EJERCICIOS CURSO JAVA (PÍLDORAS INFORMÁTICAS)

1) PRIMEROS PASOS ----------

1.1- Entrada Ejemplo 1:

Imprimir en consola el nombre y la edad del próximo año del usuario.
Los datos se deben solicitar por consola.

1.2- Entrada Ejemplo 2:

Imprimir en consola el nombre y la edad del próximo año del usuario.
Los datos se deben solicitar por ventana.

1.3- Entrada Números:

Calcular la raíz cuadrada de un número solicitado por ventana al usuario.
Los datos se deben imprimir por consola.

1.4- Evalúa Edad:

Solicitar por consola la edad del usuario y luego Imprimir en consola si es adolescente, joven, maduro o mayor.

1.5- Cálculo de Áreas:

Solicitar por consola una opción: 1(cuadrado), 2(rectángulo), 3(triángulo) y 4(círculo).
Luego solicitar por ventana los datos necesarios para calcular el área de la figura elegida.
Imprimir en consola el nombre de la figura y su área calculada.

1.6- Verificación de Email:

Solicitar la contraseña mediante ventana. Dicha contraseña debe estar establecida en el programa.
En caso de ser incorrecta imprimir el error en consola y volver a solicitar ingresar la contraseña.
Cuando la contraseña sea correcta imprimir en consola el mensaje de bienvenida.

1.7- Adivina Número:

Permitir que el usuario adivine un número aleatorio entre el 1 y el 100.
Solicitar el número por ventana.
Imprimir en consola si debe ingresar un número más alto o más bajo.
Cuando el número sea correcto imprimir el mensaje de victoria e indicar el número de intentos realizados.

1.8- Adivina Número do-while:

Permitir que el usuario adivine un número aleatorio entre el 1 y el 100.
Solicitar el número por ventana.
Imprimir en consola si debe ingresar un número más alto o más bajo.
Cuando el número sea correcto imprimir el mensaje de victoria e indicar el número de intentos realizados.
El ejercicio debe contener el condicional do-while.

1.9- Peso Ideal:

Calcular el peso ideal del usuario teniendo en cuenta su género y su altura.

Tener en cuenta que para los hombres el peso ideal es: (Altura – 110) + kg, mientras que para las mujeres es: (Altura – 120) + kg.
Los datos deben solicitarse mediante ventana.

1.10- Uso Bucle For:
Imprimir en consola el valor de i desde 0 hasta <10.
Imprimir en consola el valor de i desde 0 hasta <20, sumando cada dos.
Imprimir en consola el valor de i desde 30 hasta >20.

1.11- Comprobar la Dirección de Email:
Solicitar mediante ventana el correo electrónico al usuario.
Imprimir en consola si el email es correcto o incorrecto.
Para ser correcto debe tener al menos un punto y un solo @.

1.12- Factorial:
Solicitar un número por ventana.
Imprimir el factorial de dicho número en consola.

1.13- Uso Arrays:
Crear una matriz con 5 enteros (positivos y negativos del 20 al 99), iniciar sus valores por separado e imprimir cada uno de sus valores en líneas separadas indicando lo siguiente: El valor del índice x es:
De la misma manera crear otra matriz usando los mismos valores anteriores pero iniciándolos todos en una sola línea e imprimiendo de la misma manera que la anterior.

1.14- Uso Arrays II:
- Crear una matriz con 7 países, imprimirlos en líneas separadas y al final imprimir el total de países.
- Solicitar por ventana 6 países al usuario e imprimirlos en líneas separadas y al final imprimir el total de países.
- Crear una matriz con 150 números aleatorios e imprimirlos todos en consola en una misma línea.

1.15- Arrays Bidimensionales:
- Crear una matriz bidimensional de enteros con 4 filas y 5 columnas. Iniciar todos los valores por separado y finalmente imprimir la matriz en filas y columnas.
- Crear otra matriz bidimensional de 4 filas y 5 columnas pero iniciar todos sus valores en línea e imprimir dicha matriz usando un for simplificado.

1.16- Array 2D:
Calcular el acumulado de intereses durante 5 meses para un préstamo de 10.000€ pero para 6 tasas de interés diferentes del 10 al 15%.
Representar todos los valores en una matriz bidimensional mediante consola.

2) POO ----------

2.1- Prueba Final (Empleados):
Imprimir en consola y mediante un método el nombre, el departamento y el nro-id de tres empleados en líneas separadas.

Sus nombres deben ser ingresados en el main.
Dos tipos departamentos deben ser modificados en el main.

2.2- Coche:
Crear una Clase para crear un objeto coche con las siguientes características fijas: cantidad de ruedas, largo en cm, ancho en cm, revoluciones del motor, peso de plataforma. Establecer en el main: color, asientos cuero y climatizador. Características internas: peso de carrocería y precio final. Incluir un método que devuelva las cinco características principales más el color.

2.3- Uso Coche:
Crear un programa que use la clase Coche y pregunte al usuario mediante ventana: el color, si tiene asientos y si tiene climatizador.
Finalmente imprimir en consola: longitud en metros, color, si tiene asientos de cuero, si tiene climatización, el peso total y el precio total.

2.4- Furgoneta:
Crear una Clase que use los mismos valores de Coche pero que agregue sus propias características: capacidad de carga y plazas extra, los datos deben ser solicitados por ventana cuando se use la Clase.
Crear un método que entregue los datos propios de la furgoneta.

2.5- Uso Vehículo:
Crear un programa sin sub-clases que imprima la data de las clases Coche y Furgoneta.
Imprimir en consola: las 10 características del coche y las 12 características de la furgoneta.

2.6- Uso Persona:
- Crear un programa que tenga una clase de uso obligatorio (por si alguien desea usarla en otro programa), dicha clase debe incluir: nombre y descripción de lo que hace, ambos datos deben incluirse en el main.
- Crear una sub-clase que otorgue los datos de un empleado: sueldo, “nro-id”, fecha alta de contrato. También debe pedir en el main: nombre, sueldo, año, mes y día. Además debe incluir los datos de la clase anterior.
- Crear otra sub-clase que devuelva los datos de un alumno: carrera. El nombre y la carrera deben ser ingresados en main. Usar también los datos de la primera clase creada. Finalmente imprimir en consola en una línea: nombre de empleado, descripción que muestre el “nro-id” y el salario. En la otra línea: nombre del alumno, descripción que muestre la carrera que está cursando.

2.7- Enum Uso Tallas:
Desarrollar un programa que tenga una lista de constantes de tallas: s, m, l y xl.
Debe solicitar mediante consola la talla al usuario y luego imprimir: la talla del usuario en mayúsculas y la abreviatura de dicha talla.

2.8- Jefes y Trabajadores:
- Crear una interfaz de tipo Jefe que tenga un método simple sin implementación que solicite un dato de tipo decisión mediante main.
- Crear otra interfaz de tipo Trabajadores que tenga: un método simple para establecer bono (salario extra) el cual debe ser solicitado por main y un salario base de 1500€.

2.9- Uso Empleado:
Crear un programa con dos sub-clases:
- Una sub-clase que sea de tipo empleado que tenga a su vez sobrecarga de métodos constructores:
  - Uno de ello debe pedir en el main: nombre, sueldo, año, mes y día. Y un nro-id fijo para cada empleado.
  - El otro debe pedir en main: nombre, apellido. Y debe tener un nro-id.
  - El último sólo debe pedir el nombre en el main. Éste método incluye data fija: salario 30000, año 2000, mes 1 y día 15.

Ésta clase debe implementar la interfaz Trabajadores.

- La otra sub-clase del tipo jefatura debe poseer la misma data de la sub-clase empleado y a su vez debe tener un incentivo el cual se debe indicar en el main y debe sumarse al sueldo base. Ésta clase debe implementar la interfaz Jefes.

Finalmente se debe imprimir en consola:

- La decisión de uno de los jefes.
- Nombre y bono de un jefe.
- Nombre y bono de un empleado.
- Una lista de 7 empleados de los cuales:
  - 3 son empleados con todos los datos por main.
  - 1 es empleado con nombre en main y datos fijos.
  - 1 es empleado sólo con nombre y apellido.
  - 1 es jefe con todos los datos main (instanciado fuera del array) y un incentivo de 2.570€.
  - 1 es jefe con todos los datos main y un incentivo de 55.000€

La lista de los 7 empleados debe tener: nro-id, nombre empleado, salario y fecha de contrato. La lista debe estar ordenada por salarios de menor a mayor.

2.10- Prueba Temporizador:
Crear un programa que imprima en consola la hora y la fecha cada 5 segundos y que se cierre mediante ventana.

3) GRÁFICOS ----------

3.1- Creando y Escribiendo en Marcos:
Crear una ventana de medidas: 600x400, título: Mi Ventana Java, tenga un icono propio, esté centrada en el escritorio (usando como referencia el ancho y el alto de la pantalla) y se cierre con el botón correspondiente.
Agregar texto dentro de la ventana: Título del Contenido, sin usar JLabel.

3.2- Prueba Dibujo y Trabajando con Colores:
Crear una ventana con su título, centrada, color de fondo cyan claro y cierre correspondiente que tenga los siguientes dibujos:
- Un rectángulo vacío de la clase Graphics con su título.
- Un rectángulo relleno de clase Graphics con su título.
- Un rectángulo de la clase Graphics2D con su título.
- Un a elipse circunscripta en el anterior rectángulo y su título.

- Una línea oblicua desde el vértice superior izquierdo del rectángulo hasta el vértice inferior derecho del mismo y su título.
- Un círculo exterior al rectángulo (ajustado) y su título.
- Dibujar un cuadrado con un círculo circunscripto y rellenar ambos de color diferente y agregar su título.

3.3- Fuentes Tipo y Trabajando con Fuentes:

Crear un programa que permita al usuario consultar cualquier tipo de fuente tipográfica para saber si se encuentra o no instalada en el sistema operativo.
Recibir la data mediante ventana. Imprimir respuesta en consola.
Escribir en ventana un título, línea divisoria, sub-título y un pequeño contenido con el tipo de Fuente solicitada por el usuario. Establecer una por defecto en caso de ser negativa la consulta. Cambiar el color del fondo y de la fuente.

3.4- Prueba Imágenes:

Crear una ventana con una imagen principal y de fondo una imagen pequeña que se repita a lo largo y ancho. Cada imagen debe llevar el control de excepciones.

3.5- Prueba Eventos y Prueba Acciones:

Crear una ventana con tres botones y sus textos e iconos que permitan cambiar cada uno el color del fondo.
A su vez el fondo puede ser cambiado por combinaciones del teclado

3.6- Eventos Ventana, Foco Ventana y Eventos Teclado:

Crear dos ventanas en diferentes posiciones, cada una con su propio título.
- Cuando se cambien de ventana imprimir en consola: ventana abierta, ventana activada, ventana desactivada y ventana cerrando.
- Al cerrar la ventana 1 se cierra el programa, al cerrar la ventana 2 la 1era quedará abierta.
- Notificar en consola cuando se cambie de estado: minimizar, maximizar, ha ganado o ha perdido el foco.
- Imprimir en consola la tecla presionada del teclado y su código.

3.7- Eventos Ratón:

- Imprimir en consola si el ratón se está moviendo o arrastrando.
- Imprimir en consola cuál tecla del ratón se ha presionado, levantado o ha hecho clic.
- Imprimir la posición en la cual se ha hecho clic con el ratón.
- Imprimir en consola si el ratón ha entrado o si ha salido de la ventana.

3.8- Foco Evento:

Crear una ventana con dos campos para ingresar texto, en uno de ellos se debe comprobar un email que tenga @ y un punto, dicha comprobación se realiza al perder el foco ese campo de texto.

3.9- Varios Oyentes:

Crear una ventana con dos botones:
- Uno permite crear ventanas nuevas ubicadas en cascada y que se cierran de forma individual.
- El otro botón cierra todas las ventanas nuevas abiertas pero no cierra la ventana con los dos botones.

3.10- Layouts:
Crear una ventana que tenga en la parte superior una sección con 5 botones dispuestos en cada zona de un BorderLayout. En la parte inferior 3 botones alineados a la derecha.

3.11- Calculadora:
Crear una calculadora que realice las operaciones básicas.

3.12- Prueba Texto:
Crear una ventana con un campo de texto, su label y un botón de enviar.
Al oprimir el botón se debe activar un label que indique si el email introducido es correcto o incorrecto.
Al escribir o eliminar cada letra dentro del cuadro de texto se debe imprimir en consola: data ingresada o data eliminada.

3.13- Campo Password:
Crear una ventana de login de usuario con dos campos para email de usuario y clave con sus labels, y un botón de ingresar.
El campo de clave debe cambiar el color del fondo en vivo si el requisito de 8 a 12 caracteres no se cumple. El campo de password debe proteger las letras ingresadas en forma de puntos. También se debe notificar en texto al usuario los requisitos para corregir el error.

3.14- Ejemplo Área:
Crear un programa con dos áreas de texto, uno de ingresar y el otro sólo de salida que no permita edición.
Los campos de texto deben ajustar el texto automáticamente y en caso de pasar los límites activar barras de desplazamiento.
Crear un botón de Enviar que traslade la data del campo de texto al siguiente.
Al enviar la data activar junto al campo de salida un label que indique que el texto se ha enviado.

3.15- Prueba Área:
Crear una ventana que tenga un campo de texto que abarque toda la superficie y en la parte inferior dos botones.
Un botón que inserte un texto predeterminado y el otro botón que cambie entre: insertar salto de línea y quitar salto de línea.
Si no hay salto de línea debe haber sroll horizontal, caso contrario scroll vertical.

3.16- Prueba Checks:
Crear una ventana con un texto en el centro que pueda ser modificado a negrita, cursiva o ambos con un par de botones check.

3.17- Sintaxis Radio, Ejemplo Radio, Prueba Combo, Marco Slider, Marco Spinner:
- Crear una ventana con un texto central que pueda ser modificado por dos grupos de botones de radio: un grupo debe cambiar a: pequeño, mediano, grande y muy grande y el otro debe cambiar a: negrita, cursiva o normal.
- Al texto le puede ser cambiada su fuente desde una lista desplegable de fuentes tipo.
- Mediante un slider se puede cambiar el tamaño del texto.
- Usando dos spinners se pueden cambiar la fuente y el tamaño del texto.

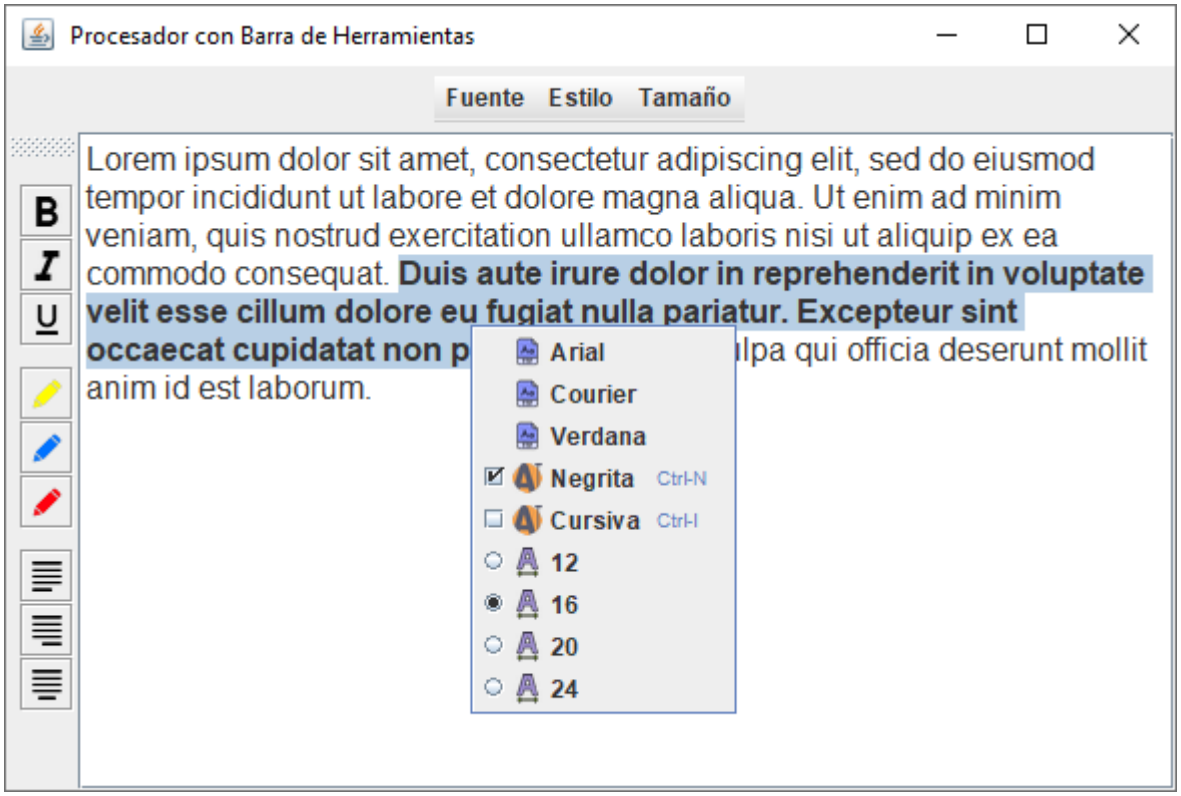
3.18- Marco Menú:

Crear una ventana con una barra de menú que tenga:

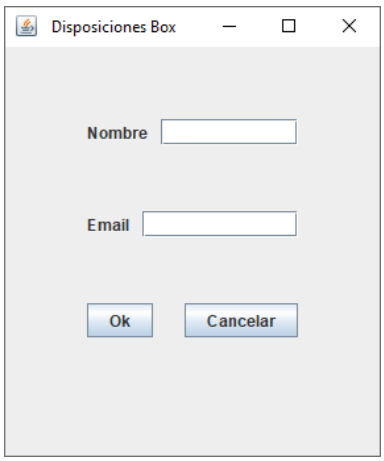
- 4 botones principales.
- Cada botón con 4 sub-botones.
- Un sub-botón con separador.
- Un sub-botón con 3 sub-botones.

4) GRÁFICOS II ----------

4.1 **Crear el siguiente programa:**



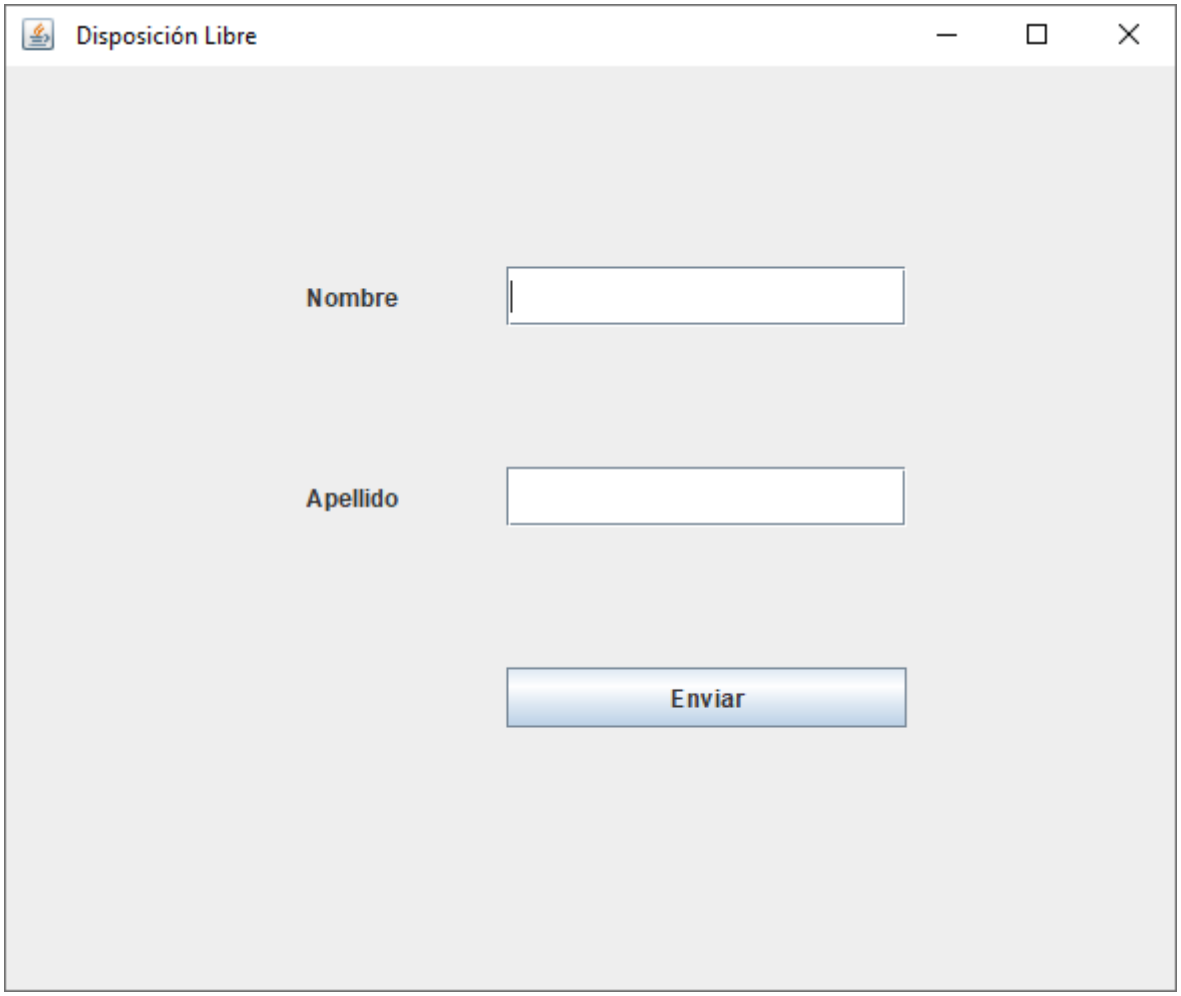
4.2 Crear el siguiente programa:



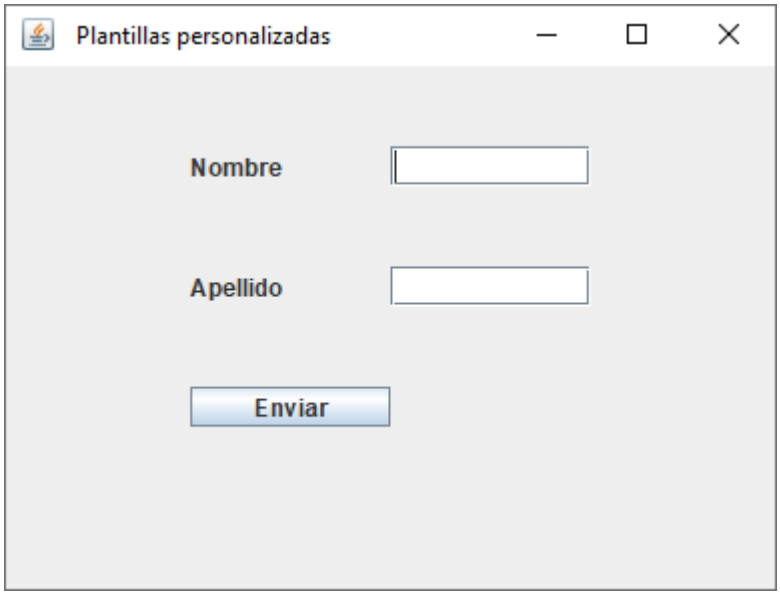
4.3 **Crear el siguiente programa:**

4.4

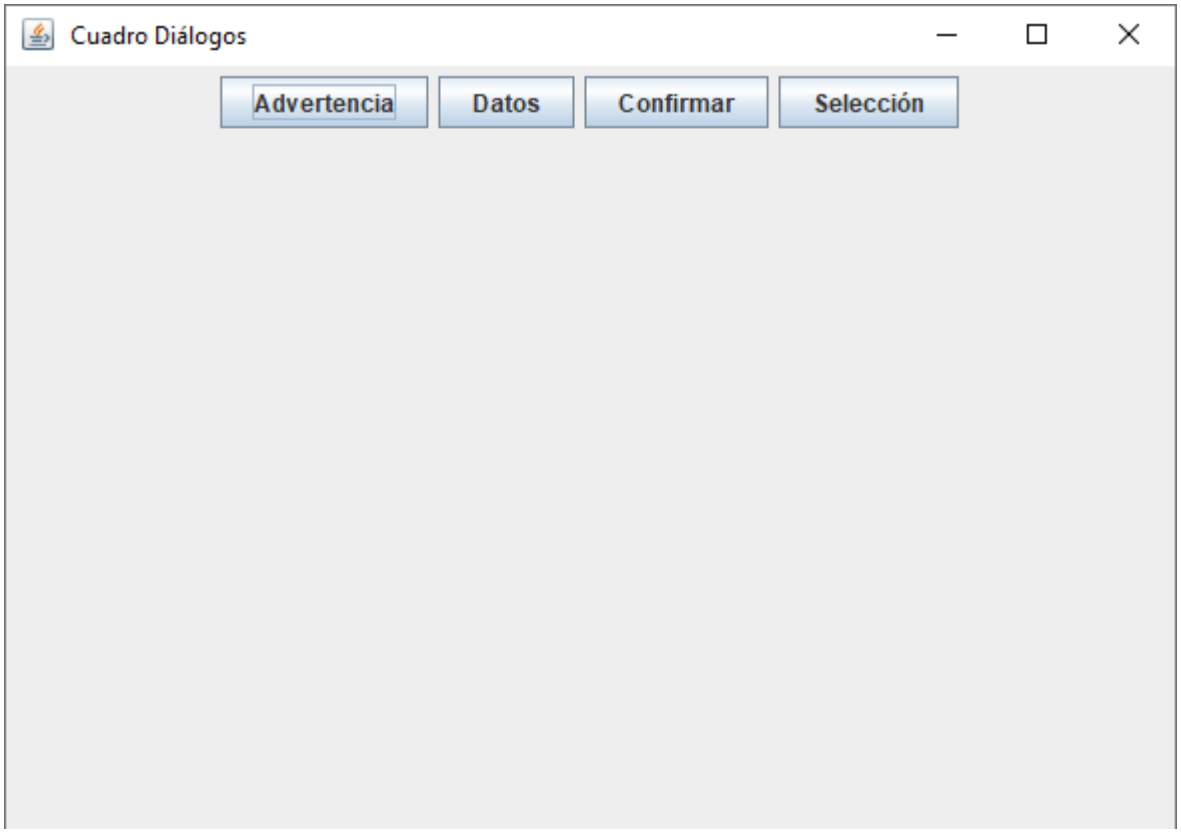
4.5 Crear el siguiente programa:



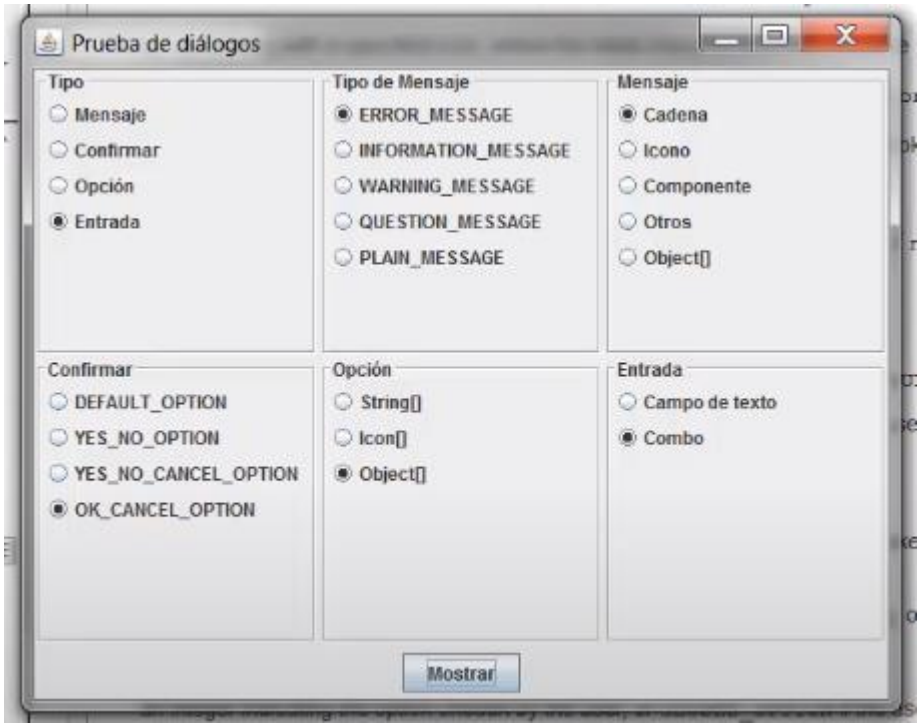
4.6 Crear el siguiente programa:



4.7 Crear el siguiente programa:



4.8 Crear el siguiente programa:



5) FICHEROS, PROG. GENÉRICA Y PROG. CONCURRENTE----------

5.1 Acceso a Ficheros:
Crear un programa que lea todos los archivos de una carpeta y los archivos de sus subcarpetas. Imprimir toda la data en consola, debe estar todo organizado, poniendo primero el nombre de la carpeta padre: y luego sus archivos. EL programa debe imprimir la ruta y luego imprimir si existe el fichero de la ruta.
Clase a usar: File.

5.2 Creando y Eliminando Ficheros y Archivos:
- Crear un programa que cree un directorio/carpeta nuevo en el ordenador, la ruta debe estar habilitada para cualquier sistema operativo (File.separator). – El programa debe crear un archivo nuevo de texto (.txt) dentro de la carpeta creada. – El programa debe escribir un párrafo en el archivo creado (debe ser letra a letra). – Luego el programa debe leer dicho archivo e imprimir el párrafo en consola. – El programa debe eliminar dicho archivo pero confirmar primero por ventana emergente. – Finalmente imprimir en consola si existe el archivo en la carpeta creada.
Nota: entre cada paso debe haber una ventana emergente que confirme la acción.
Clases a usar: File, FileWriter, FileReader, IOException, JOptionPane.

5.3 Leyendo/Escribiendo Buffer:
Primero crear en el ordenador dos archivos de texto: uno con un párrafo y el otro vacío. - Luego crear un programa con tres clases: 1 – Una clase que lea el texto creado previamente. 2 – Otro que escriba

un párrafo en el archivo vacío. 3 – Después una clase que agregue una frase después del texto del archivo anterior.
Nota: ninguna clase debe usar Buffer.
- Finalmente crear otro programa, con las mismas tres clases, pero ésta vez usando el buffer.
Clases a usar: FileReader, FileWriter, BufferedReader, BufferedWriter, FileNotFoundException, IOException.

5.4 Leyendo y Escribiendo Bytes:
Crear un programa que lea todos los bytes de una imagen localizada en una carpeta del ordenador y luego realice una copia exacta de dicha imagen en la misma carpeta.
Clases a usar: FileInputStream, FileOutputStream, FileNotFoundException, IOException.

5.5 Serialización:
Crear un programa que exporte a un archivo exterior en el ordenador un objeto de tipo array de las clases Empleado y Jefe con cuatro elementos.
Luego que el programa lea la data de dicho archivo y la almacene en un array nuevo en el main y luego imprima en consola los cuatro elementos del array.
Toda la data debe estar serializada.
Clases a usar: FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream, Serializable.

5.6 Programación Genérica: ArrayList:
Crear una clase que tenga una variable Array de tipo Objetc sin iniciar, la clase debe tener tres métodos: el método constructor principal para establecer el tamaño del array, otro de tipo Object para obtener cualquier elemento del array y el último para establecer un nuevo elemento.
Luego crear un programa que haga uso de la clase anterior: debe crear un array con seis elementos, imprimir en consola los elementos nro 2 y 4, luego crear una nueva instancia del array que reciba un elemento tipo File (ruta a un archivo) y luego lo imprima en consola.
Clases a usar: File.

5.7 ArrayList e Iterator:
Crear una clase empleado que tenga nombre, edad y salario, q devuelva nombre y q pase en un string todos los datos.
Luego crear un programa que construya un arraylist, q sea de la clase Empleado, con seis empleados (uno debe llamarse Diego), q asegure una capacidad para 11 empleados y q recorte el espacio sobrante del array, luego imprimir la cantidad (un número) de elementos que tiene array, por separado: reemplazar el empleado en la posición 1 del array por uno nuevo, imprimir el empleado nro 3, en otras líneas imprimir toda la lista de empleados del array, luego imprimir el array pero usando un iterator, después usando el iterator eliminar el empleado Diego y finalmente pasar el arraylist a un array normal e imprimirlo en consola.
Clases a usar: ArrayList, Iterator.

5.8 Clases Genéricas:
Crear una clase empleado que tenga nombre, edad y salario, solamente devuelve un string con toda la data.
Crear una clase genérica (llamarla Pareja) que sólo tenga una variable, dicha variable se puede modificar y devolver por métodos. También crear dentro de esta clase un método comodín que

imprima en consola dicha variable, dicho método tiene como parámetro un array de la clase creada (Pareja) y de tipo Empleado.
Luego usar dicha clase en el main para almacenar un String e imprimirlo en consola.
Luego crear una nueva instancia de la clase Pareja pero ahora del tipo Empleado e imprimir en consola un valor.
Crear una pequeña clase llamada Persona que devuelva un string de un nombre. Luego usando la clase Pareja imprimir la data de dicha clase Persona.

5.9 Métodos Genéricos:
Crear una clase con dos métodos genéricos: Uno estático y genérico de tipo string que reciba por parámetro un array genérico y que retorne un string con la cantidad (un número) de elementos que pueda tener dicho array. El otro método genérico retorna el valor menor del array genérico.
Crear en el main un array de varios nombres e imprimir la data usando los métodos genéricos anteriores.
Crear en el main una matriz de fechas y usar los métodos genéricos anteriores para imprimir la data.
Finalmente instanciar la clase Empleados, crear seis empleados y luego imprimir la data usando el primer método genérico.
Clases a usar: GregorianCalendar

5.10 Programación Concurrente: Sincronizando Hilos:
Crear un programa que tenga dos clases:
Una clase que herede de Thread y aplique su método run() para ejecutar un bucle de 15 hilos los cuales se tienen que ejecutar despacio (imprimir en consola el flujo).
La otra clase también hereda de Thread, tiene una variable de tipo Thread y tiene un método constructor q recibe un parámetro de tipo Thread. Luego usa el método run() de Thread e imprime en consola la ejecución de 15 hilos con una velocidad reducida.
Finalmente el programa debe crear una instancia de la primera clase y dos de la segunda clase, ejecutando sus acciones con la condición de esperar a q se ejecuten las primeras para ir pasando a las segundas (un hilo a la vez).
Al final imprimir la frase: Hilos finalizados!

5.11 Uso de Threads. Varios Botones:
Crear un programa que muestre tres bolas negras en una pantalla, rebotando de un lado al otro, luego tres botones que activan una bola a la vez y otros tres botones que detienen cada bola.
Todas las bolas se pueden ejecutar en simultáneo.
Clases a usar: Thread, BorderLayout, Component, Graphics, Graphics2D, ActionEvent, ActionListener, Ellipse2D, Rectangle2D, ArrayList, JButton, JFrame, JPanel.

5.12 Banco sin Sincronizar:
Crear un programa que realice transferencias bancarias entre cien clientes, cada cliente inicia con un saldo de 2000€ y el saldo total siempre tiene que ser de doscientos mil euros. Imprimir en consola el id de transferencia (hilo) de cada transacción, la cantidad transferida, la cuenta de origen, la cuenta destino y el saldo total del banco, todo en una misma línea.
Todas las transacciones deben realizarse, por eso las cuentas que no tengan suficiente saldo deben esperar hasta que tengan saldo y realizar dicha transferencia, imprimir si el hilo está a la espera y cuando esté liberado.
Clases a usar: Thread, Runnable, Condition, ReentrantLock.

5.13 Banco Synchronized:

Es el mismo ejercicio anterior pero usando los métodos por defecto de la clase Object (synchronized).

6) COLECCIONES Y SOCKETS ----------

6.1 HashCode() e Equals():

Crear una clase de tipo Libro que tenga: título, autor, año y código ISBN. Estos datos se piden por parámetros al instanciar la clase. Dicha clase sólo tiene un método getter que devuelve todos los datos en string.
Luego crear un programa que use la clase Libro, creando dos instancias de la misma, los dos deben tener la misma data. Luego comparar dichos libros para saber si son iguales o no usando un condicional if. En caso de que el programa no los reconozca como iguales crear un método equals() personalizado en la clase Libro para que reconozca los libros como iguales. Luego crear los métodos por defecto de Java hashCode() e equals() y comparar nuevamente los libros (Inhabilitar el método anterior para evitar errores). Finalmente imprimir en consola el HasCode de los dos libros.
Clase a usar: ninguna.

6.2 Set y HashSet:

Crear una clase de tipo Cliente que tenga: nombre, nro de cuenta y saldo. Estos datos se piden por parámetros al instanciar la clase. Dicha clase debe tener todos los setters y getters generados por eclipse así como el hashcode y el equals.
Luego crear un programa que use la clase Cliente, creando seis instancias de la misma para varios clientes. Debemos tener presente que NO pueden haber clientes duplicados, para ello se debe elegir la clase HashSet de la interface Set y crear una colección con las instancias de los clientes. Luego se debe:
- Agregar todos los clientes a la colección.
- Imprimir la data de cada cliente usando un bucle for-each.
- Crear una nueva instancia para crear un nuevo cliente (pero copiando la data de uno de los clientes que ya hay) y agregar el nuevo cliente duplicado a la colección y volver a imprimir para comprobar si se ha permitido.
- Imprimir los nombres de cada cliente pero usando un Iterator.
- Luego eliminar un cliente utilizando Iterator y volver a imprimir toda la colección para comprobar que se ha eliminado.
Clases a usar: HashSet, Set, Iterator.

6.3 LinkedList:

Crear programa que cree una colección de strings de nombres usando el LinkedList. Luego:
- Imprimir en consola el tamaño de la colección.
- Imprimir la colección usando un foreach.
- Agregar un nuevo elemento a la colección en la posición 2 usando un ListIterator y volver a imprimir para comprobar.

Clases a usar: LinkedList, ListIterator.

6.4 LinkedList Enlazada con ListIterator:

Crear un programa que cree una colección de strings de países usando el LinkedList y otra colección de capitales de cada país. Luego:
- Agregar a la colección de países las capitales pero una después de cada país usando un ListIterator.
- Imprimir en consola cada país seguido de su capital usando un foreach.
- Eliminar las capitales pares usando el ListIterator y luego imprimir las capitales restantes con un foreach.
- Finalmente las capitales que no fueron eliminadas (impares) eliminarlas de la colección países (debe ser estrictamente de la colección países).
Clases a usar: LinkedList, ListIterator.

6.5 TreeSet:

- Crear una clase de tipo Artículo con variables: número y descripción. Un constructor que pida los dos parámetros y otros que no pida nada. Un getter que retorne sólo la variable descripción. Implementar las interfaces Comparable y Comparator. Con comparable retornar el orden de los números y con comparator el orden de la descripción.
- Luego crear una colección TreeSet de tipo String y agregar varios nombres, imprimir en consola la colección y comprobar que se hayan ordenado automáticamente.
- Instanciar varios artículos usando la clase Artículo y su constructor con parámetros, agregarlos a una colección TreeSet pero en orden desordenado e imprimir con un foreach, comprobar que se hayan ordenado automáticamente según su número.
- Luego crear una instancia de Articulo pero sin parámetros, luego crear una nueva colección de TreeSet y agregar como parámetro la instancia de artículo acabada de crear. Agregar después a la nueva colección los artículos instanciados la primera vez, de forma desordenada y luego imprimir con un foreach y comprobar que se haya ordenado automáticamente según su descripción.
- Finalmente crear una tercera colección TreeSet pero como parámetro agregar una instancia de la Interface Comparator del tipo Artículo en cuyo método compare los artículos por su descripción. Luego agregar los artículo instanciados al principio e imprimir con un foreach y comprobar el orden.
Clases a usar: TreeSet, Comparable, Comparator.

6.6 Map y HashMap:

- Crear una clase de tipo empleado con variables: nombre y sueldo, un constructor que pida el nombre por parámetro y un sueldo fijo de 2000, con un método toString que devuelva toda la data.
- Crear una colección HashMap con un key tipo string para el código id de unos empleados, y como value de la colección el tipo empleado. Agregar a dicha colección varios empleados con sus respectivos códigos. Imprimir directamente en consola sin foreach.
- Luego eliminar un empleado usando su código. Imprimir en consola nuevamente la colección.
- Después sustituir un empleado por uno nuevo en base a un código. Imprimir nuevamente.
- Imprimir la colección usando el método entrySet().
- Finalmente imprimir toda la colección usando un foreach pero con el método Entry de la clase Map, se debe imprimir primero el código y luego la descripción de cada empleado.
Clases a usar: Map, HashMap.

6.7 Sockets:

- Crear un programa de chat que tenga un lado cliente y un lado servidor.
- Dos usuarios pueden escribirse mensajes. Cada uno tiene su propio Nick y dirección ip.
- En cada ventana de usuario se deben ver los nombres de quien escribe y toda la conversación.
- Cada usuario al iniciar el programa debe ingresar su Nick y los usuarios disponibles deben aparecer en una lista desplegable.
- El servidor debe recibir la data tanto de conexión de nuevos usuarios y reenviarla a todos los usuarios activos, así como de recibir y enviar al destino correspondiente los mensajes enviados.