Tarea online

Título de la tarea: Trabajando con clases, objetos y métodos.

Unidad: 03

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2023/24

¿Qué contenidos o resultados de aprendizaje trabajaremos?

El principal resultado de aprendizaje que se va a trabajar con esta tarea será:

- RA2. Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

Además, se trabaja de forma parcial:

- RA5. Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases

Esto significa que se trabajará esencialmente con los siguientes contenidos:

- Creación de objetos.
- Uso de métodos y propiedades de los objetos.
- Incorporación y uso de librerías de objetos.
- Manejo de excepciones.

1.- Descripción de la tarea.

Caso práctico

Gerd Altmann (Pixabay License)

María está entrenándose en la programación orientada a objetos. Está empezando a realizar pruebas, creando objetos, invocando métodos, consultando propiedades, etc. En fin, familiarizándose poco a poco con el lenguaje Java y algunas de las bibliotecas o librerías que lleva incorporadas el paquete básico de desarrollo.

Ahora mismo está construyendo pequeñas aplicaciones para usar algunas clases que han realizado algunos de sus compañeros (clases `Teatro` y `Dado`), así como las clases relacionadas con el uso de las fechas y las horas en Java (`LocalDate`, `LocalTime`, `LocalDateTime`). También está practicando con el uso de la **documentación javadoc** que se ha generado para esas clases y está observando lo útil que puede llegar a ser disponer de este tipo de documentación. ¡Le están literalmente "salvando la vida"!

¿Qué te pedimos que hagas?

Las actividades a realizar se centrarán en el uso y manipulación de objetos mediante operaciones sencillas, trabajando con métodos de las clases `Teatro`, `Dado` y `LocalDate` para obtener una serie de resultados, **siendo obligatorio el uso de objetos de esas clases** según corresponda en cada ejercicio. Para ello, es vital que consultes la **documentación javadoc** de cada una de esas clases. Esa información podrás encontrarla bien a través de la documentación de la API de Java (en el caso de la clase `LocalDate`), bien a través de la documentación javadoc específica que se te proporciona en esta tarea (para el caso de las clases `Teatro` y `Dado`).

Debes usar obligatoriamente el proyecto NetBeans que se te proporciona en el apartado **información de interés** de esta tarea, el cual incluye una **biblioteca específica para esta tarea** con las clases `Teatro` y `Dado`, que no forman parte de la API de Java.

1.1.- Ejercicio 1. Trabajo con teatros.

La clase `Teatro` implementa un modelo de gestión de teatros y de obras que se representan en ellos. Es importante que le eches un vistazo a la **documentación javadoc** de esta clase antes de empezar a trabajar con ella. Así podrás ver qué tipo de información contiene, qué constructores proporciona, qué restricciones se le pueden aplicar, cuáles son los métodos disponibles, qué información puedes obtener a partir de un objeto, qué información puedes obtener a partir de la clase (métodos estáticos), etc. Recuerda que tendrás que hacer el `import` correspondiente para poder usar esta clase en tu programa.

12019 (Pixabay License)

Para practicar con la creación y utilización de objetos de este tipo, vamos a trabajar con **tres instancias del tipo** `Teatro` **.**

En el proyecto de trabajo que se proporciona para esta tarea dispones de un programa `Ejercicio01` que te servirá como base y donde tendrás que llevar a cabo las siguientes acciones paso a paso:

Declarar tres variables referencia a objetos instancia de la clase `Teatro`.
Llevar a cabo una **consulta inicial de valores globales de la clase** `Teatro`, de manera que muestre en pantalla la siguiente información utilizando la herramienta de E/S `System.out.printf`:

- **número de teatros creados** hasta el momento (debería obtenerse cero);
- **número de obras que se están representando** en este momento en todos los teatros (debería obtenerse cero);
- **número de entradas vendidas** en todos los teatros y para todas las obras que se han representado hasta el momento (debería obtenerse cero). Ten en cuenta que aún no existen objetos de esta clase, por lo que está claro que tendrás que invocar a **métodos estáticos** de la clase.

Instanciar tres objetos de la clase `Teatro` a los cuales apuntarán cada una de las variables declaradas anteriormente. Pero antes, haremos algunos intentos con errores para comprobar que funciona el lanzamiento de excepciones por parte de sus constructores:

Intentar crear un teatro con un aforo inferior al mínimo aforo definido. Recuerda que para instanciar un objeto de una determinada clase debes invocar a un constructor de esa clase mediante el operador `new`. Dado que en este caso sabes que la llamada al constructor va a fallar seguro, no tendrás más remedio que utilizar un bloque `try-catch` para capturar la excepción y gestionar ese error. Dentro del bloque `catch` deberás **mostrar por pantalla el mensaje de error que te proporcione la excepción** (mediante el método `getMessage` del objeto de tipo excepción que reciba el `catch`). Si no lo hace tu código, lo hará la máquina virtual de Java y tu programa se detendrá abruptamente ante el error, cosa que nunca debe suceder. **Tu código siempre debe tener previsto este tipo de posibles errores**.

Intentar crear un teatro con un aforo superior al máximo aforo definido. Para evitar el uso de literales en la llamada al constructor, se aconseja utilizar las constantes públicas de la clase. Al igual que en el caso anterior, deberás capturar la excepción.

Intentar crear un teatro con un nombre de teatro vacío y aforo por defecto. Tendrás que emplear una versión del constructor diferente a la usada en el caso anterior, de manera que el constructor asignará un estado por omisión. Nuevamente, tendrás que insertar otro bloque `try-catch` para capturar la excepción que se va a producir al invocar al constructor. Así evitarás que tu programa "aborte". No olvides mostrar el **texto del error por pantalla** (método `getMessage` de la excepción capturada).

Crear un primer teatro con un aforo válido usando el **constructor de dos parámetros**. Lo que devuelva el operador `new` al invocar al constructor es lo que tendrás que asignar a la variable referencia con la cual podrás manipular el objeto.

Instanciar un segundo teatro sin indicar el aforo. Usa para ello el **constructor de un parámetro**, el cual asignará una aforo por omisión. Emplea una segunda variable para apuntar al nuevo objeto creado.

Crear un tercer teatro con los valores por omisión. Utiliza para ello el **constructor sin parámetros**. En este caso, recurrirás a la tercera variable para apuntar al nuevo objeto instanciado. Recuerda que cuando decimos que una variable de tipo referencia va a "apuntar" a un objeto, significa simplemente que va a almacenar en una variable (de tipo referencia, obviamente) aquello que devuelva el operador `new` tras la invocación al constructor. Es decir, una referencia a la zona de memoria donde se encuentran realmente los atributos del objeto.

Llevar a cabo las siguientes **operaciones sobre los teatros:**

Intentar **terminar de representar una obra** en un teatro que no tiene todavía una obra asignada. Eso debería provocar una excepción de tipo `IllegalStateException` que deberías controlar dentro de un bloque `try-catch`, indicando algún mensaje de error (lo más fácil es que aproveches nuevamente el texto que te devuelve el método `getMessage` de la excepción capturada).

Asignar una obra al primer teatro. Una vez asignada la obra, deberás **llenar** el teatro, mostrando a continuación el número de entradas vendidas. Recuerda que debes **comprobar antes si puedes asignarla.** Una obra puede asignarse a un teatro si este aún no tiene ninguna obra asignada.

Devolver 50 entradas del primer teatro. Puesto que acabamos de llenar el teatro, tendremos suficientes entradas para devolver y, por tanto, no será necesario comprobar si podemos devolver entradas.

Intentar **traspasar** la representación de la obra **del primer teatro al segundo teatro.** Como el segundo teatro tiene un aforo inferior al número de entradas vendidas para la representación de la obra que se va a traspasar**,** se provocará una excepción que deberías controlar dentro de un bloque `try-catch`**, indicando el mensaje de error que te devuelve el método** `getMessage`**.**

Devolver otras 50 entradas del primer teatro. En este caso, tampoco tendremos que comprobar si podemos devolver entradas, ya que todavía tenemos suficientes entradas compradas.

Volvemos a intentar **traspasar** la representación de la obra **del primer teatro al segundo teatro**. Ahora **no debería dar ninguna excepción** ya que el aforo del segundo teatro puede albergar el número de entradas compradas.

Devolver una entrada del segundo teatro. Ten en cuenta que la clase cuenta con varios métodos para devolver entradas y deberás utilizar el más conveniente para este caso.

Obtener la siguiente información del segundo teatro y mostrarla por pantalla (usando `printf`). Para ello tendrás que usar algunos de sus métodos *getter*:

Fecha de realización de la prueba con la fecha y la hora actual. Para ello, utiliza el método correspondiente de la clase `LocalDateTime` que devuelve la fecha/hora actual. Deberá mostrarse con la fecha en formato `dd/mm/aaaa` y la hora en formato `hh:mm:ss`.

Código, nombre y aforo del teatro.

Estado del teatro. Si el teatro tiene una obra asignada, se mostrará la obra de teatro que se está representando y el número de entradas vendidas. Sin embargo, si el teatro no tuviera ninguna obra asignada, se debería mostrar el mensaje "No se está representando ninguna obra actualmente."

Consulta final de valores globales de la clase `Teatro`. Una vez finalizado el paso anterior, pasaremos a mostrar por pantalla (usando siempre `printf`) los resultados de la misma consulta que tuviste que realizar al principio del programa, aunque ahora los resultados deberán ser muy diferentes (ya no serán cero):

número de teatros creados hasta el momento**;**

número de obras que se están representando en este momento en todos los teatros;

número de entradas vendidas en todos los teatros y para todas las obras que se han representado hasta el momento.

IMPORTANTE: uso de `System.out.printf`

Para la realización de **este ejercicio y el resto de ejercicios de esta tarea**, siempre que tengas que mostrar algo por pantalla, deberás hacerlo usando la herramienta de E/S `System.out.printf` en lugar de `System.out.println` o `System.out.print`. Es decir, tendrás que usar la salida con formato utilizando los indicadores de formato `%s`, `%d`, `%f`, etc.

El único caso en el que podrás usar `print` o `println` será cuando vayas a escribir por pantalla un literal de cadena (un texto entre comillas), sin mostrar el contenido de ninguna variable. En estos casos el uso de `printf` no ofrece ninguna ventaja adicional.

Resultado del ejercicio

Dado que no hay que introducir datos de entrada y todos los valores son fijos, la ejecución de tu programa siempre deberá proporcionar el mismo resultado. Aquí tienes una muestra de cómo podría quedar:

Ocultar retroalimentación

```
TRABAJO CON TEATROS
-------------------

1.-CONSULTA INICIAL DE VALORES GLOBALES
---------------------------------------

Número de teatros creados hasta el momento: 0.
Número de obras que se están representando en este momento: 0.
Número de entradas vendidas hasta el momento: 0.

2.-CREACIÓN Y USO DE TEATROS
----------------------------

Creación de teatros (constructores)
-----------------------------------
Intentando crear un teatro con un aforo inferior...
Error: Aforo incorrecto: 299

Intentando crear un teatro con un aforo superior...
Error: Aforo incorrecto: 1001

Intentando crear un teatro con un nombre vacío y aforo por defecto...
Error: El nombre del teatro no puede ser cadena vacía.
```

```
Creando teatro con aforo válido con un constructor con dos parámetros...
Teatro 1 creado, estado: { Nombre del teatro: Teatro Cervantes; Código del teatro: 1; Aforo: 900; no tiene  una obra asignada; Representando la obra:

Creando teatro con aforo por defecto usando un constructor con un parámetro...
Teatro 2 creado, estado: { Nombre del teatro: Teatro Apolo; Código del teatro: 2; Aforo: 800; no tiene  una obra asignada; Representando la obra: ---

Creando teatro con valores por defecto usando un constructor sin parámetros...
Teatro 3 creado, estado: { Nombre del teatro: Teatro 3; Código del teatro: 3; Aforo: 800; no tiene  una obra asignada; Representando la obra: ---; Er

Manipulación de teatros (métodos)
---------------------------------
Terminando de representar obra en el primer teatro...
Error: El teatro no tiene una obra asignada que se pueda finalizar.

Asignando una obra de teatro...
Se ha asignado la obra 'La vida es sueño' al teatro 'Teatro Cervantes'.Teatro lleno. Se han vendido 900 entradasIntentando traspasar una obra de un t
Error: Se supera el aforo del teatro de respaldo, no se puede realizar el traspaso.


Prueba de los getters del segundo teatro creado:
------------------------------------------------
Fecha de realización de la prueba: 19/10/2023 22:29:16

Teatro 2
    Código del teatro: 2    Nombre del teatro:"Teatro Apolo"
    Aforo: 800
    Estado:
      Se está representando la obra de teatro: "La vida es sueño"
      Entradas vendidas: 799

3.-CONSULTA FINAL DE VALORES GLOBALES
-------------------------------------

Número de teatros creados hasta el momento: 3.
Número de obras que se están representando en este momento: 1.
Número de entradas vendidas hasta el momento: 799.
```

Recuerda que para resolver esta actividad debes utilizar obligatoriamente instancias de la clase `Teatro` junto con sus métodos. Y no olvides que tendrás que hacer el `import` correspondiente para poder usar esa clase en tu programa.

1.2.- Ejercicio 2. Lanzamiento de dados.

Usando la clase `Dado`, que se incluye en el proyecto de trabajo, construir a partir de la plantilla que se os proporciona (`Ejercicio02`), un programa de prueba que simule un juego de **lanzamiento de dados**.

Alexas_Fotos (Pixabay License)

En el juego participarán **tres jugadores**, **cada uno lanzará un dado de seis caras** (que contendrá los números del 1 al 6), por tanto necesitaremos **tres dados de seis caras cada uno**.

Una vez que tengamos los dados **se lanzarán y se sumará la cifra obtenida por cada dado**, **volverán a lanzarse los dados y se acumulará la cifra** obtenida en la tirada anterior, y así sucesivamente. El juego terminará **cuando la suma de todos los lanzamientos alcance** un valor máximo, que será **un número aleatorio entre 30 y 60**.

Los **pasos que debes seguir** son:

Calcular el **número máximo de puntos**, que será un **número aleatorio comprendido entre 30 y 60**.
Crear los jugadores, es decir, **tres objetos de tipo `Dado` de seis caras cada uno**.
Lanzar todos **los dados** para sumar las puntuaciones obtenidas en el lanzamiento e ir acumulando las puntuaciones de todos los lanzamientos hasta alcanzar el número máximo de puntos:

- **Lanzar cada uno de los dados** tantas veces como sea necesario, y mostrar las puntuaciones obtenidas en cada uno de ellos. Para numerar los lanzamientos, **utiliza los métodos disponibles en la clase `Dado`**, no es necesario el uso de un contador.
- **Acumular las puntuaciones de todos los dados en todos los lanzamientos**, controlando que **no se supere el número máximo**. Utiliza para ello los métodos de la clase.

Comprobar cuál de los dados ha sido el ganador (será el dado que, en la última tirada, haya sacado la mayor puntuación). Para ello, puedes suponer que el ganador es el dado 1 y comprobar si el resto de dados superan su puntuación para actualizar los valores (no es necesario tener en cuenta los empates, en ese caso puedes designar cualquiera de los dados empatados como ganador). Una vez que lo tengas, consulta todos sus lanzamientos.
Por último, **mostrar todos los resultados**:

- El **número máximo** de puntos.
- Todos los **lanzamientos** de los tres dados.
- La **suma de puntuaciones** de todos los lanzamientos realizados.
- El **dado ganador y** los **puntos** obtenidos en el **último lanzamiento**.
- El **número de veces que ha salido en todo el juego la puntuación obtenida por el dado ganador**.
- El **número total de lanzamientos** que se han realizado entre todos los dados.
- Los **puntos de todos los lanzamientos del dado ganador**, expresados como **cadena de texto**.

No utilices más variables de las necesarias en tu programa, ya que su uso será penalizado.

Recuerda que tendrás que hacer el `import` correspondiente para poder usar la clase `Dado` en tu programa. **Toda la información que requieras la podrás obtener consultando a los métodos** apropiados de los objetos `Dado`, que para eso los tienes. La realización de cálculos para obtener información que ya ofrecen los métodos de la clase será también penalizado.

Evita al máximo el uso de literales en el código y procura obtener los valores que necesites, si es posible, a partir del objeto con el que estás trabajando.

Ejemplos de ejecución

Dado que:

1. no hay que introducir datos de entrada;
2. el número máximo de puntos que se puede alcanzar será aleatorio (entre 30 y 60);
3. cada vez que se lanza un dado, el resultado es desconocido (aleatorio);

la ejecución de tu programa será imprevisible y cada vez que lo ejecutes proporcionará un resultado diferente. Para que se repita un mismo resultado tendrá que ser mucha coincidencia.

Para que puedas comparar entre los resultados que aparezcan tras la ejecución de tu programa y algunas de la infinidad de posibles salidas que podrían obtenerse, te proporcionamos tres posibles ejemplos de las decenas de posibilidades que podrían producirse y que dependerán del azar. Recuerda que se trata de algunas de las muchas posibilidades que pueden darse, así que no te empeñes en que tus resultados coincidan con alguno de estos. Podrías necesitar decenas de ejecuciones para obtener el mismo resultado. Se trata de que los resultados que tú obtengas tengan sentido, del mismo modo que lo tienen los que te vamos a ofrecer.

Primera prueba de ejecución

Ocultar retroalimentación

```
LANZAMIENTO DE DADOS
----------------------------------------
Número máximo de puntos: 47

                  DADO1   DADO2   DADO3

Lanzamiento nº  1:   5       1       6
Lanzamiento nº  2:   6       2       2
Lanzamiento nº  3:   4       6       6
Lanzamiento nº  4:   2       4       5

Juego Terminado. La suma de los lanzamientos es: 49.
El ganador es el dado 3 con 5 puntos en la última jugada.
El valor 5 ha salido 2 veces en todo el juego y se han realizado un total de 12 lanzamientos.
Todos los lanzamientos del dado 3: SEIS DOS SEIS CINCO.
```

Como puedes observar, en este ejemplo de ejecución se han realizado **4 lanzamientos** para alcanzar una puntuación igual o superior a 47 (en concreto, 49 puntos).

Segunda prueba de ejecución

Ocultar retroalimentación

```
LANZAMIENTO DE DADOS
----------------------------------------
Número máximo de puntos: 52
```

```
                  DADO1   DADO2   DADO3

Lanzamiento nº  1:   6       4       3
Lanzamiento nº  2:   1       4       1
Lanzamiento nº  3:   3       1       1
Lanzamiento nº  4:   4       1       4
Lanzamiento nº  5:   5       6       3
Lanzamiento nº  6:   4       4       6

Juego Terminado. La suma de los lanzamientos es: 61.
El ganador es el dado 3 con 6 puntos en la última jugada.
El valor 6 ha salido 3 veces en todo el juego y se han realizado un total de 18 lanzamientos.
Todos los lanzamientos del dado 3: TRES UNO UNO CUATRO TRES SEIS.
```

En este caso, el azar ha querido que se necesiten **6 lanzamientos** para alcanzar una puntuación igual o superior a 52 puntos (concretamente 61 puntos).

Tercera prueba de ejecución

Ocultar retroalimentación

```
LANZAMIENTO DE DADOS
----------------------------------------
Número máximo de puntos: 59

                  DADO1   DADO2   DADO3

Lanzamiento nº  1:   5       6       2
Lanzamiento nº  2:   1       2       5
Lanzamiento nº  3:   6       5       1
Lanzamiento nº  4:   5       5       6
Lanzamiento nº  5:   1       4       5

Juego Terminado. La suma de los lanzamientos es: 59.
El ganador es el dado 3 con 5 puntos en la última jugada.
El valor 5 ha salido 6 veces en todo el juego y se han realizado un total de 15 lanzamientos.
Todos los lanzamientos del dado 3: DOS CINCO UNO SEIS CINCO.
```

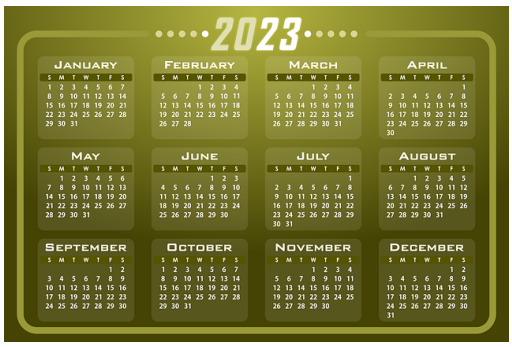
En este último ejemplo de ejecución, se han realizado **5 lanzamientos** para alcanzar una puntuación igual a 59 (momento en el que se alcanza la condición de parada).

Recuerda que a lo largo de esta actividad, siempre que tengas que mostrar algo por pantalla, deberás hacerlo usando la herramienta de E/S `System.out.printf` en lugar de `System.out.println` o `System.out.print`. Es decir, tendrás que usar la salida con formato utilizando los indicadores de formato `%s`, `%d`, etc.

Para respetar la estructura de los programas (Entrada de datos - Procesamiento - Salida de datos) utiliza el método `String.format`, que permite construir cadenas de texto con formato para mostrarlas después en la salida de datos.

1.3.- Ejercicio 3. Día de Cumpleaños.

El cumpleaños de **María** es el **18 de noviembre**. Este año, aprovechando que cae en **sábado**, quiere hacer una gran fiesta con su familia y amistades para celebrarlo. María nació en el año **1989**, la madrugada de un sábado, y se pregunta cuántas veces su cumpleaños ha caído en sábado. ¡Vamos a ayudarle a averiguarlo!



Rafael Javier (Licencia Pixabay)

Escribe un programa en Java que solicite una fecha de cumpleaños mediante la petición de tres números enteros: día, mes, año. A partir de estos datos, el programa debe mostrar el número de veces que esa fecha ha caído en el mismo día de la semana. Además, también tiene que indicar los años en los que ha ocurrido. Para ello tendrás que:

1. Solicitar el año por teclado, que debe encontrarse **entre 1900 y el año actual** (ambos incluidos). Si no se introduce un año correcto, bien porque no se introduzca un número entero válido (control de la excepción `InputMismatchException`), bien porque no esté dentro de los límites permitidos, se volverá a solicitar hasta que el año sea correcto. Se hará tantas veces como sea necesario.
2. Una vez introducido un año correcto, se procederá a la petición del mes de nacimiento. De la misma forma que hemos hecho con el año, también tenemos que validar el mes: hay que comprobar que se introduce un número entero válido (control de la excepción `InputMismatchException`) y que se encuentra dentro de los límites permitidos (entre 1 y 12). Se hará tantas veces como sea necesario.
3. A continuación, solicitamos el día de nacimiento. Aquí también llevaremos a cabo un proceso de validación: controlamos que se introduce un número entero válido (control de la excepción `InputMismatchException`) y que se encuentra en el rango correcto. Es decir, tenemos que revisar el máximo número de días dependiendo del mes (30 ó 31, y para el caso de febrero, 28 ó 29). Ten en cuenta que no es necesario que calcules de forma manual si un año es bisiesto o no, puesto que la clase `LocalDate` tiene métodos que te dan esta información. De nuevo, se hará tantas veces como sea necesario.
4. Cuando ya tenemos la fecha validada, averiguamos **qué día de la semana corresponde a esa fecha y el número de veces que el cumpleaños ha caído en ese día.** Además, imprimimos los años en los que ha ocurrido esta situación, mostrando las fechas en formato dd/mm/aaaa.

Una manera sencilla de resolver este ejercicio es ir recorriendo todos los años y comprobando (con objetos de tipo `LocalDate`) para cuáles de ellos el día y mes de cumpleaños cae en el mismo día de la semana en que se produjo el nacimiento. Ten en cuenta que, si el día de nacimiento es un 29 de febrero, cuando intentes comprobar un año que no sea bisiesto, te va a dar un error al crear el objeto de tipo `LocalDate`. Para evitarlo, antes de crear el objeto `LocalDate`, deberías asegurarte de que no es un 29 de febrero de un año no bisiesto.

Ten en cuenta que:

La clase `LocalDate` **no dispone de constructores públicos**, de manera que tendrás que usar los **métodos "fábrica" o pseudoconstructores** que proporcione para crear objetos instancia de esta clase.

Tienes que mostrar las fechas en formato *dd/mm/aa.* Para ello, estaría bien que revisaras el funcionamiento de la clase `DateTimeFormatter`, así como del método `format` de la clase `LocalDate`.

Además, la clase `LocalDate` cuenta con los siguientes métodos:

- `now`, que permite crear instancias a partir de la fecha actual.
- métodos `getXXX`, que permiten extraer cualquier parte de una fecha. Por ejemplo: `getYear` o `getDayOfWeek`. Este último método obtiene el día de la semana devolviendo un `enum` de tipo `DayOfWeek`.
- `lengthOfMonth`, que devuelve la duración del mes representado por la fecha dada, incluso contemplando los años bisiestos.

- `isLeapYear()`, que devuelve si un año es bisiesto o no.

¿Cómo obtener los nombres de los días de la semana sin tener que escribirlos en el código?

Dado un objeto instancia de la clase `LocalDate`, que representa una **fecha**, podemos obtener el día de la semana de esa fecha mediante el método `geDayOfWeek()`.

En este caso se devuelve un `enum` de tipo `DayOfWeek` (con valores de tipo `DayOfWeek.MONDAY`, `DayOfWeek.TUESDAY`, `DayOfWeek.WEDNESDAY`, etc.). Ten en cuenta que los `enum` son en el fondo clases y, por tanto, Java entiende cualquier valor `enum` como un objeto al cual se le puede aplicar el método `toString`. La cadena devuelta coincidirá con el identificador del valor específico del `enum` (normalmente el nombre del día en mayúsculas y en inglés: `"MONDAY"`, `"TUESDAY"`, `"WEDNESDAY"`, etc.). En consecuencia, para obtener el nombre del día de la semana, dada una variable `fecha`, que apunta a un objeto de tipo `LocalDate`, bastaría con hacer algo como:

```
String nombreDia= fecha.getDayOfWeek.toString();
```

Con esto es más que suficiente para resolver el ejercicio.

Ahora bien, si además quieres que los nombres de los días aparezcan en otra lengua (en este caso en español) y con algún estilo concreto (nombre completo, abreviado, etc.), entonces tienes otra posibilidad. En lugar de hacer un simple `toString`, la clase `DayOfWeek` también dispone de un método más sofisticado llamado `getDisplayName`, que permite obtener representaciones textuales del día de la semana más elaboradas:

Método `getDisplayName` de la clase `DayOfWeek`

¿Cómo lo harías usando el método `getDisplayName`?

Ocultar retroalimentación

El método `getDisplayName` necesita dos parámetros:

1. el **"estilo" del texto** (abreviado, completo, una sola letra, etc.): un `enum` de tipo `TextStyle`;
2. la **configuración regional o local**: un objeto de tipo `Locale`.

Para el primer parámetro podemos usar el valor `TextStyle.FULL`, que consiste en el nombre completo, sin abreviaturas.

Para el segundo podemos crear un objeto `Locale` con la configuración regional del español de España: `new Locale("es", "ES")`.

Por tanto, el nombre del día de la semana en español podríamos obtenerlo haciendo lo siguiente:

```
String nombreDia= fecha.getDayOfWeek().getDisplayName(TextStyle.FULL, new Locale("es", "ES")
```

Un objeto de tipo `Locale` representa una combinación particular de idioma, región y cultura. Se utiliza para implementar **programas internacionalizados**. Se trata de programas que, dependiendo de la parte del mundo en que se ejecuten, mostrarán información o formatos contextuales específicos para esa zona. Algunos ejemplos podrían ser el formato de los números reales (punto o coma decimal), los nombres de los días de la semana o los meses del año, el formato de la fecha, etc.

Puedes encontrar más información sobre la clase `Locale` en la documentación de la API de Java:

Clase `Locale` del paquete `java.util`

Ejemplos de ejecución

A continuación te proporcionamos algunos ejemplos de cómo podría quedar la ejecución del programa dependiendo de las posibles entradas que se proporcionen.

Ejemplo 1: 28/12/2000

Ocultar retroalimentación

```
DÍA DE CUMPLEAÑOS
-----------------
Introduzca año de nacimiento (1900-2023):
x
Error de lectura: año incorrecto.
Introduzca año de nacimiento (1900-2023):
1890
Error de lectura: año incorrecto.
Introduzca año de nacimiento (1900-2023):
2000
Introduzca mes de nacimiento (1-12):
13
Error de lectura: mes incorrecto.
Introduzca mes de nacimiento (1-12):
12
Introduzca día de nacimiento:
28
El día que naciste fue jueves

¿Cuántas veces tu cumpleaños ha caído en jueves?
--------------------------------------------------
 1. 28/12/2006
 2. 28/12/2017
 3. 28/12/2023

Número total de coincidencias: 3
```

Como puedes observar, si se introducen números fuera de los rangos permitidos se vuelve a solicitar la información. Por otro lado, si se introduce algo diferente a un número entero, el programa no debería "romperse" debido a una `InputMismatchException`, sino que la excepción debería ser capturada y convenientemente gestionada para a continuación volver a pedir el dato.

Ejemplo 2: 30/12/1983

Ocultar retroalimentación

```
DÍA DE CUMPLEAÑOS
-----------------
Introduzca año de nacimiento (1900-2023):
1983
Introduzca mes de nacimiento (1-12):
12
Introduzca día de nacimiento:
30
El día que naciste fue viernes

¿Cuántas veces tu cumpleaños ha caído en viernes?
-----------------------------------------------------
 1. 30/12/1988
 2. 30/12/1994
 3. 30/12/2005
 4. 30/12/2011
 5. 30/12/2016
 6. 30/12/2022

Número total de coincidencias: 6
```

Ejemplo 3: 29/02/1924

Ocultar retroalimentación

```
DÍA DE CUMPLEAÑOS
-----------------
Introduzca año de nacimiento (1900-2023):
1924
Introduzca mes de nacimiento (1-12):
2
Introduzca día de nacimiento:
29
El día que naciste fue viernes

¿Cuántas veces tu cumpleaños ha caído en viernes?
-----------------------------------------------------
 1. 29/02/1952
 2. 29/02/1980
 3. 29/02/2008

Número total de coincidencias: 3
```

2.- Información de interés.

Recursos necesarios y recomendaciones

Debes usar obligatoriamente el proyecto base NetBeans que te proporcionamos aquí. Ese proyecto incluye una **biblioteca específica para esta tarea** (`libtarea3`) que contiene las clases `Teatro` y `Dado`. Esas clases no forman parte de la API de Java.

mohamed Hassan (Licencia Pixabay)

Es vital que utilices este proyecto porque si no lo haces no podrás implementar correctamente tus programas al no disponer de esas clases ni de la configuración necesaria de tus bibliotecas.

Además, este proyecto contiene tres programas (clases `Ejercicio01`, `Ejercicio02` y `Ejercicio03`) correspondientes a cada uno a los ejercicios que debes implementar. Cada uno tendrá su propio `main` con el esqueleto del programa y algunas indicaciones de las pautas que debes seguir y así orientarte un poco el trabajo que debes realizar. ¡Aprovéchalos!

Descarga y utiliza este proyecto. ¡No crees tu propio proyecto en Netbeans! Si lo haces, tu tarea será considerada como NO APTA pues no tendrá las características que necesitamos que tenga.

Descargar proyecto base Netbeans para la tarea 3 (zip - 129 kb) .

Una vez descargado el proyecto, renómbralo con el estilo de nombre que solemos darle al paquete de entrega de la tarea:

Apellido1_Apellido2_Nombre_PROG03_Tarea

Y ya podrás ponerte a trabajar sobre él. ¡Ánimo!

Una recomendación importante: uso de `printf`

A lo largo de los distintos ejercicios de esta tarea se te exigirá utilizar la herramienta de E/S `System.out.printf` en lugar de `System.out.println` o `System.out.print`, es decir, que tendrás que usar la salida con formato utilizando los indicadores de formato `%s`, `%d`, `%f`, etc. Si no recuerdas bien cómo funciona, es un buen momento para repasar la sección correspondiente en los contenidos de la unidad.

El único caso en el que podrás usar `print` o `println` será cuando vayas a escribir por pantalla un literal de cadena (un texto entre comillas), sin mostrar el contenido de ninguna variable. En estos casos el uso de `printf` no ofrece ninguna ventaja adicional.

Aunque en el proyecto base que se te proporciona ya contiene la **documentación JavaDoc** de las clases `Teatro` y `Dado`. Te las proporcionamos también aparte por si quieres disponer de ellas para consultarlas independientemente del IDE Netbeans:

Descargar JavaDoc de las clases Teatro y Dado para la tarea 3 (zip - 336 kb)

Una vez descargado y descomprimido el paquete debes abrir con un navegador web el archivo `index.html` para empezar a navegar por el JavaDoc de esta API y consultar la documentación de cada clase.

Para consultar la documentación JavaDoc de las clases `LocalDate` y `LocalDateTime` basta con que consultes en la documentación general de la biblioteca de clases de la API de Java.

Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto NetBeans en un fichero `.zip` y nómbralo siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG03_Tarea

Recuerda que **tu proyecto Netbeans también debe llamarse así** (tienes que renombrarlo respecto al nombre original del proyecto base que vas a utilizar)

3.- Evaluación de la tarea.

Criterios de evaluación implicados

En esta unidad se va a evaluar el resultado de aprendizaje RA2. "Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos." y algunos criterios de evaluación correspondientes al resultado de aprendizaje RA5 "Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases". Ello se hará mediante los siguientes Criterios de Evaluación correspondientes a cada resultado de aprendizaje citado:

Criterios de Evaluación correspondientes al **RA2**:

- RA2.a) Se han identificado los fundamentos de la programación orientada a objetos.
- RA2.b) Se han escrito programas simples.
- RA2.c) Se han instanciado objetos a partir de clases predefinidas.
- RA2.d) Se han utilizado métodos y propiedades de los objetos.
- RA2.e) Se han escrito llamadas a métodos estáticos.
- RA2.f) Se han utilizado parámetros en la llamada a métodos.
- RA2.g) Se han incorporado y utilizado librerías de objetos.
- RA2.h) Se han utilizado constructores.
- RA2.i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.

Peggy_Marco (Pixabay Licens

Criterios de Evaluación correspondientes al **RA5**:

- RA5.a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.
- RA5.b) Se han aplicado formatos en la visualización de la información.
- RA5.c) Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.

¿Cómo valoramos y puntuamos tu tarea?

A continuación mostraremos los elementos que podrán penalizar en la evaluación de la tarea. Tenlos muy en cuenta al resolver los ejercicios:

1. **Los programas deben compilar**. Cualquier funcionalidad pedida en el enunciado debe poderse probar y ha de funcionar correctamente de acuerdo a las especificaciones del enunciado. **Si el programa ni siquiera compila, el ejercicio podría considerarse nulo y su puntuación podría llegar a ser 0.**
2. Se utilizan **estructuras de salto incondicional** para el **control del flujo**, o bien herramientas como return, exit(0) o cualquier otro mecanismo que no sea el **fin de la ejecución natural del programa**.
3. No se respeta la **estructura de bloques** propuesta en la plantilla: **entrada de datos**, **procesamiento** y **salida de resultados** (ni siquiera una versión adaptada a las necesidades del problema).
4. **Algunos identificadores no cumplen con el convenio sobre "asignación de nombres a identificadores"** establecido para el lenguaje Java para constantes, variables, métodos, clases, etc.

5. **Se declaran identificadores que no tienen nombres significativos o descriptivos** que representen de alguna manera la información que están almacenando para que el código quede lo más claro, legible y autodocumentado posible.
6. No se observa una **corrección ortográfica y gramatical**, así como la **coherencia en las expresiones lingüísticas**, tanto en los **comentarios en el código** como en los **textos de los mensajes** que aparezcan en pantalla para pedir información de entrada al usuario o para mostrar resultados de salida. Deben evitarse **mensajes de entrada de datos y/o salida de resultados** inapropiados, descontextualizados, insuficientes o incorrectos.
7. **El código no está correctamente indentado. Es fundamental para poder observar e intuir rápidamente, y de forma visual, la estructura de los programas. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "*formatear*" o "*embellecer*" y pulsa Alt+Mayús.+F).**

En esta otra tabla puedes observar la puntuación máxima asignada a cada ejercicio de la tarea así como los principales elementos que se tendrán en cuenta a la hora de corregir cada uno de ellos:

Rúbrica de la tarea	
Punto de Control 1: Se han identificado los fundamentos de la programación orientada a objetos, y utilizado en la creación de programas simples. • RA2.a) Se han identificado los fundamentos de la programación orientada a objetos. • RA2.b) Se han escrito programas simples.	9,62%
Punto de Control 2: Se han instanciado objetos a partir de clases predefinidas, la creación de estos objetos se hace con sentido para la resolución del problema planteado en cada caso, sin la creación excesiva de objetos y controlando las necesidades que solicita cada problema particular. • RA2.c) Se han instanciado objetos a partir de clases predefinidas.	3,85%
Punto de Control 3: Se han utilizado métodos y propiedades de los objetos en el desarrollo de la soluciones de los ejercicios propuestos, no sólo su definición sino también su utilización coherente. • RA2.d) Se han utilizado métodos y propiedades de los objetos.	5,77%
Punto de Control 4: Se han escrito llamadas a métodos estáticos, entendiendo su uso como métodos de clase y no de objeto, y facilitando de esta forma la resolución de cada problema concreto. • RA2.e) Se han escrito llamadas a métodos estáticos.	3,85%
Punto de Control 5: Se han utilizado parámetros en la llamada a métodos. La llamada a los método tanto de librería de clases de Java, como implementadas por el alumno/a incorporarán los parámetros correctamente definidos y verificados antes de su uso, eligiendo en cada	3,85%

caso la llamada con los parámetros que más se ajusten a las necesidades de cada problema. • RA2.f) Se han utilizado parámetros en la llamada a métodos.	
Punto de Control 6: Se han importado librerías de la API de Java descritas en la unidad que facilitan resolución de los problemas mediante la utilización de objetos y métodos conociendo su alcance y uso correcto, eligiendo el método o métodos de las librerías más apropiados. • RA2.g) Se han incorporado y utilizado librerías de objetos.	3,85%
Punto de Control 7: En la resolución de los problemas propuestos se han utilizado constructores más apropiados en cada caso particular, bien creados por el alumno/a, o bien haciendo uso de los ya definidos en las librerías de API de Java que se ha visto en la unidad. • RA2.h) Se han utilizado constructores.	3,85%
Punto de Control 8: Para la resolución de la tarea se utiliza un entorno integrado de desarrollo (IDE) en la creación y compilación de programas simples, en este caso NetBeans, presentando un proyecto agrupando la resolución de la tarea en archivos independientes dentro del proyecto. • RA2.i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples	3,85%
Punto de Control 9: La entrada salida de datos se realiza mediante la consola del IDE utilizando las clases necesarias y salvando los posibles errores que pudiesen generar de su uso en los diferentes problemas planteados. • RA5.a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información. • RA5.c) Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.	7,69%
Punto de Control 10: Los resultados y la salida de datos se presentan acorde con los formatos de visualización de la información requeridos en cada problema. • RA5.b) Se han aplicado formatos en la visualización de la información.	3,85%
Punto de Control 11: Los ejercicios desarrollados funcionan correctamente de acuerdo a las especificaciones y limitaciones requeridas en la tarea. • RA2.a) Se han identificado los fundamentos de la programación orientada a objetos. • RA2.b) Se han escrito programas simples. • RA2.c) Se han instanciado objetos a partir de clases predefinidas.	50,00%

• RA2.d) Se han utilizado métodos y propiedades de los objetos. • RA2.e) Se han escrito llamadas a métodos estáticos. • RA2.f) Se han utilizado parámetros en la llamada a métodos. • RA2.g) Se han incorporado y utilizado librerías de objetos. • RA2.h) Se han utilizado constructores. • RA2.i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples • RA5.a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información. • RA5.b) Se han aplicado formatos en la visualización de la información. • RA5.c) Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.	