Vamos a realizar una serie de pasos para comenzar con los fundamentos de Java

1. Cuenta en Github

<https://github.com>

1. Mi cuenta (Paco) se llama **serraguti**
2. Necesitamos instalar Git dentro de nuestro equipo

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Debemos indicar nuestro usuario de GIT y nuestro CORREO

Entramos en Terminal (CMD) y escribimos estos comandos

**git config --global user.name USERGIT**

**git config --global user.email** [**MAILGIT@GMAIL.COM**](mailto:MAILGIT@GMAIL.COM)



El siguiente paso es utilizar el entorno de trabajo para Java, que será Visual Studio Code

Descargamos Visual Studio Code

Visual Studio Code trabaja mediante múltiples lenguajes.

Mediante extensiones, podemos trabajar con cualquier lenguaje dentro VS Code

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Una vez que tenemos configurado Java, simplemente vamos a sincronizar nuestro proyecto individual con Github

Necesitamos instalar otra extensión para trabajar con Github dentro de Vs Code

Esta extensión se validará con nuestra cuenta de Github y podrá sincronizar nuestros proyectos.

Texto

El contenido generado por IA puede ser incorrecto.

Automáticamente nos pedirá validarnos con nuestra cuenta de Git.

Una ventaja radica en que se sincronizarán nuestras extensiones (si queremos) con el resto de equipos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Dentro de Git tenemos dos opciones:

1. **Push**: Nos permite subir nuestros ficheros a Github.com
2. **Pull**:Nos permite descargar nuestros ficheros desde Github.com

Dentro de este proceso, debemos sincronizar los códigos para no tener problemas.

Vamos a subir nuestro nuevo código a un Repositorio llamado **borrarahora**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Una vez publicado, cada vez que realicemos una modificación, siempre tendremos que hacer dos pasos:

1. Escribir un **Commit**. Un commit es una descripción de un estado de una actualización de un Branch/rama dentro de Git
2. Debemos pulsar en **Commit and Push**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Esto es respecto a realizar un Push y trabajar con un equipo (Aula)

El siguiente paso es realizar un Pull (Casa)

En casa no tenemos la carpeta **Prueba**, cerramos la carpeta en VS Code, cerramos VS Code y borramos la carpeta Prueba

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Trucos:

1. Si queremos crear otro repositorio con nuestra carpeta asociada de Github. Razones: Se ha roto, me he liado…

Ahora mismo tenemos asociado **borrarahora**, pero vamos a poner los cambios en otro repositorio distinto y nuevo llamado **borrarluego**

Necesitamos desligar nuestra carpeta de Git. Para ello, debemos eliminar la carpeta oculta **.git** del directorio del proyecto

Command + SHIFT + .

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Cada vez que cambiemos algo entre CASA y el AULA, siempre debemos hacer primero **PULL**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Eliminar los repositorios que tenemos

También podríamos cambiar su estado de visibilidad de private a public o viceversa

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Vamos a trabajar en una carpeta llamada **fundamentosjavagetafe** y abrimos dicha carpeta en VS Code

Dentro de dicha carpeta, separaremos los conceptos mediante nuestras clases de ejemplo.

Utilizaremos una sintaxis progresiva para saber el elemento en el que estamos, por ejemplo: **Class01Descripcion.java**

Si vamos a visualizar tipos de datos primitivos: **Class02TiposPrimitivos.java**

Necesitamos un para de conceptos para empezar:

* El nombre de las clases en Java debe coincidir con el nombre del fichero.
* El nombre de las clases siempre irá en mayúscula cada palabra de la clase.
* Existen elementos llamados **snippets** dentro de VS Code que nos facilitan el trabajo, por ejemplo
  + **sout**: System.out.println
  + **psvm**: public static void main

Comenzaremos con tipos primitivos:

Son clases llamada **Wrapper**, es decir, clases que para ser construidas/instanciadas no es necesario poner el constructor de forma explicita.

Simplemente, con asignar un valor, el objeto es construido.

Una variable en código nos permite almacenar valores para utilizarlos dentro del código.

Dependiendo del valor que deseemos almacenar en la variable, tendremos que declarar dicha variable con un tipado.

Lo más importante no son los tipos, sino la conversión entre los tipos.

Por ejemplo, convertir un texto a un número

Convertir una fecha a un texto

Java tiene dos elementos:

1. Tipo primitivo (pequeño): Solo almacena datos
2. Clase primitiva (grande): Almacena datos y tiene más cositas

Dependiendo del tipo de dato, se tienen una serie de características para dicho tipo de dato.

Por ejemplo, quiero almacenar el número 14:

**int numero = 14;**

GRANDE:

Integer numero = 14;

Integer.MAXVALUE: Nos indica el máximo valor para almacenar

Cada tipo de dato tiene una precisión, no puedo superar dicha precisión

byte peke = 290;

Cuando hablamos de conversión de datos, para convertir entre tipos de dato, se utilizan los **GRANDES**

Los grandes sirven para convertir entre primitivos.

Para visualizar los diferentes tipos de conversión, creamos una clase llamada **Class01Primitivos.java**

Ejemplos de tipos primitivos/Wrapper:

* byte
* short
* int
* long
* double
* decimal
* char
* String
* boolean

**CONVERSION IMPLICITA**

La conversión es automática, no tenemos que hacer nada de nada

Se realiza cuando almacenamos un tipo mayor en un tipo menor (no su valor)

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**CONVERSION ENTRE TIPOS PRIMITIVOS COMPATIBLES**

Este tipo de conversión implica convertir un tipo en otro tipo.

Se utiliza lo que se llama **Casting** y permite convertir entre tipos.

(Tipo)valor;

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Tenemos una clase que es como un Primitivo, es decir, podemos crear el objeto sin necesidad de llamar a su constructor, simplemente igualando.

Si yo quiero almacenar textos:

**S**tring texto = “Hola mundo”;

**CONVERTIR CUALQUIER OBJETO A STRING**

Para convertir un objeto a string se utiliza el método **toString()** que tienen TODOS LOS OBJETOS DE Java

El método **toString()** vienen en la clase GRANDE del primitivo.

Texto

El contenido generado por IA puede ser incorrecto.

**CONVERTIR CUALQUIER STRING A PRIMITIVO**

Para convertir cualquier String a Primitivo se utiliza un método de la clase String llamado **valueOf()** que nos permite convertir elementos a String

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**CONVERTIR UN STRING A PRIMITIVO**

Para convertir un string a primitivo necesitamos la clase Grande del primitivo.

Todas las clases primitivas tienen un método llamado **parse()** que nos permite convertir cualquier tipo de dato String a su primitivo.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Dentro de nuestros códigos tenemos tres tipos de errores:

1. Errores de compilación: Son errores de código y que los podemos visualizar de forma más o menos sencilla. (Nos falta un punto y coma)
2. Errores en ejecución: El programa compila, pero se detiene en la ejecución porque tengo algún error. (Problemas al convertir)
3. Errores lógicos: Todo funciona, no nos da ningún error y el programa no hace lo que queremos.

Error de ejecución:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Tenemos objetos especializados como acabamos de visualizar, objetos para almacenar textos, números o decimales.

Cada tipo tiene una funcionalidad, por ejemplo, si declaro un tipo int, puedo sumar, restar o dividir.

Si declaro un tipo String, puedo contar el número de caracteres que contiene.

Tenemos una clase que nos permite “pedir” información al usuario mediante teclado.

Al estar en consola, se utiliza una clase llamada **Scanner** que nos permite pedir información por teclado.

Dentro de Java todo está en librerías. Las librerías están especializadas en sus componentes.

Por ejemplo, existen librerías para ficheros o librerías para acceso a datos o librerías para entornos Web.

Existe una librería llamada **java.util** que nos permite trabajar con herramientas del entorno de Consola

Para poder crear un Scanner necesitamos dos elementos:

1. Utilizar la librería java.util
   1. java.util.Scanner

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

* 1. import java.util.Scanner

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

1. Instanciar el objeto Scanner



Para pedir datos al usuario tenemos un método llamado **nextLine()** que nos permite recuperar lo que el usuario ha escrito.

Para concatenar (unir dos textos) se utiliza el símbolo **+** en Java

Texto

El contenido generado por IA puede ser incorrecto.

Vamos a realizar un ejemplo muy sencillo en el que pediremos dos números al usuario por teclado y mostraremos la suma de los dos números.

Creamos un nuevo programa llamado **Class03SumarNumeros.java**

Texto

El contenido generado por IA puede ser incorrecto.

**import java.util.Scanner;**

**public class Class03SumarNumeros {**

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

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

**System.out.println("Introduzca número 1");**

**//Almacenamos el primer numero**

**//El numero debemos almacenarlo como entero**

**//Voy a capturar el dato como String**

***String* dato = teclado.nextLine();**

**//Convertimos el dato a int**

***int* numero1 = Integer.parseInt(dato);**

**System.out.println("Introduzca número 2");**

**//Volvemos a utilizar la variable dato**

**dato = teclado.nextLine();**

**//Convertimos el dato a int**

***int* numero2 = Integer.parseInt(dato);**

**//Realizamos la suma**

***int* suma = numero1 + numero2;**

**//Dibujamos la suma**

**System.out.println("La suma es " + suma);**

**System.out.println("Fin de programa");**

**}**

**}**

**CONDICIONALES**

Los condicionales en cualquier lenguaje nos permiten que nuestro código sea dinámico y que pueda realizar diferentes acciones dependiendo de lo que vayamos preguntando en las condiciones.

Operadores de comparación

== Igual

> Mayor

>= Mayor o igual

< Menor

<= Menor o igual

!= Distinto

Tenemos varias posibilidades en el momento de preguntar, pero la sintaxis es la misma.

**Sintaxis simple de un Condicional (IF)**

if (condición == true)

{

//Solamente entra si es TRUE

}

**Sintaxis Si no en condición**

if (condición == true)

{

//Acciones para el true

}else {

//Acciones para el false

}

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**Sintaxis con múltiples preguntas**

Se utiliza cuando necesitamos preguntar por más elementos en una condición

if (condiciónA == true)

{

//Acciones para condición A

}else if (condicionB == true) {

//Acciones para condición B

}else {

//Ninguna condición se cumple

}

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Vamos a realizar un programa para preguntar si un número es positivo, negativo o cero.

Creamos un nuevo programa llamado **Class04PositivoNegativo.java**

**Texto

El contenido generado por IA puede ser incorrecto.**

**import java.util.Scanner;**

**public class Class04PositivoNegativo {**

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

**//PODEMOS COMBINAR CON SCANNER**

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

**System.out.println("Introduzca un número");**

***String* dato = teclado.nextLine();**

**//Convertimos dato a int**

***int* numero = Integer.parseInt(dato);**

**if (numero > 0){**

**System.out.println("El número es POSITIVO");**

**}else if (numero == 0) {**

**System.out.println("El número es CERO");**

**}else {**

**System.out.println("El número es NEGATIVO");**

**}**

**System.out.println("Fin de programa");**

**}**

**}**

Vamos a realizar un programa en el que pediremos un número del 1 al 4 al usuario.

Debemos indicar la estación dependiendo del número

1.- Primavera

2.- Verano

3.- Invierno

4.- Otoño

Otra opción, escribimos un mensaje diciendo Incorrecto

Llamamos al programa **Class05Estaciones.java**

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

**import java.util.Scanner;**

**public class Class05Estaciones {**

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

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

**System.out.println("Introduzca un número (1-4)");**

***String* dato = teclado.nextLine();**

***int* numero = Integer.parseInt(dato);**

**if (numero == 1){**

**System.out.println("Primavera");**

**}else if(numero == 2){**

**System.out.println("Verano");**

**}else if (numero == 3){**

**System.out.println("Invierno");**

**}else if(numero == 4){**

**System.out.println("Otoño");**

**}else {**

**System.out.println("Opción incorrecta");**

**}**

**}**

**}**

Operadores relacionales

Los operadores relacionales nos permiten preguntar por más de un elemento dentro de una misma pregunta

&& And: Todas las condiciones deben cumplirse

|| Or: Solamente una condición debe cumplirse

! Not: Negación de una condición. Mejor utilizar siempre un Operador

Quiero preguntar si un número es diferente a 1

if (**!** (numero == 1))

if (numero !=1)

Vamos a realizar un programa para calcular el mayor, el menor y el intermedio de tres números.

Creamos un nuevo programa llamado **Class06MayorTresNumeros.java**

Tabla

El contenido generado por IA puede ser incorrecto.

**import java.util.Scanner;**

**public class Class06MayorTresNumeros {**

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

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

**System.out.println("Introduzca número 1");**

***String* dato = teclado.nextLine();**

***int* num1 = Integer.parseInt(dato);**

**System.out.println("Introduzca número 2");**

**dato = teclado.nextLine();**

***int* num2 = Integer.parseInt(dato);**

**System.out.println("Introduzca número 3");**

**dato = teclado.nextLine();**

***int* num3 = Integer.parseInt(dato);**

***int* mayor = 0;**

**//PREGUNTAMOS POR LAS CONDICIONES**

**//22 , 88, 244**

**if (num1 > num2 && num1 > num3){**

**mayor = num1;**

**}else if (num2 > num1 && num2 > num3){**

**mayor = num2;**

**}else {**

**mayor = num3;**

**}**

**System.out.println("El número mayor es " + mayor);**

**}**

**}**

Solución Versión 2

**import java.util.Scanner;**

**public class Class06MayorTresNumeros {**

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

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

**System.out.println("Introduzca número 1");**

***String* dato = teclado.nextLine();**

***int* num1 = Integer.parseInt(dato);**

**System.out.println("Introduzca número 2");**

**dato = teclado.nextLine();**

***int* num2 = Integer.parseInt(dato);**

**System.out.println("Introduzca número 3");**

**dato = teclado.nextLine();**

***int* num3 = Integer.parseInt(dato);**

***int* mayor = 0;**

***int* menor = 0;**

***int* intermedio = 0;**

**//PREGUNTAMOS POR LAS CONDICIONES**

**//22 , 88, 244**

**if (num1 >= num2 && num1 >= num3){**

**mayor = num1;**

**}else if (num2 >= num1 && num2 >= num3){**

**mayor = num2;**

**}else {**

**mayor = num3;**

**}**

**if (num1 <= num2 && num1 <= num3){**

**menor = num1;**

**}else if (num2 <= num1 && num2 <= num3){**

**menor = num2;**

**}else{**

**menor = num3;**

**}**

**//sumamos todos los numeros**

***int* suma = num1 + num2 + num3;**

**intermedio = suma - mayor - menor;**

**System.out.println("El número menor es " + menor);**

**System.out.println("El número mayor es " + mayor);**

**System.out.println("El número intermedio es " + intermedio);**

**}**

**}**

**CALCULAR DIA DE NACIMIENTO DE LA SEMANA**

**Class07DiaNacimiento.java**

Pedir una fecha al usuario para calcular el día de la semana que nació. Tenemos que tener la tabla de días de la semana para la correspondencia comenzando en sábado:

|  |  |
| --- | --- |
| **DIA** | **NÚMERO** |
| **Sábado** | 0 |
| **Domingo** | 1 |
| **Lunes** | 2 |
| **Martes** | 3 |
| **Miércoles** | 4 |
| **Jueves** | 5 |
| **Viernes** | 6 |

Debemos pedir el **día**, el número de **mes** y el **año** que el usuario haya nacido.

A partir de esto datos hay que calcular lo siguiente para averiguar el día de la semana de nacimiento:

Ejemplo 🡪 15/06/1997

Hay que tener en cuenta el mes para realizar el cálculo, si el mes es Enero, el Mes será 13 y restaremos uno al año. Si el Mes es Febrero, el Mes será 14 y restaremos uno al año.

Para poder calcular las el número final de la semana debemos seguir los siguientes pasos:

1. Multiplicar el Mes más 1 por 3 y dividirlo entre 5

((6 + 1) \* 3) / 5 🡪 4

1. Dividir el año entre 4

1997 / 4 🡪 499

1. Dividir el año entre 100

1997 / 100 🡪 19

1. Dividir el año entre 400

1997 / 400 🡪 4

1. Sumar el dia, el doble del mes, el año, el resultado de la operación 1, el resultado de la operación 2, menos el resultado de la operación 3 más la operación 4 más 2.

15 + (6 \* 2) + 1997 + 4 + 499 - 19 + 4 + 2 🡪 2514

1. Dividir el resultado anterior entre 7.

2514 / 7 🡪 359

1. Restar el número del paso 5 con el número del paso 6 por 7.

2514 – (359 \* 7) 🡪 1

1. Miramos la tabla y vemos que el número 1 corresponde a **Domingo**

SOLUCION

**import java.util.Scanner;**

**public class Class07DiaNacimiento {**

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

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

**System.out.println("Introduzca día");**

***String* dato = teclado.nextLine();**

***int* dia = Integer.parseInt(dato);**

**System.out.println("Introduzca mes");**

**dato = teclado.nextLine();**

***int* mes = Integer.parseInt(dato);**

**System.out.println("Introduzca año");**

**dato = teclado.nextLine();**

***int* anyo = Integer.parseInt(dato);**

**if (mes == 1){**

**mes = 13;**

**anyo -= 1;**

**}else if (mes == 2){**

**mes = 14;**

**anyo--;**

**}**

**//1. Multiplicar el Mes más 1 por 3 y dividirlo entre 5**

***int* op1 = ((mes + 1) \* 3) / 5;**

***int* op2 = anyo / 4;**

***int* op3 = anyo / 100;**

***int* op4 = anyo / 400;**

**//Comentar en bloque. Seleccionamos lo que necesitemos**

**// Command + K + C**

**//Descomentar codigo: Command + K + U**

**// 5. Sumar el dia, el doble del mes, el año**

**// , el resultado de la operación 1, el resultado de la operación 2**

**// , menos el resultado de la operación 3 más la operación 4 más 2.**

***int* op5 = dia + (mes \* 2) + anyo + op1 + op2 - op3 + op4 + 2;**

**//6. Dividir el resultado anterior entre 7.**

***int* op6 = op5 / 7;**

**// 7. Restar el número del paso 5 con el número del paso 6 por 7.**

***int* resultado = op5 - (op6 \* 7);**

***String* diaSemana = "";**

**if (resultado == 0){**

**diaSemana = "SABADO";**

**}else if (resultado == 1){**

**diaSemana = "DOMINGO";**

**}else if (resultado == 2){**

**diaSemana = "LUNES";**

**}else if (resultado == 3){**

**diaSemana = "MARTES";**

**}else if (resultado == 4){**

**diaSemana = "MIERCOLES";**

**}else if (resultado == 5){**

**diaSemana = "JUEVES";**

**}else if (resultado == 6){**

**diaSemana = "VIERNES";**

**}else {**

**diaSemana = "Hemos pinchado...";**

**}**

**System.out.println("El día de la semana es " + diaSemana);**

**}**

**}**

Una vez que tenemos el programa, debemos aprender a Depurar/Debug del código.

La depuración es algo esencial a medida que vayamos creciendo en desarrollo.

Para depurar se utiliza la tecla F5 o Debug Java

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Nuestro código se detendrá dónde nosotros deseemos, es decir, debemos incluir Puntos de interrupción para “saber” analizar qué sucede en el código en un momento dado.

Los puntos de interrupción se incluyen a la izquierda de nuestro código (mas allá de los números)

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

En el momento que ejecutemos como **Debug**, se detendrá en el punto de interrupción

Tendremos dos opciones para ir paso a paso

* F10: Cuando tenemos solamente un código que analizar
* F11: Cuando tenemos llamadas en dicho código y deseamos ir a visualizar esas llamadas.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

CALCULAR SALARIO

**Class08CalcularSalario.java**

* Realizar una aplicación que obtenga el salario neto de los empleados de una empresa teniendo en cuenta el número de horas trabajadas, el importe a pagar por hora y el lugar al que ha sido destinado el trabajador (establecido en Km).
* Cada hora extra se pagará 2€ más que las normales, (la hora extra comienza a computarse a partir de la 36 hora semanal trabajada)
* Sobre el salario base anterior, si el destino del empleado está:
  + - Entre 101 Km y 900Km: La dieta será “NACIONAL”
    - Por encima de 900Km: Dieta “INTERNACIONAL”
    - Por debajo de 101 Km Dieta “PROVINCIAL”
* Sobre el precio final obtenido del salario:
  + - Cantidades menores o iguales a 250 € no soportan retención 0%
    - Cantidades por encima de 250 € y menores o iguales a 500 € la retención es del 20%
    - Cantidades por encima de 500 € la retención es del 50%
* Aplicaremos el IVA (16%) al salario bruto y se lo restaremos para conseguir el salario neto del trabajador.

Horas trabajadas: 40

Precio Hora: 20

Horas extras: 4

PrecioBase: 36 \* 20

PrecioExtra: 4 \* 22

* **Al final debemos mostrar el siguiente informe:**

INTRODUZCA HORAS SEMANALES

51

INTRODUZCA IMPORTE HORA:

24

INTRODUZCA KILOMETROS:

269

---------------------------------------------------

HORAS TRABAJADAS: 51

HORAS EXTRAS: 15

IMPORTE DE LA HORA: 24

DISTANCIA EN KM: 269

DESTINO: NACIONAL

RETENCION: 50%

SALARIO BASE: 864.00

SALARIO HORAS EXTRA: 540.00

SALARIO BRUTO: 1404.00

IVA (16%): 224.64

---------------------------------------------------

SALARIO TOTAL: 1179.36

---------------------------------------------------

FIN DE PROGRAMA

**SOLUCION**

**import java.util.Scanner;**

**public class Class08CalcularSalario {**

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

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

***int* horas = 0;**

***int* kilometros = 0;**

***int* horasExtra = 0;**

***int* importeHora = 0;**

***int* salario = 0;**

***int* salarioExtra = 0;**

***String* destino = "";**

***String* retencion = "";**

***float* iva = 0;**

**System.out.println("Introduzca horas semanales");**

***String* dato = teclado.nextLine();**

**horas = Integer.parseInt(dato);**

**System.out.println("Introduzca precio/hora");**

**dato = teclado.nextLine();**

**importeHora = Integer.parseInt(dato);**

**System.out.println("Kilometros recorridos");**

**dato = teclado.nextLine();**

**kilometros = Integer.parseInt(dato);**

**if (horas > 36){**

**//Tenemos horas extra**

**horasExtra = horas - 36;**

**salario = 36 \* importeHora;**

**salarioExtra = horasExtra \* (importeHora + 2);**

**salario = salario + salarioExtra;**

**}else {**

**salario = horas \* importeHora;**

**}**

**//comprobamos el destino segun km**

**if (kilometros > 101 && kilometros < 900){**

**destino = "NACIONAL";**

**}else if (kilometros > 900){**

**destino = "INTERNACIONAL";**

**}else {**

**destino = "PROVINCIAL";**

**}**

**//comprobamos el salario para la retencion**

**if (salario <= 250){**

**retencion = "0%";**

**}else if (salario > 500){**

**retencion = "50%";**

**}else {**

**retencion = "20%";**

**}**

**iva = (*float*)(salario \* 0.16);**

**System.out.println("Horas trabajadas: " + horas);**

**System.out.println("Horas extras: " + horasExtra);**

**System.out.println("Importe hora: " + importeHora);**

**System.out.println("Kilometros: " + kilometros);**

**System.out.println("Destino: " + destino);**

**System.out.println("Retención: " + retencion);**

**System.out.println("Salario base: " + salario);**

**System.out.println("Salario extra: " + salarioExtra);**

**System.out.println("Iva (16%): " + iva);**

**System.out.println("-------------------------");**

**System.out.println("Salario final: " + (salario - iva));**

**}**

**}**

**BUCLES EN JAVA**

Un bucle es una secuencia repetitiva en código.

Un bucle es un bucle, no importa el lenguaje.

Las secuencias pueden ser de dos tipos:

* Bucle Contadores: Son bucles que tienen un inicio y un final y están administrados. (for)

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

* Bucles condicionales: Son bucles que dependen de una condición, no sabemos cuando van a finalizar, dependerá de la condición. Este tipo de bucles son los más peligrosos porque podríamos tener un bucle infinito. (while)

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Creamos una clase llamada **Class09Bucles.java**

**import java.util.Scanner;**

**public class Class09Bucles {**

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

**//bucles contador (for)**

**//Debemos indicar un inicio, un final (while), un incremento**

**//La variable contador se declara dentro del bucle y**

**//solamente se puede utilizar dentro de dicho bucle**

**// for (int i = 1; i <= 5; i++){**

**// System.out.println("For: " + i);**

**// }**

**// //Podemos tener un incremento distinto**

**// //de 3 en 3**

**// for (int i = 1; i <= 10; i+=3){**

**// System.out.println("Tres en tres " + i);**

**// }**

**// //Podemos hacer que el bucle sea inverso**

**// for (int i = 7; i > 0; i--){**

**// System.out.println("Inverso " + i);**

**// }**

**// //Bucles de condición**

**// //Necesitan una condición para salir**

**// //bucles infinitos**

**// //Necesitamos una variable de salida del bucle**

**// int contador = 1;**

**// while (contador <= 5) {**

**// System.out.println("While: " + contador);**

**// //Debemos hacer "algo" para que la variable**

**// //del bucle cambie y llegue a la condición de salida**

**// contador++;**

**// }**

**//Necesito mostrar los 20 primeros números pares**

**// for (int i = 1; i <= 20; i++){**

**// //Preguntamos si un numero es par**

**// //Se utiliza el operador del resto: %**

**// if (i % 2 == 0){**

**// System.out.println("Par: " + i);**

**// }**

**// }**

**//Vamos a pedir al usuario textos hasta que escriba stop**

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

***String* dato = "";**

**//Java en sus String no compara con el operador == ni !=**

**//Estoy comparando objetos y con String debo comparar el valor**

**//del objeto**

**//se utiliza el método equals() de la clase String**

**// if (dato.equals("")){**

**// System.out.println("Texto vacio!!!");**

**// }else{**

**// System.out.println("Contenido");**

**// }**

**while (dato.equals("stop") == false){**

**System.out.println("Dime algo bonito (stop para salir)");**

**dato = teclado.nextLine();**

**}**

**}**

**}**

Creamos una nueva clase llamada **Class10TablaMultiplicar.java**

Pediremos un número al usuario y mostraremos la tabla de multiplicar de dicho número.

5\*1 = 5

5\*2 = 10

…

5\*10 = 50

**SOLUCION**

**import java.util.Scanner;**

**public class Class10TablaMultiplicar {**

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

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

**System.out.println("Introduzca un número");**

***String* dato = teclado.nextLine();**

***int* numero = Integer.parseInt(dato);**

**for (*int* i = 1; i <= 10; i++){**

***int* operacion = numero \* i;**

**//5 \* 1 = 5**

**System.out.println(numero + " \* " + i + " = " + operacion);**

**}**

**}**

**}**

Vamos a intentar demostrar la Conjetura de Collatz.

Esto es una teoría matemática que dice lo siguiente:

Todo número positivo llegará siempre a 1 siguiendo una serie de condiciones:

* Si el número es par, se divide entre 2
* Si el número es impar, se multiplica por 3 y se suma 1

6, 3, 10, 5, 16, 8, 4, 2, 1

**Class11Collatz.java**

**import java.util.Scanner;**

**public class Class11Collatz {**

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

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

**System.out.println("Número para Collatz: ");**

***String* dato = teclado.nextLine();**

***int* numero = Integer.parseInt(dato);**

**while (numero != 1){**

**if (numero % 2 == 0){**

**numero = numero / 2;**

**}else{**

**numero = numero \* 3 + 1;**

**}**

**System.out.println(numero);**

**}**

**}**

**}**

**CLASES JAVA**

Todo en Java son clases (Excepto los primitivos).

Todas las clases están compuestas por métodos, dichos métodos pueden ser acciones o características de un objeto.

Por ejemplo, pongamos el ejemplo de un Coche:

Coche car;

car.arrancar(); //ACCION

car.girar(); //ACCION

car.color; //ATRIBUTO

Los métodos en las clases también pueden recibir parámetros, por ejemplo:

car.frenar(MUCHO);

car.frenar(POCO);

Dentro de las clases y del concepto de POO existen varias características.

Una de las características se llama Polimorfismo, que indica que un método puede tener múltiples formas.

Por ejemplo, no tenemos tres pedales de coche…

car.frenarMucho();

car.frenarPoco();

car.frenarNormal();

Aplicado a Polimorfismo, tenemos un solo método (frenar) y varias acciones sobre él

car.frenar(MUCHO);

car.frenar(POCO);

car.frenar();

Vamos a visualizar los métodos y características de la clase String.

Un String es una clase que contiene un conjunto de caracteres.

Tenemos multitud de métodos, para convertir a mayúsculas, para buscar caracteres, para eliminar caracteres…

Todos los conjuntos dentro de Java comienzan en CERO.

String texto = “Hola Cara Cola”;

* **equals():** Comparar un String con otro String (valor)
* **equalsIgnoreCase():** Comparar un String con otro String sin importar mayúsculas/minúsculas
* **isEmpty():** Pregunta si el texto es vacío.
* **toLowerCase():** Convierte a minúsculas un texto
* **toUpperCase():** Converte a mayúsculas un texto
* **contains(caracter):** Indica si un carácter está dentro de un String
* **length():** La longitud del texto del String
* **charAt(índice):** Devuelve la letra que se corresponde con dicho índice
* **indexOf(caracter):** Busca un carácter dentro del String y devuelve la posición donde se encuentra dicho carácter (la primera coincidencia).

Si no encuentra el carácter, devuelve -1.

* **indexOf(carácter, posición):** Busca un carácter dentro del String y devuelve la posición donde se encuentra dicho carácter a partir de la posición que estemos indicando.

Si no encuentra el carácter, devuelve -1.

* **lastIndexOf(caracter**): Realiza lo mismo que indexOf, pero comienza a buscar al final del texto, es decir, de derecha a izquierda.

Las posiciones de los caracteres no cambian busquemos por la izquierda o la derecha.

Si no encuentra el carácter devuelve -1.

* **endsWith(texto):** Devuelve true/false indicando si el String finaliza con el texto
* **startsWith(texto):** Devuelve true/false indicando si el String comienza con el texto indicado.
* **replace(old char, new char):** Reemplaza el antiguo texto por el nuevo
* **substring(inicio):** Recupera una subcadena del String a partir del inicio que indiquemos.
* **substring(inicio, final):** Recupera una subcadena del String a partir del inicio que indiquemos hasta el final.
* **trim():** Elimina los espacios al inicio y al final del texto.
* **String.valueOf(integer):** Sirve para convertir objetos primitivos a String

Para probar esto, creamos una nueva clase llamada **Class12String.java**