EMPRESA DE MARKETING TELEFÓNICO

Crea un proyecto llamado *Marketing*.

El programa **simulará** la gestión de llamadas telefónicas diarias de una pequeña empresa de marketing telefónico. Mediante un menú se podrá dar de alta a nuevos contactos y se podrán obtener los contactos a los que vamos a telefonear hoy.

Se utilizarán las siguientes clases:

Clase Dirección

Atributos de la clase: población (String) y provincia (String).

Clase Contacto

Atributos de la clase: nombre (String), teléfono (String), dirección (Direccion), numeroDeLlamadas (int): es un contador del número de llamadas realizadas a esta persona.

La clase contiene el atributo static *limiteLLamadas*. Su valor inicial es 10.

Escribe en las clases los métodos que consideres necesarios: Constructor o constructores adecuados, Setters y Getters, toString(), equals, etc.

Clase Menu

El programa utilizará la siguiente clase Menu:

```
public class Menu {
    private int opcion;
    public void mostrar() {
        System.out.println(" 1. Nuevo contacto");
        System.out.println(" 2. Llamar hoy");
        System.out.println(" 3. Mostrar contactos");
        System.out.println(" 4. Mostrar contactos de una población");
        System.out.println(" 5. Inicializar");
        System.out.println(" 6. Modificar límite de llamadas");
        System.out.println(" 0. FIN");
    }
    public int leer() {
        Scanner sc = new Scanner(System.in);
        do {
            System.out.print("Introduzca opción: ");
            opcion = sc.nextInt();
        } while (opcion < 0 || opcion > 6);
        return opcion;
    }
}
```

Clase Principal

Se proporciona una clase principal básica que se deberá completar para conseguir que el programa funcione correctamente:

```
public class Marketing {
    static ArrayList<Contacto> contactos = new ArrayList<>();
    static ArrayList<Contacto> llamarHoy = new ArrayList<>();
    static ArrayList<Contacto> noDisponibles = new ArrayList<>();
    public static void main(String[] args) {
       ////Mostrar el menú y llamar al método correspondiente según la opción elegida.
       ////El programa finaliza cuando se introduzca 0 como opción de menú.
    }

    /// Métodos correspondientes a cada opción de menú
    /// y resto de métodos que consideres necesarios
}
```

Los contactos se almacenan en el ArrayList *contactos*.

El ArrayList *llamarHoy* contendrá los contactos a los que vamos a llamar hoy. Su contenido se genera de forma aleatoria a partir del ArrayList *contactos*.

El ArrayList *noDisponibles* contiene los contactos a los que hemos llamado un número de veces igual al límite de llamadas. Se añadirá un contacto a este ArrayList cuando su número de llamadas supere el valor del atributo static limiteLLamadas.

Métodos que se deben ejecutar correspondientes a cada opción del menú:

Opción 1: método *nuevoContacto()*. Añade un nuevo contacto al ArrayList *contactos*. Se introducen por teclado los datos del contacto y se añade al ArrayList. **El número de llamadas no se introduce**. Inicialmente su valor será 0.

No puede haber contactos repetidos. Si se intenta añadir un contacto que ya existe se mostrará un mensaje indicándolo y se vuelven a pedir los datos del contacto. Consideramos que dos contactos son iguales si tienen el mismo nombre.

También se debe comprobar que el teléfono es válido. Se considera que un teléfono es válido si tiene una longitud de 9 caracteres, empieza por 6 ó 7 y solamente contiene dígitos. Si no es válido se mostrará un mensaje indicándolo y se vuelve a pedir.

Opción 2: método *llamadasParaHoy()*. Mediante este método se obtienen los clientes que llamaremos hoy. El proceso será el siguiente:

Se eliminan todos los elementos del ArrayList *llamarHoy.*

Se pide al usuario que introduzca el número N de clientes a los que vamos a llamar hoy. Este número deberá estar comprendido entre 1 y el número máximo de contactos disponibles para llamar. El número máximo de contactos será el total de contactos menos el número de contactos no disponibles. Si no hay contactos disponibles para llamar se muestra un mensaje y se vuelve al menú.

Se seleccionan al azar los N contactos para llamar hoy de entre todos los contenidos en el ArrayList de *contactos* y se añaden al ArrayList *llamarHoy*. Si durante este proceso se elige un contacto que ya ha sido seleccionado previamente o se selecciona un contacto que no está disponible, se vuelve a elegir otro en su lugar. A cada contacto seleccionado se le incrementará en 1 el valor del atributo *numeroDeLLamadas.* Si después de incrementar el número de llamadas se alcanza el límite de llamadas, el contacto se añade al ArrayList *noDisponibles*.

Opción 3: método *mostrarContactos()*. Este método muestra lo siguiente:

El valor actual del límite de llamadas.

El contenido del ArrayList *contactos* ordenados alfabéticamente por nombre.

El contenido del ArrayList *noDisponibles* ordenado alfabéticamente por nombre.

El contenido del ArrayList *llamarHoy* ordenado por número de llamadas de mayor a menor. Si varios contactos tienen el mismo número de llamadas aparecerán ordenados alfabéticamente por nombre.

Opción 4: método *contactosDeUnaPoblacion()*.

Se introduce por teclado el nombre de una población y se muestran los contactos de esa población. Si no hay contactos en esa población se muestra un mensaje indicándolo.

Opción 5: método *inicializar()*.

Se elimina el contenido del ArrayList noDisponibles.

Se pone a cero el valor del número de llamadas en todos los contactos del ArrayList contactos.

Antes de realizar este proceso se pedirá la confirmación del usuario.

Opción 6: método *modificarLimiteLLamadas()*.

Se muestra el valor actual del límite de llamadas, se pide el nuevo valor y se modifica.

El límite de llamadas debe ser mayor que 0.

Este cambio afecta al ArrayList noDisponibles: se elimina el contenido actual del ArrayList noDisponibles y se pasan a este ArrayList todos los contactos que superen el nuevo límite de llamadas.