado, calcular su promedio. Mostrar un mensaje si dicho promedio es mayor o igual a 5 o menor.

3 – Generar 3 valores aleatorios comprendidos entre 0 y 10. Mostrar el menor de los mismos.

Estructuras condicionales (switch)

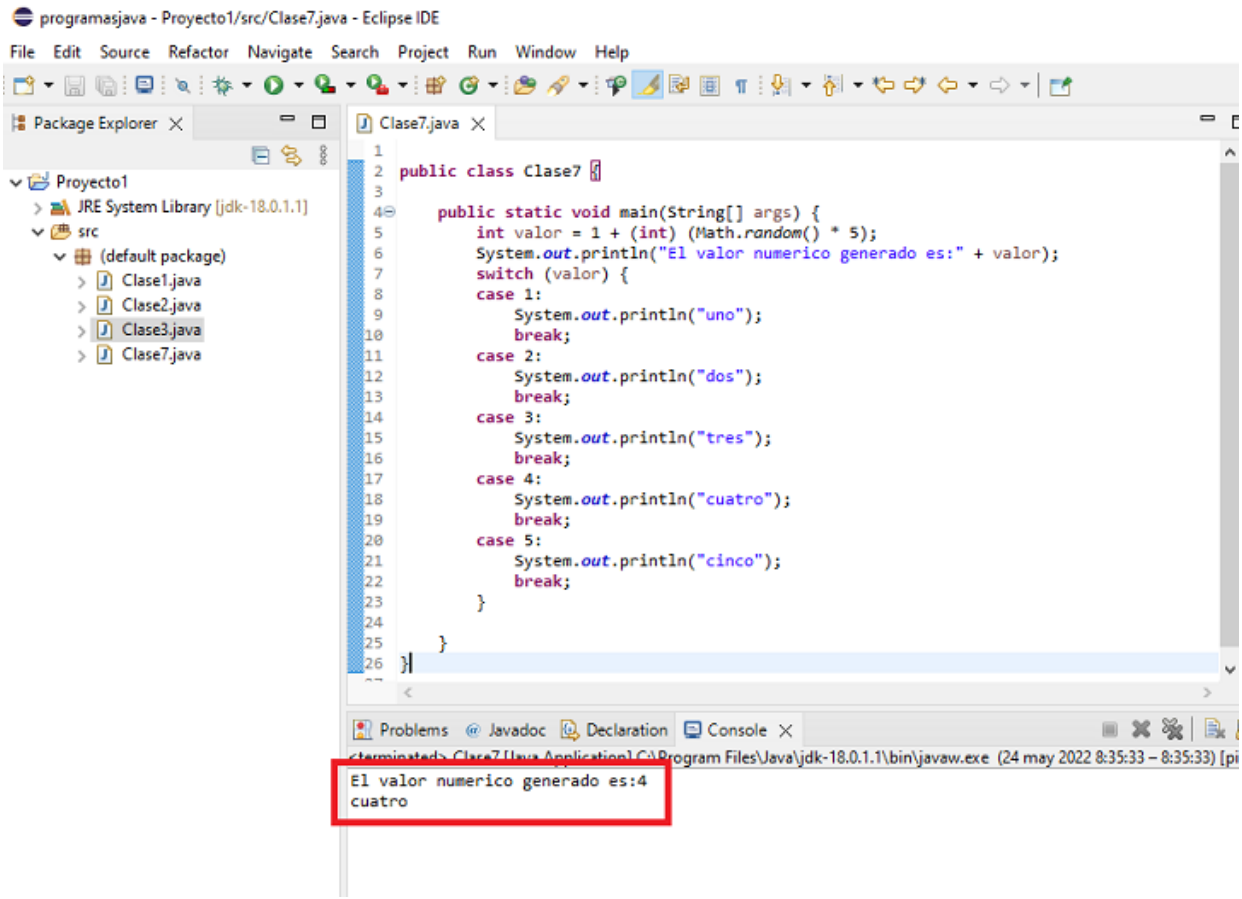
Otra estructura condicional existente en Java y que no existe en Python es la estructura switch.

La estructura switch remplaza en algunas situaciones cuando tenemos una serie de estructuras if anidadas.

Problema:

Generar un valor aleatorio comprendido entre 1 y 5. Mostrar un mensaje por pantalla de dicho nuúmero en castellano.

Si bien podemos resolver este problema utilizando una serie de if anidados, veamos la sintaxis para mostrar dicho valor con la instrucción switch:



El código fuente es:

```
public class Clase7 {

	public static void main(String[] args) {
```

```
		int valor = 1 + (int) (Math.random() * 5);
		System.out.println("El valor numerico generado es:" + valor);
		switch (valor) {
		case 1:
			System.out.println("uno");
			break;
		case 2:
			System.out.println("dos");
			break;
		case 3:
			System.out.println("tres");
			break;
		case 4:
			System.out.println("cuatro");
			break;
		case 5:
			System.out.println("cinco");
			break;
		}

	}
}
```

La instrucción switch analiza si una variable entera toma una serie de valores indicados en casa uno de los 'case', por ejemplo si la variable 'valor' almacena el 4 luego se ejecuta el case:

```
		case 4:
			System.out.println("cuatro");
			break;
```

Es importante la palabra clave break luego de las instrucciones que se ejecutarán si la variable almacena un 4.

Hay una parte opciona en la estructura switch similar a un else de un if. Veamos un ejemplo.

Problema:

Generar un valor aleatorio entre 1 y 10. Si sale un 5 mostrar que "gano un auto", si sale un 7 mostrar que "gano una heladera", si sale un 9 mostrar que "gano un televisor", en cualquier otro caso mostrar un mensaje que no ganó.

```
public class Clase8 {


	public static void main(String[] args) {
		int valor = 1 + (int) (Math.random() * 10);
		System.out.println("El valor numerico generado es:" + valor);
		switch (valor) {
		case 5:
			System.out.println("Gano un auto");
			break;
		case 7:
			System.out.println("Gano una heladera");
```

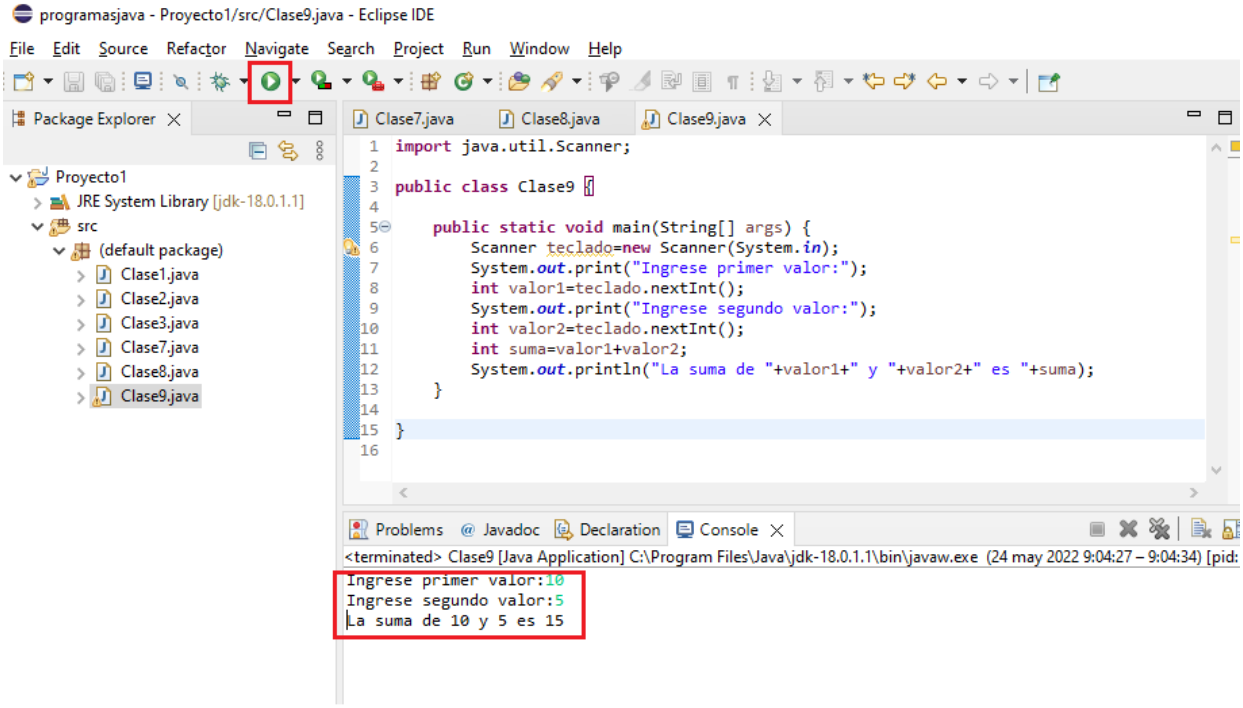
```
                break;
            case 9:
                System.out.println("Gano un televisor");
                break;
            default:
                System.out.println("No gano");
            }

        }

}
```

El bloque del ‘default’ se ejecuta cuando no se verifico verdadero ninguno de los case.

Entrada de datos por teclado

Problema:

Cargar dos valores enteros por teclado, proceder a sumarlos y mostrar dicha suma por pantalla.




```
import java.util.Scanner;


public class Clase9 {

	public static void main(String[] args) {
		Scanner teclado=new Scanner(System.in);
	  System.out.print("Ingrese primer valor:");
	  int valor1=teclado.nextInt();
	  System.out.print("Ingrese segundo valor:");
	  int valor2=teclado.nextInt();
	  int suma=valor1+valor2;
	  System.out.println("La suma de "+valor1+" y "+valor2+" es "+suma);
	}

}
```

Para hacer la entrada de datos por teclado en Java se complica. Utilizaremos una clase llamada Scanner que nos facilita el ingreso de datos. Por eso debemos importar la clase Scanner que se encuentra en el paquete java.util en la primer línea de nuestro programa.

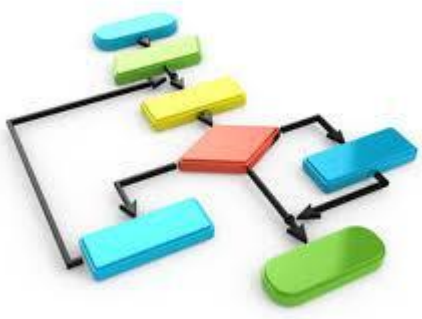
En la función main debemos crear un objeto de la clase Scanner con la siguiente sintaxis:

```
Scanner teclado=new Scanner(System.in);
```

Luego para cargar valores enteros por teclado debemos implementar la siguiente sintaxis:

```
int valor1=teclado.nextInt();
```

Si se necesita cargar un float se debe llamar al método 'nextFloat()'.



Problemas propuestos

1 - Calcular el sueldo mensual de un operario conociendo la cantidad de horas trabajadas y el pago por hora.

2 - Realizar la carga del lado de un cuadrado, mostrar por pantalla el perímetro del mismo (El perímetro de un cuadrado se calcula multiplicando el valor del lado por cuatro)

3 - Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente. Mostrar lo que debe abonar el comprador.

4 - Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".

5 - Se ingresa por teclado un valor entero, mostrar una leyenda que indique si el número es positivo, nulo o negativo.

JDK (Java Development Kit)

Al comienzo de esta clase procedimos a instalar el JDK.

JDK (Java Development Kit) contiene tanto la JVM (Java Virtual Machine) como el JRE (Java Runtime Environment), además de otras herramientas y utilidades para el desarrollo de aplicaciones Java.

JVM (Java Virtual Machine)

¿Qué es la JVM ?

La JVM es un componente fundamental en el ecosistema de Java. Es una máquina virtual que proporciona un entorno de ejecución aislado para programas escritos en Java. Cuando escribes un programa en Java y lo compilas, no se traduce directamente a código máquina que el hardware pueda entender. En cambio, se compila en un formato llamado bytecode, que es un código de nivel intermedio específico de Java.

La JVM es responsable de tomar este bytecode y ejecutarlo en cualquier sistema operativo o arquitectura de hardware compatible. Es decir, actúa como una capa de abstracción entre el código Java y el sistema subyacente. Esto significa que un programa Java puede ser ejecutado en diferentes sistemas operativos (Windows, macOS, Linux, etc.) sin necesidad de modificar el código fuente.

La JVM realiza varias tareas críticas durante la ejecución de un programa Java, como la gestión de memoria, la gestión de hilos, la administración de recursos y la ejecución del bytecode.

JRE (Java Runtime Environment)

El JRE es un conjunto de herramientas y bibliotecas que proporcionan todo lo necesario para ejecutar programas Java. Incluye la JVM, así como bibliotecas Java estándar y otros archivos necesarios para ejecutar aplicaciones Java. En esencia, el JRE es un entorno de tiempo de ejecución completo para Java.

Cuando instalas el JRE en tu sistema, estás instalando todo lo necesario para ejecutar aplicaciones Java, pero no estás configurando el entorno para desarrollar programas Java. Por lo tanto, si solo deseas ejecutar aplicaciones Java en tu sistema, instalar el JRE es suficiente.

Es decir que en la computadora de nuestro cliente solo debe tener instalado el JRE para que se puedan ejecutar nuestros programas. No es necesario instalar el JDK.

Entornos de desarrollo para programar en Java.

Si bien hemos instalado y vamos a trabajar con el entorno de programación Eclipse, hay otras plataformas populares para programar en Java.

Eclipse: Eclipse es un entorno de desarrollo integrado (IDE) muy popular para Java. Es altamente personalizable y cuenta con una amplia gama de complementos y extensiones para facilitar el desarrollo en Java.

IntelliJ IDEA: Desarrollado por JetBrains, IntelliJ IDEA es otro IDE muy popular entre los desarrolladores de Java. Ofrece una amplia gama de características avanzadas, como refactorización inteligente, análisis de código y soporte para varios marcos y tecnologías.

https://www.jetbrains.com/es-es/idea/download/?section=windows

NetBeans: NetBeans es un IDE de código abierto y gratuito que también es ampliamente utilizado para el desarrollo en Java. Ofrece soporte completo para el desarrollo de aplicaciones Java, así como soporte para otras tecnologías como JavaFX y HTML5.

https://netbeans.apache.org/front/main/download/index.html

Visual Studio Code (con extensiones para Java): Aunque inicialmente se desarrolló como un editor de código ligero, Visual Studio Code se ha convertido en una opción popular para el desarrollo en Java con la ayuda de extensiones como "Java Extension Pack" proporcionado por Microsoft.

https://code.visualstudio.com/

BlueJ: BlueJ es un entorno de desarrollo integrado (IDE) diseñado específicamente para principiantes en la programación Java. Es ligero y tiene una interfaz simple y amigable.

https://www.bluej.org/