UNA
UNIVERSIDAD NACIONAL
COSTA RICA

Diseño y Programación de Dispositivos Móviles

EIF 204

Profesor: Ms.C Gregorio Villalobos Camacho

Correo: gregorio.villalobos.camacho@una.ac.cr

Android Studio: Carpetas

- manifest
  - AndroidManifest.xml
    - Tiene configuraciones básicas del programa
    - Todos los activities se deben registar aquí
    - Note cómo existe ya el registro del Activity Main
- Java
  - Carpeta con el nombre del pack y el app
    - Tiene todas las Activities de mi app (Aquí se agregan)

Android Studio: Carpetas (Cont)

- Java
  - AndroidTest
    - AKA test de instrumentación
    - Se colocan aquí todas las pruebas que ocupan android
  - Test
    - AKA tests unitarios
    - Esta se usa para pruebas directas a clases
    - No necesita la estructura completa de Android

Android Studio: Carpetas(Cont)

- Res: AKA resources (Recursos)
  - Drawable: graficos que usa la aplicación
  - Layout
    - Contiene todos los XML de los diferentes Activities
  - Mipmap: para gráficos que no son rescalables
    - Están los íconos con distintas resoluciones
    - También están los íconos redondeados

Android Studio: Carpetas(Cont)

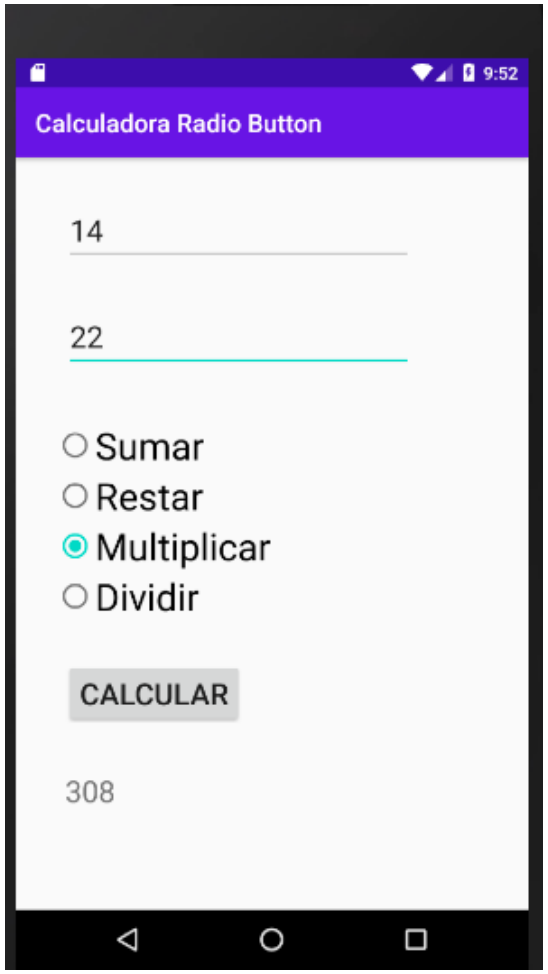
- Res: AKA resources (Recursos)
  - Values
    - Colors: tiene todos los colores RGB del APP
    - String: se puede definir todos los que usa el APP
    - Styles: Acá se define los estilos que utiliza el APP

Android Studio: Carpetas(Cont)

- Gradle Scripts
  - Project: es el principal del Proyecto
  - Module
    - Hay uno por cada modulo que vamos a crear
    - Tiene versión de
      - Compilación
      - Herramientas
      - Versión minima del SDK
      - Versión target
      - Se puede habilitar encripción en minifyEnable
  - Cualquier cambio solicita sincronización

Ejercicio

- Hagamos una calculadora utilizando
  - Radio Buttons



Eliminar: hardcoded string

- Ingresamos a res.values.strings.xml
- Se debe agregar todo el texto que se necesita
- Veamos un Ejemplo

```
<string name="valor1_string">Valor1</string>
```

- El anterior se puede invocar en un elemento

```
@string/valor1_string
```

- Se coloca ya sea en los atributos
  - Text
  - Hint

Multiples Activities

- Se agrega un nuevo ”Empty Activity”
- En el mainActivity se agrega un botón
- En el onCreate del mainActivity
  - Se declara una variable botón y se conecta
  - Se debe habilitar al botón setOnClickListener

```
boton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openActivity2();
    }
});
```

  - Se debe crear el método openActivity2()

Multiples Activities

- Se debe crear el método openActivity2()

```
public void openActivity2() {
    Intent intent = new Intent(this, Activity2.class);
    startActivity(intent);
}
```

- Ejercicio
  - Ejecute esta modificación a un proyecto actual

Envío de valores entre Activities

- En la clase principal del Activity1
  - Se crea un valor String constante para cada valor
  - Se recomienda usar como prefijo el nombre del pack

```
public static final String EXTRA_TEXT = ”packName.EXTRA_TEXT";
public static final String EXTRA_NUMBER = ”packName.EXTRA_NUMBER";
```

- En el método que abre el Activity2
  - Se extrae la información que se necesita
    - Puede ser en variables (string, int,…..)
  - Se envía la información con intent.putExtra

```
intent.putExtra(EXTRA_TEXT, nombreSTR);
intent.putExtra(EXTRA_NUMBER, idInt);
```

Recepción de Valores entre Activities

- En el activity2 en el onCreate
  - Se crean las variables donde se recibe la info

```
String nombre_str;
int id_int;
Intent intent = getIntent();
```

  - Se carga la información en dichas variables

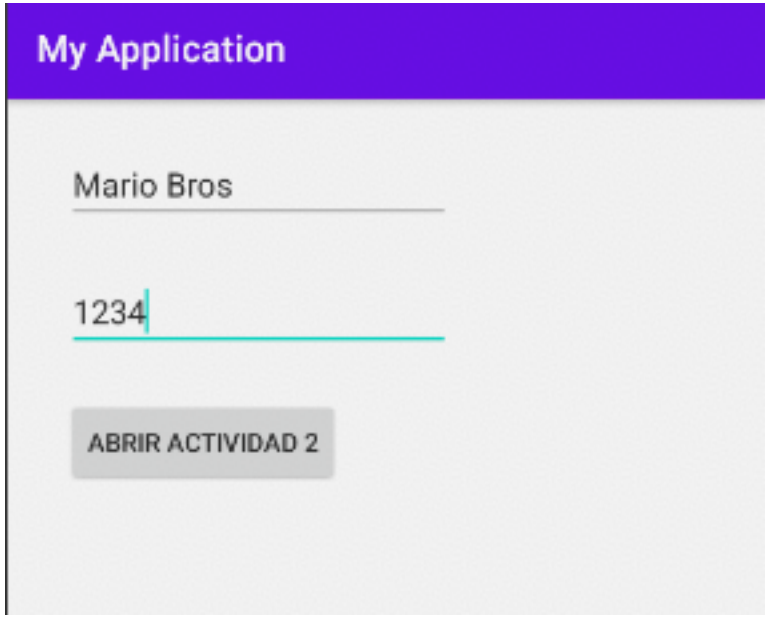
```
nombre_str = intent.getStringExtra(MainActivity.EXTRA_TEXT);
id_int = intent.getIntExtra(MainActivity.EXTRA_NUMBER,0);
```

  - Se puede utilizar

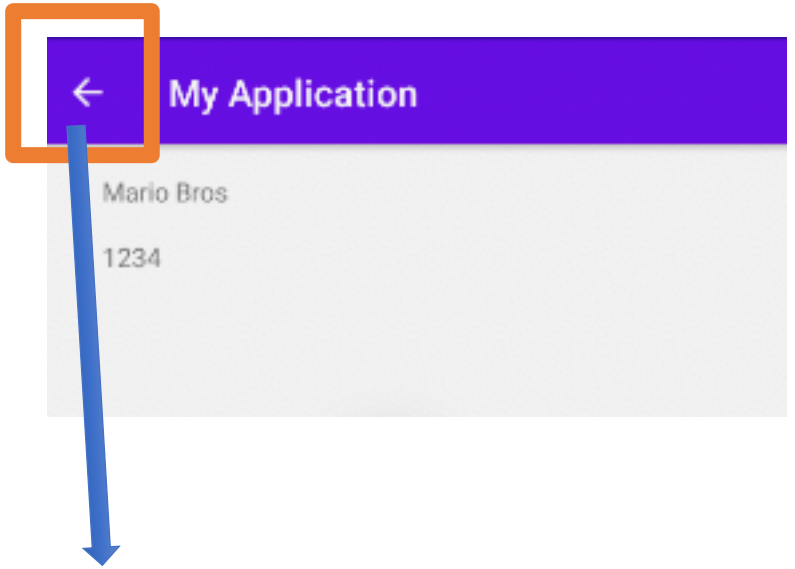
```
tvId.setText(""+ id_int);
tvNombre.setText(nombre_str);
```

Ejercicio: Navegación

Activity Principal



Activity 2



- En el Android.Manifest.xml
- Modifique el tag de Activity2
- Agregue
- android:parentActivityName

Para recibir datos de vuelta

- En el activity inicial (Parent)
  - Al llamar al segundo activity
  - Se construye igual el intent
  - Se envían los datos que necesita (putExtra)
  - Se invoca el nuevo activity con

  startActivityForResult(intent, 1);

Para recibir datos de vuelta(Cont)

- En el activity secundario (child)
  - Se recibe el intent, se extrae su información
  - Se puede regresar con un botón este debe
    - Crear un nuevo Intent para devolver
    - Colocar los parámetros en el intent (putExtra)
    - Ejecutar lo siguiente al final

```
setResult(RESULT_OK, resIntent);
finish();
```

Para recibir datos de vuelta(Cont)

- En el activity principal se recibe en
  - El método onActivityResult
  - Se debe verificar el valor de

    ResultCode == 1

    RequestCode == RESULT_OK
  - Se extrae los datos desde data

    data.getIntExtra("resultado", 0);
  - Se utilizan los valores devueltos del activity child

Android Components: Spinner

- En otros ambientes se conoce cómo
  - Drop Down Button
  - Se debe crear un arreglo tipo String
  - Esto se hace al inicio de la clase principal

```
String [] opciones = {"Sumar", "Restar"};
```

  - En el método onCreate se crea un ArrayAdapter

```
ArrayAdapter <String> adaptador1 = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, opciones);
```

  - Luego se configura el spinner con el adaptador

Android Compnents: Spinner

- Cuando vamos a evaluar el Spinner
  - Preguntamos

  spinner1.getSelectedItem().equals("Sumar")
  - Note que es sensitivo a mayúsculas y minúsculas
  - De esta forma sabemos que está seleccionado
  - Con un simple if ejecutamos lo que se requiere

- Ejercicio
  - Resuelva la Calculadora usando Spinner

Spinner + Custom Layout

- Modifica cómo se ve el Layout del Spinner
- Debemos agregar un nuevo Layout
  - En Res-Layout cree un nuevo layout
  - Ponga un nombre descriptivo y ponemos el código
  - Agregue la siguiente información

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:textColor="#253FD0"
    android:padding="10sp"
    android:textSize="24sp"
    xmlns:android="http://schemas.android.com/apk/res/android" />
```

Spinner + Custom Layout (Cont)

- En el .Java de la ventana principal
- Se debe modificar el Spinner
- Cambiamos lo siguiente

```
ArrayAdapter <String> adaptador = new ArrayAdapter<String>(this, R.layout.spinner_intem_prueba, opciones);
spinner1.setAdapter(adaptador);
```

- En su ejercicio
  - Ejecute los cambios y visualice el nuevo spinner

Android Components: ListView

- Es similar al Spinner
  - Lo agregamos
  - Los ítems se deben agregar en el .java
  - Se modifica su despliegue con nuevo Layout
- Ejercicio
  - Cree un proyecto que contenga
    - 1 textView y 1 ListView (conecte XML con java)
    - Cree 2 arreglos: nombres y edades (igual num elementos)
    - Cree el nuevo ArrayAdapter
    - Configure el ArrayAdapter con el ListView
    - Ejecute y visualice (Vea su Layout aplicado)

Android Components: ListView

- Codifique en el OnCreate
  - Al ListView creado le agregamos
  - setOnItemClickListener
    - Como es una función le agregamos entre paréntesis

    new AdapterView.OnItemClickListener()
    - Esta opción anterior nos agrega una función anónima
    - Con toda su estructura lista para usarse
    - En el código agregamos

```
tv1.setText("La edad de " + lv1.getItemAtPosition(position) + " es " +
edades[position] + " años");
```

Control ImageButton

- Se debe conseguir imágenes pequeñas
  - Por ejemplo 50X50 pixels
  - Pueden ser jpeg o png
- Se debe salvar en la carpeta
  - Res-mipmap
  - Para esto se puede copiar desde el OS
  - Y dentro de la carpeta se da pegar
  - Los nombres de las imágenes debe ser solo minúsculas

Control ImageButton

- Con las imágenes en la carpeta
- Se puede agregar un ImageButton
  - Al agregar el botón en la ventana de configuración
  - Se pasa al tab de Mip Map
  - Ahí están las imágenes que pegamos
  - Se seleccionan y se puede visualizar
  - El botón funciona igual por dentro

Control ImageButton

- Ejercicio
  - Cree un proyecto nuevo
  - Agregue 2 imágenes al proyecto en mipmap
  - Agregue dos ImageButton y le configura las imágenes
  - En MainActivity.java
    - Agregue dos métodos que ejecuten un Toast c/u
    - Configure los botones con dichos métodos
  - Ejecute y vea el funcionamiento

Validación de EditText

- Las validaciones se deben programar
- En el MainActivity.java
  - Se conectan los campos de texto XML/java
  - Se configura la validación en el botón
    - Se obtiene la info del EditText dentro de un String
- String tiene
  - .isEmpty()
  - .length()
  - .equals()

Validación de EditText

- Ejercicio
  - Verifique 2 campos de texto (Usuario y password)
  - Al presionar un botón

Email/Password

- Correo
  - Podemos cargar el campo en un String
  - Comparamos el campo con el patrón predefinido
  - El patrón está definido con ***Regular Expression***

```
if (!Patterns.EMAIL_ADDRESS.matcher(correo).matches())
```

  - Si no concuerda con el patrón

```
etCorreo.setError("El formato del correo no es correcto");
```

  - Si el patrón se cumple

```
etCorreo.setError(null);
```

Email/Password

- Password
  - El patrón de correo no existe
  - Al inicio de la clase MainActivity se agrega

```
private static final Pattern PASSWORD_PATTERN =
        Pattern.compile("^" +
                "(?=.*[0-9])" +         //al menos un dígito
                "(?=.*[a-z])" +         //al menos una minúscula
                "(?=.*[A-Z])" +         //al menos una mayúscula
                "(?=.*[a-zA-Z])" +      //cualquier letra
                "(?=.*[@#$%^&+=])" +    //caracteres especiales
                "(?=\\S+$)" +           //si espacios en blanco
                ".{4,}" +               //minimo 4 caracteres
                "$");
```

Email/Password

- Password
  - Podemos cargar el campo en un String
  - Comparamos el campo con el patrón que creamos

```
if (!PASSWORD_ADDRESS.matcher(password).matches())
```

  - Si no concuerda con el patrón
    - Coloque todo el mensaje con lo que debe contener

```
etPass.setError(”Password débil!");
```

  - Si el patrón se cumple

```
etPass.setError(null);
```