PROGRAMACIÓN IN1W examen problemas, 1ª ev. 01.12.20

NOMBRE .....

Escribir los programas en JAVA que se indican a continuación:

Créate un único proyecto con el nombre **ApellidosEval1**, con tus apellidos obviamente.

Las clases que contengan el método main() para cada ejercicio deben llamarse **ApellidosExNum.java** donde Num es el número del ejercicio (por ejemplo LopezPerezEx1.java).

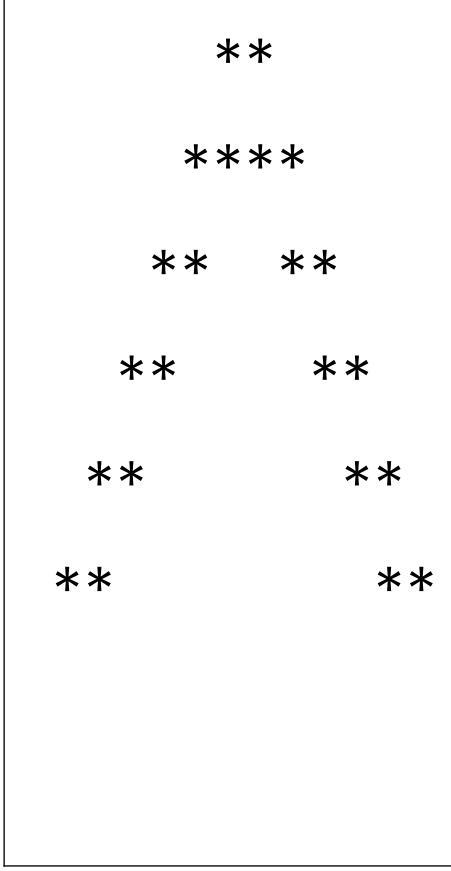
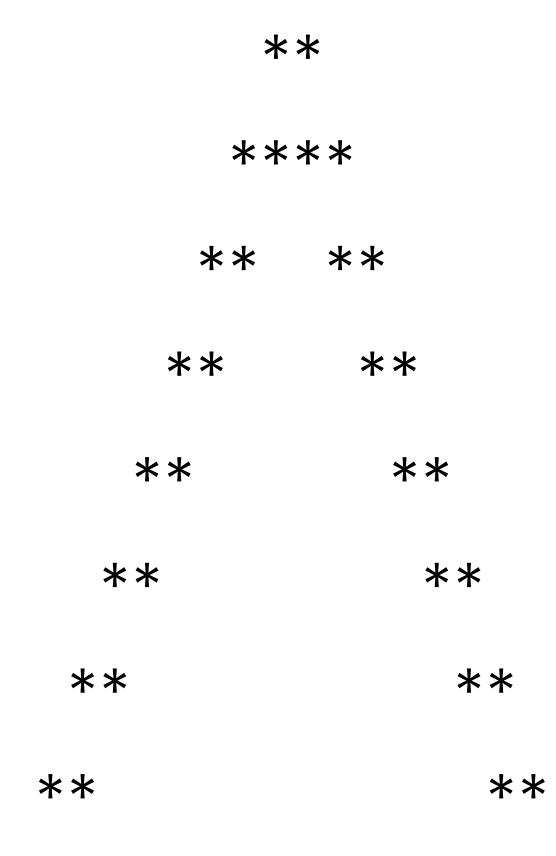
Al finalizar el examen, genera un fichero comprimido que contenga todo el **proyecto** (no el workspace) y lo envías por correo electrónico a la dirección y **y** daniel.val@cpilosenlaces.com

Usa la clase **Leer** suministrada y guárdala en el mismo proyecto. Si tienes tu propia clase Leer, cópiala en el proyecto de tu examen y envíala al final dentro del fichero zip.

ES RESPONSABILIDAD TUYA ENVIAR CORRECTAMENTE LA SOLUCIÓN. Si no es legible o no cumple lo anteriormente expuesto, el examen no podrá ser corregido y la nota será 1.

Ejercicio 1.- (3 puntos)

Crea un programa que solicite un número entero mayor que 3, que será la altura de la letra lambda mayúscula y deberá generar la siguiente salida para los valores del entero introducido:

6	8
	

PROGRAMACIÓN IN1W examen problemas, 1ª ev. 01.12.20

NOMBRE .....

Ejercicio 2.- (7 puntos)

En una tienda de bicicletas se quiere llevar el control de existencias y ventas que se realizan.

De las bicicletas se guarda el **modelo**, nº de **platos**, nº de **piñones**, tipo de **ruedas** (descripción de las mismas: llantas, radios, aros), **precio** y **cliente**. El cliente estará a null cuando la bicicleta esté sin vender.

Del cliente se guarda un **código** que se **autoincremente** a partir de un atributo static, **nombre**, **teléfono** y la lista de **bicicletas** compradas. Al crear el cliente no tendrá bicicletas compradas.

Haced una carga inicial con los datos facilitados antes de iniciar el menú.

La aplicación **ApellidosEx2.java** tendrá **7** opciones: Crear cliente, Crear bicicleta, Modificar bicicleta, Vender bicicleta, Listado de bicicletas sin vender, Listado de bicicletas vendidas, Listado de clientes con sus compras.

La opción de modificar bicicleta se recomienda gestionarla con un submenú.

Datos iniciales para cliente:

- Miguel Induráin, 676 234 123
- Shakira, 665 234 111
- Joane Somarriba, 664 212 278

Datos iniciales para bicicletas:

- Mountain BK1, 3, 2, llantas anti-pinchazo de 40 radios y aros dobles, 345.23
- Carretera M1, 4, 3, cámara anti-pinchazo de 36 radios y aros dobles, 280.56
- Pista P3, 3, 3, Lenticulares, 470.66
- Paseo S2, 2, 2, 30 radios aro simple, 180.20
- Iniciación, 1, 1, 24 radios aro simple, 80.12
- Carretera M2, 3, 2, cámara anti-pinchazo de 36 radios y aros simples, 210.23

Recomendación de orden para el desarrollo:

1. Crea las clases Bicicleta y Cliente con los métodos básicos habituales.
2. Crea la clase **ApellidosEx2.java** con el método main() con el menú.
3. Crea la opción de rellenar los vectores de Bicicleta y Cliente con los datos facilitados.
4. Desarrolla las opciones de los listados formateados.
5. Desarrolla las opciones de Crear bicicletas y Crear cliente.
6. Crea la opción de vender.
7. Crea la opción de modificar la bicicleta.

Valoración: (Modificación de bicicleta y el listado de bicicletas por cliente suman más allá del 7)

- **Diseño de las clases 2 puntos.**
- **Listados 1.5** (+1) puntos.
- **Vender 1 punto.**
- **Carga inicial 1 punto.**
- **Crear clientes y bicicletas 1.5 punto.**
- Modificación 1 punto.

PROGRAMACIÓN IN1W examen problemas, 1ª ev. 01.12.20

NOMBRE .....

Escribir los programas en JAVA que se indican a continuación:

Créate un único proyecto con el nombre **ApellidosEval1**, con tus apellidos obviamente.

Las clases que contengan el método main() para cada ejercicio deben llamarse **ApellidosExNum.java** donde Num es el número del ejercicio (por ejemplo LopezPerezEx1.java).

Al finalizar el examen, genera un fichero comprimido que contenga todo el **proyecto** (no el workspace) y lo envías por correo electrónico a la dirección y **marta.peribanez@cpilosenlaces.com** y **daniel.val@cpilosenlaces.com**

Usa la clase **Leer** suministrada y guárdala en el mismo proyecto. Si tienes tu propia clase Leer, cópiala en el proyecto de tu examen y envíala al final dentro del fichero zip.

ES RESPONSABILIDAD TUYA ENVIAR CORRECTAMENTE LA SOLUCIÓN. Si no es legible o no cumple lo anteriormente expuesto, el examen no podrá ser corregido y la nota será 1.

Ejercicio 1.- (3 puntos)

Crea un programa **ApellidosEx1.java** que solicite un número entero mayor que 3, que será la altura y anchura del símbolo +. La aplicación deberá generar la siguiente salida para el valor del entero introducido:

<table>
<tr>
<td>3:
  *
* * *
  *</td>
<td>4:
 * *
* * * *
* * * *
 * *</td>
<td>5:
  *
  *
* * * * *
  *
  *</td>
</tr>
<tr>
<td>6:
 * *
 * *
* * * * * *
* * * * * *
 * *
 * *</td>
<td>7:
  *
  *
  *
* * * * * * *
  *
  *
  *</td>
<td>8:
 * *
 * *
 * *
* * * * * * * * *
* * * * * * * * *
 * *
 * *
 * *</td>
</tr>
</table>

Pista: La anchura del símbolo + depende de si el número introducido es par o impar.

PROGRAMACIÓN IN1W examen problemas, 1ª ev. 01.12.20

NOMBRE .....

Ejercicio 2.- (7 puntos)

En una inmobiliaria se quiere llevar el control de existencias y ventas de pisos que se realizan.

De los apartamentos se guarda la **calle**, **número**, **código postal**, número de **habitaciones, metros** cuadrados, **precio** y **propietario.** El propietario se inicia a null cuando el piso esté sin vender.

Del propietario se quiere guardar un código que se autoincremente a partir de un atributo static, **nombre**, **apellido**, **dni**, **teléfono** y la lista de **pisos** comprados. Al crear el cliente no tendrá pisos comprados.

Haced una carga inicial con los datos facilitados antes de iniciar el menú. (Carga.java)

La aplicación **ApellidosEx2.java** tendrá **7** opciones: Crear propietario, Modificar propietario, Crear apartamento, Vender apartamento, Listado de apartamentos sin vender, Listado de apartamentos vendidos y Listado de propietarios con sus compras.

La opción de modificar apartamento se recomienda gestionarla con un submenú. Se pedirán los datos del propietario, se cotejará con el existente y si difiere algún dato, se actualiza.

Datos iniciales para propietario:

- Miguel Induráin, 72853112H, 676 234 123
- Shakira Val, 73875132J, 665 234 111
- Joane Somarriba, 74986124K, 664 212 278

Datos iniciales para apartamentos:

- Av. Madrid 82, 50017, 3, 70, 172000
- Vía Hispanidad 153, 50012, 5, 140, 325000
- Av. Pablo Gargallo, 50003, 4, 110, 210000
- Av. Valencia 25, 50005, 3, 85, 183000
- Av. Alcalde Gómez Laguna, 3, 50009, 4, 130, 265000

Recomendación de orden para el desarrollo:

1. Crea las clases Propietario y Apartamento con los métodos básicos habituales.
2. Crea la clase **ApellidosEx2.java** con el método main() con el menú.
3. Crea la opción de rellenar los vectores de apartamento y propietario con los datos facilitados.
4. Desarrolla las opciones de los listados formateados.
5. Desarrolla las opciones de Crear apartamento y Crear propietario.
6. Crea la opción de vender.
7. Crea la opción de modificar propietario.

Valoración:

- **Diseño de las clases 2 puntos.**
- **Listados 1.5** (+1.5 por formato) puntos.
- **Vender 1 punto.**
- **Carga inicial 1 punto.**
- **Crear propietario y apartamento 1 punto.**
- Modificación **1 punto**.
- Código bien estructurado, con comentarios y eficiente: **1 punto**.

Hoja Estáticos. Ejercicios tema 4: Clases y objetos

1) Hacer un programa que lea un número N y nos escriba en la pantalla mediante un mensaje si es ó no primo (NOTA: Número primo es aquél que es divisible solo por sí mismo y por, la unidad)
Desarrollar este programa creando un método llamado primo que devuelva true si el número es primo y false si no lo es.

2) Dados dos números P y Q que leeremos por teclado y que deben ser positivos, hacer un programa que nos diga cuál de los dos tiene más divisores, con un mensaje que diga "P tiene más divisores que Q" ó viceversa.
Este programa debe llamar a un método que calcule los divisores de un numero, y lo aplique para P y para Q y luego compare y decida.

3) Se define el factorial de un número N como N*(N-1)*(N-2)*.....*3*2*1 . Hacer un programa que lea un número N filtrando a que sea mayor que cero y calcule su factorial.
La función a construir aquí se llamara factorial. El método main solo debe leer un valor y devolver su factorial (versión iterativa).

4) Dado un número N que se pedirá por teclado y debe ser positivo, imprimir la lista de todos los numero primos hasta dicho número incluido.
Hacer este programa usando el método primo que ya tenéis construido.

5) Escribir un método que con dos parámetros enteros p y q nos devuelva el m.c.d (máximo común divisor) de ambos.

6) Escribir un método que reciba dos parámetros enteros p y q y devuelva el mínimo común múltiplo de ambos.

7) Escribir un método al que se le dé como parámetro un valor r que representa el radio de una, figura una opción y un dato entero que será 1/2/3. El método debe devolver:

a. En el caso opcion1 la longitud del circulo de radio r dada por la expresión 2*pi*r
b. En el caso opción 2 la superficie del circulo dada por la expresión pi*r2
c. En el caso opción 3 el volumen de una esfera de radio r dada por la expresión 4/3*pi*r3

NOTA:pi debe defnirse como una constante;

8) Dados m elementos de un conjunto que se desean agrupar de n en n el número combinatorio que nos define dicho numero de combinaciones posibles es m!/(n!*(m-n)!), siendo siempre m >=n (hay que filtrarlo)
Escribir un método para que dados como parámetros m y n nos calcule el número de combinaciones de m sobre n. Usad el método factorial del ejercicio 3.

9) Escribir un programa que llame a un método con los coeficientes de una ecuación de segundo grado a, b, c y devuelva sus raíces en el caso de ser reales.
Nota : las raíces de una ecuación son:
R1=(-b+raíz(b_2-4*a_c))/(2*a)
R2=(-b-raíz(b_2-4*a_c))/(2*a)
Para la raíz llamar a la clase Math y el método raíz que es sqrt.
Esta función solo se puede aplica en el caso de que valor sea positivo es decir b_2-*a*c > 0.

Hoja 1. Ejercicios tema 4: Clases y objetos

1) Desarrollar un programa en Java que utilice una clase que se llame ***Alumno*** que sea la representación de un alumno.
La clase deberá tener los atributos nombre, edad. En el momento de la creación de cada objeto se asignarán todos los elementos que no podrán ser nulos. Entre los métodos que se programarán deberán estar: imprimir el nombre, imprimir la edad e incrementar la edad.
En ese programa (el principal), crearás un vector de elementos de tipo Alumno y se podrá escoger el alumno a tratar.

2) Desarrollar un programa en Java que utilice una clase que se llame ***Hora*** con miembros de tipo ***Integer*** para hora, minutos y segundos. Deberá tener un constructor para inicializar la hora a 0 y otro para incializar a una hora determinada (hora, minutos, segundos). La hora deberá ser una hora con valores posibles que hemos de controlar).Se deberá poder sumar y restar horas, así como imprimir una hora, ver la conversión a segundos de una hora dada, sumar segundos a una hora dada.
El formato de impresión y lectura será hh:mm:ss, todo en modo 24 horas.

3) Diseña la clase TragaBolas que tiene los siguientes atributos y métodos:

TragaBolas
- color: **String**. Color del tragabolas. Sólo puede ser azul, amarillo, rojo o verde. **- bolasComidas**: **Integer**. Número de bolas que ha comido hasta el momento. **- maxBolas**: I**nteger**. La cantidad máxima de bolas que puede comer.
+ TragaBolas(String, int): Pide por teclado el color y maxBolas. Las bolasComidas se inicializan a 0. **+ visualizar()**: Muestra los datos del tragabolas por pantalla. **+ comer()**: sólo puede comer si bolasComidas es menor que maxBolas, esta acción sumará 1 a bolasComidas y mostrará por pantalla "He comido una bola"**.** **+ trotar()**: sólo puede trotar si bolasComidas es mayor o igual que 1, esta acción restará 1 de bolasComidas y mostrará por pantalla "Estoy trotando". **+ dormir():** sólo puede dormir si bolasComidas es igual a maxBolas. Mostrará en pantalla “Tripa llena. ZZZZZZ” y rebajará bolasComidas a la mitad. Si no cumple la condición para poder dormir mostrará en pantalla: “No quiero dormir”.

En el método main de la clase Principal hay que mostrar un menú con las siguientes opciones:

1: Crear tragaBolas.
2: Darle de comer.
3: Hacerle trotar.
4: Hacerle dormir.
5: Ver estado.
0: Fin.

4) Vamos a crear la clase CuentaCorriente, con las siguientes propiedades y comportamiento:

CuentaCorriente
- numCuenta: **String**.Será el número de la cuenta corriente. **- saldo**: **Double**. Saldo actual de la cuenta. - **cliente: String**. Nombre de cliente. **- numSiguiente:Integer=1** Número de cuenta siguiente
+ CuentaCorriente(double cantidad, String cliente) El número de la cuenta se creará a partir de numSiguiente **+ ingresaEfectivo(double cantidad)** : el parámetro que recibe se lo suma al saldo. **+ retiraEfectivo(double cantidad)**: **Boolean** el parámetro indica la cantidad que queremos retirar. Si hay saldo, restará el importe y devolverá true, en caso contrario devolverá false y no realizará ninguna operación. **+ visualiza()** : Mostrará por pantalla la información de la cuenta corriente: Número de cuenta y saldo.

El método **main** de la clase principal (**GestionCuentas**) creará las cuentas que se deseen a partir del número de numSiguiente, con 0.00 € de saldo y clientes distintos, posteriormente mostrará el siguiente menú por pantalla:

1: Crear nueva cuenta
2: Ingresar en cuenta
3: Retirar de cuenta
4: Visualizar cuenta
5: Visualiza todas las cuentas
0: Fin

Dicho menú se ejecutará realizando las operaciones oportunas para cada opción hasta que el usuario elija la opción de fin. El saldo se debe presentar con solo dos decimales.

NOTA: en los métodos de la clase CuentaCorriente no se piden datos ni se imprime

Hoja 2. Ejercicios tema 4: Clases y objetos

1) Lee el **enunciado completo** del ejercicio antes de empezar a codificar. Crea el proyecto **`FusionCuentas`** en el que hay que utilizar la clase `CuentaCorriente` del ejercicio 4 de la hoja 1. Modifica el código de la clase para poder hacer lo que se pide:

- El número de las cuentas será correlativo y se asignará automáticamente desapareciendo del constructor el número de cuenta.
- El banco quiere saber cuántas cuentas activas existen. Añade los elementos necesarios para poder controlar esto.
- Si un cliente cierra una cuenta, en un Boolean se marcará como verdadero y se pondrá el saldo a cero – al hacer esto el número de cuentas activas disminuye en una unidad-.
- Añade un método estático que se llame **`fusiona`** que reciba como parámetros dos objetos de la clase `CuentaCorriente` y devuelve un objeto de clase `CuentaCorriente`. Este método debe crear una nueva cuenta – que devolverá como parámetro de salida - cuyo saldo será la suma de los saldos de las cuentas que se le han pasado en los parámetros de entrada y, a continuación cerrará dichas cuentas. Al hacer la fusión de dos cuentas en una, hay que tener en cuenta que:
    - Ambas deben pertenecer al mismo cliente. La nueva cuenta tendrá como cliente al cliente de las cuentas antiguas, pero se le asignará un número de cuenta nuevo.
    - No se pueden fusionar dos cuentas si tienen el mismo número de cuenta porque eso nos facilitaría duplicar nuestro dinero de la nada.

  Hay que comprobar que se cumplen estas condiciones antes de crear la nueva cuenta. Si falla una de las dos no se crea la nueva cuenta, no se cierran las antiguas y se devuelve `null`.

2) Crea un nuevo proyecto **CuentasCliente** a partir del enunciado anterior y crea la clase Cliente con los atributos que consideres necesarios.

Cambia la lógica del programa de forma que el atributo **String cliente** sea ahora **Cliente cliente** en la clase CuentaCorriente.

En la clase Cliente añade como atributo un vector de CuentasCorriente en el que se guarden las cuentas activas de un cliente.

Modifica los métodos de forma que se puedan gestionar los elemento nuevos que se han introducido cada uno en su clase correspondiente.

El programa que contiene el main deberá gestionar todos los clientes y todas las cuentas.

NOTA: Como siempre, las métodos de las clases definidas no pedirán datos ni mostrarán mensajes.

3) Crea el proyecto **`LibroUnAutor`**, que estará formado por las siguientes clases:

Autor
-nombre: String (no tiene valor por defecto) -email: String (no tiene valor por defecto) -genero: Character (los posibles valores son 'm' o 'f')
+Autor(nombre: String, email: String, genero: String) +getNombre(): String +getEmail: String +setEmail(email: String): void +getGenero(): Character +toString(): String

El método toString devolverá la siguiente cadena de caracteres (por ejemplo):

"Autor[nombre = Arturo Perez Reverte, email = apreverte@correo.com, genero = m]"

Libro
-titulo: String (no tiene valor por defecto) -autor: Autor -precio: Double -cantidad: Integer = 0
+Libro(titulo: String, autor: Autor, precio: double) +Libro(titulo: String, autor: Autor, precio: double, cantidad: int) +getTitulo(): String +getAutor: Autor +getPrecio(): Double +setPrecio (precio: double): void +getCantidad(): Integer +setCantidad(cantidad: int): void +toString(): String

Supondremos que un libro sólo tiene un autor y que el método toString devolverá una cadena de caracteres como la siguiente (por ejemplo):

"Libro [titulo = El asedio,
Autor [nombre = Arturo Perez Reverte, email = apreverte@correo.com, genero = m]
precio = 18,90
cantidad = 3]"

Escribe la clase GestLibroAutor, que además del método main deberá tener otros métodos estáticos para que al ejecutar la aplicación muestre un menú con las siguientes opciones:

1- Crear libros.
2- Crear autor.
3- Modificar autor.
4- Modificar libro.
5- Listado de libros.
6- Listado de autores.
0- Salir.

4) Crea un nuevo proyecto, al que llamarás **`LibroVariosAutores`** que utilice las clases del ejercicio anterior y modifica la clase Libro para que un libro pueda tener varios autores.

El método toString debe devolver, por ejemplo:

"Libro [titulo = Cuentan que cuentan

{ Autor [nombre = Silvia Schujer, email = schujer@correo.com, genero = f], [Laura Devetach, ldevet@edu.com, genero = f], [Emma Wolf, wolf@correo.com , genero = f]}

precio = 14
cantidad = 6]"

Se pide hacer un programa para jugar al típico juego de piedra, papel o tijera.

- Un primer menu con las opciones: Jugar, puntuaciones y salir.

--- Jugar: Cada partida se compone de 10 rondas.
Para cada ronda, se mostrará un menu facilitando la elección del usuario.

--- puntuaciones: Se mostrará el contenido de un vector del tipo Score de 5 posiciones.
Crear la clase Score compuesta por un nombre y puntuación.
Actualizar los datos para que quede el de mayor puntuación en la parte superior.
Si el usuario gana la ronda, se incrementa su puntuacion en 10, si pierde se le descuentan 3 puntos
y en caso de empate, no se modifica.
No hay puntuaciones negativas, la puntuación de cada partida estará comprendida entre 0 y 100 puntos.

EJERCICIO 1:

Escribe una Clase con un método que genere números aleatorios entre 0 y 9 inclusive.

Escribe un programa que genere una tabla de 4 filas y 4 columnas utilizando el método anterior.

Genera una segunda tabla de 4 filas y 4 columnas llamada tabla 2.

Implementa un método que devuelva una tabla de 4 filas y 4 columnas que sea el resultado de sumar el valor de las dos tablas anteriores (Posición a posición)

EJERCICIO 2:

Escribe una clase **Coche** con los atributos, **marca**, **modelo** y **matrícula** (del tipo que consideres oportuno).

Crea un vector con 3 coches y muéstralos en un listado.

EJERCICIO 3:

Escribe un programa que genere nombres y apellidos aleatorios. Para ello:

- Utiliza un vector de nombres preestablecidos (10 nombres)
- Utiliza un vector de apellidos preestablecidos (10 apellidos)
- Genera 5 conjuntos de nombre y apellido de tal forma que sólo pueden asignarse una única vez los nombres y los apellidos.

EJERCICIO 4:

Escribe un programa que genere nombres, apellidos y notas aleatorios. Para ello:

- Utiliza un vector de nombres preestablecidos (10 nombres)
- Utiliza un vector de apellidos preestablecidos (10 apellidos)
- Las notas (entre 1 y 10) tendrán 2 decimales.
- Genera 5 conjuntos de nombre, apellido y nota, de tal forma que sólo pueden asignarse una única vez los nombres y los apellidos, las notas se pueden repetir.
- Una vez generados esos 5 alumnos, ordénalos por nota (de mayor a menor).