NOMBRE Y APELLIDOS		CICLO FORMATIVO	CURSO
		DAW/DAM	1º
DNI	FECHA	FECHA	CONVOCATORIA
	5 DE JUNIO DE 2024	PROGRAMACIÓN	ORDINARIA

IMPORTANTE: CADA ENTREGABLE DEBE COMPILAR PARA SER CORREGIDO.

LA RESOLUCIÓN DEL PROBLEMA SE HA DE REALIZAR OBLIGATORIAMENTE SIGUIENDO LA PROGRAMACIÓN ORIENTADA A OBJETOS

Una empresa dedicada a los juegos de azar desea desarrollar una aplicación para sus locales. Para tal fin, desean empezar por diseñar una aplicación para el bingo. En el bingo, podrán jugar un número ilimitado (solamente limitado por los recursos del equipo informático) de jugadores, donde cada jugador dispondrá de un numero de cartones sin determinar. Se desea desarrollar la aplicación en base a las restricciones descritas más abajo.

Modelo de datos (4p)

a. El bingo, contendrá diferentes bolas, que además de un número, se quiere que se muestren con un color, que puede variar según desee el dueño del local. Recordad que los colores se almacenan como enteros (podéis usar los valores de los colores verde y rojo según si el número de la bola es par o impar).

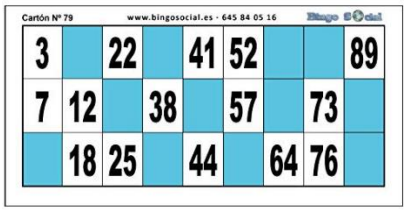
b. Los jugadores, independientemente de si juegan al bingo o a otro juego, dispondrán de un nombre, apellidos y un saldo. Los jugadores de bingo dispondrán de una cantidad ilimitada de cartones diferentes.

c. Se desea diseñar un bombo de tal manera que pueda ser utilizado tanto para bolas de bingo como para en un futuro albergar bolas de lotería u otro tipo de objeto todavía por definir. Se ha de diseñar de tal manera que un bombo no pueda contener 2 bolas iguales, entendiendo por bolas de bingo iguales, aquellas que tienen el mismo número independientemente de su color. **Para evitar este problema, se debe desarrollar el bombo utilizando la colección Set**.

NOMBRE	APELLIDOS

d. Los bombos utilizados para el bingo, que tienen 90 bolas. El bombo sólo puede sacer una bola cada vez. Se debe asegurar que cada 5 bolas extraídas, se girará el bombo del bingo para que se mezclen de nuevo las bolas.

e. Los cartones contendrán 10 números que pueden ir desde el 1 al 90 sin repetición y ordenados. **Para evitar este problema, se debe desarrollar el cartón utilizando la colección Set.** Los cartones heredarán de la vista LinearLayout, y estará formado por una serie de text views que contendrán los números.

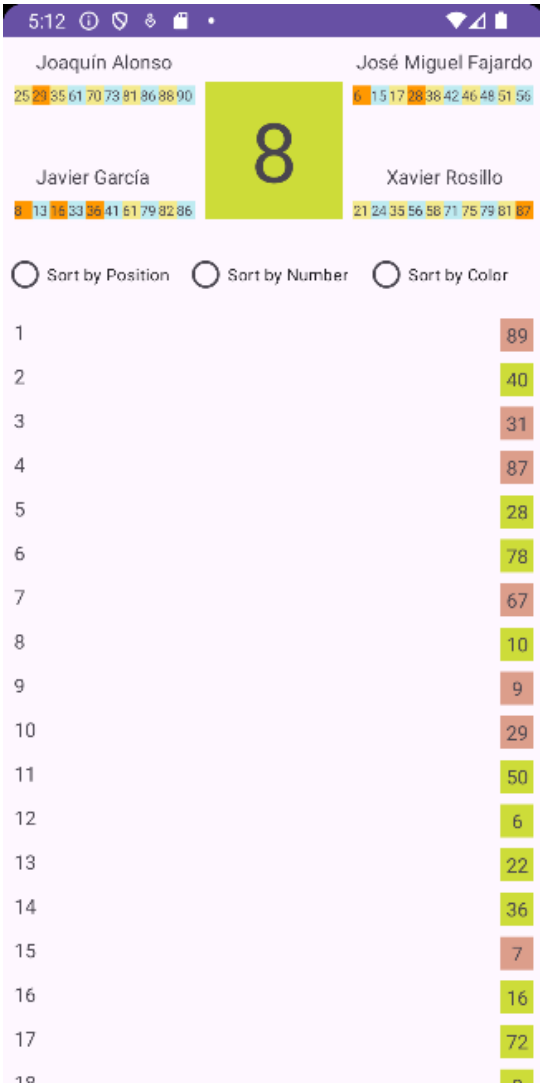


NOMBRE	APELLIDOS

Para poder comprobar el funcionamiento del sistema, se desea crear una App con las siguientes características.

Aplicación (4p).

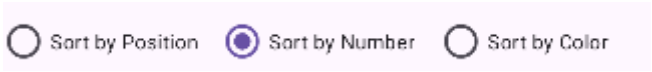
- Diseña una app con una interfaz similar a la siguiente:



- La app tendrá 3 partes:
    1. Se debe visualizar la bola que sale del bombo. deberá mostrar tanto el numero como el color. Para volver a sacar otra bola, bastara con hacer clic sobre el TextView.

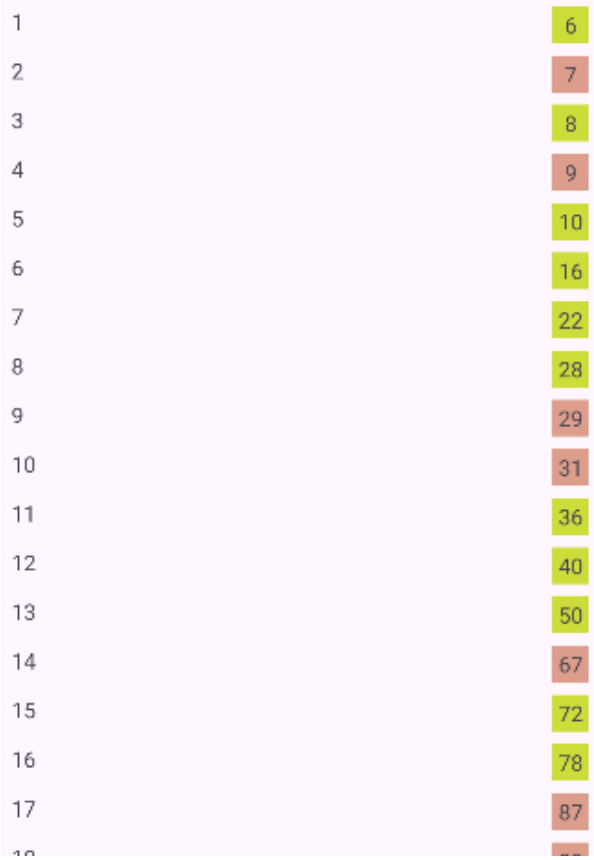
NOMBRE	APELLIDOS

2. Se dispondrán de una serie de RadioButtons para ordenar el RecyclerView de tres maneras diferentes:



   - **Sort by Position**: El RecyclerView mostrará las diferentes bolas según el orden en el que han ido saliendo
   - **Sort by Number**: El RecyclerView mostrará las bolas ordenadas por número.
   - **Sort by Color:** El RecyclerView mostrará las bolas ordenadas **por su orden natural**, este es: primero por color y ante dos bolas del mismo color, por número.

3. Se dispondrá de un RecyclerView, donde podremos ver las bolas que han salido. Estas deberán ordenarse cuando el usuario haga clic en los RadioButtons anteriores:



NOMBRE	APELLIDOS

- Se pide hacer una simulación para 4 jugadores. Aunque las clases puedan utilizar más de un cartón y deban estar desarrolladas para ello, en la app, para simplificar solo se podrá jugar con un cartón.

- A medida que el juego vaya discurriendo, cada vez que se obtenga una bola del bombo, se debe:
    1. Mostrar la bola en el display
    2. Añadir una entrada al RecyclerView.
    3. Que cada jugador marque en sus cartones el número que ha salido.
    4. Si hay alguien que tenga bingo, se muestre un Toast indicando el/los ganador/es, impidiendo que el juego siga. Esto último, se puede conseguir, simplemente ocultando el display.

NOMBRE	APELLIDOS

Annex I. Obtencion color desde colors.xml

```
getResources().getColor(R.color.black, getContext().getTheme());

Recuerda que para establecer el color de fondo puedes utilizar la función:
setBackground(context.getDrawable(aquí el color);
```

Annex II. Esquema incompleto de un adaptador para un RecyclerView

```
public class MyRecyclerViewAdapter extends
RecyclerView.Adapter<MyRecyclerViewAdapter.MyViewHolder> {

    private LayoutInflater inflater;
    private Context context;

    public MyRecyclerViewAdapter(@NonNull Context context) {
        this.context = context;
        inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @NonNull
    @Override
    public MyRecyclerViewAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view = inflater.inflate(R.layout.simple_element,parent,false);
        return new MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyRecyclerViewAdapter.MyViewHolder
holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }

    public class MyViewHolder extends RecyclerView.ViewHolder{

        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
}
```