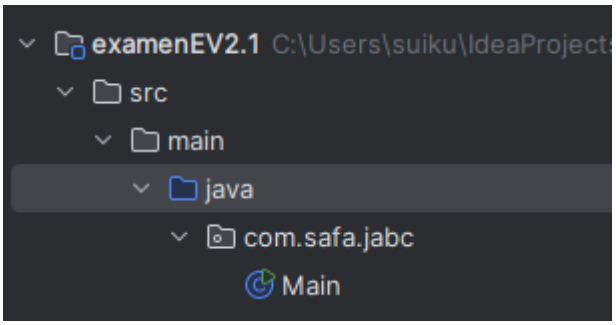
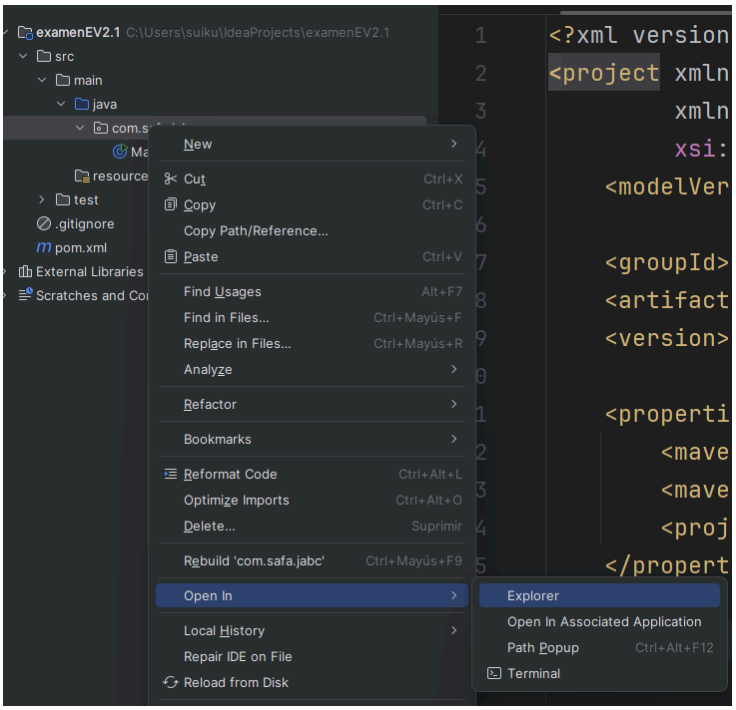
Examen de Programación: Segunda evaluación

Para realizar este examen se generará una única clase Java de nombre Main. Tanto esta clase Main como el resto de clases que se necesiten se generarán en src>main>java y luego un paquete que cumpla la siguiente nomenclatura: **com.safa.<tus_iniciales>** Por ejemplo, en mi caso sería **com.safa.jabc**.



El contenido de este paquete será lo que se comprima en un fichero .ZIP y se adjunte como entrega en Moodle.



En cada ejercicio se indicará el **nombre exacto de las clases y los métodos** (ya que podrán probarse con clases ajenas a vuestro main también). Si la función tiene otro nombre (aunque sea el mismo cambiando mayúsculas o minúsculas), no se contabilizará el ejercicio. **Salvo que se indique lo contrario, ninguna función imprimirá nada por consola**.

Ejercicio 0: Menú por pantalla (0.5 puntos)

En Main, muestra por pantalla un menú para elegir el ejercicio que se desea ejecutar (del 1 al 5, o 6 para terminar). Mientras la entrada no sea una entrada válida se deberá volver a mostrar el menú y preguntar qué se quiere hacer.
Cuando termine de ejecutarse una opción debe volver a mostrarse el menú.

Ejercicio 1: Saludo según la hora (1 punto)

Bien en un método de Main o bien en el bloque que se ejecute **al seleccionar la opción 1**, se inicializarán variables para pasar como parámetro al siguiente método, que será el que se llame (haz varias llamadas).

Hay que crear un método **public static String saludaSegunHora(List<String> listaNombres)** en la clase (que debéis crear) **UtilidadesExamen**, que admita una Lista con un número indeterminado de nombres de personas (String).

Si la hora actual es antes de las 6 o las 20 o más, devolverá "Buenas noches".
Si la hora actual son las 6 o más, pero menos que las 12, devolverá: "Buenos días".
Si la hora actual son las 12 o más, pero menos que las 20, devolverá: "Buenas tardes".

Además, dependiendo del número de parámetros, devolverá solo el saludo o lo acompañará de los nombres de la siguiente manera:

```
System.out.println(UtilidadesExamen.saludaSegunHora( listaNombres: null));
System.out.println(UtilidadesExamen.saludaSegunHora(new ArrayList<>()));
System.out.println(UtilidadesExamen.saludaSegunHora(List.of( e1: "Juan")));
System.out.println(UtilidadesExamen.saludaSegunHora(List.of("Juan","Nati","Vanesa")));
```

Deberá **devolver**:

```
Buenas noches.
Buenas noches.
Buenas noches, Juan
Buenas noches, Juan, Nati y Vanesa
```

Es importante respetar los signos de puntuación (terminar el saludo con el punto, separar el vocativo con una coma y, si hay más de un nombre, anteponer “y” al último elemento de la lista).

NOTA: Antepondremos “y” al último elemento, aunque este empiece por el sonido /i/. Lo correcto (salvo que sea un diptongo) sería poner “e”, pero para no complicar más el ejercicio.

Ej: “Buenas tardes, Juan y Inés.” (aunque lo correcto sería “Buenas tardes, Juan e Inés.”).

Ejercicio 2: Valida nombre de jugador (1 punto)

En la clase **UtilidadesExamen** se requiere la construcción de una función **validaNombreJugador** que tendrá un único parámetro de entrada (String) y devolverá un booleano según considere que el nombre de usuario es válido en la plataforma online de gaming. La validación deberá hacerse mediante el uso de una expresión regular.

Los criterios que se considerarán para ver si el nombre es válido serán:

1) Empiezan con una letra mayúscula o minúscula (A-Z, a-z).

2) Pueden contener números (0-9), guiones bajos (_) o puntos (.) en medio.

3) No pueden terminar con un guion bajo (_) o un punto (.).

4) Deben tener entre 5 y 15 caracteres en total.

Llama en la clase Main a este método con variedad de entradas para ver resultados **cuando se seleccione la opción 2**.

Ejercicio 3: Clase Producto (2.5 puntos)

Crea una clase **Producto** que tenga dos atributos nombre y codigo (ambos String). Esta clase deberá tener un constructor con todos los parámetros, un constructor copia y un constructor vacío. Además, habrá que implementar todo lo que se considere oportuno.

Los objetos de esta clase, se ordenarán por defecto por su nombre y, en caso de empate, por el código. Además, si se intentan añadir a un Set, no se podrán añadir duplicados (se considerará que, a todos los efectos, dos productos son iguales si su **código** es el mismo).

Crea, además, una clase **Valoracion**, que tendrá de atributos el producto al que se valora, una puntuacion (que será un enum de 1 a 5 estrellas) y un comentario, de tipo String. Las valoraciones se ordenarán por producto (por su misma ordenación).

En el método correspondiente de la clase Main **(la opción 3**) crea los ejemplos de producto que consideres necesarios para mostrar la ordenación en una lista, la inserción en un conjunto y el mostrarlos por consola (con formato **Nombre (Código)**).

Crea después las Valoraciones que consideres necesarias en base a los productos. Deben tener el siguiente formato:

```
Valoracion de: Televisor (TV-23X53_R)
        Puntuación: 4 estrellas
        Comentario: estoy muy contento con las nuevas funcionalidades de esta smart-TV.
```

Ejercicio 4: Uso de SoporteHeroes (3 puntos)

Utilizando la librería que se suministra heroes.jar (ver Anexo para documentación) crea, en **UtilidadesExamen**:

1. Un método **listAutoresVivos()** que devuelva una lista con los autores que no han fallecido. (0.5 puntos)
2. Un método **listObrasVivos()** que devuelva una lista con las obras cuyo autor o autores (todos si hay más de uno) estén vivos. (0.5 puntos)
3. Un método **listPersonajesFemeninos()** que devuelva una lista con los personajes de sexo femenino. Este método deberá utilizar un Iterador para eliminar los personajes no femeninos de la lista original. (0.5 puntos)
4. Un método **getPorcentajeFemeninos()** que devuelva un double (redondeado a 2 decimales) con el porcentaje de personajes femeninos respecto al total de personajes. (0.5 puntos)
5. Un método **muestraNoHumanos()**, que no devuelva nada, pero imprima por pantalla el resultado de listar los personajes no humanos que recupere, ordenados en principio por fecha de publicación de la obra (ascendente) y luego por nombre del personaje (ascendente). La información del personaje que habrá que mostrar será:

   Si tiene nombreIdentidad -> nombreIdentidad (nombre) - nombre de la Obra

   Si no tiene nombreIdentidad -> nombre - nombre de la Obra

   Ej:

   ```
   Ideáfix - Astérix
   Thor (Thor Odinson) - Los Vengadores
   ```

   El formato de impresión ha de ser como se muestra en la captura (1 punto)

Al elegir la **opción 4**, haz las llamadas necesarias para mostrar por consola los resultados de los 5 métodos, consecutivamente.

Ejercicio 5: Saiyanos (2 puntos)

Crea una clase Transformacion, que tendrá como atributos nombre (String) y poder (int).

Crea una clase Saiyano, que herede de Personaje.

Tendrá un atributo listaTransformaciones de tipo List<Transformacion>.

El método toString de Saiyano mostrará el nombre del personaje y, luego, entre parántesis, el poder de su Transformacion más poderosa.

Crea (en la clase Main o en una de utilidades que uses, a tu elección), una serie de transformaciones:

Transformaciones	
Nombre	**Poder**
SSJ	1000
SSJ2	10000
SSJ3	100000
SSJG	1000000
SSJGSJ	10000000
UI	100000000
UE	100000000
Beast	100000000
Definitivo	100000

De forma programática, recupera a todos los Personaje cuya raza es Saiyano o Saiyano Mestizo.

Crea una Lista de Saiyano que los incluya a todos, de modo que:

- **Son Goku** tiene SSJ, SSJ2, SSJ3, SSJG, SSJGSJ y UI
- **Vegeta** tiene SSJ, SSJ2, SSJG, SSJGSJ y UE
- **Son Gohan** tiene SSJ, SSJ2, Beast y Definitivo
- **Son Goten** y **Trunks** solo tienen SSJ.

Se creará en **UtilidadesExamen** un metodo **devuelveMapaSaiyano** que en base a la lista de Saiyanos con las características indicadas devolverá un Map cuya clave será el número de transformaciones y los valores la lista de Saiyanos con dicho número de transformaciones. El Map devuelto deberá estar ordenado por su clave y se mostrará su contenido por pantalla en la **opción 5**.

Anexo: librería heroes.jar

La librería heroes.jar incluye las siguientes clases (con atributos):

Autor

- String nombre
- Sexo sexo
- LocalDate fechaNacimiento
- LocalDate fechaDefuncion
- Pais pais

Obra

- String nombre
- List<Autor> autoria
- LocalDate primeraPublicacion
- TipoObra tipoObra

Personaje

- String nombre
- String nombreIdentidad
- Sexo sexo
- Raza raza
- Obra obra
- Bando bando
- Protagonismo protagonismo

NOTA nombreIdentidad refleja una identidad secreta o alterna del personaje (por ejemplo, de Usagi Tsukino sería Sailor Moon, o de Peter Parker sería Spider-Man).

Además, está la clase **SoporteHeroes** (de tipo Singleton), cuya única instancia se recupera mediante SoporteHeroes.getInstance()
Esta clase proporciona los métodos:

public List<Autor> getListaAutores(): devuelve la lista de autores precargados.

public List<Obra> getListaObras(): devuelve la lista de obras precargadas.

public List<Personaje> getListaPersonajes(): devuelve la lista de personajes precargados.

public Obra getObraPorNombre(String nombre): recupera una Obra por su nombre.

public Autor getAutorPorNombre(String nombre): recupera un Autor por su nombre