

Asignatura

Programación básica

| Solucionario de ejercicios



UNIVERSAE



ÍNDICE

| | |
|--------------------|----|
| Unidad 1..... | 3 |
| Unidad 2 | 10 |
| Unidad 3 | 16 |
| Unidad 4 y 5 | 27 |
| Unidad 6 | 44 |
| Unidad 7 | 53 |
| Unidad 8 | 71 |
| Unidad 9 | 79 |
| Unidad 10..... | 90 |
| Unidad 12 | 92 |

Unidad 1

Ejercicio 1.

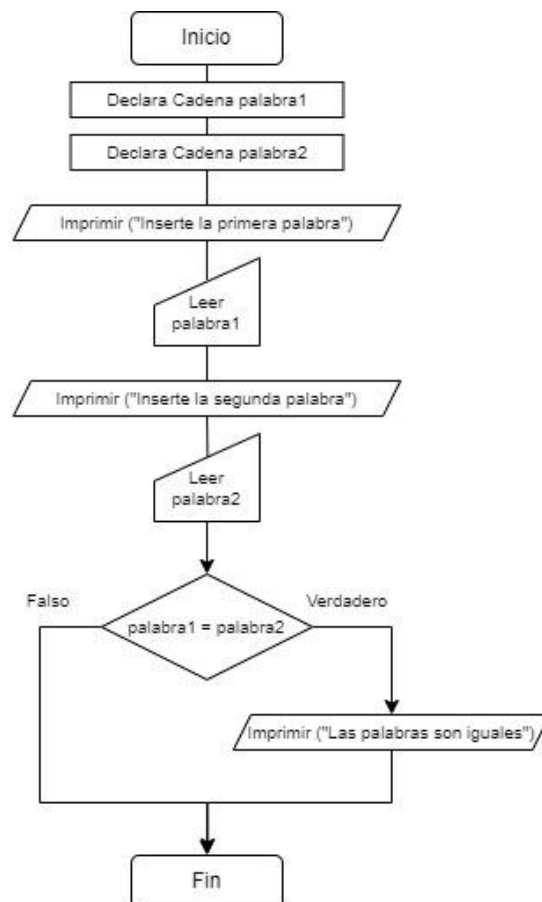
Pedir dos palabras por teclado, indicar si son iguales.

Pseudocódigo

```
Inicio
    Cadena palabra1
    Cadena palabra2

    Imprimir ("Inserte la primera palabra")
    Leer (palabra1)
    Imprimir ("Inserte la segunda palabra")
    Leer (palabra2)
    Si palabra1 = palabra2 entonces
        Imprimir ("Las palabras son iguales")
    Fin si
Fin
```

Diagrama



Ejercicio 2.

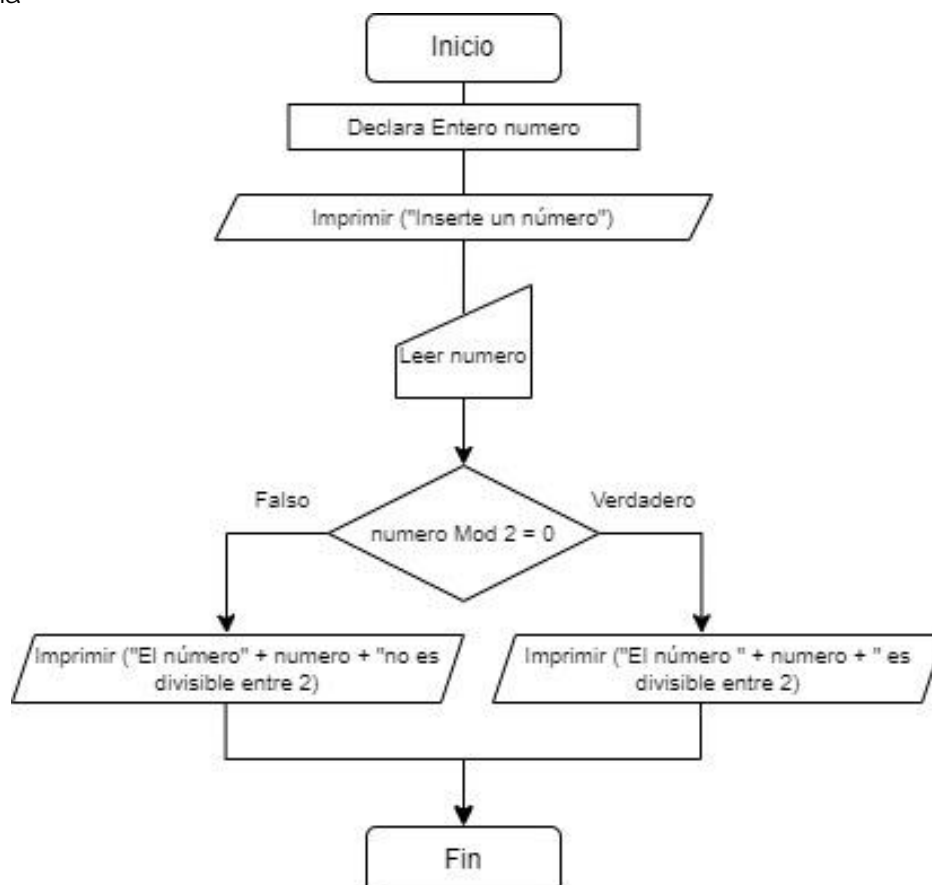
Lee un número por teclado e indica si es divisible entre 2 (resto = 0). Si no lo es, también debemos indicarlo.

Pseudocódigo

```
Inicio
    Entero numero

    Imprimir ("Inserte un número")
    Leer (numero)
    Si numero mod 2 = 0 entonces
        Imprimir ("El número "+ numero + " es divisible entre 2")
    Si no
        Imprimir ("El número " + numero + " no es divisible entre 2")
    Fin si
Fin
```

Diagrama



Ejercicio 3.

Pide un número por teclado e indica si es un número primo o no. Un número primo es aquel solo puede dividirse entre 1 y sí mismo. Por ejemplo: 25 no es primo, ya que 25 es divisible entre 5, sin embargo, 17 si es primo. NOTA: Si se introduce un número menor o igual que 1, directamente es no primo.

Pseudocódigo

```
Inicio
    Entero numero
    Booleano esPrimo ← Verdadero

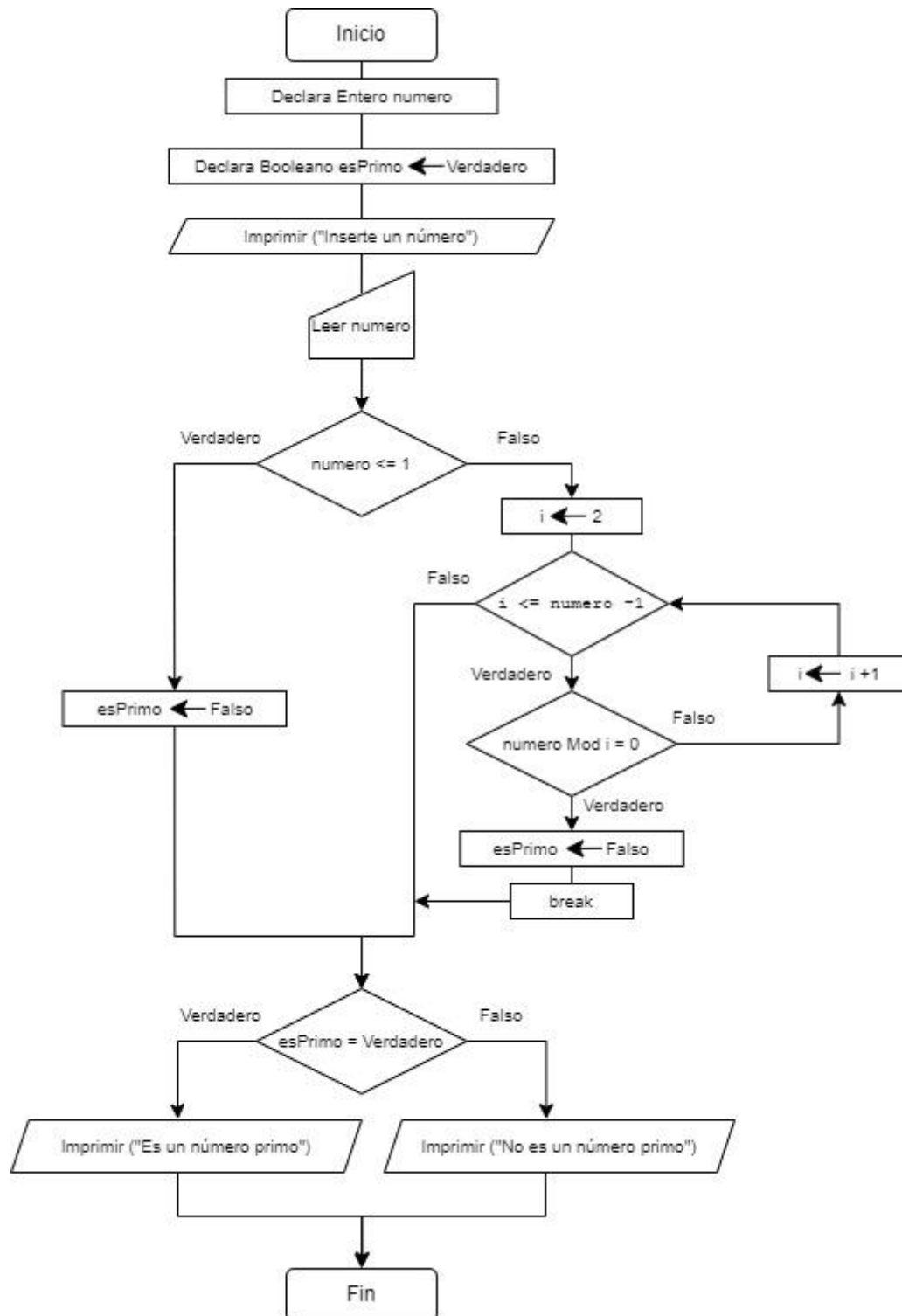
    Imprimir ("Inserte un número")
    Leer (numero)

    Si numero <= 1 Entonces
        esPrimo ← Falso
    Si no
        Para i ← 2 Hasta numero-1 Hacer
            Si numero Mod i = 0 entonces
                esPrimo ← Falso
                // No se continua con el bucle
                Salir del bucle (break)
            Fin si
        Fin para

    Fin Si

    Si esPrimo = Verdadero entonces;
        Imprimir ("Es un número primo")
    Sino
        Imprimir ("No es un número primo")
    Fin si
Fin
```

Diagrama



Ejercicio 4.

Realizar la suma del 1 al número que indiquemos, este debe ser mayor que 1.

Pseudocódigo

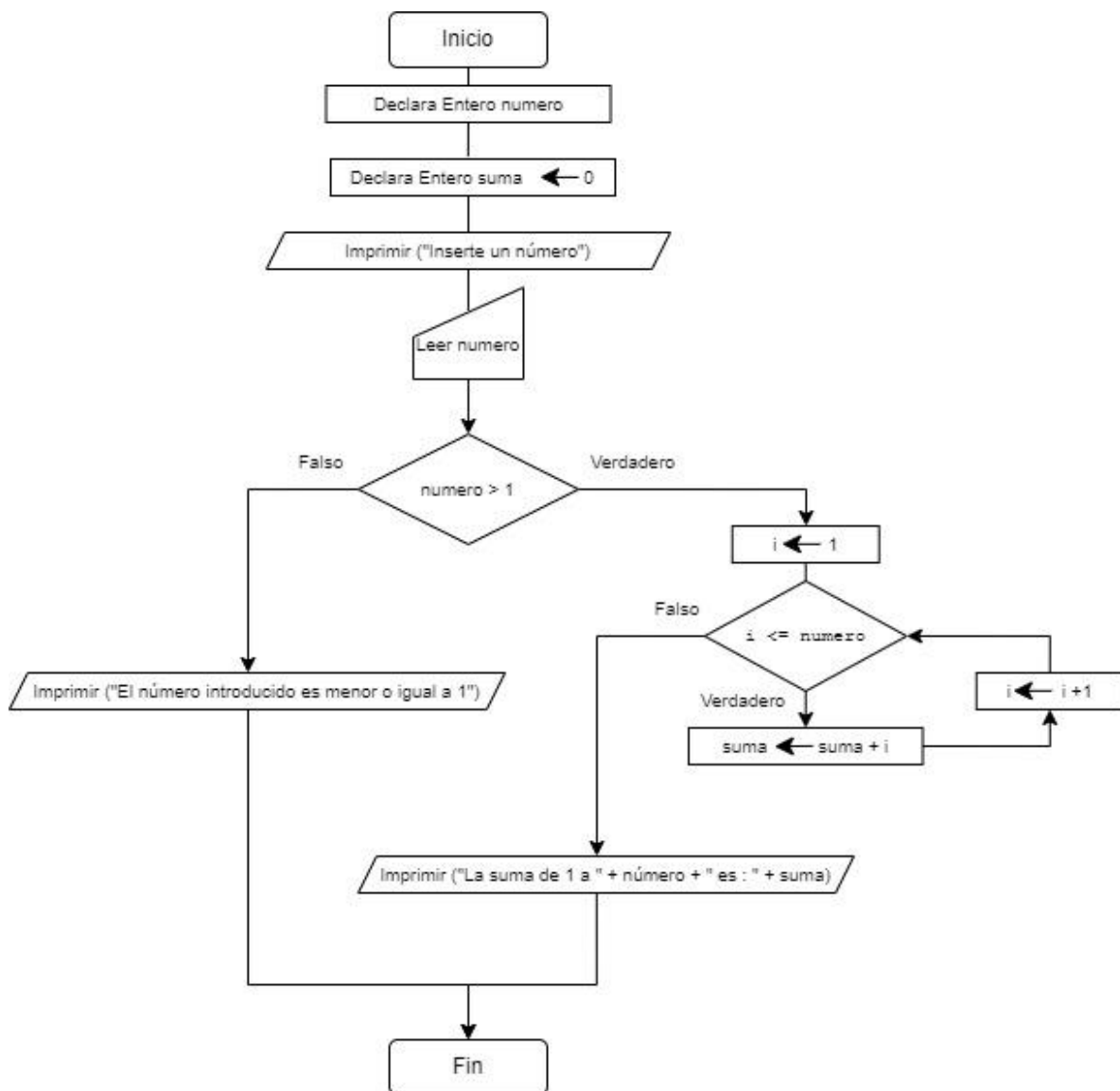
```
Inicio
    Entero numero
    Entero suma ← 0

    Imprimir ("Inserte un número")
    Leer (numero)

    Si (numero>1) Entonces
        Para i ← 1 Hasta numero Hacer
            suma ← suma + i
        Fin para

        Imprimir ("La suma de 1 a " + numero + " es: " + suma)
    Si no
        Imprimir ("El numero introducido es menor o igual a 1)
    Fin Si
Fin
```

Diagrama



Ejercicio 5.

Haz un algoritmo que calcule el área de un círculo ($\pi \cdot R^2$). El radio se pedirá por teclado.

El cálculo del área se debe realizar en una función.

Pseudocódigo

```

Funcion CalcularAreaCirculo(radio)
    Decimal PI ← 3.14159265359
    Decimal area ← PI * radio * radio

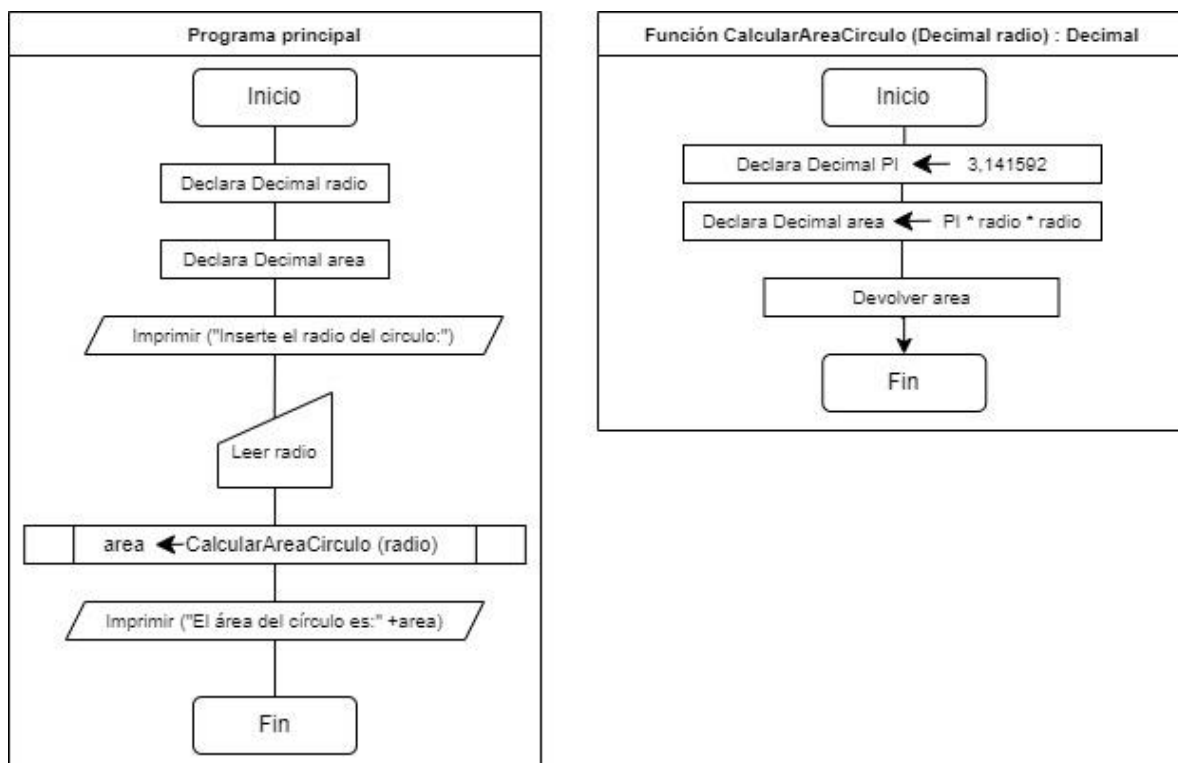
    Retornar area
Fin función
  
```



```

Inicio
    Decimal radio
    Decimal area
    Imprimir ("Inserte el radio del círculo:")
    Leer (radio)
    area ← CalcularAreaCirculo(radio)
    Imprimir ("El área del círculo es : " + area)
Fin
  
```

Diagrama



Unidad 2

Ejercicio 1.

Indica una sola clase y los campos necesarios que debe tener para representar a los cuatro elementos principales que aparecen en la imagen y muestra sus objetos.



Animal (nombre, nombreCientifico, tipo, tipoAlimentacion)

Los objetos que se muestran en la imagen.

Animal1("Burro", "Equus asinus", "Mamífero", "Herbívoro")

Animal2("Perro", "Canis lupus familiaris", "Mamífero", "Carnívoro")

Animal3("Gato", "Felis catus", "Mamífero", "Carnívoro")

Animal4("Gallo", "Gallus gallus domesticus", "Ave", "Omnívoro")

Ejercicio 2.

Saca las clases, campos, propiedades y relaciones del siguiente enunciado:

"Un sistema de comercio electrónico simple que permita a los usuarios navegar por productos, agregar artículos a un carrito de compras y realizar pedidos. El sistema debe contemplar: usuarios, productos, carritos de compras y pedidos. Cada usuario debe tener un nombre de usuario y una contraseña, y cada producto debe tener un nombre, descripción y precio. El carrito de compras debe permitir a los usuarios agregar y eliminar artículos, y los pedidos debe contener información sobre los artículos pedidos, el coste total y la dirección de envío."

Muestra un objeto de cada clase.



Clases

Usuario (nombreUsuario, contraseña, login())

Producto (nombre, descripción, precio)

Carrito (listaProductos[], añadirProducto(Producto), quitarProducto(Producto), realizarPedido())

Pedido(listaProductos, costeTotal, direccion)

Relaciones

Un Usuario puede tener un Carrito.

Un Usuario puede realizar múltiples Pedidos

Un Carrito pertenece a un Usuario.

Un Carrito puede contener múltiples Productos.

Un Pedido pertenece a un Usuario.

Un Pedido puede contener múltiples Productos.

Objetos

Usuario1 ("juan123", "contraseña")

Producto1 ("Zapatillas deportivas", "Zapatillas para correr y hacer ejercicio.", 59,99)

Producto1 ("Camisa deportiva", "Camisa transpirable de color blanco", 29,98)

Carrito (listaProductos: [Producto1, Producto2])

Pedido (listaProductos: [Producto1, Producto2], 89,97, "Calle Tranv 123, 1ºA, Madrid, España")

Ejercicio 3.

Saca las clases, campos, propiedades y relaciones del siguiente enunciado:

“Un sistema escolar que realice un seguimiento de la información de los estudiantes y su rendimiento académico. El sistema debe contemplar: estudiantes, cursos y calificaciones. Cada estudiante debe tener un nombre, fecha de nacimiento e identificación. Cada curso debe tener un código de curso, nombre y descripción, y cada calificación debe contener información sobre el rendimiento del estudiante en un curso en particular, como la calificación recibida y la fecha en que se obtuvo. El sistema también debe incluir profesores y clases, con profesores asignados a cursos específicos y clases que contengan grupos de estudiantes matriculados en el mismo curso.”

Muestra un objeto de cada clase.



Clases

Estudiante (identificación, nombre, fechaNacimiento)
 Curso (codigo, nombre, descripción)
 Calificación (Estudiante, Curso, calificación, fecha, registrarNota())
 Profesor (identificación, nombre)
 Clase (curso, listaEstudiantes)

Relaciones

Un Estudiante puede estar matriculado en múltiples Clases
 Un Estudiante puede tener múltiples Calificaciones
 Un Curso puede tener múltiples Clases
 Un Curso puede ser impartido por múltiples Profesores
 Una Calificación pertenece a un Estudiante
 Una Calificación está relacionada con un Curso
 Un Profesor puede impartir múltiples Cursos
 Una Clase puede tener múltiples Estudiantes
 Una Clase está relacionada con un Curso

Objetos

EstudianteAna ("11223344Y", "Ana Pérez", "01/05/2005")
 EstudiantePedro ("22334455W", "Pedro Carrasco", "15/08/2000")

CursoMath ("MATH101", "Matemáticas Básicas", "Este curso cubre conceptos básicos de matemáticas, como aritmética, álgebra y geometría.")

Calificación1 (EstudianteAna, CursoMath, 8,5 "16/06/2022")

Profesor ("33445556R", "Carlos Sánchez")

Clase (CursoMath, [EstudianteAna, EstudiantePedro])

Ejercicio 4.

Saca las clases, campos, propiedades y relaciones del siguiente enunciado:

"Un zoológico que incluya información sobre varios animales y sus hábitats. El sistema debe contemplar: animales, hábitats y cuidadores. Cada animal debe tener un nombre, especie y edad, y cada hábitat debe tener un nombre, ubicación y descripción. El cuidador debe contener información sobre su nombre, sus responsabilidades y su horario. Cada animal debe ser asignado a un hábitat, y cada hábitat debe tener uno o más animales viviendo en él. El sistema también debe incluir exhibiciones, donde cada exhibición contiene uno o más hábitats y proporciona información sobre los animales y sus hábitats a los visitantes."

Muestra un objeto de cada clase.

Clases

Animal (nombre, especie, edad)
Habitat (nombre, ubicación, descripción)
Cuidador (nombre, responsabilidades, horario)
Exhibición (nombre, listaHabitats)
Relaciones

Un Animal está asignado a un Hábitat
Un Hábitat puede tener uno o más Animales
Un Hábitat puede formar parte de una Exhibición
Una Exhibición contiene uno o más Hábitats

Objetos

Animal1 ("Simba", "León", 4)

HabitatSabana ("Sabana africana", "Sector B", "Una recreación de la sabana africana con árboles y pastos para albergar animales como leones, jirafas y cebras.")

HabitatDesierto ("Desierto", "Sector A", "Una recreación del desierto con palmeras y oasis para albergar animales como camellos, felinos y coyotes")

CuidadorMariaP ("María Pérez", "Alimentar a los animales de la Sabana africana, mantener el hábitat limpio y monitorear la salud de los animales.", "Lunes a viernes de 8:00 a 17:00")

Exhibición1 ("Safari africano", [HabitatSabana, HabitatDesierto])

Ejercicio 5.

Saca las clases, campos, propiedades y relaciones que veas oportuno. El tema principal de la imagen es el trabajo de tres trabajadores sobre la construcción de una casa.





Clases

Trabajador (nombre, rol, experiencia)

Tarea (descripción, estado, listadoTrabajadores, asignarTrabajador())

Relaciones

Un Trabajador puede estar asignado a una Tarea

Una Tarea la pueden realizar uno o más trabajadores

Objetos

Trabajador1 ("Laura González", "Arquitecto", 10)

Trabajador2 ("Carlos Martínez", "Financiero", 7)

Trabajador3 ("Juan Pérez", "Delineante", 8)

Tarea1 ("Realizar cálculos de costes de construcción", "En proceso", [Trabajador1, Trabajador2])

Tarea2 ("Dibujar plano de construcción", "En proceso", [Trabajador1, Trabajador3])

Ejercicio 6.

Obtener el diagrama de clases de un sistema que gestiona la agenda de contactos.

La agenda se identifica por un nombre y contiene el número total de contactos. A cada agenda se puede acceder por una única cuenta con un usuario y contraseña, las cuentas deben tener propiedades para acceder a sus campos y una cuenta solo puede tener una única agenda.

Un contacto contiene un identificador único de tipo numérico, nombre, apellidos, teléfono y email. Además, debe tener propiedades para acceder a sus campos. Un contacto puede pertenecer a más de una agenda y una agenda disponer de más de un contacto.

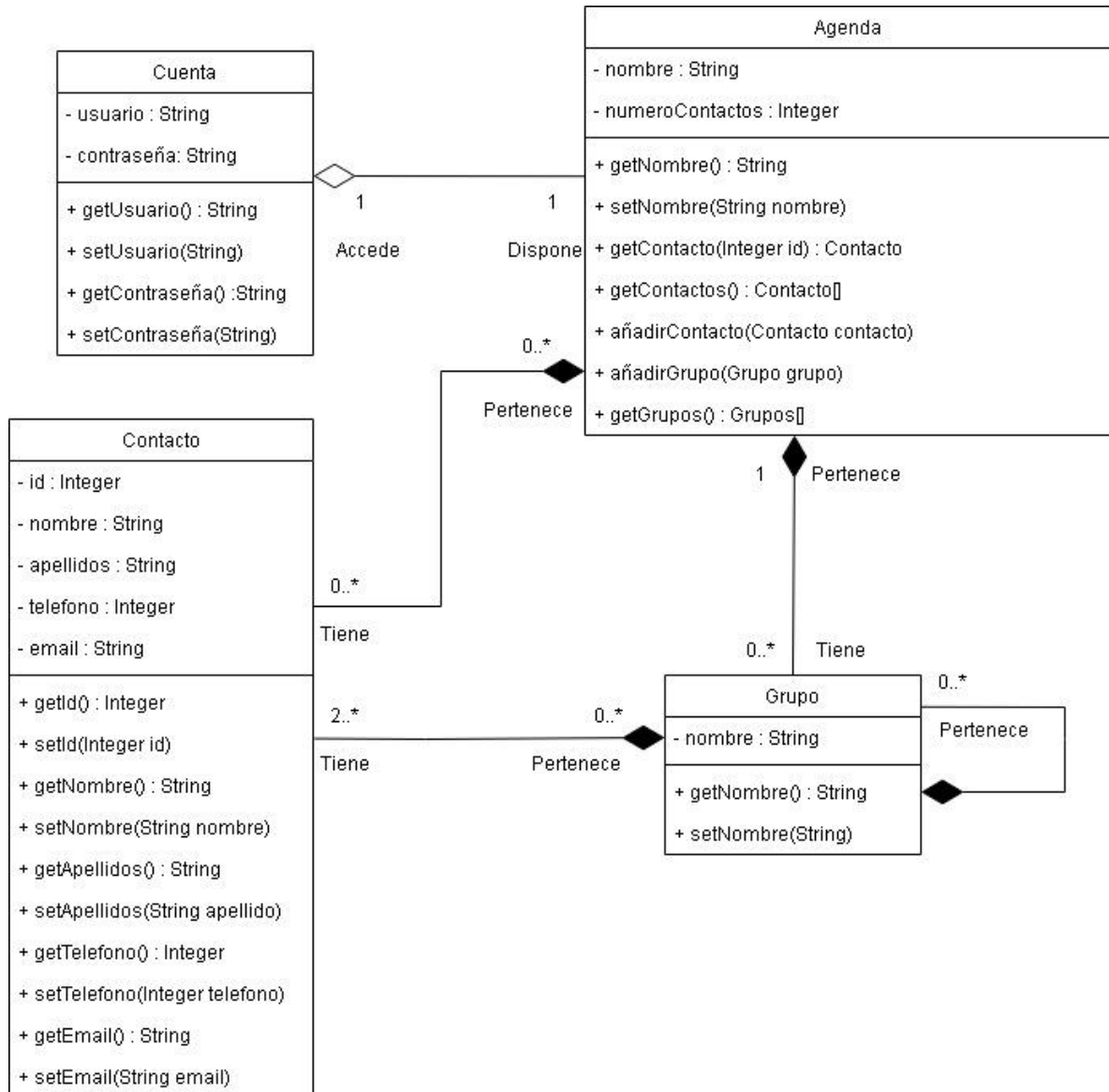
A parte se quiere dar la posibilidad de realizar grupos de contactos identificados por un nombre de cadena de caracteres. Un grupo puede surgir a partir de otro, esto quiere decir, que puede haber un grupo padre y subgrupos que dependan de él. Como funciones debe tener las propiedades del campo nombre. Los grupos están compuestos como mínimo de dos contactos o más. A su vez, un contacto puede ser que no esté en ningún grupo o en varios.

Un grupo solo puede pertenecer a una agenda y una agenda puede tener más de un grupo. La agenda debe tener las propiedades para el campo nombre y además las siguientes funcionalidades:

- Obtener un contacto a partir de su identificador.
- Obtener el listado de contactos
- Añadir un contacto a la agenda.
- Añadir un grupo a la agenda

- Obtener el listado de grupos

Se tiene que considerar que, si se borra una agenda, los contactos y grupos que le pertenecen se tiene que borrar. Sucede lo mismo si se borra un grupo, sus contactos y subgrupos dejan de existir.





Unidad 3

Ejercicio 1. Instalación de la JDK de java.

Procedimiento

1. Descargar de la página oficial de Oracle, la última versión disponible de la JDK para Windows
2. Proceder con la instalación y verificar que se instala correctamente.
3. Buscar la ruta donde se ha instalado.
4. Revisa las variables de entorno PATH y CLASSPATH. Se puede acceder a las variables de entorno mediante:

Panel de control/Sistema y seguridad/Sistema/Configuración avanzada del sistema

En el Path debe de aparecer un registro que apunta a la ruta donde está instalado la JDK

Responde a las siguientes preguntas

- ¿Qué diferencia existe entre la JDK y la JRE? ¿Como programadores que entorno necesitamos la JDK o JRE? ¿Y si solo voy a ejecutar programas en java?
 - Existe un entorno denominado OpenJDK. Busca porque existe una versión denominada OracleJDK y otra OpenJDK
 - ¿Por qué es necesario poner la ruta de la instalación en el PATH?
 - Abre el símbolo del sistema (Inicio y escribir cmd o símbolo del sistema) y ejecutar el siguiente comando `java -version`. ¿Qué resultado da? Nota si aparece un error revisar de nuevo el punto 4. de la instalación
-
- A continuación, explicamos que es la JDK y la JRE:

JDK (Java Development Kit): Es el conjunto completo de herramientas y utilidades que los desarrolladores necesitan para desarrollar, compilar, depurar y ejecutar aplicaciones Java. La JDK incluye la JRE, así como el compilador de Java (javac), el depurador (jdb), otras herramientas de desarrollo, utilidades y bibliotecas estándar de Java.

JRE (Java Runtime Environment): Es el entorno mínimo necesario para ejecutar aplicaciones Java. La JRE incluye la máquina virtual de Java (JVM, Java Virtual Machine) y las bibliotecas estándar de Java. La JRE no incluye herramientas de desarrollo como el compilador ni el depurador.

Con lo cual, para desarrollar será necesario tener la JDK y para ejecutar programas sin desarrollar JRE



- Versiones OpenJDK y OracleJDK

OpenJDK y Oracle JDK son dos implementaciones diferentes de Java Development Kit (JDK), pero ambas están basadas en el mismo código fuente y ofrecen prácticamente la misma funcionalidad. La principal diferencia entre ellas radica en la licencia, el soporte y algunas características adicionales.

OpenJDK: Es una implementación de código abierto del Java Platform, Standard Edition (Java SE). OpenJDK es el resultado de un esfuerzo de colaboración entre varios desarrolladores y organizaciones, incluida Oracle, para crear una implementación de Java de código abierto. El proyecto OpenJDK se creó en 2006 cuando Sun Microsystems (adquirida por Oracle en 2010) decidió liberar gran parte del código fuente de su JDK bajo una licencia de código abierto (la Licencia Pública General de GNU, versión 2, con la "Cláusula de excepción de la Clase"). OpenJDK es la base oficial para la implementación de referencia de Java SE, y sus actualizaciones se sincronizan con las versiones de Oracle JDK.

Oracle JDK: Antes de la versión 11 de Java, Oracle JDK era una implementación comercial separada de JDK que incluía algunas características adicionales, mejoras en el rendimiento y herramientas en comparación con OpenJDK. Sin embargo, a partir de la versión 11 de Java, Oracle JDK y OpenJDK están casi al mismo nivel en términos de características y rendimiento. Oracle JDK ahora se distribuye bajo la Licencia de Oracle Technology Network (OTN) para Oracle Java SE, que permite el uso gratuito para desarrollo y pruebas, pero requiere una suscripción comercial para uso en producción. Además, Oracle proporciona soporte a largo plazo (LTS) solo para las versiones de Oracle JDK cubiertas por una suscripción comercial.

- Ruta de instalación en el PATH

Este paso es necesario para que el sistema operativo conozca donde está ubicado los ejecutables de las herramientas de java cuando se ingresan comandos en la línea de comandos o terminal.

Las últimas versiones de la JDK o JRE incluyen en su asistente de instalación que se añade automáticamente el registro en el PATH.

Se puede averiguar si no tenemos configurado correctamente el PATH abriendo un terminal del sistema operativo y escribiendo java. Si nos aparece un mensaje que no reconoce el comando, es que el PATH no está bien.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Usuario>java
"java" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\Usuario>
```

- Ejemplo de ejecutar el comando java -version

Muestra la versión instalada de la JRE

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Usuario>java -version
java version "18.0.1.1" 2022-04-22
Java(TM) SE Runtime Environment (build 18.0.1.1+2-6)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.1.1+2-6, mixed mode, sharing)

C:\Users\Usuario>
```

Ejercicio 2. Ejecución de una aplicación sin IDE

Procedimiento

1. Crear una carpeta llamada *Ejercicios* dentro de C:
2. Crear un fichero de texto llamado *Ejercicio2.java* dentro de la carpeta del paso 1. Es importante que la primera letra este en mayúsculas.
Nota: Si tenéis oculto que muestre las extensiones protegidas por el sistema operativo, os creará un fichero tal que así Ejercicio2.java.txt. Se debe habilitar que el sistema operativo muestre las extensiones y borrar del nombre la extensión .txt
3. Abrir el fichero con el bloc de notas y añadir el siguiente código:

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class Ejercicio2 {

    public static void main(String[] args) {

        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDateTime fecha = LocalDateTime.now();

        System.out.println("Hoy es : " + dtf.format(fecha) + ", un fantastico día!");

        System.out.println("Indicamento como te llamas:");
        Scanner entrada = new Scanner(System.in);
```



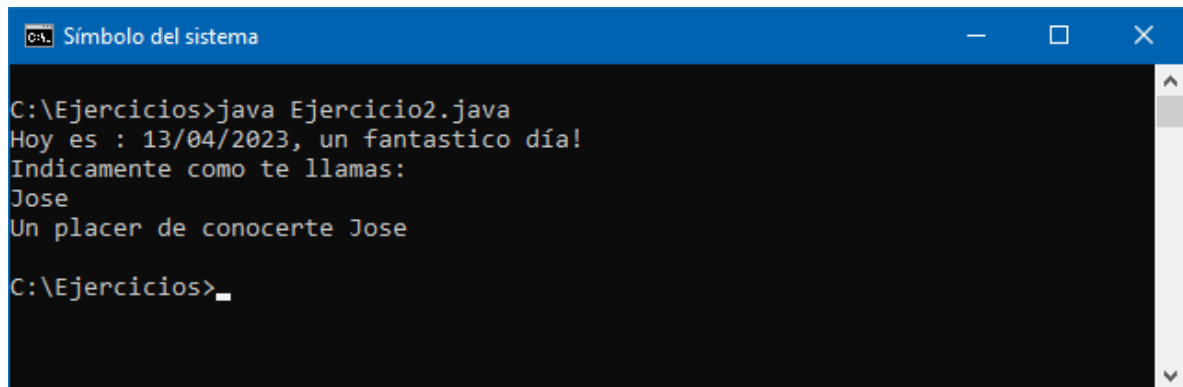
```
String contenido = entrada.next();  
System.out.println("Un placer de conocerte " + contenido);  
}  
}
```

4. Abrir el símbolo del sistema (Inicio y escribir cmd o símbolo del sistema) y ejecutar: `cd C:\Ejercicios` y nos ubicará en la carpeta que habíamos creado. Ahora ejecutar un segundo comando: `java Ejercicio2.java`

Responde a las siguientes preguntas:

- Muestra una captura de pantalla que se pueda ver el resultado de la ejecución del último programa.
- Explica que hace el comando `System.out.println`
- Ejecuta el comando: `javac Ejercicio2.java` y mirar en la carpeta, ¿Ha creado un nuevo fichero? Explica para que sirve.

Al ejecutar el comando `java Ejercicio2.java` obtenemos la siguiente respuesta por consola:





```
C:\Ejercicios>java Ejercicio2.java  
Hoy es : 13/04/2023, un fantastico día!  
Indicame como te llamas:  
Jose  
Un placer de conocerte Jose  
C:\Ejercicios>
```

`System.out.println` es un comando en Java que se utiliza para imprimir un mensaje en la consola y, a continuación, agregar un salto de línea.

Al ejecutar el comando `javac Ejercicio2.java` se genera un nuevo fichero con extensión `.class`

Nombre

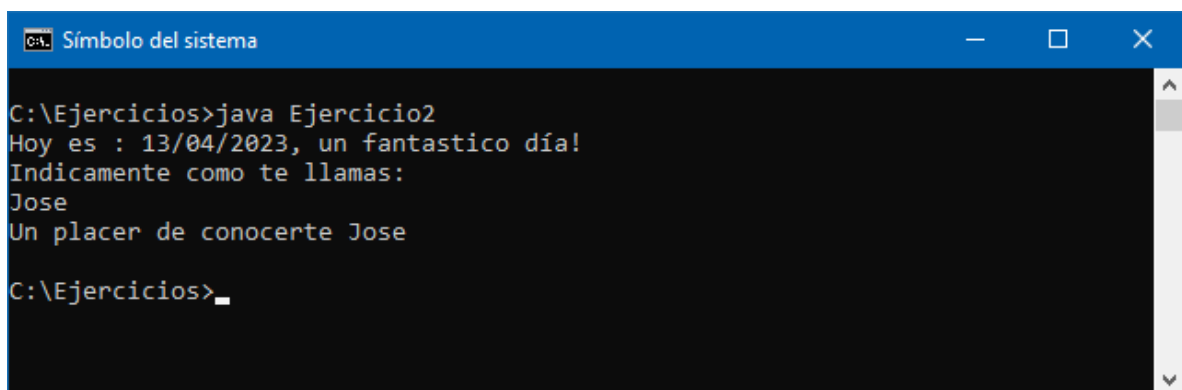
 Ejercicio2.class
 Ejercicio2.java

Los ficheros `.class` son archivos binarios que contienen el bytecode generado después de compilar el código fuente de Java (archivos `.java`). El bytecode es una representación

intermedia del código fuente que es independiente de la plataforma y se puede ejecutar en la Máquina Virtual de Java (JVM).

Cuando escribes un programa en Java y lo guardas como un archivo .java, el siguiente paso es compilarlo usando el compilador de Java (javac). El compilador convierte tu código fuente en archivos .class que contienen bytecode. Esto permite que se pueda ejecutar el programa en cualquier entorno que este instalado la JRE, sin tener en cuenta el sistema operativo o cualquier requisito exclusivo.

Ahora ya no es necesario disponer del fichero .java. Es posible ejecutar directamente el .class mediante el comando `java Ejercicio2`



```
C:\Ejercicios>java Ejercicio2
Hoy es : 13/04/2023, un fantastico día!
Indicame como te llamas:
Jose
Un placer de conocerte Jose

C:\Ejercicios>_
```

Se puede observar que no es necesario indicar la extensión .class, simplemente se pone el nombre del fichero.

Ejercicio 3. Instalación de un entorno de desarrollo

Existen diferentes entornos de desarrollo para Java, busca información sobre Eclipse y Netbeans, descárgate ambos y procede a instalarlos.

Responde a las siguientes preguntas:

- ¿Cuál te gusta más? Indica el por qué
- ¿Qué diferencia encuentras entre trabajar con un entorno gráfico o no hacerlo?
- A parte de Eclipse y Netbeans, ¿Existen otros entornos gráficos? Mencionalos.
- Crea un proyecto de cero, explica como lo has hecho.
- Existen diferentes tipos de proyecto. Para una aplicación normal de java. ¿Qué tipo de proyecto tenemos que crear?

Eclipse y NetBeans son dos entornos de desarrollo integrado (IDE) utilizados para desarrollar aplicaciones en varios lenguajes de programación, incluido Java.

- Eclipse: Es un proyecto de código abierto desarrollado y mantenido por la Eclipse Foundation. Eclipse cuenta con una gran comunidad de desarrolladores y contribuyentes, lo que permite una amplia gama de plugins y extensiones.



Admite una amplia gama de lenguajes de programación, como C, C++, Python, PHP, JavaScript y muchos más a través de plugins y extensiones.

Es altamente personalizable y se puede configurar para adaptarse a las necesidades de los desarrolladores. Sin embargo, su flexibilidad a veces puede dificultar la configuración y el uso, especialmente para principiantes. Es por tal motivo que eclipse es considerado más lento en comparación con NetBeans.

- NetBeans: Originalmente desarrollado por Sun Microsystems, NetBeans es ahora un proyecto de código abierto de la Apache Software Foundation. También cuenta con una comunidad activa, pero su ecosistema de plugins y extensiones es generalmente más pequeño que el de Eclipse. Admite múltiples lenguajes de programación, incluidos Java, C, C++, PHP, JavaScript y otros. Sin embargo, su enfoque principal sigue siendo Java y las tecnologías relacionadas.

La principal diferencia con Eclipse es su facilidad de uso y configuración "lista para usar". Ofrece una experiencia más sencilla y directa, lo que lo hace especialmente atractivo para los desarrolladores principiantes o aquellos que prefieren un enfoque más simple.

La elección de un IDE u otro dependerá de cada uno.

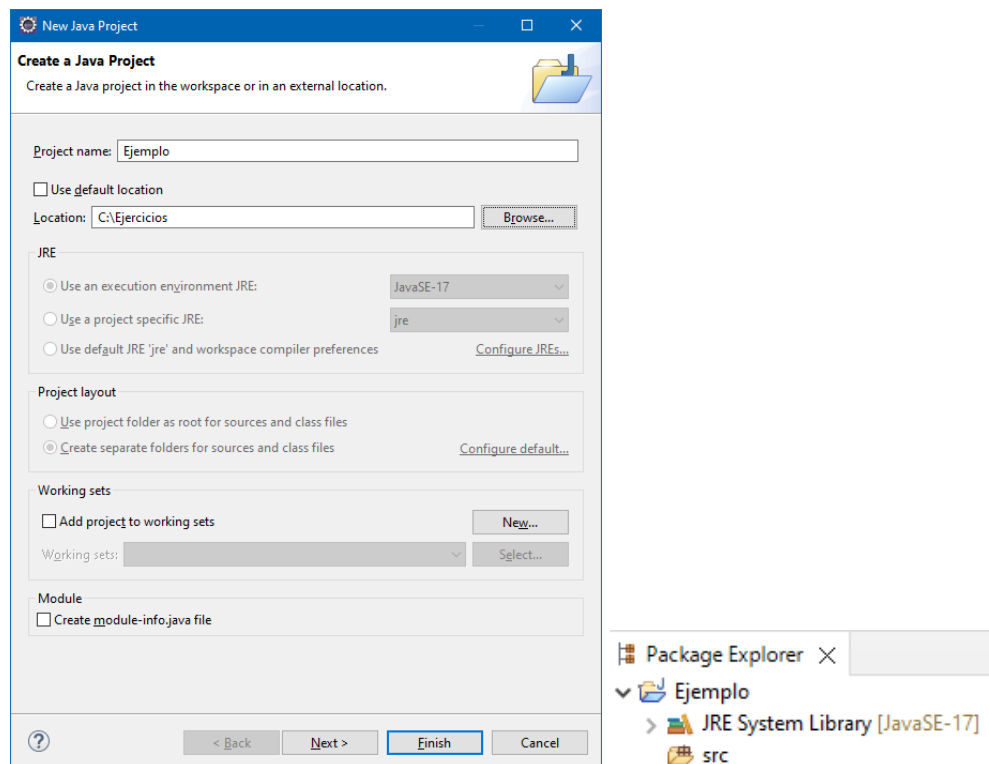
La elección de trabajar con un IDE o sin, dependerá en gran medida de las preferencias personales, requisitos de proyecto y nivel de experiencia. Los IDE ofrecen una experiencia de desarrollo más integrada y fácil de usar, mientras que trabajar sin un IDE puede proporcionar más flexibilidad y un enfoque más ligero.

A parte de eclipse y netbeans, es posible utilizar los siguientes IDEs para Java:

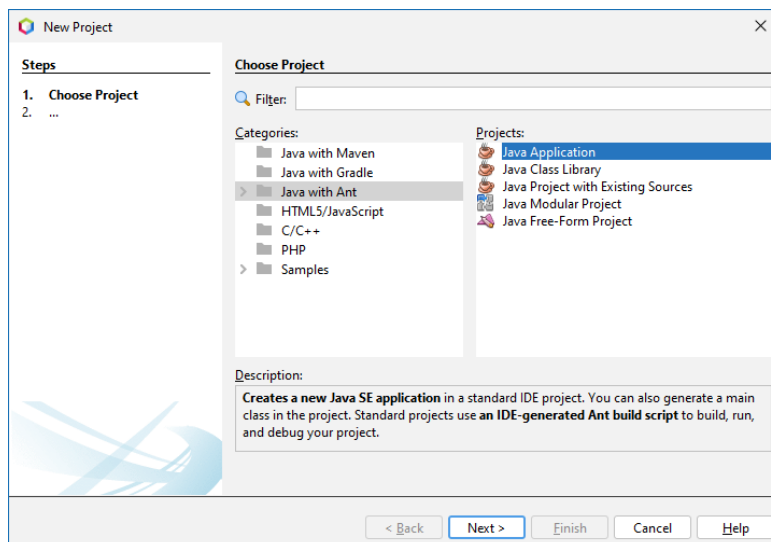
- IntelliJ IDEA
- Visual Studio Code
- JDeveloper
- BlueJ
- DrJava

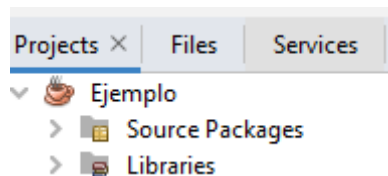
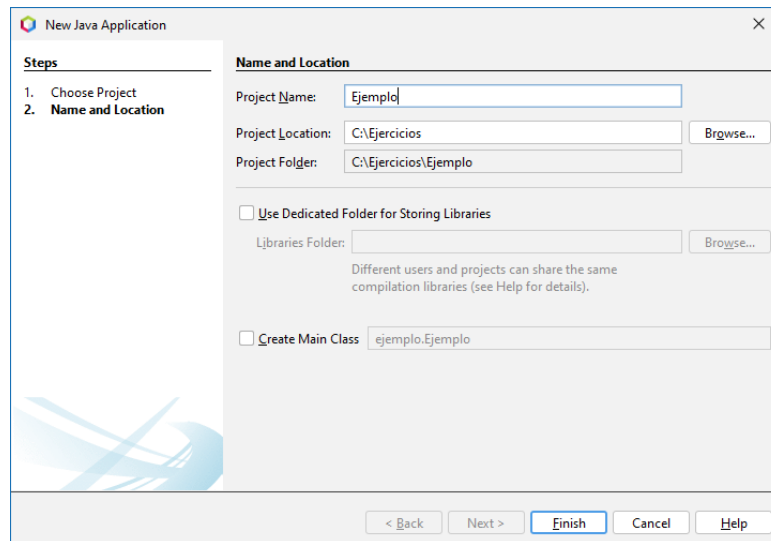
Para crear un proyecto de cero, debemos abrir el entorno concreto.

- Desde Eclipse. Menú File/New/Java Project
Debemos indicar un nombre al proyecto, especificar que versión de java se utilizará para ejecutar y la ubicación del proyecto.



- Desde Netbeans. Menú File/New Project
Se selecciona Java Application como proyecto dentro de la categoría Java with Ant. Se indica el nombre del proyecto y la ubicación.





Para una aplicación normal o de escritorio se utilizará los tipos de proyecto "Java Application" o "Java Project". Además de la aplicación existen otro tipo de proyectos como:

- Aplicaciones Web "Static Web Project". Estos proyectos utilizan tecnologías como Java Servlets, JavaServer Pages (JSP) y JavaServer Faces (JSF) para desarrollar aplicaciones que se ejecutan en servidores web y de aplicaciones, como Tomcat, Jetty o GlassFish.
- Aplicaciones empresariales "Enterprise Application Project". Estos proyectos se basan en la plataforma Java EE (Enterprise Edition) e incluyen tecnologías como EJB (Enterprise JavaBeans), JPA (Java Persistence API) y JMS (Java Message Service) para desarrollar aplicaciones empresariales más complejas y escalables.
- Proyectos Maven "Maven Project". Maven es una herramienta de gestión de proyectos y compilación que facilita la gestión de dependencias y la construcción de proyectos Java. Un proyecto Maven utiliza una estructura de directorios específica y un archivo POM (Project Object Model) para definir las dependencias y configuraciones del proyecto.
- Proyectos Gradle "Gradle Project". Gradle es otra herramienta de automatización de compilación y gestión de proyectos, similar a Maven pero con un enfoque en la flexibilidad y el rendimiento. Gradle utiliza un lenguaje de programación basado en Groovy o Kotlin para definir la configuración del proyecto y las tareas de compilación.

Ejercicio 4. Ejecución de una aplicación con IDE

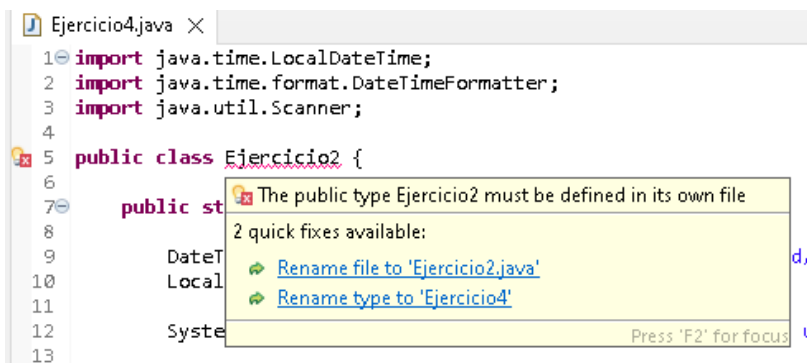
Procedimiento

1. Crea un proyecto de cero
2. Crea una clase y nómbrala *Ejercicio4*. Introduce el código que hay del ejercicio 2

Responde a las siguientes preguntas:

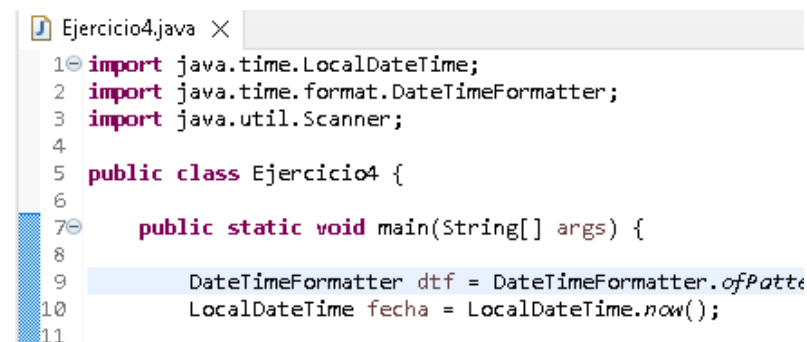
- Al finalizar el paso 2, ¿El IDE os muestra alguna advertencia o error? Si es así, explica que está sucediendo y como solucionarlo.
- Investiga como se ejecuta el programa y muestra una captura de pantalla con el resultado.
- ¿Por qué dentro de la clase existe un método llamado *main*? ¿Pueden existir más de uno en diferentes clases?

En este ejercicio se ha utilizado el IDE Eclipse. Una vez creado el proyecto y copiado el código del ejercicio 2, el IDE muestra un error.



Este error sucede debido a que hemos nombrado a la clase Ejercicio4.java y en el código se está nombrando Ejercicio2. El nombre de la clase debe coincidir con el nombre del fichero .java. Para solucionarlo podemos optar por dos vías:

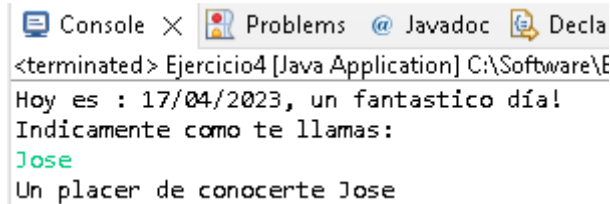
1. Cambiar el nombre del fichero a Ejercicio2.java
2. Cambiar el nombre en el código a Ejercicio4



Para ejecutar el programa podemos hacerlo de diferentes maneras:

1. Pulsar sobre el icono que hay dibujado un triángulo play con fondo verde
2. Desde la opción Run del menú y luego Run
3. Pulsar Control +F11 (Esto puede cambiar según el IDE)

Al ejecutar el programa se muestran los mensajes en una pestaña denominada Consola que simula el terminal del sistema operativo para mostrar cualquier salida de la aplicación.



```
<terminated> Ejercicio4 [Java Application] C:\Software\E
Hoy es : 17/04/2023, un fantastico día!
Indicame como te llamas:
Jose
Un placer de conocerte Jose
```

El método `main(String[] args)` en Java es el punto de entrada para la ejecución de una aplicación Java. Cuando se ejecuta una aplicación Java, la máquina virtual Java (JVM) busca y ejecuta este método específico para iniciar la aplicación.

Cada clase solo puede tener su propio método `main` y no más. Aún que cada clase puede definir su propio método `main`, es recomendable disponer de un solo punto de entrada a la aplicación.

Ejercicio 5. Ha llegado tu turno.

Realiza un pequeño programa que pida dos números, realice su división y muestre el resultado. Realizar primero el pseudocódigo para que os sea más fácil. Realizar algunas pruebas de ejecución y probar con estos datos número 1 = 5, número 2 = 0. ¿Qué sucede? ¿Cómo lo puedes solucionar?

```
import java.util.Scanner;

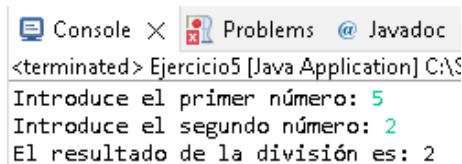
public class Ejercicio5 {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.print("Introduce el primer número: ");
        int num1 = entrada.nextInt();

        System.out.print("Introduce el segundo número: ");
        int num2 = entrada.nextInt();

        int resultado = num1 / num2;
        System.out.println("El resultado de la división es: " + resultado);
    }
}
```

Algún ejemplo de ejecución:



```
<terminated> Ejercicio5 [Java Application] C:\S
Introduce el primer número: 5
Introduce el segundo número: 2
El resultado de la división es: 2
```

Al introducir los datos 5 y 0 el programa no da el resultado esperado y muestra un error.



```
Console X Problems Javadoc Declaration Progress Coverage
<terminated> Ejercicio5 [Java Application] C:\Software\Eclipse\plugins\org.eclipse.justj.open
Introduce el primer número: 5
Introduce el segundo número: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Ejercicio5.main(Ejercicio5.java:14)
```

Según la descripción del error se produce por una operación aritmética inválida, en concreto, es debido a intentar hacer una división por cero. Para solucionar este problema, podemos comprobar si se introduce un 0 en el segundo número antes de realizar la operación. Existe una estructura condicional (if/else) que nos permite verificar una condición, en este caso que el número sea igual a 0 y realice una acción concreta, en caso contrario, realice otra acción.

```
import java.util.Scanner;

public class Ejercicio5 {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);

        System.out.print("Introduce el primer número: ");
        int num1 = entrada.nextInt();

        System.out.print("Introduce el segundo número: ");
        int num2 = entrada.nextInt();

        if(num2 == 0) {
            System.out.println("Error: No se puede dividir por 0");
        } else {
            int resultado = num1 / num2;
            System.out.println("El resultado de la división es: " + resultado);
        }
    }
}
```

Unidad 4 y 5

Ejercicio 1.

Declara dos variables numéricas (con el valor que desees), muestra por consola la suma, resta, multiplicación, división y módulo (resto de la división).

```
public static void main(String[] args) {  
  
    int num1 = 5;  
    int num2 = 2;  
  
    System.out.println("Suma : " + (num1+num2));  
    System.out.println("Resta : " + (num1-num2));  
    System.out.println("Multiplicación : " + (num1*num2));  
    System.out.println("División : " + (num1/num2));  
    System.out.println("Modulo : " + (num1%num2));  
  
}
```

Ejercicio 2.

Declara 2 variables numéricas (con el valor que desees), he indica cual es mayor de los dos. Si son iguales indicarlo también. Ves cambiando los valores para comprobar que funciona.

```
public static void main(String[] args) {  
  
    int num1 = 5;  
    int num2 = 2;  
  
    if(num1>num2) {  
        System.out.println("El número 1 es mayor que el número 2");  
    } else {  
        if(num1<num2) {  
            System.out.println("El número 2 es mayor que el número 1");  
        } else {  
            System.out.println("El número 1 es igual que el número 2");  
        }  
    }  
  
}
```

Ejercicio 3.

Declara un String que contenga tu nombre, después muestra un mensaje de bienvenida por consola. Por ejemplo: si introduzco «Ángel», me aparezca «Bienvenid@ Ángel».

```
public static void main(String[] args) {  
  
    String nombre = "Jose";  
  
    System.out.println("Bienvenid@ "+nombre);  
  
}
```



Ejercicio 4.

Modifica la aplicación anterior, para que nos pida el nombre que queremos introducir.

```
public static void main(String[] args) {  
  
    System.out.println("Introduce un nombre");  
    Scanner entradaTeclado = new Scanner(System.in);  
    String nombre = entradaTeclado.nextLine();  
  
    System.out.println("Bienvenid@ "+nombre);  
  
    entradaTeclado.close();  
  
}
```

Ejercicio 5.

Haz una aplicación que calcule el área de un círculo($\pi \cdot R^2$). El radio se pedirá por teclado (recuerda pasar de String a double con Double.parseDouble). Usa la constante PI y el método pow de Math.

```
public static void main(String[] args) {  
  
    final float PI = 3.14f;  
  
    System.out.println("Introduce el radio de un círculo");  
    Scanner entradaTeclado = new Scanner(System.in);  
    double radio = Float.parseFloat(entradaTeclado.nextLine());  
  
    double area = PI * Math.pow(radio, 2);  
  
    System.out.println("El area del círculo es: "+area);  
  
    entradaTeclado.close();  
  
}
```

Ejercicio 6.

Lee un número por teclado e indica si es divisible entre 2 (resto = 0). Si no lo es, también debemos indicarlo.

```
public static void main(String[] args) {  
  
    System.out.println("Introduce un número");  
    Scanner entradaTeclado = new Scanner(System.in);  
    int numero = Integer.parseInt(entradaTeclado.nextLine());  
  
    if(numero%2 == 0) {  
        System.out.println("El número introducido es divisible entre 2");  
    } else {  
        System.out.println("El número introducido no es divisible entre 2");  
    }  
  
    entradaTeclado.close();  
  
}
```



Ejercicio 7.

Lee un número por teclado y muestra por consola, el carácter al que pertenece en la tabla ASCII. Por ejemplo: si introduzco un 97, me muestre una a.

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

    System.out.println("Introduce un número");
    Scanner entradaTeclado = new Scanner(System.in);
    int numero = Integer.parseInt(entradaTeclado.nextLine());

    // Al transformar el numero a tipo char hace la conversion con los valores de la tabla ASCII
    char caracter = (char) numero;

    System.out.println("El número introducido equivale al carácter : "+caracter+" de la tabla ASCII");

    entradaTeclado.close();

}
```

Ejercicio 8.

Modifica el ejercicio anterior, para que en lugar de pedir un número, pida un carácter (char) y muestre su código en la tabla ASCII.

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

    System.out.println("Introduce un número");
    Scanner entradaTeclado = new Scanner(System.in);
    char caracter = entradaTeclado.next().charAt(0);

    int numero = (int) caracter;

    System.out.println("El caracter introducido equivale al código : "+numero+" de la tabla ASCII");

    entradaTeclado.close();

}
```

Ejercicio 9.

Lee un número por teclado que pida el precio de un producto (puede tener decimales) y calcule el precio final con IVA. El IVA será una constante que será del 21%.

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

    final float IVA = 21;

    System.out.println("Introduce el precio de un producto");
    Scanner entradaTeclado = new Scanner(System.in);
    float precio = entradaTeclado.nextFloat();

    float preciofinal = precio * (1+(IVA/100));
    System.out.println("El precio final del producto es: "+preciofinal);

    entradaTeclado.close();

}
```

Ejercicio 10.

Muestra los números del 1 al 100 (ambos incluidos). Usa un bucle while.

```
public static void main(String[] args) {  
  
    int contador = 1;  
  
    while (contador<=100) {  
        System.out.print(contador + " ");  
        contador++;  
    }  
  
}
```

Ejercicio 11.

Haz el mismo ejercicio anterior con un bucle for.

```
public static void main(String[] args) {  
  
    for (int contador = 1; contador <=100; contador++) {  
        System.out.print(contador + " ");  
    }  
  
}
```

Ejercicio 12.

Muestra los números del 1 al 100 (ambos incluidos)divisibles entre 2 y 3. Utiliza el bucle que desees.

```
public static void main(String[] args) {  
  
    String numerosDivisibles2 = "";  
    String numerosDivisibles3 = "";  
  
    for (int contador = 1; contador <=100; contador++) {  
        if(contador%2 == 0 )  
            numerosDivisibles2 += contador + " ";  
        if(contador%3 == 0 )  
            numerosDivisibles3 += contador + " ";  
    }  
  
    System.out.println("Lo numero divisibles entre 2 del 1 al 100 son: "+numerosDivisibles2);  
    System.out.println("Lo numero divisibles entre 3 del 1 al 100 son: "+numerosDivisibles3);  
  
}
```

Ejercicio 13.

Realiza una aplicación que nos pida un número de ventas a introducir, después nos pedirá tantas ventas por teclado como número de ventas se hayan indicado. Al final mostrara la suma de todas las ventas. Piensa que es lo que se repite y lo que no.

```
public static void main(String[] args) {  
  
    System.out.println("Introduce un número de ventas realizadas");  
    Scanner entradaTeclado = new Scanner(System.in);  
    int totalVentas = entradaTeclado.nextInt();  
  
    float importeTotal = 0;  
  
    for (int i = 1; i <= totalVentas; i++) {  
        System.out.println("Introduce el importe de la venta número " + i);  
    }  
  
}
```

```
        importeTotal += entradaTeclado.nextFloat();
    }

    System.out.println("El importe total de todas las ventas ha sido: "+importeTotal);

    entradaTeclado.close();
}
```

Ejercicio 14.

Realiza una aplicación que nos calcule una ecuación de segundo grado. Debes pedir las variables a, b y c por teclado y comprobar antes que el discriminante (operación en la raíz cuadrada). Para la raíz cuadrada usa el método `sqrt` de `Math`. Te recomiendo que uses mensajes de traza.

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

    Scanner entradaTeclado = new Scanner(System.in);
    double a, b, c;

    System.out.println("Introduce el valor de la variable a");
    a = entradaTeclado.nextDouble();

    System.out.println("Introduce el valor de la variable b");
    b = entradaTeclado.nextDouble();

    System.out.println("Introduce el valor de la variable c");
    c = entradaTeclado.nextDouble();

    double discriminante = Math.pow(b, 2) - 4 * a * c;
    System.out.println("Discriminante: " + discriminante);

    if (discriminante > 0) {
        double x1 = (-b + Math.sqrt(discriminante)) / (2 * a);
        double x2 = (-b - Math.sqrt(discriminante)) / (2 * a);
        System.out.println("Raíces reales y distintas:");
        System.out.println("x1 = " + x1);
        System.out.println("x2 = " + x2);
    } else if (discriminante == 0) {
        double x = -b / (2 * a);
        System.out.println("Raíces reales e iguales:");
        System.out.println("x1 = x2 = " + x);
    } else {
        System.out.println("Raíces complejas y conjugadas.");
        double parteReal = -b / (2 * a);
        double parteImaginaria = Math.sqrt(-discriminante) / (2 * a);
        System.out.println("x1 = " + parteReal + " + " + parteImaginaria + "i");
        System.out.println("x2 = " + parteReal + " - " + parteImaginaria + "i");
    }

    entradaTeclado.close();
}
```



Ejercicio 15.

Lee un número por teclado y comprueba que este número es mayor o igual que cero, si no lo es lo volverá a pedir (do while), después muestra ese número por consola.

```
public static void main(String[] args) {  
  
    Scanner scanner = new Scanner(System.in);  
    double numero;  
  
    do {  
        System.out.print("Introduce un número mayor o igual que cero: ");  
        numero = scanner.nextDouble();  
  
        if (numero < 0) {  
            System.out.println("El número ingresado es menor que cero.  
            Por favor, vuelve a intentarlo.");  
        }  
    } while (numero < 0);  
  
    System.out.println("El número ingresado es: " + numero);  
  
    scanner.close();  
  
}
```

Ejercicio 16.

Escribe una aplicación con un String que contenga una contraseña cualquiera. Después se te pedirá que introduzcas la contraseña, con 3 intentos. Cuando aciertes ya no pedirá más la contraseña y mostrará un mensaje diciendo «Enhorabuena». Piensa bien en la condición de salida (3 intentos y si acierta sale, aunque le queden intentos).

```
public static void main(String[] args) {  
  
    Scanner scanner = new Scanner(System.in);  
    String contrasena = "contraseña";  
    String entrada;  
    int intentos = 3;  
    boolean acertado = false;  
  
    System.out.println("Adivina la contraseña en 3 intentos");  
  
    while (intentos > 0 && !acertado) {  
        System.out.print("Introduce la contraseña: ");  
        entrada = scanner.nextLine();  
        intentos--;  
  
        if (entrada.equals(contrasena)) {  
            acertado = true;  
            System.out.println("Enhorabuena, has acertado la contraseña.");  
        } else if (intentos > 0) {  
            System.out.println("Contraseña incorrecta. Te quedan " + intentos + "  
            intento(s).");  
        }  
    }  
  
    if (!acertado) {  
        System.out.println("Lo siento, no has adivinado la contraseña en 3 intentos.");  
    }  
  
    scanner.close();  
  
}
```




Ejercicio 17.

Crea una aplicación que nos pida un día de la semana y que nos diga si es un día laboral o no. Usa un switch para ello.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce un día de la semana (en minúsculas): ");
    String dia = scanner.nextLine();

    switch (dia) {
        case "lunes":
        case "martes":
        case "miercoles":
        case "miércoles":
        case "jueves":
        case "viernes":
            System.out.println("Es un día laboral.");
            break;

        case "sabado":
        case "sábado":
        case "domingo":
            System.out.println("No es un día laboral.");
            break;

        default:
            System.out.println("Por favor, introduce un día de la semana válido.");
            break;
    }
    scanner.close();
}
```

Ejercicio 18.

Pide por teclado dos números y genera 10 números aleatorios entre esos números. Usa el método Math.random para generar un número entero aleatorio (recuerda el casting de double a int).

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    double num1, num2;

    System.out.print("Introduce el primer número: ");
    num1 = scanner.nextDouble();

    System.out.print("Introduce el segundo número: ");
    num2 = scanner.nextDouble();

    if (num1 > num2) {
        double temp = num1;
        num1 = num2;
        num2 = temp;
    }

    System.out.println("Generando 10 números aleatorios entre " + num1 + " y " + num2 + ":");

    for (int i = 0; i < 10; i++) {
        int numeroAleatorio = (int) (Math.random() * ((int)num2 - (int)num1 + 1)) +
            (int)num1;
        System.out.println("Número " + (i + 1) + ": " + numeroAleatorio);
    }

    scanner.close();
}
```



Ejercicio 19.

Pide por teclado un número entero positivo (debemos controlarlo) y muestra el número de cifras que tiene. Por ejemplo: si introducimos 1250, nos muestre que tiene 4 cifras. Tendremos que controlar si tiene una o más cifras, al mostrar el mensaje.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int numero;

    do {
        System.out.print("Introduce un número entero positivo: ");
        numero = scanner.nextInt();

        if (numero < 0) {
            System.out.println("El número ingresado no es positivo. Por favor, vuelve a intentarlo.");
        }
    } while (numero < 0);

    int contadorCifras = 0;
    while (numero > 0) {
        contadorCifras++;
        numero /= 10;
    }

    if (contadorCifras == 1) {
        System.out.println("El número tiene 1 cifra.");
    } else {
        System.out.println("El número tiene " + contadorCifras + " cifras.");
    }

    scanner.close();
}
```

Ejercicio 20.

Pide un número por teclado e indica si es un número primo o no. Un número primo es aquel solo puede dividirse entre 1 y sí mismo. Por ejemplo: 25 no es primo, ya que 25 es divisible entre 5, sin embargo, 17 sí es primo. Un buen truco para calcular la raíz cuadrada del número e ir comprobando que si es divisible desde ese número hasta 1.

NOTA: Si se introduce un número menor o igual que 1, directamente es no primo.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int numero;

    System.out.print("Introduce un número: ");
    numero = scanner.nextInt();

    if (numero <= 1) {
        System.out.println("El número ingresado no es primo.");
    } else {
        boolean esPrimo = true;
        int limite = (int) Math.sqrt(numero);

        for (int i = 2; i <= limite; i++) {
            if (numero % i == 0) {
                esPrimo = false;
                break;
            }
        }

        if (esPrimo) {

```



```

        System.out.println("El número ingresado es primo.");
    } else {
        System.out.println("El número ingresado no es primo.");
    }
}

scanner.close();
}

```

Ejercicio 21.

Muestra los números primos entre 1 y 100.

```

public static void main(String[] args) {
    System.out.println("Números primos entre 1 y 100:");

    for (int numero = 2; numero <= 100; numero++) {
        boolean esPrimo = true;
        int limite = (int) Math.sqrt(numero);

        for (int i = 2; i <= limite; i++) {
            if (numero % i == 0) {
                esPrimo = false;
                break;
            }
        }

        if (esPrimo) {
            System.out.print(numero + " ");
        }
    }
}

```

Ejercicio 22.

Del siguiente String «La lluvia en Madrid es una maravilla» cuenta cuantas vocales hay en total (recorre el String con charAt).

```

public static void main(String[] args) {
    String frase = "La lluvia en Madrid es una maravilla";
    int contadorVocales = 0;

    for (int i = 0; i < frase.length(); i++) {
        char caracter = Character.toLowerCase(frase.charAt(i));

        if (caracter == 'a' || caracter == 'e' || caracter == 'i' || caracter == 'o' ||
            caracter == 'u') {
            contadorVocales++;
        }
    }

    System.out.println("El número de vocales en la frase \"" + frase + "\" es: " +
        contadorVocales);
}

```

Ejercicio 23.

Reemplaza todas las a del String anterior por una e.

```

public static void main(String[] args) {
    String frase = "La lluvia en Madrid es una maravilla";
    String fraseModificada = frase.replace('a', 'e');

    System.out.println("Frase original: " + frase);
    System.out.println("Frase modificada: " + fraseModificada);
}

```



Ejercicio 24.

Recorre el String del ejercicio 22 y transforma cada carácter a su código ASCII. Muéstralos en línea recta, separados por un espacio entre cada carácter.

```
public static void main(String[] args) {
    String frase = "La lluvia en Madrid es una maravilla";

    for (int i = 0; i < frase.length(); i++) {
        char caracter = frase.charAt(i);
        int codigoASCII = (int) caracter;
        System.out.print(codigoASCII + " ");
    }
}
```

Ejercicio 25.

Crea una aplicación llamada CalculadoraInversa, nos pedirá 2 operandos (int) y un signo aritmético (String), según este último se realizará la operación correspondiente. Al final mostrara el resultado en un cuadro de dialogo. Los signos aritméticos disponibles son:

- + : suma los dos operandos.
- - : resta los operandos.
- * : multiplica los operandos.
- / : divide los operandos, este debe dar un resultado con decimales (double)
- ^ : 1º operando como base y 2º como exponente.
- % : módulo, resto de la división entre operando1 y operando2.

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

    Scanner entradaTeclado = new Scanner(System.in);
    int operando1, operando2;
    String signoAritmetico;

    System.out.print("Introduce el primer operando: ");
    operando1 = entradaTeclado.nextInt();
    System.out.print("Introduce el segundo operando: ");
    operando2 = entradaTeclado.nextInt();
    entradaTeclado.nextLine(); // Consumir la línea restante
    System.out.print("Introduce el signo aritmético (+, -, *, /, ^, %): ");
    signoAritmetico = entradaTeclado.nextLine();

    double resultado = 0;
    boolean operacionValida = true;

    switch (signoAritmetico) {
        case "+":
            resultado = operando1 + operando2;
            break;
        case "-":
            resultado = operando1 - operando2;
            break;
        case "*":
            resultado = operando1 * operando2;
            break;
        case "/":
            resultado = (double) operando1 / operando2;
            break;
        case "^":
            resultado = Math.pow(operando1, operando2);
            break;
        case "%":

```

```

        resultado = operando1 % operando2;
        break;
    default:
        operacionValida = false;
        break;
    }

    if (operacionValida) {
        System.out.println("El resultado de la operación es: " + resultado);
    } else {
        System.out.println("El signo aritmético ingresado no es válido.");
    }

    entradaTeclado.close();
}

```

Ejercicio 26.

Realizar la suma del 1 al número que indiquemos, este debe ser mayor que 1.

```

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int numero = 0;

    System.out.print("Introduce un número mayor que 1: ");
    numero = scanner.nextInt();

    if(numero <= 1) {
        System.out.println("El número introducido es menor o igual a 1");
    } else {
        int suma = 0;

        for (int i = 1; i <= numero; i++) {
            suma += i;
        }

        System.out.println("La suma de los números del 1 al " + numero + " es: " + suma);
    }

    scanner.close();
}

```

Ejercicio 27.

Crear una aplicación que nos permite insertar números hasta que insertemos un -1. Calcular el número de números introducidos.

```

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int numero = 0;
    int contador = 0;

    System.out.println("Introduce números. Para terminar, introduce -1.");

    while (numero != -1) {
        System.out.print("Introduce un número: ");
        numero = scanner.nextInt();
        if (numero != -1) {
            contador++;
        }
    }

    System.out.println("Has introducido " + contador + " números.");

    scanner.close();
}

```



Ejercicio 28.

Eliminar los espacios de una frase pasada por consola por el usuario.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce una frase: ");
    String frase = scanner.nextLine();

    String fraseSinEspacios = frase.replace(" ", "");

    System.out.println("Frase original: " + frase);
    System.out.println("Frase sin espacios: " + fraseSinEspacios);

    scanner.close();
}
```

Ejercicio 29.

Pedir al usuario que nos escriba frases de forma infinita hasta que insertemos una cadena vacía. Mostrar la cadena resultante.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    StringBuilder frasesConcatenadas = new StringBuilder();

    System.out.println("Escribe frases. Para terminar, introduce una cadena vacía.");

    while (true) {
        System.out.print("Introduce una frase: ");
        String frase = scanner.nextLine();
        if (frase.isEmpty()) {
            break;
        }

        frasesConcatenadas.append(frase);
    }

    System.out.println("La cadena resultante es: " + frasesConcatenadas.toString());

    scanner.close();
}
```

Ejercicio 30.

Convertir una frase a mayúsculas o minúsculas, que daremos opción a que el usuario lo pida y mostraremos el resultado por pantalla.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce una frase: ");
    String frase = scanner.nextLine();

    System.out.println("Elige una opción:");
    System.out.println("1. Convertir a mayúsculas");
    System.out.println("2. Convertir a minúsculas");

    int opcion = scanner.nextInt();

    String fraseConvertida;

    switch (opcion) {
        case 1:
            fraseConvertida = frase.toUpperCase();
    }
```

```
        System.out.println("Frase en mayúsculas: " + fraseConvertida);
        break;
    case 2:
        fraseConvertida = frase.toLowerCase();
        System.out.println("Frase en minúsculas: " + fraseConvertida);
        break;
    default:
        System.out.println("Opción no válida.");
        break;
}

scanner.close();
}
```

Ejercicio 31.

Mostrar la longitud de una cadena.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce una cadena: ");
    String cadena = scanner.nextLine();

    int longitud = cadena.length();
    System.out.println("La longitud de la cadena es: " + longitud);

    scanner.close();
}
```

Ejercicio 32.

Pedir dos palabras por teclado, indicar si son iguales.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce la primera palabra: ");
    String palabra1 = scanner.nextLine();

    System.out.print("Introduce la segunda palabra: ");
    String palabra2 = scanner.nextLine();

    if (palabra1.equals(palabra2)) {
        System.out.println("Las palabras son iguales.");
    } else {
        System.out.println("Las palabras no son iguales.");
    }

    scanner.close();
}
```

Ejercicio 33.

Dada una cadena, extraer la cuarta y quinta letra usando el método substring.

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce una cadena (mínimo 5 caracteres): ");
    String cadena = scanner.nextLine();

    if (cadena.length() >= 5) {
        String letras = cadena.substring(3, 5);
        System.out.println("La cuarta y quinta letra de la cadena son: " + letras);
    } else {
        System.out.println("La cadena debe tener al menos 5 caracteres.");
    }
}
```

```
    }  
    scanner.close();  
}
```

Ejercicio 34.

Dada una frase, separarlo en palabras.

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.print("Introduce una frase: ");  
    String frase = scanner.nextLine();  
  
    String[] palabras = frase.split(" ");  
  
    System.out.println("Las palabras en la frase son:");  
    for (String palabra : palabras) {  
        System.out.println(palabra);  
    }  
  
    scanner.close();  
}
```

Ejercicio 35.

Crea un enum con los días de la semana, pide un día de la semana e indica si es laboral o no (en el main).

```
enum Dia {  
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.print("Introduce un día de la semana: ");  
    String entrada = scanner.nextLine().toUpperCase();  
  
    Dia dia = Dia.valueOf(entrada);  
  
    switch (dia) {  
        case SABADO:  
        case DOMINGO:  
            System.out.println("No es un día laboral.");  
            break;  
        default:  
            System.out.println("Es un día laboral.");  
            break;  
    }  
  
    scanner.close();  
}
```




Ejercicio 36.

Modifica el anterior enum para indicar que es día laborable directamente (usar toString).

```
enum Dia {
    LUNES("Laborable"), MARTES("Laborable"), MIERCOLES("Laborable"), JUEVES("Laborable"),
    VIERNES("Laborable"), SABADO("No laborable"), DOMINGO("No laborable");

    private final String tipoDia;

    // Constructor del enum. Permite crea el elemento concreto y su valor (Laborable
    // o No laborable)
    Dia(String tipoDia) {
        this.tipoDia = tipoDia;
    }

    // Método para obtener el valor (Laborable o No laborable)
    public String esLaborable() {
        return this.tipoDia;
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Introduce un día de la semana: ");
    String entrada = scanner.nextLine().toUpperCase();

    Dia dia = Dia.valueOf(entrada);

    System.out.println("El día " + dia.toString() + " es " + dia.esLaborable());

    scanner.close();
}
```

Ejercicio 37.

Crea el enum Mes, que contenga como parámetros el orden(1,2,3,etc)y el número de días (febrero tendrá 28 días siempre).Estos datos pueden pedirse por separado, así que tienes que hacer sus respectivos get. No son necesarios los setters. Crear un array de Mes (mírate la función values), pide un numero por teclado e indica que meses tienen ese número de días (toda su información).Por ejemplo, si escribes un 28, este te devolverá la información de FEBRERO.

```
enum Mes {
    ENERO(1, 31), FEBRERO(2, 28), MARZO(3, 31), ABRIL(4, 30), MAYO(5, 31), JUNIO(6, 30),
    JULIO(7, 31), AGOSTO(8, 31), SEPTIEMBRE(9, 30), OCTUBRE(10, 31), NOVIEMBRE(11, 30),
    DICIEMBRE(12, 31);

    private final int orden;
    private final int dias;

    Mes(int orden, int dias) {
        this.orden = orden;
        this.dias = dias;
    }

    public int getOrden() {
        return orden;
    }

    public int getDias() {
        return dias;
    }
}
```

```

        public String toString() {
            return this.name() + ": " + this.orden + " - " + this.dias + " días";
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Mes[] meses = Mes.values();

        System.out.print("Introduce un número de días: ");
        int numDias = scanner.nextInt();

        for (Mes mes : meses) {
            if (mes.getDias() == numDias) {
                System.out.println(mes.toString());
            }
        }

        scanner.close();
    }

```

Ejercicio 38.

Pedir números al usuario y cuando el usuario meta un -1 se terminará el programa. Al terminar, mostrará lo siguiente:

- mayor numero introducido
- menor número introducido
- suma de todos los números
- suma de los números positivos
- suma de los números negativos
- media de la suma (la primera que pido) El número -1 no contara como número.

```

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        double numero, mayor = 0, menor = 0, suma = 0, sumaPositivos = 0, sumaNegativos = 0;
        int contador = 0;
        double media;

        do {
            System.out.print("Introduce un número, -1 para finalizar: ");
            numero = scanner.nextDouble();

            if (numero != -1) {
                suma += numero;
                contador++;

                if (numero > mayor) {
                    mayor = numero;
                }

                if (numero < menor) {
                    menor = numero;
                }

                if (numero > 0) {
                    sumaPositivos += numero;
                } else {
                    sumaNegativos += numero;
                }
            }
        } while (numero != -1);
    }

```



```
        if (contador > 0) {
            media = (double) suma / contador;
        } else {
            media = 0;
        }

        System.out.println("El mayor número introducido es: " + mayor);
        System.out.println("El menor número introducido es: " + menor);
        System.out.println("La suma total de los números es: " + suma);
        System.out.println("La suma de los números positivos es: " + sumaPositivos);
        System.out.println("La suma de los números negativos es: " + sumaNegativos);
        System.out.println("La media de la suma de los números es: " + media);

        scanner.close();
    }
}
```

Ejercicio 39.

Realiza un reloj digital que muestre la hora sin parar.

```
public static void main(String[] args) {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");

    while (true) {
        LocalDateTime currentTime = LocalDateTime.now();
        String timeString = currentTime.format(formatter);
        System.out.print("\r" + timeString);

        // Esta parte permite dormir la aplicacion 1 segundo
        // Si no estaría imprimiendo por consola constantemente.
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Unidad 6

Ejercicio 1.

Realiza una clase llamada calculadora, añade los campos numero1 y numero2. Estos campos se deben poder consultar y modificar. Determinar cada acción (Suma, Resta, Producto y División) de la calculadora sobre los campos. No se pide hacer un método general con todas las operaciones, si no, uno por cada acción.

```
public class Calculadora {
    private double numero1;
    private double numero2;

    public double getNumero1() {
        return numero1;
    }

    public void setNumero1(double numero1) {
        this.numero1 = numero1;
    }

    public double getNumero2() {
        return numero2;
    }

    public void setNumero2(double numero2) {
        this.numero2 = numero2;
    }

    public double sumar() {
        return numero1+numero2;
    }

    public double restar() {
        return numero1-numero2;
    }

    public double multiplicar() {
        return numero1*numero2;
    }

    public double dividir() {
        return numero1/numero2;
    }
}
```

```
public class Ej01 {
    public static void main(String[] args) {

        Calculadora calculadora = new Calculadora();

        calculadora.setNumero1(10);
        calculadora.setNumero2(8);

        System.out.println("La calculadora tiene como numero1 : "+calculadora.getNumero1());
        System.out.println("La calculadora tiene como numero2 : "+calculadora.getNumero2());
        System.out.println("El resultado de sumar es: "+calculadora.sumar());
        System.out.println("El resultado de restar es: "+calculadora.restar());
        System.out.println("El resultado de multiplicar es: "+calculadora.multiplicar());
        System.out.println("El resultado de dividir es: "+calculadora.dividir());

    }
}
```

Ejercicio 2.

Realiza una clase que se llame `EcuacionPrimerGrado`, debe disponer un campo para almacenar la ecuación. Además, debe poder realizar el cálculo de la ecuación.

```
public class EcuacionPrimerGrado {  
  
    private double a;  
    private double b;  
  
    public double getA() {  
        return a;  
    }  
  
    public void setA(double a) {  
        this.a = a;  
    }  
  
    public double getB() {  
        return b;  
    }  
  
    public void setB(double b) {  
        this.b = b;  
    }  
  
    public double calcularSolucion() {  
  
        double solucion = Double.POSITIVE_INFINITY;  
  
        if (a != 0) {  
            solucion = -b / a;  
        }  
  
        return solucion;  
    }  
  
    // Campo calculado  
    public String getEcuacion() {  
        return a + "x + " + b + " = 0";  
    }  
}
```

```
public class Ej02 {  
  
    public static void main(String[] args) {  
        EcuacionPrimerGrado ecuacionPrimerGrado = new EcuacionPrimerGrado();  
        ecuacionPrimerGrado.setA(3);  
        ecuacionPrimerGrado.setB(5);  
        System.out.println("Ecuación: " + ecuacionPrimerGrado.getEcuacion());  
        System.out.println("Solución: x = " + ecuacionPrimerGrado.calcularSolucion());  
    }  
}
```

Ejercicio 3.

Realiza una clase que se llame `EcuacionSegundoGrado` con los campos `a`, `b` y `c` e incluya una función que nos calcule la ecuación. Se debe de poder crear el objeto con los valores de `a`, `b` y `c` sin utilizar los métodos `set` de cada campo. Nota. Para devolver más de un valor se debe emplear una `Array`.

```
public class EcuacionSegundoGrado {

    private double a;
    private double b;
    private double c;

    public EcuacionSegundoGrado(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public double getA() {
        return a;
    }

    public void setA(double a) {
        this.a = a;
    }

    public double getB() {
        return b;
    }

    public void setB(double b) {
        this.b = b;
    }

    public double getC() {
        return c;
    }

    public void setC(double c) {
        this.c = c;
    }

    public double[] calcularSoluciones() {

        double discriminante = Math.pow(b, 2) - 4 * a * c;

        if (discriminante > 0) {
            double solucion1 = (-b + Math.sqrt(discriminante)) / (2 * a);
            double solucion2 = (-b - Math.sqrt(discriminante)) / (2 * a);
            return new double[]{solucion1, solucion2}; // Dos soluciones reales
        } else if (discriminante == 0) {
            double solucion = -b / (2 * a);
            return new double[]{solucion}; // Una solución real
        } else {
            return null; // No hay soluciones reales
        }
    }

    public String getEcuacion() {
        return a + "x^2 + " + b + "x + " + c + " = 0";
    }
}
```

```

public class Ej03 {

    public static void main(String[] args) {
        EcuacionSegundoGrado ecuacion2g = new EcuacionSegundoGrado(1, -3, 2);
        System.out.println("Ecuación: " + ecuacion2g.getEcuacion());

        double[] soluciones = ecuacion2g.calcularSoluciones();
        if (soluciones == null) {
            System.out.println("No hay soluciones reales.");
        } else {
            System.out.print("Soluciones: ");
            for (int i = 0; i < soluciones.length; i++) {
                System.out.print("x" + (i + 1) + " = " + soluciones[i]);
                if (i < soluciones.length - 1) {
                    System.out.print(", ");
                }
            }
            System.out.println();
        }
    }
}

```

Ejercicio 4.

Realiza una clase llamada reloj para poder gestionar la fecha y hora. El reloj es completamente modificable, con lo cual, se debe poder de establecer la fecha y hora, además modificarla. Aparte debe de tener la funcionalidad de devolver la fecha según un formato concreto por ejemplo dd/MM/yyyy, yyyy/MM/dd o cualquier otro. Y disponer de la funcionalidad de un cronometro, poder arrancarlo y pararlo cuando se quiera.

```

import java.text.SimpleDateFormat;
import java.util.Date;

public class Reloj {

    private Date fechaHora;
    private long inicioCronometro;
    private long paradaCronometro;

    public Reloj() {
        fechaHora = new Date();
    }

    public void establecerFechaHora(Date nuevaFechaHora) {
        fechaHora = nuevaFechaHora;
    }

    public void modificarFechaHora(int anos, int meses, int dias, int horas, int minutos, int segundos) {
        long tiempoActual = fechaHora.getTime();
        long tiempoModificar = (((anos * 365L) + meses * 30L + dias) * 24L + horas) * 60L + minutos * 60L + segundos;
        tiempoModificar *= 1000;
        fechaHora.setTime(tiempoActual + tiempoModificar);
    }

    public String obtenerFechaHora(String formato) {
        SimpleDateFormat formateador = new SimpleDateFormat(formato);
        return formateador.format(fechaHora);
    }

    public void iniciarCronometro() {
        inicioCronometro = System.currentTimeMillis();
    }
}

```

```
public void pararCronometro() {
    paradaCronometro = System.currentTimeMillis();
}

public String obtenerTiempoCronometrado() {

    long tiempo = paradaCronometro - inicioCronometro;;

    int segundos = (int) (tiempo / 1000) % 60 ;
    int minutos = (int) ((tiempo / (1000*60)) % 60);
    int horas = (int) ((tiempo / (1000*60*60)) % 24);

    return horas + "h:" + minutos + "m:" + segundos + "s";
}
}

public class Ej04 {

    public static void main(String[] args) {

        Reloj reloj = new Reloj();

        System.out.println("La fecha que tiene el reloj es : " +
            reloj.obtenerFechaHora("dd/MM/yyyy"));
        System.out.println("La hora que tiene el reloj es : " +
            reloj.obtenerFechaHora("hh:mm:ss"));

        System.out.println("Iniciamos el cronómetro");
        reloj.iniciarCronometro();
        //Esperamos 15 segundo para ver que funciona el cronometro
        try {
            Thread.sleep(15000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Paramos el cronómetro");
        reloj.pararCronometro();

        System.out.println("El tiempo cronometrado ha sido: " +
            reloj.obtenerTiempoCronometrado());
    }
}
```


Ejercicio 5.

Realiza una clase que refleje la información de una cuenta bancaria. Debe de tener un número de cuenta y poder gestionar el titular, la cantidad y la última operación realizada. Además, debe tener las siguientes funcionalidades:

- Debe poder consultar el saldo actual.
- Debe de poder ingresar dinero.
- Debe de poder retirar dinero. La función tiene que verificar si ha sido posible la retirada, si no hay saldo suficiente no debe permitir la operación.
- Debe de poder realizar transferencias hacia otra cuenta.

```
public class CuentaBancaria {
    private String numeroCuenta;
    private String titular;
    private double saldo;
    private String ultimaOperacion;

    public CuentaBancaria(String numeroCuenta, String titular) {
        this.numeroCuenta = numeroCuenta;
        this.titular = titular;
        this.saldo = 0.0;
    }

    public String getNumeroCuenta() {
        return numeroCuenta;
    }

    public void setNumeroCuenta(String numeroCuenta) {
        this.numeroCuenta = numeroCuenta;
    }

    public String getTitular() {
        return titular;
    }

    public void setTitular(String titular) {
        this.titular = titular;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public void ingresar(double cantidad) {
        System.out.println("Ingresar: " + cantidad);
        ultimaOperacion = "Ingreso: " + cantidad;
        saldo += cantidad;
    }

    public boolean retirar(double cantidad) {
        System.out.println("Retirar: " + cantidad);

        ultimaOperacion = "Retirada: " + cantidad;
        if (saldo >= cantidad) {
            saldo -= cantidad;
            return true;
        }
        return false;
    }
}
```



```

    public boolean transferir(CuentaBancaria cuentaDestino, double cantidad) {
        System.out.println("Transferir: " + cantidad);
        ultimaOperacion = "Transferencia a " + cuentaDestino.getNumeroCuenta() + ": " +
            cantidad;
        if (retirar(cantidad)) {
            cuentaDestino.ingresar(cantidad);
            return true;
        }

        return false;
    }

    public String consultarUltimaOperacion() {
        return ultimaOperacion;
    }
}

```

```

public class Ej05 {

    public static void main(String[] args) {
        CuentaBancaria cuenta1 = new CuentaBancaria("222233344123456", "Juan Pérez");
        CuentaBancaria cuenta2 = new CuentaBancaria("555444556654321", "María García");

        cuenta1.ingresar(1000);
        cuenta1.retirar(200);
        System.out.println("Saldo actual : " + cuenta1.getSaldo());

        if (!cuenta1.retirar(5000)) {
            System.out.println("Error: No se ha podido retirar el importe de 5000");
        }

        System.out.println("La última operación ha sido " +
            cuenta1.consultarUltimaOperacion());

        cuenta1.transferir(cuenta2, 300);
        System.out.println("La última operación ha sido " +
            cuenta1.consultarUltimaOperacion());
    }
}

```

Ejercicio de la videoclase

Queremos tener una aplicación destinada a la matriculación de alumnos en un centro educativo. Un centro educativo está formado por aulas, alumnos, docentes y compuesto por cursos.

Diseña las clases que veas oportunas, teniendo en cuenta sus campos y métodos. Una vez finalizado mira de construir el centro educativo y que tenga su lógica.

```

public class Alumno {
    private String nombre;
    private String dni;
    private Curso curso;

    public Alumno(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
        this.curso = null;
    }

    // Métodos getter y setter
}

```

```
public class Aula {
    private String nombre;
    private int capacidad;

    public Aula(String nombre, int capacidad) {
        this.nombre = nombre;
        this.capacidad = capacidad;
    }

    // Métodos getter y setter
}
```

```
public class Docente {
    private String nombre;
    private String dni;
    //Estructura de colección permite agrupar varios objetos del tipo Curso
    private List<Curso> cursos;

    public Docente(String nombre, String dni) {
        this.nombre = nombre;
        this.dni = dni;
        this.cursos = new ArrayList<>();
    }

    // Métodos para agregar y eliminar cursos asignados al docente
    public void asignarCurso(Curso curso) {
        // Accion para añadir un curso al docente
    }
}
```

```
public class Curso {

    private String nombre;
    //Estructura de colección permite agrupar varios objetos del tipo Alumnos
    private List<Alumno> alumnos;
    private Docente docente;
    private Aula aula;

    public Curso(String nombre, Docente docente, Aula aula) {
        this.nombre = nombre;
        this.docente = docente;
        this.aula = aula;
        this.alumnos = new ArrayList<>();
    }

    public void añadirAlumno(Alumno alumno) {
        // Accion para añadir un alumno
    }

    public void añadirDocente(Docente docente) {
        // Accion para añadir un docente al curso
    }

    public void añadirAula(Aula aula) {
        // Accion para añadir un aula al curso
    }

    // Otros metodos ...
}
```

```
public class CentroEducativo {
    private String nombre;
    //Estructura de colección permite agrupar varios objetos del tipo Aula
    private List<Aula> aulas;
    //Estructura de colección permite agrupar varios objetos del tipo Alumno
    private List<Alumno> alumnos;
    //Estructura de colección permite agrupar varios objetos del tipo Docente
    private List<Docente> docentes;
    //Estructura de colección permite agrupar varios objetos del tipo Curso
}
```

```

private List<Curso> cursos;

public CentroEducativo(String nombre) {
    this.nombre = nombre;
    this.aulas = new ArrayList<>();
    this.alumnos = new ArrayList<>();
    this.docentes = new ArrayList<>();
    this.cursos = new ArrayList<>();
}

public void añadirAlumno(Alumno alumno) {
    // Accion para añadir alumnos al centro educativo
}

public void añadirDocente(Docente docente) {
    // Accion para añadir docentes al centro educativo
}

public void añadirAula(Aula aula) {
    // Accion para añadir una aula al centro educativo
}

public void añadirCurso(Curso curso) {
    // Accion para añadir un curso al centro educativo
}

// Otros métodos ...
}

```

```

public class Test {
    public static void main(String[] args) {

        Alumno alumno1 = new Alumno("Juan Pérez", "12345678A");
        Alumno alumno2 = new Alumno("Ana Sánchez", "23456789B");
        Aula aula1 = new Aula("Aula 101", 30);
        Docente docente1 = new Docente("Carlos García", "34567890C");

        CentroEducativo centroEducativo = new CentroEducativo("Instituto de Enseñanza");

        Curso curso1 = new Curso("Matemáticas", docente1, aula1);

        curso1.añadirAlumno(alumno1);
        curso1.añadirAlumno(alumno2);

        docente1.asignarCurso(curso1);

        centroEducativo.añadirAula(aula1);
        centroEducativo.añadirAlumno(alumno1);
        centroEducativo.añadirAlumno(alumno2);
        centroEducativo.añadirDocente(docente1);
        centroEducativo.añadirCurso(curso1);

    }
}

```

Unidad 7

Ejercicio 1.

Desarrolla una aplicación que tenga una representación de un sistema de gestión de inventario para un almacén de componentes informáticos.

Deberás crear una clase base llamada "Componente" que tiene tres atributos: referencia (tipo long), marca (tipo String) y modelo (tipo String). Deberás incluir en esta clase un constructor que inicialice estos tres atributos, métodos de acceso (getters y setters) para cada uno de ellos, y el método toString().

Además, se te pide que desarrolles dos clases que heredan de la clase "Componente", estas clases son "DiscoDuro" y "Microprocesador".

La clase "DiscoDuro" debe tener un atributo adicional: capacidad (tipo int). Deberás incluir un constructor en esta clase que inicialice los cuatro atributos (los tres de la clase "Componente" y el nuevo atributo), métodos de acceso para este nuevo atributo y el método toString().

La clase "Microprocesador" debe tener un atributo adicional: frecuencia (tipo int). Deberás incluir un constructor en esta clase que inicialice los cuatro atributos (los tres de la clase "Componente" y el nuevo atributo), métodos de acceso para este nuevo atributo, y el método toString().

Finalmente, crea una clase principal denominada "Ej1" que contenga un método main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

```
public class Componente {  
  
    private long referencia;  
    private String marca, modelo;  
  
    public Componente(long referencia, String marca, String modelo){  
        this.referencia = referencia;  
        this.marca = marca;  
        this.modelo = modelo;  
    }  
  
    public long getReferencia() {  
        return referencia;  
    }  
  
    public void setReferencia(long referencia) {  
        this.referencia = referencia;  
    }  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String marca) {  
        this.marca = marca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

```

    }

    @Override
    public String toString() {
        return "Componente [referencia=" + referencia + ", marca=" + marca + ", modelo="
+ modelo + "]\n";
    }
}

```

```

public class DiscoDuro extends Componente {

    private int capacidad;

    public DiscoDuro(long referencia, String modelo, String marca, int capacidad){
        super(referencia, marca, modelo);
        this.capacidad = capacidad;
    }

    public int getCapacidad() {
        return capacidad;
    }

    public void setCapacidad(int capacidad) {
        this.capacidad = capacidad;
    }

    @Override
    public String toString() {
        return "DiscoDuro [referencia=" + getReferencia() + ", marca=" + getMarca() + ",
        modelo=" + getModelo() + ", capacidad=" + capacidad + "]\n";
    }
}

```

```

public class Microprocesador extends Componente {

    private int frecuencia;

    public Microprocesador(long referencia, String modelo, String marca, int frecuencia){
        super(referencia, marca, modelo);
        this.frecuencia = frecuencia;
    }

    public int getFrecuencia() {
        return frecuencia;
    }

    public void setFrecuencia(int frecuencia) {
        this.frecuencia = frecuencia;
    }

    @Override
    public String toString() {
        return "Microprocesador [referencia=" + getReferencia() + ", marca=" +
        getMarca()+ " , modelo=" + getModelo() + ", frecuencia=" +
        frecuencia + "]\n";
    }
}

```

```

public class Ej1 {
    public static void main(String[] args) {
        Microprocesador micro = new Microprocesador(123,"Intel","i3",2300);
        DiscoDuro discoduro = new DiscoDuro(456,"Seagate","Barracuda",500);

        System.out.println(micro);
        System.out.println(discoduro);
    }
}

```



Ejercicio 2.

Realiza una aplicación para gestionar vehículos que permite saber si están arrancado o no. Para ello crea una clase "Vehículo" que tenga los atributos marca, modelo, precio y estaArrancado. Esta clase debe tener:

- Un constructor que tome tres argumentos para inicializar estos valores y estaArrancado como false.
- Un método arrancar() y parar() que cambie el estado de estaArrancado y que imprima si el vehículo está arrancado o apagado. Estos métodos deben ser capaz de reconocer que clase hija le llama para mostrar un mensaje diferente, según el tipo de vehículo. Por ejemplo, "El [coche/moto/camión/etc..] está arrancando o parado"
- El método toString().

Crea dos subclases: Coche y Moto. El coche solo tiene un sistema de arranque eléctrico, en cambio, la moto puede ser arrancada manualmente o de forma eléctrica, la lógica de la moto debe ser capaz de distinguir estos dos tipos de arranque.

Ambas clases deben tener:

- Un constructor que tome los mismos atributos que tiene la clase Vehículo.
- Métodos arrancar() según los sistemas de arranque que tengan. Deben llamar al método arrancar() de Vehículo y mostrar un mensaje inicial con el mensaje "(El coche/La moto) ha utilizado el encendido (manual/eléctrico)".
- Un método parar(). Debe llamar al método parar() de Vehículo.
- El método toString().

Finalmente, crea una clase principal denominada "Ej2" que contenga un método main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

```
public class Vehiculo {  
  
    private String marca;  
    private String modelo;  
    private double precio;  
    private boolean estaArrancado;  
  
    public Vehiculo(String marca, String modelo, double precio) {  
        this.marca = marca;  
        this.modelo = modelo;  
        this.precio = precio;  
        this.estaArrancado = false;  
    }  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public void setMarca(String marca) {  
        this.marca = marca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

```

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public boolean isEstaArrancado() {
        return estaArrancado;
    }

    public void setEstaArrancado(boolean estaArrancado) {
        this.estaArrancado = estaArrancado;
    }

    public void arrancar() {
        this.estaArrancado = true;

        /* Con el polimorfismo podemos conocer quien llama al método
           this representa a la instancia actual y mediante instanceof
           podemos saber que clase es */
        if(this instanceof Coche) {
            System.out.println("El coche está arrancando.");
        } else if (this instanceof Moto) {
            System.out.println("La moto está arrancando.");
        } else {
            System.out.println("El vehiculo está arrancando.");
        }
    }

    public void parar() {
        this.estaArrancado = false;

        /* Con el polimorfismo podemos conocer quien llama al método
           this representa a la instancia actual y mediante instanceof
           podemos saber que clase es */
        if(this instanceof Coche) {
            System.out.println("El coche está parado.");
        } else if (this instanceof Moto) {
            System.out.println("La moto está parada.");
        } else {
            System.out.println("El vehiculo está parado.");
        }
    }

    @Override
    public String toString() {
        return "Vehiculo [marca=" + marca + ", modelo=" + modelo + ", precio=" + precio +
            ", estaArrancado=" + estaArrancado + "];"
    }
}

```

```

public class Coche extends Vehiculo {
    public Coche(String marca, String modelo, double precio) {
        super(marca, modelo, precio);
    }

    @Override
    public void arrancar() {
        super.arrancar();
        System.out.print("El coche ha utilizado encendido electrónico. ");
    }

    @Override
    public void parar() {

```




```

        super.parar();
        System.out.println("El coche ha sido parado");
    }

    @Override
    public String toString() {
        return "Coche [marca=" + getMarca() + ", modelo=" + getModelo() + ", precio=" +
            getPrecio() + ", estaArrancado=" + isEstaArrancado() + "]";
    }
}

```

```

public class Moto extends Vehiculo {

    public Moto(String marca, String modelo, double precio) {
        super(marca, modelo, precio);
    }

    @Override
    public void arrancar() {
        super.arrancar();
        System.out.print("La moto ha utilizado el encendido manual. ");
    }

    public void arrancarElectrico() {
        super.arrancar();
        System.out.print("La moto ha utilizado el encendido eléctrico. ");
    }

    @Override
    public void parar() {
        super.parar();
        System.out.println("La moto ha sido parada.");
    }

    @Override
    public String toString() {
        return "Moto [marca=" + getMarca() + ", modelo=" + getModelo() + ", precio=" +
            getPrecio() + ", estaArrancado=" + isEstaArrancado() + "]";
    }
}

```

```

public class Ej2 {
    public static void main(String[] args) {

        Coche coche = new Coche("Ford", "Mustang", 35000.0);
        coche.arrancar();
        coche.parar();

        System.out.println();

        Moto moto = new Moto("Honda", "CBR", 15000.0);
        moto.arrancar();
        moto.parar();

        System.out.println();

        moto.arrancarElectrico();

        System.out.println("Se muestra el estado de los objetos");
        System.out.println(coche);
        System.out.println(moto);
    }
}

```

Ejercicio 3.

Desarrolla una aplicación que permite calcular el área y perímetro de diferentes figuras geométricas para ello se pide crear una interfaz llamada "FiguraGeometrica" que declare los siguientes métodos: "calcularArea()" y "calcularPerimetro()".

Crea tres clases que implementen esta interfaz: "Circulo", "Rectangulo" y "Triangulo". Cada clase debe tener los atributos necesarios para poder calcular el área y el perímetro (por ejemplo, un círculo necesita el radio, un rectángulo necesita largo y ancho, y un triángulo necesita base y altura para el área, y los tres lados para el perímetro).

Implementa los métodos calcularArea() y calcularPerimetro() en cada clase de forma acorde a la figura que representan.

Finalmente, crea una clase denominada "Ej3" que tenga la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

```
public interface FiguraGeometrica {  
    /**  
     * Función para calcular un área de una figura  
     * @return  
     */  
    public double calcularArea();  
  
    /**  
     * Función para calcular un perímetro de una figura  
     * @return  
     */  
    public double calcularPerimetro();  
}
```

```
public class Circulo implements FiguraGeometrica {  
    private double radio;  
  
    public Circulo(double radio) {  
        this.radio = radio;  
    }  
  
    public double calcularArea() {  
        return Math.PI * Math.pow(radio, 2);  
    }  
  
    public double calcularPerimetro() {  
        return 2 * Math.PI * radio;  
    }  
}
```

```
public class Rectangulo implements FiguraGeometrica {  
    private double ancho;  
    private double alto;  
  
    public Rectangulo(double ancho, double alto) {  
        this.ancho = ancho;  
        this.alto = alto;  
    }  
  
    public double calcularArea() {  
        return ancho * alto;  
    }  
  
    public double calcularPerimetro() {  
        return 2 * (ancho + alto);  
    }  
}
```

```
}

```

```
public class Triangulo implements FiguraGeometrica {
    private double base;
    private double altura;
    private double lado1;
    private double lado2;

    public Triangulo(double base, double altura, double lado1, double lado2) {
        this.base = base;
        this.altura = altura;
        this.lado1 = lado1;
        this.lado2 = lado2;
    }

    public double calcularArea() {
        return (base * altura) / 2;
    }

    public double calcularPerimetro() {
        return base + lado1 + lado2;
    }
}
```

```
public class Ej3 {

    public static void main(String[] args) {
        FiguraGeometrica circulo = new Circulo(5);
        FiguraGeometrica rectangulo = new Rectangulo(4, 5);
        FiguraGeometrica triangulo = new Triangulo(3, 4, 5, 6);

        System.out.println("Área del círculo: " + circulo.calcularArea());
        System.out.println("Perímetro del círculo: " + circulo.calcularPerimetro());

        System.out.println("Área del rectángulo: " + rectangulo.calcularArea());
        System.out.println("Perímetro del rectángulo: " +
            rectangulo.calcularPerimetro());

        System.out.println("Área del triángulo: " + triangulo.calcularArea());
        System.out.println("Perímetro del triángulo: " + triangulo.calcularPerimetro());
    }
}
```

Ejercicio 4.

Desarrolla una aplicación que permita gestionar instrumentos musicales. Crear una interfaz llamada "InstrumentoMusical" que declare los siguientes métodos: "tocar()", "afinar()" y "tipoInstrumento()".

Crea tres clases que implementen esta interfaz: "Guitarra", "Piano" y "Trompeta". Cada una de estas clases debe implementar los métodos de la interfaz de la siguiente manera:

- **tocar():** debe imprimir un mensaje que diga "Estoy tocando [instrumento]".
- **afinar():** debe imprimir un mensaje que diga "Estoy afinando [instrumento]".
- **tipoInstrumento():** debe retornar una cadena de texto que describa el tipo de instrumento (puedes ser creativo aquí, o simplemente retornar el nombre del instrumento).

Crea una clase denominada "Ej4" con la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

```
public interface InstrumentoMusical {  
  
    void tocar();  
    void afinar();  
    String tipoInstrumento();  
  
}
```

```
public class Guitarra implements InstrumentoMusical {  
  
    public void tocar() {  
        System.out.println("Estoy tocando la guitarra");  
    }  
  
    public void afinar() {  
        System.out.println("Estoy afinando la guitarra");  
    }  
  
    public String tipoInstrumento() {  
        return "Guitarra";  
    }  
  
}
```

```
public class Piano implements InstrumentoMusical {  
  
    public void tocar() {  
        System.out.println("Estoy tocando el piano");  
    }  
  
    public void afinar() {  
        System.out.println("Estoy afinando el piano");  
    }  
  
    public String tipoInstrumento() {  
        return "Piano";  
    }  
  
}
```

```
public class Trompeta implements InstrumentoMusical {  
  
    public void tocar() {  
        System.out.println("Estoy tocando la trompeta");  
    }  
  
    public void afinar() {  
        System.out.println("Estoy afinando la trompeta");  
    }  
  
    public String tipoInstrumento() {  
        return "Trompeta";  
    }  
  
}
```

```
public class Ej4 {  
  
    public static void main(String[] args) {  
        InstrumentoMusical guitarra = new Guitarra();  
        InstrumentoMusical piano = new Piano();  
        InstrumentoMusical trompeta = new Trompeta();  
  
        System.out.println("Instrumento: " + guitarra.tipoInstrumento());  
        guitarra.tocar();  
        guitarra.afinar();  
  
        System.out.println("Instrumento: " + piano.tipoInstrumento());  
        piano.tocar();  
        piano.afinar();  
    }  
}
```



```

        System.out.println("Instrumento: " + trompeta.tipoInstrumento());
        trompeta.tocar();
        trompeta.afinar();
    }
}

```

Ejercicio 5.

Desarrolla una aplicación para representar un sistema de archivos básico. En este sistema de archivos, hay dos tipos de objetos con los que puedes interactuar: archivos y directorios.

Cada archivo tiene un nombre y un tamaño en bytes. Cada directorio tiene un nombre y puede contener una cantidad arbitraria de archivos y directorios. Además, cada directorio tiene un tamaño, que se define como la suma del tamaño de todos los archivos y directorios contenidos dentro de él.

Se te pide diseñar este sistema y proporcionar una manera de calcular el tamaño total de un directorio dado. Ten en cuenta que los archivos pueden ser añadidos a un directorio y cada objeto dentro del sistema de archivos tiene un nombre.

Crea una clase denominada "Ej5" con la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

```

public abstract class SistemaFicheros {
    private String nombre;

    // Se crea un constructor que es necesario para las subclases.
    // La clase abstracta no puede instanciar objetos
    public SistemaFicheros(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public abstract int getTamaño();
}

```

```

public class Fichero extends SistemaFicheros {

    private int tamaño;

    public Fichero(String nombre, int tamaño) {
        super(nombre);
        this.tamaño = tamaño;
    }

    @Override
    public int getTamaño() {
        return tamaño;
    }
}

```

```

import java.util.ArrayList;
import java.util.List;

public class Directorio extends SistemaFicheros {

```

```

private List<SistemaFicheros> contenido;

public Directorio(String nombre) {
    super(nombre);
    contenido = new ArrayList<>();
}

public void añadir(SistemaFicheros sistemaFicheros) {
    contenido.add(sistemaFicheros);
}

@Override
public int getTamaño() {
    int tamañoTotal = 0;
    for (SistemaFicheros sistemaFicheros : contenido) {
        tamañoTotal += sistemaFicheros.getTamaño();
    }
    return tamañoTotal;
}
}

```

```

public class Ej5 {
    public static void main(String[] args) {

        Fichero fichero1 = new Fichero("Fichero1", 1000);
        Fichero fichero2 = new Fichero("Fichero2", 2000);

        Directorio directorio1 = new Directorio("Directorio1");
        directorio1.añadir(fichero1);
        directorio1.añadir(fichero2);

        Directorio directorioRaiz = new Directorio("/");
        directorioRaiz.añadir(directorio1);

        Fichero fichero3 = new Fichero("Fichero3", 3000);
        directorioRaiz.añadir(fichero3);

        System.out.println("El tamaño total de " +
            directorioRaiz.getNombre() + ": " + directorioRaiz.getTamaño());
    }
}

```

Ejercicio 6.

Desarrolla una aplicación de un sistema de gestión de productos para una tienda que vende varios tipos de productos: electrónicos, alimentos y ropa. Todos los productos tienen algunas características en común: un código de identificación, un nombre y un precio. Sin embargo, cada tipo de producto tiene diferentes maneras de calcular su precio final para el cliente.

Para los productos electrónicos, se agrega una tarifa de reciclaje al precio; para los alimentos, se agrega un impuesto sobre las ventas; y para la ropa, si es de una determinada marca, se añade un precio adicional.

Crea una clase abstracta **Producto** con las características comunes de todos los productos y un método abstracto **calcularPrecioFinal**. Luego, crea las clases **Electronico**, **Alimento** y **Ropa** que extienden a **Producto** y proporciona una implementación para el método **calcularPrecioFinal** en cada una de estas subclases.

Finalmente, escribe un programa de prueba que cree al menos un objeto de cada subclase, calcule el precio final para cada producto e imprima la información del producto junto con su precio final.

```
public abstract class Producto {
    protected String id;
    protected String nombre;
    protected double precio;

    public Producto(String id, String nombre, double precio) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
    }

    public abstract double calcularPrecioFinal();
}
```

```
public class Electronico extends Producto {
    private static final double TARIFA_RECICLAJE = 10.0;

    public Electronico(String id, String nombre, double precio) {
        super(id, nombre, precio);
    }

    @Override
    public double calcularPrecioFinal() {
        return precio + TARIFA_RECICLAJE;
    }
}
```

```
public class Alimento extends Producto {
    private static final double IMPUESTO_VENTAS = 0.15;

    public Alimento(String id, String nombre, double precio) {
        super(id, nombre, precio);
    }

    @Override
    public double calcularPrecioFinal() {
        return precio + (precio * IMPUESTO_VENTAS);
    }
}
```

```
public class Ropa extends Producto {
    private static final double PRECIO_MARCA = 20.0;
    private boolean esDeMarca;

    public Ropa(String id, String nombre, double precio, boolean esDeMarca) {
        super(id, nombre, precio);
        this.esDeMarca = esDeMarca;
    }

    @Override
    public double calcularPrecioFinal() {
        return esDeMarca ? precio + PRECIO_MARCA : precio;
    }
}
```

```
public class Ej6 {

    public static void main(String[] args) {
        Producto p1 = new Electronico("1", "TV", 200);
        Producto p2 = new Alimento("2", "Manzana", 1);
        Producto p3 = new Ropa("3", "Camisa", 50, true);
    }
}
```

```
System.out.println("Precio final del producto " + p1.nombre + ": " +  
                    p1.calcularPrecioFinal());  
System.out.println("Precio final del producto " + p2.nombre + ": " +  
                    p2.calcularPrecioFinal());  
System.out.println("Precio final del producto " + p3.nombre + ": " +  
                    p3.calcularPrecioFinal());  
}  
}
```

Ejercicio 7.

Desarrolla una aplicación de un sistema para un banco que ofrece varios tipos de cuentas: cuenta de ahorro, cuenta corriente y cuenta de inversión. Todas las cuentas tienen un número de cuenta, un titular y un saldo. Sin embargo, cada tipo de cuenta tiene diferentes maneras de calcular el interés.

Crea una clase abstracta *Cuenta* con las características comunes de todas las cuentas y un método abstracto *calcularInteres*. Luego, crea las clases *CuentaAhorro*, *CuentaCorriente* y *CuentaInversion* que extienden a *Cuenta* y proporciona una implementación para el método *calcularInteres* en cada una de estas subclases.

Finalmente, escribe un programa de prueba que cree al menos un objeto de cada subclase, calcule el interés para cada cuenta e imprima la información de la cuenta junto con el interés calculado.

```
public abstract class Cuenta {  
  
    protected String numero;  
    protected String titular;  
    protected double saldo;  
  
    public Cuenta(String numero, String titular, double saldo) {  
        this.numero = numero;  
        this.titular = titular;  
        this.saldo = saldo;  
    }  
  
    public abstract double calcularInteres();  
  
    public String getNumeroCuenta() {  
        return numero;  
    }  
  
    public void setNumeroCuenta(String numero) {  
        this.numero = numero;  
    }  
  
    public String getTitular() {  
        return titular;  
    }  
  
    public void setTitular(String titular) {  
        this.titular = titular;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public void setSaldo(double saldo) {  
        this.saldo = saldo;  
    }  
}
```




```

    }
}

public class CuentaAhorro extends Cuenta {
    private static final double TASA_INTERES = 0.03;

    public CuentaAhorro(String numero, String titular, double saldo) {
        super(numero, titular, saldo);
    }

    @Override
    public double calcularInteres() {
        return saldo * TASA_INTERES;
    }
}

public class CuentaCorriente extends Cuenta {
    private static final double TASA_INTERES = 0.01;

    public CuentaCorriente(String numero, String titular, double saldo) {
        super(numero, titular, saldo);
    }

    @Override
    public double calcularInteres() {
        return saldo * TASA_INTERES;
    }
}

public class CuentaInversion extends Cuenta {

    private static final double INTERES = 0.06;

    CuentaInversion(String numeroCuenta, String titular, double saldo) {
        super(numeroCuenta, titular, saldo);
    }

    @Override
    public double calcularInteres() {
        return saldo * INTERES;
    }
}

public class Ej7 {

    public static void main(String[] args) {
        Cuenta[] cuentas = new Cuenta[3];

        cuentas[0] = new CuentaAhorro("12345", "John Doe", 10000);
        cuentas[1] = new CuentaCorriente("67890", "Jane Doe", 20000);
        cuentas[2] = new CuentaInversion("11223", "Richard Roe", 30000);

        for (Cuenta cuenta : cuentas) {
            double interes = cuenta.calcularInteres();
            System.out.printf("Cuenta: %s, Titular: %s, Saldo: %.2f, Interés: %.2f\n",
                cuenta.getNumeroCuenta(), cuenta.getTitular(), cuenta.getSaldo(), interes);
        }
    }
}

```



Ejercicio 8.

Vamos a diseñar una simulación de un jardín. En este jardín, hay una variedad de organismos, incluyendo varias especies de plantas, insectos y aves. Por otro lado, podemos añadir distintas casas o casetas para que puedan vivir los organismos.

Una particularidad del jardín es que cualquier objeto que exista, sea un organismo o casa, puede tener la capacidad de moverse por sí mismo o que se pueda ser trasladado por acción nuestra. La forma de mover o trasladar será a partir de recibir dos coordenadas, x e y que representa una ubicación del jardín. Con lo cual, debe de existir dos métodos denominados, `mover(ubicación)` y `trasladar(ubicación)` comunes a todas nuestras clases, y que permita modificar su funcionalidad según la necesidad de cada una de ellas. Cabe destacar que, por las necesidades de nuestra aplicación, pensamos que la funcionalidad de la movilidad se pueda extender a futuros requerimientos que no sea solo del jardín.

Cada organismo tiene un conjunto de características comunes: un nombre y una ubicación en el jardín, como se ha indicado en el anterior párrafo, las ubicaciones se representan con coordenadas, x e y . Según los requisitos, no es posible tener organismos genéricos, solo podemos tener plantas, insectos, aves u otro tipo que se presente en un futuro.

Dentro de los organismos tenemos las plantas que tienen como característica propia la fecha en la que se plantó y las acciones de crecer y realizar la fotosíntesis. A diferencia de los demás organismos, las plantas no pueden moverse por sí mismo, solo se pueden trasladar por acción nuestra.

Otro de los organismos, los insectos no tienen características propias. Pueden moverse libremente por el jardín según la ubicación que quieran y tienen la capacidad de alimentarse de plantas y de otros insectos, pero no del resto de organismos. Aún que los insectos son libres, los podemos atrapar y trasladarlos a otra ubicación por nuestra cuenta.

Las aves son otro organismo que no tienen características propias. Pueden moverse libremente por el jardín según la ubicación que quieran, aún que esta acción no existe como tal, si no, que su acción se denomina volar y no es posible trasladar las aves a otra ubicación.

Hay dos estructuras construidas en el jardín: una caseta para pájaros y una casa de insectos. Ambas estructuras comparten características, como el nombre, el material con el que están construidas y la ubicación. En cambio, se diferencia en que la caseta de pájaros permite saber la cantidad aves que puede tener y la casa de insectos permite conocer el tipo de insectos que alberga. Cabe indicar que nuestra aplicación si permite tener casas genéricas que no sean de pájaros ni de insectos. Aún que las casas no tienen ninguna acción peculiar, todas se comportan igual, realizando acciones de mover y trasladar:

- Cuando se mueva, se mostrará un mensaje indicando: "La casa no puede moverse por sí sola."

- Cuando se traslade, se mostrará un mensaje indicando: "La casa se traslada a la ubicación x,y".

Además, debes implementar un método denominado *muestraCaracteristicas()* en cada clase que imprima un mensaje mostrando los atributos y los valores que tiene de cada organismo o casa. En el caso, que en un futuro se generen nuevas clases que representen objetos del jardín, tenemos que forzar que se implemente este método con el mismo nombre.

Tu tarea es diseñar las clases necesarias para representar a los diferentes organismos y estructuras del jardín, pensando si es necesario crear clases abstractas o interfaces, así como, definir las acciones que pueden realizar. No es necesario realizar la funcionalidad de las acciones, solo es suficiente con mostrar un mensaje. Por ejemplo, la acción de mover puede mostrar un mensaje como el siguiente: "El objeto se mueve a la ubicación x e y". Debes asegurarte de que tu diseño es lo suficientemente flexible como para poder añadir nuevas especies de organismos y nuevas acciones en el futuro.

Por último, debes implementar un método en tu programa principal que utilice tus clases para crear una pequeña simulación del jardín. Este método debe crear una variedad de organismos y casas, y luego debe utilizar los métodos de tus clases para hacer que los organismos realicen varias acciones.

```
public class Ubicacion {  
  
    private int x;  
    private int y;  
  
    public Ubicacion(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    public int getX() {  
        return x;  
    }  
  
    public void setX(int x) {  
        this.x = x;  
    }  
  
    public int getY() {  
        return y;  
    }  
  
    public void setY(int y) {  
        this.y = y;  
    }  
  
    @Override  
    public String toString() {  
        return "x=" + x + ", y=" + y;  
    }  
}
```



```
public interface Movilidad {  
  
    public void mover(Ubicacion ubicacion);  
    public void trasladar(Ubicacion ubicacion);  
  
}
```

```
public abstract class Organismo {  
  
    private String nombre;  
    private Ubicacion ubicacion;  
  
    public Organismo(String nombre, Ubicacion ubicacion) {  
        this.nombre = nombre;  
        this.ubicacion = ubicacion;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public Ubicacion getUbicacion() {  
        return ubicacion;  
    }  
  
    public void setUbicacion(Ubicacion ubicacion) {  
        this.ubicacion = ubicacion;  
    }  
  
    public abstract void muestraCaracteristicas();  
  
}
```

```
public class Planta extends Organismo {  
  
    private Date fechaPlantacion;  
  
    public Planta(String nombre, Ubicacion ubicacion, Date fechaPlantacion) {  
        super(nombre, ubicacion);  
        this.fechaPlantacion = fechaPlantacion;  
    }  
  
    public void trasladar(Ubicacion ubicacion) {  
        System.out.println(getNombre() + " se traslada de sitio a:" + ubicacion);  
    }  
  
    public void crece() {  
        System.out.println(getNombre() + " ha crecido.");  
    }  
  
    public void fotosintesis() {  
        System.out.println(getNombre() + " realiza la fotosíntesis.");  
    }  
  
    public Date getFechaPlantacion() {  
        return fechaPlantacion;  
    }  
  
    @Override  
    public void muestraCaracteristicas() {  
        System.out.println("Planta: " + getNombre() + " Ubicacion: " + getUbicacion() + "  
fechaPlantacion: " + getFechaPlantacion() );  
    }  
  
}
```



```
public class Insecto extends Organismo implements Movilidad {

    public Insecto(String nombre, Ubicacion ubicacion) {
        super(nombre, ubicacion);
    }

    public void come(Organismo organismo) {
        if(organismo instanceof Insecto) {
            System.out.println(getNombre() + " se alimenta de insectos.");
        } else if(organismo instanceof Planta) {
            System.out.println(getNombre() + " se alimenta de plantas.");
        } else {
            System.out.println("No puede comer este tipo de organismo");
        }
    }

    @Override
    public void mover(Ubicacion ubicacion) {
        System.out.println("Se mueve libremente a la ubicacion" + ubicacion);
    }

    @Override
    public void trasladar(Ubicacion ubicacion) {
        System.out.println("El insecto se traslada a la ubicacion" + ubicacion);
    }

    @Override
    public void muestraCaracteristicas() {
        System.out.println("Insecto: " + getNombre() + " Ubicacion: " + getUbicacion());
    }
}
```

```
public class Ave extends Organismo {

    public Ave(String nombre, Ubicacion ubicacion) {
        super(nombre, ubicacion);
    }

    public void volar(Ubicacion ubicacion) {
        System.out.println("El ave " + getNombre() + " vuela a la ubicación: " + ubicacion);
    }

    @Override
    public void muestraCaracteristicas() {
        System.out.println("Ave: " + getNombre() + " Ubicacion: " + getUbicacion());
    }
}
```

```
public class Casa implements Movilidad {

    private String nombre;
    private String material;
    private Ubicacion ubicacion;

    public Casa(String nombre, String material, Ubicacion ubicacion) {
        this.nombre = nombre;
        this.material = material;
        this.ubicacion = ubicacion;
    }

    @Override
    public void mover(Ubicacion ubicacion) {
        System.out.println("La casa no puede moverse por sí sola.");
    }
}
```

```
@Override
public void trasladar(Ubicacion ubicacion) {
    System.out.println("La casa se traslada a la ubicación: "+ubicacion);
}

public void muestraCaracteristicas() {
    System.out.println("Casa: " + nombre + " Material: " + material + " Ubicacion: "
        + ubicacion );
}
}
```

```
public class CasaInsectos extends Casa {

    private String tipoInsectos;

    public CasaInsectos(String nombre, String material, Ubicacion ubicacion, String
        tipoInsectos) {
        super(nombre, material, ubicacion);
        this.tipoInsectos = tipoInsectos;
    }

    @Override
    public void muestraCaracteristicas() {
        super.muestraCaracteristicas();
        System.out.print(" TipoInsectos: " + tipoInsectos +"\n");
    }
}
```

```
public class CasetaPajaros extends Casa {

    private int cantidadPajaros;

    public CasetaPajaros(String nombre, String material, Ubicacion ubicacion, int cantidadPajaros)
    {
        super(nombre, material, ubicacion);
        this.cantidadPajaros = cantidadPajaros;
    }

    @Override
    public void muestraCaracteristicas() {
        super.muestraCaracteristicas();
        System.out.print(" CantidadPajaros: " + cantidadPajaros + "\n");
    }
}
```

Unidad 8

Ejercicio 1.

Crea un array de enteros que contenga una secuencia numérica del 1 al 100. Muestra por pantalla todos los elementos pares del array.

```
public class Ej01 {  
  
    public static void main(String[] args) {  
        int[] array = new int[100];  
  
        for (int i = 0; i < array.length; i++) {  
            array[i] = i + 1;  
        }  
  
        System.out.println("Los números pares son: ");  
        for (int i = 0; i < array.length; i++) {  
            if (array[i] % 2 == 0) {  
                System.out.println(array[i]);  
            }  
        }  
    }  
}
```

Ejercicio 2.

Crea un array para almacenar las 10 frutas que más te gusten. Haz una pequeña aplicación con un menú para solicitar la posición de la fruta se quiere mostrar por pantalla.

```
public class Ej02 {  
  
    public static void main(String[] args) {  
  
        String[] frutas = {"Manzana", "Naranja", "Plátano", "Fresa", "Melón", "Kiwi", "Piña",  
"Mango", "Uva", "Melocotón"};  
  
        Scanner entradaTeclado = new Scanner(System.in);  
  
        while (true) {  
            System.out.println("Elige una opción:");  
            System.out.println("1. Mostrar fruta por posición");  
            System.out.println("2. Salir");  
  
            int option = entradaTeclado.nextInt();  
  
            switch (option) {  
                case 1:  
                    System.out.println("Por favor, introduce la posición de la fruta (1-  
10):");  
  
                    int indice = entradaTeclado.nextInt();  
                    if (indice >= 1 && indice <= 10) {  
                        System.out.println("La fruta en la posición " + indice + " es " +  
frutas[indice - 1]);  
                    } else {  
                        System.out.println("Posición no válida. Introduce un número entre  
1 y 10.");  
                    }  
                    break;  
                case 2:  

```

```

        System.out.println("Fin de programa");
        entradaTeclado.close();
        return;
    default:
        System.out.println("Opción no válida.");
    }
    System.out.println();
}
}
}

```

Ejercicio 3

Crea un array de enteros de 20 elementos. Una vez inicializado, queremos dar la oportunidad al usuario de ampliar los elementos del array y ordenarlos en forma ascendente. El programa debe trabajar sobre una sola array y debe permitir solicitar al usuario cuantos elementos nuevos se van a guardar en el array, mostrar en cualquier momento los valores del array y la posibilidad de ordenar de forma ascendente todos sus elementos. Cada elemento se generará a partir de un número aleatorio de 100. Nota. Para generar un número aleatorio utiliza el objeto *Random* que aparece en el ejemplo de la práctica.

Genera el menú necesario para que el usuario pueda operar tal y como se indican en los requisitos.

```

public class Ej03 {

    public static void main(String[] args) {

        int[] numeros = new int[20];
        for (int i = 0; i < numeros.length; i++) {
            numeros[i] = new Random().nextInt(100);
        }

        Scanner entradaTeclado = new Scanner(System.in);

        while (true) {
            System.out.println("Elige una opción:");
            System.out.println("1. Añadir más elementos al array");
            System.out.println("2. Mostrar los elementos del array");
            System.out.println("3. Ordenar los elementos del array");
            System.out.println("4. Salir");

            int option = entradaTeclado.nextInt();

            switch (option) {
                case 1:
                    System.out.println("¿Cuántos elementos nuevos quieres añadir?");

                    int nuevosElementos = entradaTeclado.nextInt();

                    int[] nuevaArray = new int[numeros.length +
nuevosElementos];
                    System.arraycopy(numeros, 0, nuevaArray, 0,
numeros.length);
                    for (int i = numeros.length; i < nuevaArray.length; i++) {
                        nuevaArray[i] = new Random().nextInt(100);
                    }
                    numeros = nuevaArray.clone();
                    break;
                case 2:
                    imprimirArray(numeros);
                    break;
            }
        }
    }
}

```



```

        case 3:
            java.util.Arrays.sort(numeros);
            System.out.println("Se han ordenado los elementos del
array");
            break;
        case 4:
            System.out.println("Fin de programa");
            entradaTeclado.close();
            return;
        default:
            System.out.println("Opción no válida.");
            break;
    }
    System.out.println();
}

private static void imprimirArray(int[] array) {
    System.out.println("El array contiene los siguientes números:");
    for (int i = 0; i < array.length; i++) {
        System.out.print(array[i]);
        if (i < array.length - 1) {
            System.out.print(", ");
        }
    }
    System.out.println();
}
}

```

Ejercicio 4.

Supongamos que tienes una lista de conexiones entre varios puntos en un mapa. Cada conexión se representa por un par de números (por ejemplo, la conexión entre el punto 2 y el punto 3 se puede representar como 2,3). Crea una matriz de adyacencia para representar estas conexiones.

Por ejemplo, la matriz de adyacencia de los puntos {1, 2}, {2, 3}, {3, 1}, {4, 5}, {5, 1} es:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 | 0 |

```

public class Ej04 {
    public static void main(String[] args) {

        int[][] conexiones = { { 1, 2 }, { 2, 3 }, { 3, 1 }, { 4, 5 }, { 5, 1 } };

        int numeroPuntos = 5;

        int[][] matriz = new int[numeroPuntos][numeroPuntos];

        for (int[] conexion : conexiones) {

            int i = conexion[0] - 1;
            int j = conexion[1] - 1;

            matriz[i][j] = 1;
        }
    }
}

```

```

        matriz[j][i] = 1;
    }

    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[i].length; j++) {
            System.out.print(matriz[i][j] + " ");
        }
        System.out.println();
    }
}

```

Ejercicio 5.

Crea un programa que reciba un string y encuentre la palabra más larga en el string. Ignora los signos de puntuación y distingue entre mayúsculas y minúsculas. En caso de empate se indicará la primera palabra que se encuentre con más caracteres.

```

public class Ej05 {

    public static void main(String[] args) {
        String cadena = "Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. "
            + "Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) "
            + "desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especimen. No sólo sobrevivió 500 años, "
            + "sino que también ingresó como texto de relleno en documentos electrónicos, quedando esencialmente igual al original. Fue popularizado en los 60s "
            + "con la creación de las hojas \"Letraset\", las cuales contenían pasajes de Lorem Ipsum, y más recientemente con software de autoedición, "
            + "como por ejemplo Aldus PageMaker, el cual incluye versiones de Lorem Ipsum";

        String palabraMasLarga = "";
        String[] palabras = cadena.split(" ");

        for (String palabra : palabras) {
            String palabraLimpia = palabra.replaceAll("[^a-zA-Z]", "");
            if (palabraLimpia.length() > palabraMasLarga.length()) {
                palabraMasLarga = palabraLimpia;
            }
        }

        System.out.println("La palabra más larga es: " + palabraMasLarga);
    }
}

```

Ejercicio 6.

Escribe un programa que verifique si una frase es un palíndromo, ignorando los espacios, la puntuación y las diferencias entre mayúsculas y minúsculas. Un palíndromo es aquella expresión que se lee de izquierda a derecha que de derecha a izquierda.

```

public class Ej06 {

    public static void main(String[] args) {
        String cadena = "Anita lava la tina";
        System.out.println("La frase es un palíndromo: " + esPalindromo(cadena));
    }

    public static boolean esPalindromo(String str) {

```

```
String cadenaSinEspacios = str.toLowerCase().replace(" ", "");

String cadenaAlreves = "";
for (int i = cadenaSinEspacios.length() - 1; i >= 0; i--) {
    cadenaAlreves += cadenaSinEspacios.charAt(i);
}

return cadenaSinEspacios.equals(cadenaAlreves);
}
}
```

Ejercicio 7.

A partir de una cadena de texto cualesquiera que contenga dígitos y caracteres, se pide extraer todos los números enteros y realizar la suma de todos ellos. Por ejemplo, de la siguiente cadena "El tigre vive alrededor de 10 años, suele medir 70 cm y puede llegar a comer 40kg de carne" el resultado tiene que ser (10+70+40) 120.

Nota: Se puede utilizar la función *Character.isDigit* para comprobar si un carácter es un dígito o no.

```
public class Ej07 {
    public static void main(String[] args) {
        String cadena = "5El tigre vive alrededor de 10 años, suele medir 70 cm y puede llegar a comer 40kg de carne14";

        int suma = 0;
        String temporal = "";

        for (int i = 0; i < cadena.length(); i++) {
            char ch = cadena.charAt(i);

            if (Character.isDigit(ch)) {
                temporal += ch;
            } else if (!temporal.equals("")) {
                suma += Integer.parseInt(temporal);
                temporal = "";
            }
        }

        if (!temporal.equals("")) {
            suma += Integer.parseInt(temporal);
        }

        System.out.println("La suma de los números es: " + suma);
    }
}
```

Ejercicio 8.

Realiza un programa que tome una frase e imprima por pantalla cada palabra en una nueva línea, junto con su longitud y su posición inicial y final en la cadena original.

```
public class Ej08 {

    public static void main(String[] args) {
        String cadena = "The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, "
            + "for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase";

        int posicion = 0;
```

```
String[] palabras = cadena.split(" ");

for (String palabra : palabras) {
    int longitud = palabra.length();
    System.out.println("Palabra: " + palabra);
    System.out.println("Longitud: " + longitud);
    System.out.println("Posición inicial: " + posicion);

    System.out.println("Posición final: " + (posicion + longitud - 1));
    System.out.println("-----");

    posicion += longitud + 1;
}
}
```

Ejercicio 9.

Imagina que tienes que desarrollar un sistema para manejar las reservas de un pequeño hotel.

El hotel tiene tres pisos, cada piso tiene 10 habitaciones y las reservas se pueden hacer para cualquier día del año (asumiremos 365 días).

Para este ejercicio, el hotel se representa como un array tridimensional donde los tres ejes representan el piso, la habitación y el día del año. Cada celda en el array puede ser true (la habitación está reservada para ese día) o false (la habitación está libre).

Deberás implementar la siguiente funcionalidad:

- Realizar una reserva para un cliente en una habitación específica para un día específico, si la habitación está disponible.
- Cancelar una reserva existente.
- Comprobar si una habitación está disponible en un día específico.
- Comprobar qué habitaciones están disponibles en un día específico.
- Mostrar un informe de la ocupación del hotel para un día específico.

```
public class Ej09 {

    public static void main(String[] args) {

        Hotel hotel = new Hotel();

        int piso = 1;
        int habitacion = 2;
        int dia = 100;

        // Realiza una reserva
        if (hotel.hacerReserva(piso, habitacion, dia)) {
            System.out.println("Reserva realizada para el piso " + piso + ", habitacion " +
                habitacion + " el día " + dia);
        } else {
            System.out.println("No se ha podido hacer la reserva para el piso " + piso + ",
                habitación " + habitacion + " el día " + dia);
        }

        // Comprueba la disponibilidad
        if (!hotel.disponibilidad(piso, habitacion, dia)) {
            System.out.println("La habitación no esta disponible");
        } else {
            System.out.println("La habitación esta disponible");
        }

        // Muestra un informe de ocupación
    }
}
```

```

        hotel.reporteOcupacion(dia);

        // Cancela la reserva
        hotel.cancelarReserva(piso, habitacion, dia);
        System.out.println("Reserva cancelada para el piso " + piso + ", habitacion " +
            habitacion + " el día " + dia);

        // Comprueba la disponibilidad nuevamente
        if (!hotel.disponibilidad(piso, habitacion, dia)) {
            System.out.println("La habitación no esta disponible");
        } else {
            System.out.println("La habitación esta disponible");
        }
    }
}

```

Ejercicio 10.

Imagina que tienes una aplicación de música y quieres implementar una funcionalidad para manejar playlists. Cada playlist se representará como una clase Playlist con las siguientes propiedades:

- **nombre:** El nombre de la playlist.
- **canciones:** Un array de strings, donde cada string representa el nombre de una canción. El formato de cada string será el siguiente: "nombre canción – duración". Por ejemplo, "Bohemian Rhapsody - 5.55". La cantidad de canciones que puede contener el playlist la establece el usuario cuando se crea.

Deberás implementar los siguientes métodos en la clase Playlist:

- Un método para agregar una nueva canción a la playlist.
- Un método para eliminar una canción de la playlist por su nombre.
- Un método para imprimir todas las canciones en la playlist.
- Una función que reciba el nombre de una canción y devuelva true si la canción está en la playlist, false en caso contrario.
- Una función que devuelva la canción más larga de la playlist.
- Una función que devuelva el total de la duración de todas las canciones en la playlist.

```

public class Ej10 {
    public static void main(String[] args) {
        Playlist playlist = new Playlist("Música favorita", 5);

        playlist.agregarCancion("Bohemian Rhapsody - 5.55");
        playlist.agregarCancion("Stairway to Heaven - 8.02");
        playlist.agregarCancion("Hotel California - 6.30");

        System.out.println("Canciones en la playlist:");
        playlist.imprimirCanciones();

        System.out.println("¿Contiene la playlist la canción 'Stairway to Heaven'? " +
            playlist.contieneCancion("Stairway to Heaven"));
        System.out.println("Canción más larga: " + playlist.cancionMasLarga());
        System.out.println("Duración total: " + playlist.duracionTotal());

        playlist.eliminarCancion("Hotel California - 6.30");

        System.out.println("Canciones en la playlist después de eliminar una canción:");
        playlist.imprimirCanciones();
    }
}

```



}

Unidad 9

Ejercicio 1.

Crea un programa que permita al usuario introducir nombres de personas y guardarlos en una colección. Se quiere almacenar solo los nombres únicos y no repetidos (No hay que realizar ninguna lógica para verificar si se introducen nombres repetidos). Una vez que se han introducido todos los nombres, el programa debe imprimir todos los nombres únicos.

```
public class Ej01 {  
  
    public static void main(String[] args) {  
        Set<String> nombres = new HashSet<>();  
        Scanner entradaTeclado = new Scanner(System.in);  
  
        while (true) {  
            System.out.println("Introduce un nombre (Para terminar escribe una 's'): ");  
            String name = entradaTeclado.nextLine();  
  
            if (name.equalsIgnoreCase("s"))  
                break;  
  
            nombres.add(name);  
        }  
  
        System.out.println("Los nombres únicos son: " + nombres);  
        entradaTeclado.close();  
    }  
}
```

Ejercicio 2.

Crea un programa que gestione una lista de tareas pendientes. El usuario debe poder agregar tareas, eliminar tareas y ver todas las tareas. Piensa adecuadamente que colección corresponde según la funcionalidad descrita.

```
public class Ej02 {  
  
    public static void main(String[] args) {  
        List<String> tareas = new ArrayList<>();  
        Scanner entradaTeclado = new Scanner(System.in);  
  
        while (true) {  
            System.out.println("1. Agregar tarea\n2. Eliminar tarea\n3. Ver  
tareas\n4. Salir");  
  
            int opcion = entradaTeclado.nextInt();  
            entradaTeclado.nextLine();  
  
            switch (opcion) {  
                case 1:  
                    System.out.println("Introduce la tarea: ");  
                    String task = entradaTeclado.nextLine();  
                    tareas.add(task);  
                    break;  
  
                case 2:  
                    System.out.println("Indica el índice de la tarea  
a eliminar: ");  
  
                    int index = entradaTeclado.nextInt();  
                    tareas.remove(index);  
                    break;  
  
                case 3:  
                    System.out.println("Todas las tareas pendientes:");  
                    for (String task : tareas) {  
                        System.out.println(task);  
                    }  
                    break;  
  
                case 4:  
                    return;  
            }  
        }  
    }  
}
```

```

        +tareas.get(i));

        case 3:
            if (index < 0 || index >= tareas.size()) {
                System.out.println("Índice no válido.");
            } else {
                tareas.remove(index);
            }
            break;

        case 4:
            System.out.println("Tareas: ");
            for(int i= 0; i<tareas.size(); i++) {
                System.out.println("\t"+ i + " - "

            }
            break;

        default:
            entradaTeclado.close();
            System.out.println("Fin de programa");
            return;

            System.out.println("Opción no válida.");
        }
        System.out.println();
    }

}

```

Ejercicio 3.

Crea un simulador de una cola de un banco. Los clientes deben ser añadidos a la cola y procesados en orden (FIFO, primero en entrar primero en salir). Asegúrate de que la cola funciona correctamente cuando llegan nuevos clientes y cuando un cliente es atendido, para ello, haz un menú para poder agregar nuevos clientes a la cola, atender a clientes y mostrar los clientes que actualmente están esperando.

```

public class Ej03 {

    public static void main(String[] args) {

        Queue<String> colaBanco = new LinkedList<>();
        Scanner entradaTeclado = new Scanner(System.in);
        Random random = new Random();

        while (true) {

            System.out.println("1. Añadir cliente\n2. Atender cliente\n3. Mostrar cola\n4. Salir");

            int opcion = entradaTeclado.nextInt();
            entradaTeclado.nextLine();

            switch (opcion) {
                case 1:
                    String nuevoCliente = "Cliente_" +
                    (random.nextInt(100) + 1);
                    colaBanco.add(nuevoCliente);
                    System.out.println(nuevoCliente + " se ha
                    incorporado en la cola");
                    break;
                case 2:
                    String cliente = colaBanco.poll();
                    if (cliente != null) {
                        System.out.println(cliente + " ha sido atendido");
                    } else {
                        System.out.println("No hay clientes en la cola");
                    }
            }
        }
    }
}

```



```

        }
        break;
        case 3:
            colaBanco);
            System.out.println("Clientes esperando: " +
                                colaBanco.size());
            break;
        case 4:
            entradaTeclado.close();
            System.out.println("Fin de programa");
            return;
        default:
            System.out.println("Opción no válida.");
    }
}
}
}

```

Ejercicio 4.

Implementa un sencillo diccionario español-inglés. El usuario puede introducir palabras en español y el programa traducirá al inglés. Piensa adecuadamente como almacenar el diccionario para que sea lo más eficiente posible.

```

public class Ej04 {

    public static void main(String[] args) {

        Map<String, String> diccionario = new HashMap<>();
        diccionario.put("hola", "hello");
        diccionario.put("adiós", "goodbye");
        diccionario.put("mundo", "world");
        diccionario.put("amor", "love");
        diccionario.put("amigo", "friend");

        Scanner entradaTeclado = new Scanner(System.in);

        while (true) {
            System.out.println("Escribe una palabra para buscarla en el diccionario (Para
terminar escribe una 's'): ");
            String palabra = entradaTeclado.nextLine();
            if (palabra.equalsIgnoreCase("s")) {
                break;
            }
            String traduccion = diccionario.get(palabra.toLowerCase());
            if (traduccion != null) {
                System.out.println("La traducción de : " + palabra + " es: " + traduccion);
            } else {
                System.out.println("La palabra no esta en el diccionario");
            }
        }

        entradaTeclado.close();
    }
}

```

Ejercicio 5.

Desarrollar un programa para solicitar números enteros, guardarlos en una lista hasta que se decida finalizar. Una vez finalizado, ordenar la lista de números de menor a mayor utilizando el algoritmo de ordenación burbuja y mostrar el resultado por pantalla.

El algoritmo de ordenación burbuja tiene el siguiente pseudocódigo:

```

procedimiento ordenacionBurbuja(lista)
    para i desde 0 hasta tamaño de lista - 1
        para j desde 0 hasta tamaño de lista - i - 1
            // Si el elemento actual es mayor que el siguiente
            si lista[j] > lista[j + 1] entonces
                // Intercambiar los elementos
                temp = lista[j]
                lista[j] = lista[j + 1]
                lista[j + 1] = temp
            fin si
        fin para
    fin para
fin procedimiento
  
```

```

public class Ej05 {

    public static void main(String[] args) {
        List<Integer> listaNumeros = new ArrayList<>();
        Scanner entradaTeclado = new Scanner(System.in);

        System.out.println("Escribe números (Para terminar escribe una 's'): ");
        while (true) {
            String input = entradaTeclado.nextLine();
            if (input.equalsIgnoreCase("s")) {
                break;
            }
            listaNumeros.add(Integer.parseInt(input));
        }

        ordenacionBurbuja(listaNumeros);

        System.out.println("Números ordenados: " + listaNumeros);

        entradaTeclado.close();
    }

    private static void ordenacionBurbuja(List<Integer> lista){
        for (int i = 0; i < lista.size(); i++) {
            for (int j = 0; j < lista.size() - i - 1; j++) {
                if (lista.get(j) > lista.get(j + 1)) {
                    int temp = lista.get(j);
                    lista.set(j, lista.get(j + 1));
                    lista.set(j + 1, temp);
                }
            }
        }
    }
}
  
```

Ejercicio 6.

Implementa un programa que, dado dos conjuntos de números enteros, calcule la intersección y la unión de estos. Cada conjunto puede tener un número indeterminado de elementos y no puede haber repetidos. El programa debe imprimir ambos conjuntos, la intersección y la unión.

La intersección de dos conjuntos es un nuevo conjunto que contiene todos los elementos que son comunes a ambos conjuntos originales



La unión de dos conjuntos es un nuevo conjunto que contiene todos los elementos que están en cualquiera de los dos conjuntos originales

Por ejemplo, si hay dos conjuntos:

- Conjunto A: {1, 2, 3, 4, 5}
- Conjunto B: {4, 5, 6, 7, 8, 9}

La intersección de A y B es: {4, 5} y la unión de A y B es: {1, 2, 3, 4, 5, 6, 7, 8, 9}

```
public class Ej06 {  
  
    public static void main(String[] args) {  
        Set<Integer> conjunto1 = new HashSet<>();  
        Set<Integer> conjunto2 = new HashSet<>();  
  
        System.out.println("Escribe números para el conjunto 1 (Para terminar escribe una  
's'):" );  
        conjunto1 = solicitarNumeros();  
  
        System.out.println("Escribe números para el conjunto 2 (Para terminar escribe una  
's'): ");  
        conjunto2 = solicitarNumeros();  
  
        // Intersección  
        Set<Integer> interseccion = new HashSet<>(conjunto1);  
        interseccion.retainAll(conjunto2);  
  
        // Unión  
        Set<Integer> union = new HashSet<>(conjunto1);  
        union.addAll(conjunto2);  
  
        System.out.println("Numeros del conjunto 1: " + conjunto1);  
        System.out.println("Numeros del conjunto 2: " + conjunto2);  
        System.out.println("Intersección: " + interseccion);  
        System.out.println("Unión: " + union);  
    }  
  
    private static Set<Integer> solicitarNumeros() {  
        Set<Integer> conjunto = new HashSet<>();  
  
        Scanner entradaTeclado = new Scanner(System.in);  
  
        while (true) {  
            String input = entradaTeclado.nextLine();  
            if (input.equalsIgnoreCase("s")) {  
                break;  
            }  
            conjunto.add(Integer.parseInt(input));  
        }  
  
        return conjunto;  
    }  
}
```

Ejercicio 7.

Implementa un programa que simule la gestión de procesos en un sistema operativo utilizando una cola. Los procesos tienen un ID único y un tiempo de ejecución en milisegundos. El programa debe permitir añadir procesos, ejecutar los procesos en orden de llegada y mostrar los procesos que están en cola. No es posible ejecutar el siguiente proceso, si el anterior aún no ha acabado su tiempo de ejecución.

```

public class Ej07 {

    private static class Proceso {
        private static int count = 1;
        int id;
        long tiempoEjecucion;

        public Proceso(long tiempoEjecucion) {
            this.id = count++;
            this.tiempoEjecucion = tiempoEjecucion;
        }

        @Override
        public String toString() {
            return "Proceso{" + "id=" + id + ", tiempoEjecución=" + tiempoEjecucion +
        }

    }

    public static void main(String[] args) {

        Queue<Proceso> colaProcesos = new LinkedList<>();
        Scanner entradaTeclado = new Scanner(System.in);

        Proceso procesoEnEjecucion = null;
        long comienzo = 0;

        while (true) {
            System.out.println("1. Añadir proceso\n2. Ejecutar de proceso\n3. Mostrar
cola de procesos\n4. Salir");
            int opcion = entradaTeclado.nextInt();
            entradaTeclado.nextLine();

            switch (opcion) {
                case 1:
                    System.out.println("Indica el tiempo de ejecución
en milisegundos: ");
                    long tiempoEjecucion = entradaTeclado.nextInt();
                    Proceso proceso = new Proceso(tiempoEjecucion);
                    colaProcesos.add(proceso);
                    System.out.println("Proceso añadido: " +
proceso);
                    break;
                case 2:
                    if(procesoEnEjecucion != null) {
                        if(System.currentTimeMillis() - comienzo
<= procesoEnEjecucion.tiempoEjecucion) {
                            System.out.println("No es posible
ejecutar un nuevo proceso. El proceso: "
+ procesoEnEjecucion + " lleva en ejecución: " + (System.currentTimeMillis() -
comienzo));
                            break;
                        }
                    }
                    procesoEnEjecucion = colaProcesos.poll();
                    if (procesoEnEjecucion != null) {
                        comienzo = System.currentTimeMillis();
                        System.out.println("Ejecutando proceso: "
+ procesoEnEjecucion);
                    } else {
                        System.out.println("No hay procesos en la
cola");
                    }
                    break;
                case 3:
                    System.out.println("Procesos en la cola: " +
colaProcesos);
                    break;
                case 4:
            }
        }
    }
}

```

```

                                entradaTeclado.close();
                                System.out.println("Fin de programa");
                                return;
                                default:
                                System.out.println("Opción no válida.");
                                }
                                System.out.println();
                                }
                                }
}

```

Ejercicio 8.

Escribe un programa que cuente el número de apariciones de cada carácter en una cadena de texto. La cadena de texto será introducida por teclado y finalmente se debe de mostrar por cada carácter la cantidad que aparece en la cadena de texto.

```

public class Ej08 {

    public static void main(String[] args) {

        Scanner entradaTeclado = new Scanner(System.in);

        System.out.println("Introduce un texto: ");
        String texto = entradaTeclado.nextLine();

        Map<Character, Integer> caracteres = new HashMap<>();
        for (char character : texto.toCharArray()) {
            caracteres.put(character, caracteres.getOrDefault(character, 0) + 1);
        }

        System.out.println("Caracteres y cantidad de apariciones");
        for (Map.Entry<Character, Integer> entry : caracteres.entrySet()) {
            System.out.println("- Caracter: " + entry.getKey() + ", nº de
apariciones: " + entry.getValue());
        }

        entradaTeclado.close();
    }
}

```

Ejercicio 9.

Crea un programa que maneje un conjunto de estudiantes y sus notas. Cada estudiante tiene un nombre y una nota media. El programa debe permitir al usuario añadir estudiantes sin que existan repetidos y ver todos los estudiantes ordenados ascendentemente por su nombre alfabéticamente. Además, también se debe poder buscar estudiantes por su nombre.

```

public class Ej09 {

    public static class Estudiante implements Comparable<Estudiante> {
        String nombre;
        double nota;

        Estudiante(String nombre, double nota) {
            this.nombre = nombre;
            this.nota = nota;
        }
    }
}

```

```

        @Override
        public int compareTo(Estudiante otroEstudiante) {
            return this.nombre.compareTo(otroEstudiante.nombre);
        }

        @Override
        public String toString() {
            return "Estudiante{" + "nombre='" + nombre + '\'' + ", nota=" + nota +
        }';
    }

    public static void main(String[] args) {
        TreeSet<Estudiante> estudiantes = new TreeSet<>();
        Scanner entradaTeclado = new Scanner(System.in);

        while (true) {
            System.out.println("1. Añadir estudiante\n2. Mostrar estudiante\n3.
Buscar estudiante\n4. Salir");
            int opcion = entradaTeclado.nextInt();
            entradaTeclado.nextLine();

            switch (opcion) {
                case 1:
                    System.out.print("Introduce el nombre: ");
                    String nombre = entradaTeclado.nextLine();
                    System.out.print("Introduce la nota: ");
                    double nota = entradaTeclado.nextDouble();
                    estudiantes.add(new Estudiante(nombre, nota));
                    break;

                case 2:
                    for (Estudiante estudiante : estudiantes) {
                        System.out.println(estudiante);
                    }
                    break;

                case 3:
                    System.out.print("Introduce el nombre del
estudiante a buscar: ");
                    entradaTeclado.nextLine();

                    String nombreEstudiante =
                    for (Estudiante student : estudiantes) {
                        if
                            System.out.println("Found: " +
                                student);
                    }
                    break;

                case 4:
                    entradaTeclado.close();
                    System.out.println("Fin de programa");
                    return;

                default:
                    System.out.println("Opción no válida.");
            }

            System.out.println();
        }
    }
}

```

Ejercicio 10.

Implementa un sistema de votación para una elección política. El programa debe contemplar los siguientes requisitos:

- Los candidatos disponen de su nombre y el nombre del partido político que representa. Se debe de inicializar los candidatos al inicio del sistema de votación hasta un máximo de 5 y no puede existir candidatos repetidos.
- Los votantes son añadidos a una cola. Cada votante se identifica por un ID único y vota por un candidato. Aún que el voto debe ser anónimo, es necesario que se quede guardado el voto que ha realizado. Se debe poder ingresar nuevos votantes y ejercer su voto según el orden de la cola.
- Los votos de cada candidato se almacenan en una lista para facilitar la cuenta final. Un candidato solo puede ejercer un único voto.
- Al finalizar la votación, el programa debe mostrar el candidato ganador y el recuento total de votos. En el caso de empate en votos el ganador será el primer candidato que se le hizo el recuento.

```
public class Ej10 {

    public static class Votante {
        private static int count = 1;

        int id;
        Candidato candidato;

        public Votante(Candidato candidato) {
            this.id = count++;
            this.candidato = candidato;
        }

        @Override
        public String toString() {
            return "Votante{" + "id='" + id + '\'' + ", candidato=" + candidato +
        }

    }

    public static class Candidato {
        String nombre;
        String partido;

        public Candidato(String nombre, String partido) {
            this.nombre = nombre;
            this.partido = partido;
        }

        @Override
        public String toString() {
            return "Candidato{" + "nombre='" + nombre + '\'' + ", partido=" + partido
        + '\'';
        }

    }

    public static void main(String[] args) {

        Scanner entradaTeclado = new Scanner(System.in);

        Set<Candidato> candidatos = new HashSet<>();
        Queue<Votante> colaVotacion = new LinkedList<>();
        Map<Candidato, List<Votante>> listadoVotacion = new HashMap<>();
    }
}
```

```

// Inicializar candidatos
boolean continuar = true;
while (continuar) {
    System.out.println("1. Añadir candidato\n2. Mostrar candidatos\n3.
Terminar el registro de
candidatos");
    int opcion = entradaTeclado.nextInt();
    entradaTeclado.nextLine();

    switch (opcion) {
        case 1:
            if (candidatos.size() < 5) {
                System.out.print("Introduce el nombre del
candidato: ");
                entradaTeclado.nextLine();
                System.out.print("Introduce el partido
politico al que pertenece: ");
                entradaTeclado.nextLine();
                Candidato candidato = new
                Candidato(nombre, partido);
                candidatos.add(candidato);
                listadoVotacion.putIfAbsent(candidato,
                new ArrayList<>());
            } else {
                System.out.println("No se puede añadir
                más de 5 candidatos");
                break;
            }
        case 2:
            System.out.println("Los candidatos son: " +
            candidatos);
            break;
        case 3:
            continuar = false;
            break;
        default:
            System.out.println("Opción no válida.");
    }
    System.out.println();
}

// Registrar votos
continuar = true;

while (continuar) {
    System.out.println("1. Añadir votante\n2. Registrar voto\n3. Mostrar cola
de votación\n4. Terminar el proceso de votación");
    int opcion = entradaTeclado.nextInt();
    entradaTeclado.nextLine();

    switch (opcion) {
        case 1:
            System.out.print("Introduce el nombre del
candidato a votar: ");
            String candidato = entradaTeclado.nextLine();
            boolean insertado = false;
            for (Iterator<Candidato> iterator =
            candidatos.iterator(); iterator.hasNext() && !insertado;) {
                Candidato c = (Candidato)
                iterator.next();
                if(c.nombre.equalsIgnoreCase(candidato))
                {
                    colaVotacion.add(new Votante(c));
                }
            }
        }
    }
}

```



```

                                insertado = true;
                                }
                                }
                                if(insertado) {
                                    System.out.println("El votante ha pasado
a la cola de votación");
                                } else {
                                    System.out.println("No existe ningun
candidato con ese nombre");
                                }
                                break;
                                case 2:
                                    Votante votante = colaVotacion.poll();
                                    if (votante != null) {
                                        listadoVotacion.get(votante.candidato).add(votante);
                                        System.out.println("El votante " +
votante.id + " ha ejercido su voto");
                                    } else {
                                        System.out.println("No hay votantes en la
cola");
                                    }
                                    break;
                                    case 3:
                                        System.out.println("La cola actual de votantes: "
+ colaVotacion);
                                    break;
                                    case 4:
                                        continuar = false;
                                        break;
                                    default:
                                        System.out.println("Opción no válida.");
                                }
                                System.out.println();
                            }
                            int votosMaximos = 0;
                            Candidato ganador = null;
                            System.out.println("Recuento de votos:");
                            for (Candidato candidato : candidatos) {
                                int votos = listadoVotacion.get(candidato).size();
                                if (votos > votosMaximos) {
                                    votosMaximos = votos;
                                    ganador = candidato;
                                }
                                System.out.println("- Candidato: " + candidato.nombre + ", Votos: " +
votos);
                            }
                            System.out.println("El ganador es: " + ganador + ", Votos: " + votosMaximos);
                            entradaTeclado.close();
                        }
                    }
}

```

Unidad 10

Ejercicio de la videoclase

- El código de las imágenes ¿Hacen lo mismo?
- ¿Qué diferencia existe entre utilizar throws o no utilizarlo?
- ¿Qué diferencia hay entre el uso del throws y throw?
- ¿Qué código es mejor?


```
public static int calculadora(int operacion, int param, int param2)
    throws IllegalArgumentException {
    switch (operacion) {
        case 0:
            return param + param2;
        case 1:
            return param - param2;
        case 2:
            return param * param2;
        case 3:
            return param / param2;
        default:
            throw new IllegalArgumentException("Operacion no reconocida");
    }
}

public static int calculadora(int operacion, int param, int param2) {
    switch (operacion) {
        case 0:
            return param + param2;
        case 1:
            return param - param2;
        case 2:
            return param * param2;
        case 3:
            return param / param2;
        default:
            throw new OperacionException("Operacion no reconocida");
    }
}
```


El código de las imágenes realiza lo mismo, en el caso que se envíe un número no reconocido como operación lanzará una excepción con el mensaje Operación no reconocida.

En este caso, al utilizar throws estamos informando que excepción puede producirse y que se enviara al siguiente nivel. En cambio, si no utilizamos throws, el usuario quien llame al método no puede saber que excepción puede surgir.

`calculadora(4,0,0);`

 `int ejercicio.Ej1.calculadora(int operacion, int param, int param2) throws
IllegalArgumentException`

`calculadora(4,0,0);`

 `int ejercicio.Ej1.calculadora(int operacion, int param, int
param2)`

Como se puede ver en las dos imágenes, si no se puede ver el código que tiene el cuerpo del método, solo la cabecera, en la primera opción veo que excepción se puede suceder, en cambio en la segunda opción, no.

La principal diferencia entre throws y throw, es saber de quién es la responsabilidad de gestionar y no la excepción. Con throws estamos indicando que la excepción se propagará y que nuestro método no va a gestionar nada, se encargará el siguiente nivel



de decidir qué hacer con la excepción. En cambio, con `throw`, somos nosotros quien indica cuando se produce la excepción y podemos gestionarla o propagarla.

El código que es considerado mejor que el otro es el que utiliza `throws` en la cabecera del método, ya que estamos informando a cualquier usuario que llame al método que excepción es la que puede provocar y que puede manejar o propagar.

Unidad 12

Ejercicio 1 de la videoclase

Crear un fichero con el abecedario completo. Se pide que se lea el fichero y donde haya una consonante se sustituya por un asterisco *. Se tiene que trabajar siempre con el mismo fichero.

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

    String archivoNombre = "abecedario.txt";
    String contenido = "abcdefghijklmnopqrstuvwxyz";

    // Crear el archivo con el abecedario completo
    File ficheroAbecedario = null;
    FileWriter writer = null;
    try {
        ficheroAbecedario = new File("abecedario.txt");
        System.out.println("¿Se ha creado el fichero? " +
            ficheroAbecedario.createNewFile());

        writer = new FileWriter(archivoNombre);
        writer.write(contenido);
        System.out.println("Contenido grabado en el fichero");
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(writer != null)
            writer.close();
    }

    // Leer el archivo y reemplazar las consonantes por asteriscos
    BufferedReader reader = null;
    StringBuilder contenidoModificado = new StringBuilder();
    try {
        reader = new BufferedReader(new FileReader(archivoNombre));
        int c;
        while ((c = reader.read()) != -1) {
            char caracter = (char) c;
            if ("bcdfghjklmnpqrstvwxyz".indexOf(caracter) != -1) {
                contenidoModificado.append('*');
            } else {
                contenidoModificado.append(caracter);
            }
        }
        System.out.println("Lectura finalizada");
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(reader != null)
            reader.close();
    }

    // Guardar el contenido modificado en el mismo archivo
    try {
        writer = new FileWriter(archivoNombre);
        writer.write(contenidoModificado.toString());
        System.out.println("Contenido modificado");
        System.out.println("El nuevo contenido es el siguiente: "+contenidoModificado);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(writer != null)
            writer.close();
    }
}
```

```

        writer.close();
    }
}

```

Ejercicio 2 de la videoclase

Dada una ruta (path) cualquiera, se solicita que muestre por pantalla:

- El total de ficheros.
- El total de carpetas.

A tener en cuenta, se pide todos los ficheros que hay en la ruta y dentro de cada carpeta que hubiera. A demás puede ser que se introduzca mal una ruta (path).

```

public static void main(String[] args) {

    String ruta = "C:\\Ejercicios";
    File directorioRaiz = new File(ruta);

    if (!directorioRaiz.exists()) {
        System.out.println("La ruta proporcionada no existe.");
        return;
    }

    System.out.println("La ruta proporcionada es: "+ruta);

    System.out.println("Calculando el total de fichero y carpetas.....");
    int totalArchivos = 0;
    int totalCarpetas = 0;

    Queue<File> queue = new LinkedList<>();
    queue.add(directorioRaiz);

    while (!queue.isEmpty()) {
        File current = queue.poll();

        if (current.isDirectory()) {
            totalCarpetas++;
            File[] subFiles = current.listFiles();
            if (subFiles != null) {
                for (File subFile : subFiles) {
                    queue.add(subFile);
                }
            }
        } else {
            totalArchivos++;
        }
    }

    System.out.println("Total de ficheros: " + totalArchivos);
    System.out.println("Total de carpetas: " + totalCarpetas);
}

```



UNIVERSAE

— CHANGE YOUR WAY —

