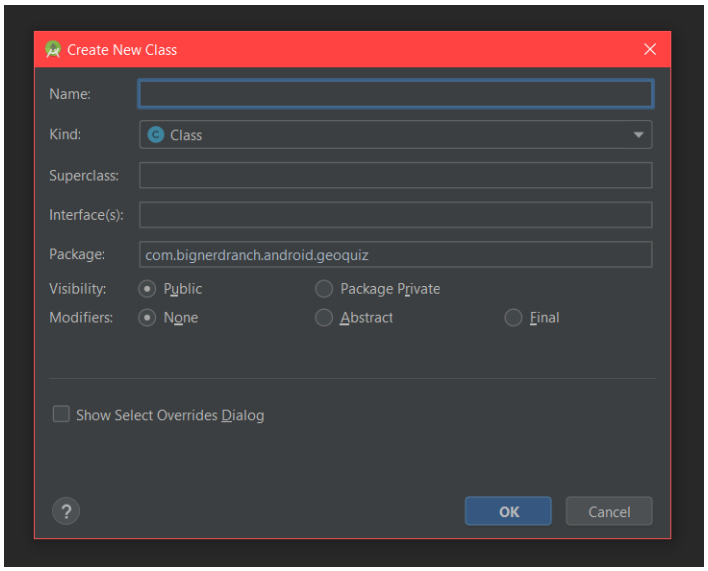
2. Android y Model-View-Controller

En este capítulo se verá actualizado el proyecto de GeoQuiz para presentar más de una pregunta al usuario.

Para poder tener distintas preguntas que contestar, será necesario implementar un pequeño modelo de datos.

Crearemos la clase Question, que contendrá el texto a mostrar de la pregunta y el valor de la respuesta. El texto de la pregunta lo guardaremos como un identificador de recurso, para mantener la aplicación bien estructurada.

La forma con la que crear una nueva clase, deberemos seleccionar el paquete en el que se encuentra nuestra QuizActivity, clicar con el botón derecho y seleccionar New → Java Class.



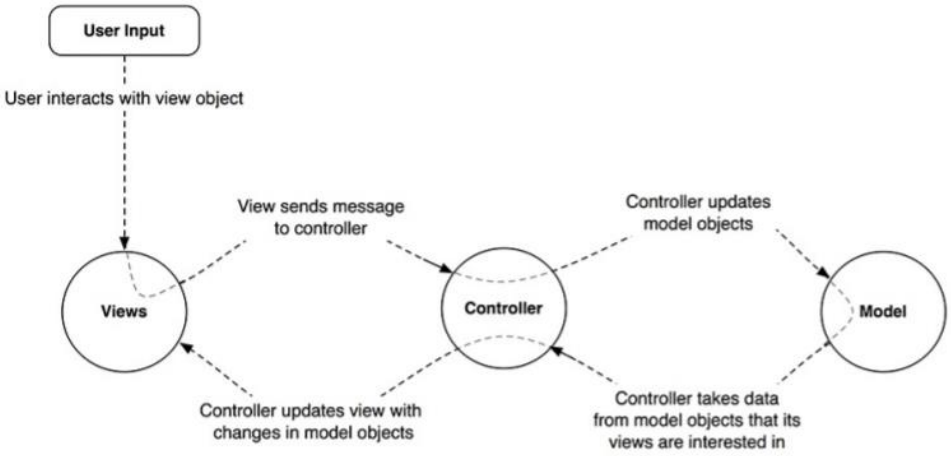
En el campo Name pondremos el nombre de la clase, en el campo Superclass pondremos la clase de la que hereda la que estamos creando (si es que hereda de alguna).

Una vez creada la clase, podremos definir los atributos de la clase junto con su constructor y los *getters* y *setters*[1].

Ya hemos creado la clase Question, pero para seguir la estructura MVC deberemos modificar la clase QuizAtivity para que actúe como controlador. De esta forma, tendremos la siguiente estructuración del proyecto:

- Model: La clase Question.
- Controller: La clase QuizActivity.
- View: Todos aquellos recursos que diseñamos, como por ejemplo las pantallas.

A partir de ahora, el flujo de nuestra aplicación será el descrito en el siguiente esquema.



[1] Una forma fácil de crear las funciones básicas de la clase es pulsando las teclas Alt+Insert, y nos propondrá las distintas funciones predeterminadas que podremos crear.

Ahora que ya hemos indagado en el MVC, modificamos la vista de la aplicación para mostrar un botón para pasar a la siguiente pregunta.

Como podemos ver en el código del layout de la Quiz Activity, a partir de ahora no habrá más texto hardcoded, sino que todo texto estará declarado como recurso en el archivo res/values/strings.xml. ¡Esto ayudará a depurar mejor el código de la aplicación y en un futuro a poder traducir la aplicación a cualquier idioma!

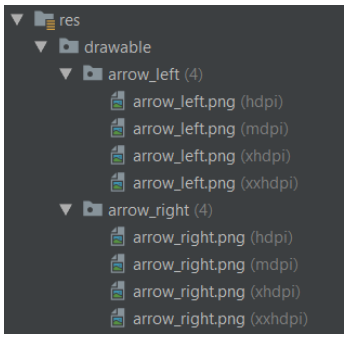
El siguiente paso para poder albergar distintas preguntas, será definirlas de forma estática en el controlador, a forma de array, junto con un índice que nos indicará la pregunta en la que nos encontramos. Cuando el usuario seleccione alguna respuesta, deberemos mirar el valor correcto de la respuesta y mostrar el mensaje en la parte inferior de la pantalla. En cambio, cuando el usuario apriete el botón de Siguiente pregunta, deberemos actualizar el índice de la pregunta a la siguiente, y actualizar la pantalla para mostrar la siguiente pregunta.

Añadir un icono

Como bien hemos podido apreciar, el botón para pasar a la siguiente pregunta dispone de un icono de una flecha hacia la derecha. Los iconos que utilicemos en nuestra aplicación deberán encontrarse guardados en el directorio ***res/drawable***, donde podremos especificar distintos recursos para un mismo icono, de forma que cada uno de ellos sirva para una densidad de pantalla distinta. Veremos que cada uno de estos recursos tendrá distintas densidades, caracterizadas por el sufijo del propio recurso:

- **mdpi**: Densidad media (160dpi)
- **hdpi**: Densidad alta (240dpi)
- **xhdpi**: Extra-densidad alta (320dpi)
- **xxhdpi**: Extra-extra-densidad alta (480dpi)

Existen más valores de **densidades de pantalla**, pero por el momento utilizaremos tan sólo estas.



Nosotros como programadores no deberemos especificar cuál de los recursos del icono escogeremos para según que pantallas, sino que el sistema operativo lo hará por nosotros y escogerá la mejor resolución en función del dispositivo que corra la aplicación.

Nota: Si el dispositivo que corre la aplicación tiene una densidad de pantalla distinta a las que hemos incluido en el proyecto, **Android escalará automáticamente la imagen al tamaño apropiado para el dispositivo.**

Nota: El recurso que añadamos al proyecto debe ser del tipo **.png, .jpg o .gif**, y el nombre debe estar **en minúsculas y sin espacios**.

Referenciaremos el icono para que aparezca en la parte derecha del botón de esta forma:

```
<Button
    android:id="@+id/next_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Next"
    android:drawableEnd="@drawable/arrow_right"
    android:drawablePadding="4dp"/>
```

Como podemos ver, el propio botón ya nos deja poner un recurso a la derecha de este, como si de otro atributo se tratara. Tan sólo deberemos llamar a su recurso y listo.

En capítulos posteriores se tratará más a fondo cómo trabajar con el directorio */res*.

Si probamos la app, veremos que, aunque cambiemos de pregunta, *si rotamos la pantalla, nos vuelve a aparecer la primera pregunta.* ¿Cómo es eso posible? La respuesta a esa pregunta es el **ciclo de vida de una actividad**, que se tratará en los próximos capítulos.

Challenges

Si te ves con ganas de descubrir un poco más por tu cuenta, puedes intentar:

- Añadir un Listener al TextView con tal de que, cuando el usuario apriete la pregunta, pase a la siguiente.
- Añadir un botón de “Previous” para volver a la pregunta anterior.
- Substituir los botones de “Previous” y “Next” por dos iconos de flechas hacia la derecha y hacia la izquierda, tal que así: