PROYECTO DE PROGRAMACIÓN ORIENTADO A OBJETOS – GESTIÓN TELEFONÍA

Descripción del proyecto

En este proyecto tendrás que desarrollar una **aplicación Java en modo consola que permita gestionar un conjunto de facturas telefónicas** empleando los fundamentos de la Programación Orientada a Objetos. La descripción completa y detallada de la implementación de este proyecto se encuentra en las siguientes secciones.

Debes aplicar todas las buenas prácticas aprendidas, entre otras:

- Implementar los métodos de las clases realizando control de errores en los parámetros de entrada.
- Validar todos los datos de entrada, tanto los introducidos por teclado por el usuario, como los parámetros de las funciones.
- Implementar los constructores y métodos de la clase reutilizando todo el código posible.
- Evitar código duplicado y reutilizar código mediante funciones.
- Implementar las clases sin modificar la interfaz pública de la misma.
- Nombrar las variables, clases y métodos de acuerdo a las convenciones del lenguaje, y usar identificadores descriptivos y representativos de lo que almacena o realiza la variable/función.
- Escribir la documentación Javadoc de todas las clases del proyecto de forma completa y detallada, siguiendo la guía de estilo.
- Seguir las especificaciones descritas en este guion sin cambiarlas, salvo justificación y previa consulta con el profesor.

Importante: No se corregirá ningún proyecto en el que haya errores de compilación.

La tarea se califica entre 0 y 10 puntos desglosados de la siguiente manera:

- Clase ***Llamada*** (1 punto).
- Clase ***LineaTelefono*** (4 puntos).
- Programa principal ***AplicacionGestionTelefonia*** (4 puntos).
- Documentación Javadoc de las clases (1 punto).

Se trata de desarrollar una aplicación Java en consola que permita gestionar una lista de facturas telefónicas. El programa mostrará un primer menú con las siguientes opciones:

1. **Crear nueva línea de teléfono**.
2. **Eliminar línea de teléfono**.
3. **Gestionar línea de teléfono**.
4. **Consultar gastos totales**.

5. **Salir del programa**.

En caso de seleccionar el menú de **gestión de línea telefónica**, se mostrará el siguiente **submenú** para la línea seleccionada.

1. **Mostrar el número de teléfono completo.**
2. **Mostrar el nombre del titular de la línea.**
3. **Mostrar la fecha de permanencia.**
4. **Modificar la contraseña.**
5. **Realizar una llamada.**
6. **Consultar llamadas.**
7. **Consultar gasto total.**
8. **Volver al menú principal.**

Se creará un proyecto llamado **GestionTelefonia** que contendrá tres ficheros: **LineaTelefono.java**, **Llamada.java** y **AplicacionGestionTelefonia.java**.

- El primero de ellos contendrá la clase `LineaTelefono` con los atributos y métodos necesarios para trabajar con un objeto ***LineaTelefono***.
- El segundo contendrá la clase `Llamada` con los atributos y métodos necesarios para trabajar con un objeto ***Llamada***.
- El tercero contendrá la función principal *main* que gestionará la aplicación en modo consola de comandos.

Especificación del diseño de las clases

A continuación se detalla el diseño de cada una de las clases de la aplicación.

La clase Llamada

Una llamada telefónica se compone de los siguientes atributos privados: *duracion*, *destino* y *fecha.*

A continuación se muestran los requisitos que deben cumplir los atributos:

- La duracion en segundos de la llamada. Debe ser un valor positivo.
- El destino es el número de teléfono al que se llama. Se debe comprobar su validez mediante expresiones regulares. El Anexo I explica cómo puede ser el formato.
- La fecha en la que se ha realizado la llamada. Se establecerá automáticamente con la fecha y hora actual del sistema.

Se implementarán los siguientes constructores y métodos públicos:

- **(0,3 puntos)** Constructor con los parámetros del objeto. Si alguno de los parámetros no es válido se lanzará la excepción `IllegalArgumentException.`

```
public Llamada(double duracion, String destino);
```

- **(0,3 puntos)** Constructor copia.

```
public Llamada(Llamada ll);
```

- **(0,10 puntos)** Métodos ***get*** para los atributos: *duracion*, *destino*, *fecha*.
- **(0,10 puntos)** Método ***set*** para el atributo: *duracion*. Se debe comprobar su validez y no modificar el atributo si no se verifica.
- **(0,10 puntos)** Método ***toString*** para representar en modo texto el contenido del objeto.
- **(0,10 puntos)** Método ***equals*** para comparar dos objetos de tipo *Llamada*, que se consideran iguales si todos sus atributos son iguales.

La clase LineaTelefono

Una línea de teléfono se compone de los siguientes atributos privados: *titular*, *nif*, *contraseña*, *limite*, *mesPermanencia*, *añoPermanencia*, *numeroTelefono*, *llamadas* y *tarifa*.

A continuación se muestran los requisitos que deben cumplir los atributos:

- El nombre del titular debe validarse adecuadamente mediante una expresión regular y debe tener una longitud máxima de 80 caracteres y una longitud mínima de 15 caracteres.
- El NIF debe validarse adecuadamente mediante una expresión regular y podrá ser tanto un NIF, CIF o NIE.
- La contraseña debe validarse adecuadamente mediante una expresión regular y deberá tener una longitud mínima de 8 caracteres con al menos una mayúscula, una minúscula, un dígito y un carácter especial de entre los siguientes: $._#=%*.
- El límite de gasto será un valor entre 10 y 5000 euros.
- El mes de permanencia coincidirá con el mes actual (se obtendrá de la fecha actual al crear el objeto).
- El año de permanencia será 2 años después del año actual (se calculará a partir de la fecha actual al crear el objeto).
- El número de teléfono consta en general de 9 dígitos: los números de teléfono fijos empiezan por 9 u 8, y los móviles empiezan por 6 o 7. Este es el formato de número de teléfono que se permite crear para una línea telefónica.
- El atributo llamadas será un array de 50 elementos de objetos *Llamada*.
- La tarifa será una enumeración de tipo *TarifaTelefonica*. Los tipos de tarifas disponibles en el programa son: BASICA, NORMAL y PREMIUM. Cada tarifa tendrá asociado un coste distinto en función de la llamada que se realice.

Se implementarán los siguientes constructores y métodos públicos:

- **(1 punto)** Constructor con los parámetros de la línea de teléfono. Si alguno de los parámetros y/o el número de teléfono no es válido se lanzará la excepción `IllegalArgumentException`.

```
public LineaTelefono(String titular, String nif, String password, int limite, String numeroTelefono, TarifaTelefonica tarifa);
```

- **(0,5 puntos)** Constructor copia.

```
public LineaTelefono(LineaTelefono Linea);
```

- **(0,10 puntos)** Métodos ***get*** para los atributos: *titular, nif, contraseña, limite, mesPermanencia, añoPermanencia, numeroTelefono y tarifa.*
- **(0,10 puntos)** Métodos ***set*** para los atributos: *contraseña, limite* y *tarifa*. Se debe comprobar la validez de cada uno de ellos y no modificar el atributo si no se verifica.
- **(0,4 puntos)** Método ***llamar***. Realizará una llamada telefónica. La duración debe ser positiva y el destino un número de teléfono válido, en caso contrario se lanzará la excepción `IllegalArgumentException`. Debe comprobar que con la nueva llamada no se supera el límite de gasto de la línea usando la función *calcularCosteLlamada()*. La nueva llamada se añadirá al array de llamadas de la línea. Devolverá true si se ha podido realizar la llamada o false si no se puede realizar al superar el límite de gasto.

```
public boolean llamar(double duracion, String destino);
```

- **(0,20 puntos)** Método **gastado**. Devuelve la cantidad total gastada con las llamadas de la línea. Para cada llamada, debe obtener la duración y el destino de la llamada y obtener el coste de la misma utilizando la función *calcularCosteLlamada()*.

```
public double gastado();
```

- **(0,20 puntos)** Método **llamadas**. Devuelve la información en formato String de las llamadas realizadas en la línea. Recibe por parámetro el número de las últimas llamadas que se desean consultar. Debe mostrarse la fecha de la llamada, el teléfono al que se ha llamado, la duración y el coste de la llamada. Si no es un valor válido se lanzará la excepción `IllegalArgumentException`.

```
public String llamadas(int numero);
```

- **(0,10 puntos)** Método **número de llamadas**. Devuelve cuántas llamadas se han realizado en la línea.

```
public int numeroLlamadas();
```

- **(0,20 puntos)** Comprobar la validez de un número de teléfono. El método devolverá *true* o *false* indicando si el número es válido o no. Se deben utilizar expresiones regulares.

```
public static boolean comprobarNumeroTelefono(String numero);
```

- **(0,5 puntos)** Calcular el tipo de número de teléfono. El método devuelve el tipo de número de teléfono, un valor de la enumeración *TipoTelefono* que puede ser: FIJO, MOVIL, GRATUITO, COSTE_ADICIONAL, COSTE_COMPARTIDO. Se debe implementar usando expresiones regulares. En el Anexo I se explica cómo se determina el tipo de teléfono.

```
public static TipoTelefono obtenerTipoTelefono(String numero);
```

- **(0,5 puntos)** Calcular el coste de una llamada. El método devuelve el coste en euros de una llamada de teléfono en función del tipo de tarifa, el tipo de teléfono al que se está llamando y la duración. Si la duración no es un entero positivo se lanzará la excepción *IllegalArgumentException*. La fórmula para realizar el cálculo del coste de una llamada se encuentra en el Anexo II al final del documento.

```
private static double calcularCosteLlamada(TarifaTelefonica tarifa,
                    TipoTelefono tipo, int duracion);
```

- **(0,10 puntos)** Método **String toString()**. Devolverá una cadena con los datos del titular, el NIF, mes y año de permanencia, el número de teléfono, el límite de gasto y la cantidad de dinero que se ha gastado en las llamadas.
- **(0,10 puntos)** Método **boolean equals(LineaTelefono t)**. Compara dos líneas de teléfono, que serán iguales si tienen el mismo número de teléfono.

Puedes implementar los **atributos y métodos privados** que creas necesarios para mejorar la encapsulación y evitar la duplicación de código.

Clase AplicacionGestionTelefonia

El programa principal se encarga de gestionar la solicitud de datos al usuario y la creación de los objetos en un **array de objetos *LineaTelefono***. Para ello, el programa creará inicialmente un array vacío de **10 elementos de capacidad**. Durante la gestión del programa, se deberá comprobar en todo momento que no se supera el límite de creación de líneas telefónicas, avisando al usuario en caso de que desee crear una nueva línea de teléfono y no haya más espacio de almacenamiento.

Las opciones de la aplicación se mostrarán en un menú como el mostrado al principio de esta tarea. Se implementará del siguiente modo:

- **(1 punto) Crear línea telefónica**. Se llamará a una función (si hay espacio en el array) que se encargará de solicitar todos los datos al usuario y devolverá un objeto *LineaTelefono*. El programa guardará el objeto en la última posición libre del array siempre y cuando no exista otra línea creada con el mismo NIF.
- **(1,5 puntos) Eliminar línea telefónica**. Se pedirá al usuario el NIF del titular de la línea que se desea borrar. Se utilizará una función que recibirá por parámetro el NIF y el array de líneas telefónicas, y devolverá la posición del array donde se ha localizado (buscar una línea de teléfono). A continuación, se llamará a otra función para eliminar la línea que recibirá por parámetro el array de líneas telefónicas y el índice del elemento a borrar. La función de borrado deberá mover los objetos del array para ocupar el hueco y compactar el array.
- **(1,25 puntos) Gestionar línea telefónica**. Pedirá al usuario el NIF de la línea que desea gestionar y su contraseña. Para buscar la línea se reutilizará la función de búsqueda descrita en la opción anterior. Una vez seleccionado el objeto, si la contraseña es correcta, se mostrará al usuario el submenú de opciones de gestión de líneas telefónicas que se mostró al inicio de la descripción del proyecto. Este submenú es bastante autoexplicativo con lo que debe realizarse en cada una

de sus opciones. En caso de que no se introduzca la contraseña correctamente se mostrará un mensaje de error.

- **(0,25 puntos) Consultar gastos totales**. Se utilizará una función que recibe por parámetro el array de líneas telefónicas y devolverá el gasto total realizado en el programa.
- **Salir del programa**. Finaliza la aplicación.

Se deben comprobar y manejar los errores de forma adecuada para evitar que el programa termine de forma inesperada. Por tanto se comprobará que el usuario introduce valores correctos en todo momento.

Se valorará el diseño modular y estructurado, la claridad y simplicidad del código y su documentación. Todas las validaciones de las cadenas de caracteres se realizarán mediante expresiones regulares. Para ello, se reutilizará lo máximo posible el código creando funciones en los lugares donde creas que son más adecuados para realizar esta tarea.

Anexo I. Tipo de número de teléfono.

- **Fijo** (9 dígitos) : 9AX XX XX XX o 8AX XX XX XX (A!=0 y A!=9)
- **Móvil** (9 dígitos) : 6XX XX XX XX o 7AX XX XX XX (A = 1, 2, 3 o 4).
- **Gratuito** (9 dígitos): 800…, 900…,
- **Gratuitos especiales** (3 dígitos): 112, 0XX,
- **Coste adicional** (9 dígitos): 803…, 806…, 807…, 905…, 907…
- **Coste compartido** (9 dígitos): 901…, 902…

Más información: https://es.wikipedia.org/wiki/N%C3%BAmeros_de_tel%C3%A9fono_de_Espa%C3%B1a

Anexo II. Cálculo del coste de una llamada telefónica.

La siguiente tabla muestra los valores a aplicar para calcular el coste de una llamada telefónica en función del tipo de teléfono del destinatario de la llamada y de la tarifa telefónica del usuario.

	TIPO DE TARIFA		
TIPO TELÉFONO	**BÁSICA**	**NORMAL**	**PREMIUM**
FIJO	0,13 €/seg	0,1 €/seg	0 €/seg
MÓVIL	0,22 €/seg	0,2 €/seg	0,05 €/seg
GRATUITO	0 €/seg	0 €/seg	0 €/seg
COSTE_ADICIONAL	0,30 €/seg	0,30 €/seg	0,30 €/seg
COSTE_COMPARTIDO	0,15 €/seg	0,15 €/seg	0,15 €/seg

Entrega de la tarea

El proyecto se desarrollará con el IDE Netbeans. El nombre del proyecto, de las clases y métodos deben llamarse de la forma indicada en la descripción de la tarea. Se comprimirá la carpeta del proyecto Netbeans en un único fichero en formato .ZIP o 7z que se subirá al buzón de la tarea en la plataforma Moodle.

El archivo se nombrará siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG_ProyectoGestionTelefonia.zip

Resultados de aprendizaje y criterios de evaluación relacionados

En esta actividad se evalúan los siguientes resultados de aprendizaje con los criterios de evaluación que se relacionan para cada uno de ellos:

- RA 3. Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.
  - a) Se ha escrito y probado código que haga uso de estructuras de selección.
  - b) Se han utilizado estructuras de repetición.
  - c) Se han reconocido las posibilidades de las sentencias de salto.
  - d) Se ha escrito código utilizando control de excepciones.
  - e) Se han creado programas ejecutables utilizando diferentes estructuras de control.
  - f) Se han probado y depurado los programas.
  - g) Se ha comentado y documentado el código.
- RA 4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.
  - a) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.
  - b) Se han definido clases.
  - c) Se han definido propiedades y métodos.
  - d) Se han creado constructores.
  - e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.
  - f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.
  - h) Se han creado y utilizado métodos estáticos.

- RA 5. Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

  a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

  b) Se han aplicado formatos en la visualización de la información.

  c) Se han reconocido las posibilidades de entrada / salida del lenguaje y las librerías asociadas.

- RA 6. Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

  a) Se han escrito programas que utilicen arrays.

  g) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.

Rúbrica de evaluación

Cada apartado puntuable del proyecto se valorará con la siguiente rúbrica.

#	Criterio	Porcentaje
1	El programa/función implementado no cumple los requisitos, no soluciona de forma algorítmica el ejercicio o las soluciones obtenidas por el programa no son las esperadas, el código no compila o contiene errores.	0%
2	El programa/función implementado soluciona de forma algorítmica el ejercicio pero falla con datos de entrada no permitidos.	25%
3	El programa/función implementado tiene fallos inesperados en situaciones específicas o concretas, es decir, falla para un determinado caso o valor de entrada, pero en general el resultado obtenido es válido.	50%
4	El programa/función implementado cumple los requerimientos pero: • El código no es legible o no está bien estructurado.	75%
5	El programa/función implementado se ajusta perfectamente a la especificación: • Se validan los datos de entrada. • El resultado obtenido es válido para cualquier dato de entrada. • El código es modular y se emplean funciones/métodos adecuadamente. • El código es legible y usa comentarios relevantes y/o Javadoc.	100%