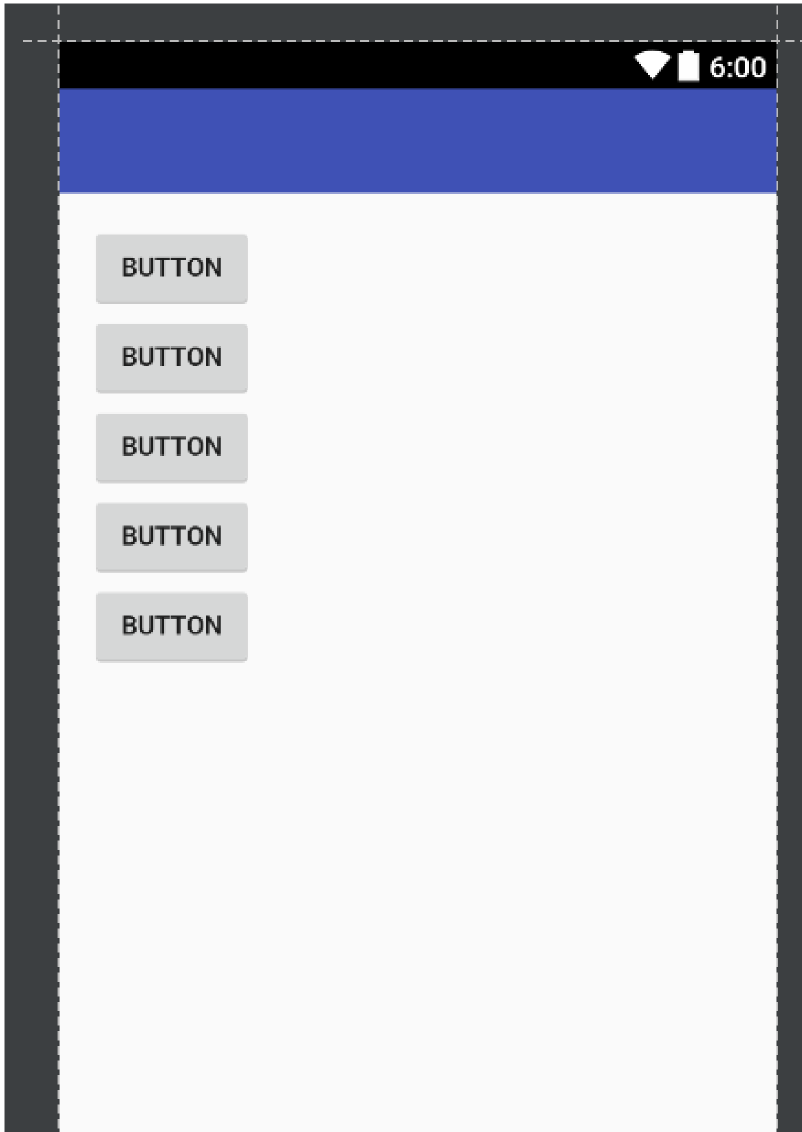
Java Android

BasicViewOptions

BasicViewOptions

Create user interface looking like this:



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/content_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.amen.basicviewoptions.MainActivity"
    tools:showIn="@layout/activity_main">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_alignParentTop="true"
            android:layout_alignParentEnd="true">
            <Button
                android:text="Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button2" />
            <Button
                android:text="Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button3" />

            <Button
                android:text="Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button6" />
            <Button
                android:text="Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button4" />
            <Button
                android:text="Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button5" />
        </LinearLayout>
    </ScrollView>
</RelativeLayout>
```

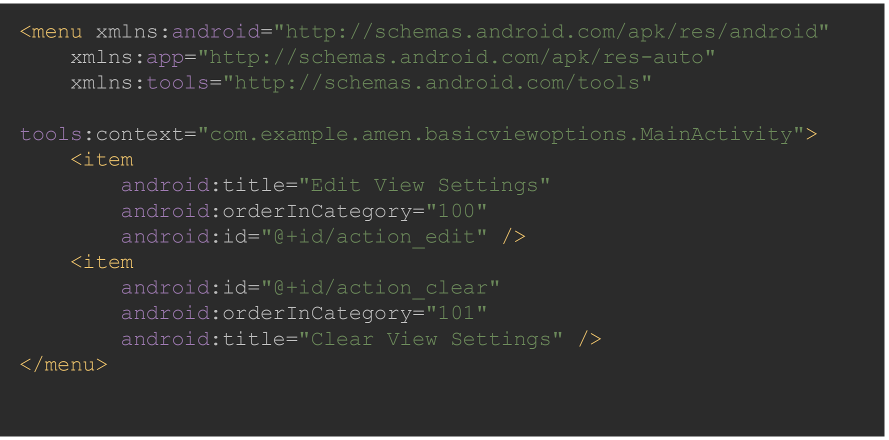
BasicViewOptions

In the **MainActivity** bind your controls:

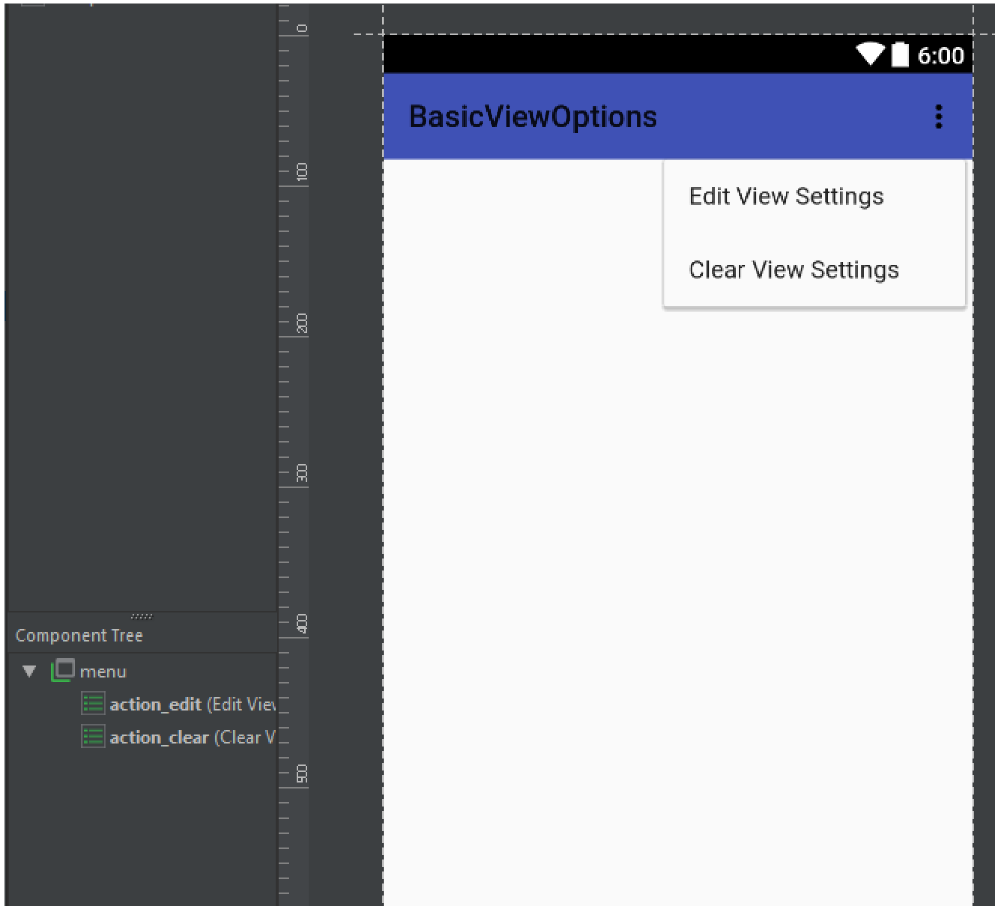
```
private Button btn1, btn2, btn3, btn4, btn5;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btn1 = (Button) findViewById(R.id.button6);
    btn2 = (Button) findViewById(R.id.button2);
    btn3 = (Button) findViewById(R.id.button3);
    btn4 = (Button) findViewById(R.id.button4);
    btn5 = (Button) findViewById(R.id.button5);

}
```

BasicViewOptions

We want our application to have context menu in the top right corner of the screen. To do so, we need to override onCreateOptionsMenu, but first, we need to define menu in the **res/menu** folder. Call it **menu_main.xml**.



```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"

tools:context="com.example.amen.basicviewoptions.MainActivity">
    <item
        android:title="Edit View Settings"
        android:orderInCategory="100"
        android:id="@+id/action_edit" />
    <item
        android:id="@+id/action_clear"
        android:orderInCategory="101"
        android:title="Clear View Settings" />
</menu>
```



BasicViewOptions

Now use this menu and call menuInflater when activity creates it’s menu:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Handling menu actions is very simple. I gave my menu options simple id, and whenever they are pressed MenuItem calls event on Activity and invokes OnOptionsMenuSelected:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_clear) {

        // obsługa czyszczenia
        // ...
        return true;
    } else if (id == R.id.action_edit) {

        // obsługa edycji
        // ...
        return true;
    }

    return super.onOptionsItemSelected(item);
}
```

BasicViewOptions

Now, i will implement clearing shared preferences:

```java
private void clearPreferences() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();

    editor.clear();

    editor.commit();
}
```

Also, for the method on the previous slide we need to implement calling second activity. Let’s do this:

```java
private void openPreferencesEditor() {
    Intent intent = new Intent(this, PrefsEditorActivity.class);
    startActivity(intent);
}
```

BasicViewOptions

Let’s use them:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_clear) {

        // obsługa czyszczenia
        clearPreferences();
        return true;
    } else if (id == R.id.action_edit) {

        // obsługa edycji
        openPreferencesEditor();
        return true;
    }

    return super.onOptionsItemSelected(item);
}
```

BasicViewOptions

Ok, now again – we want our Activity to refresh UI and load shared preferences whenever it comes back to the front, so, let's implement reading shared preferences, and put this method in onResume method. But first – we need to define sharedPreferences name, and key for our preferences:

```
public static final String KEY_WIDTH = "view.width";
public static final String KEY_HEIGHT = "view.height";
public static final String KEY_FONT_SIZE = "view.font_size";
public static final String KEY_BTN_COLOR = "view.btn_color";

public static final String PREFS_NAME = "view.prefs";
```

Now getStylePreferences method:

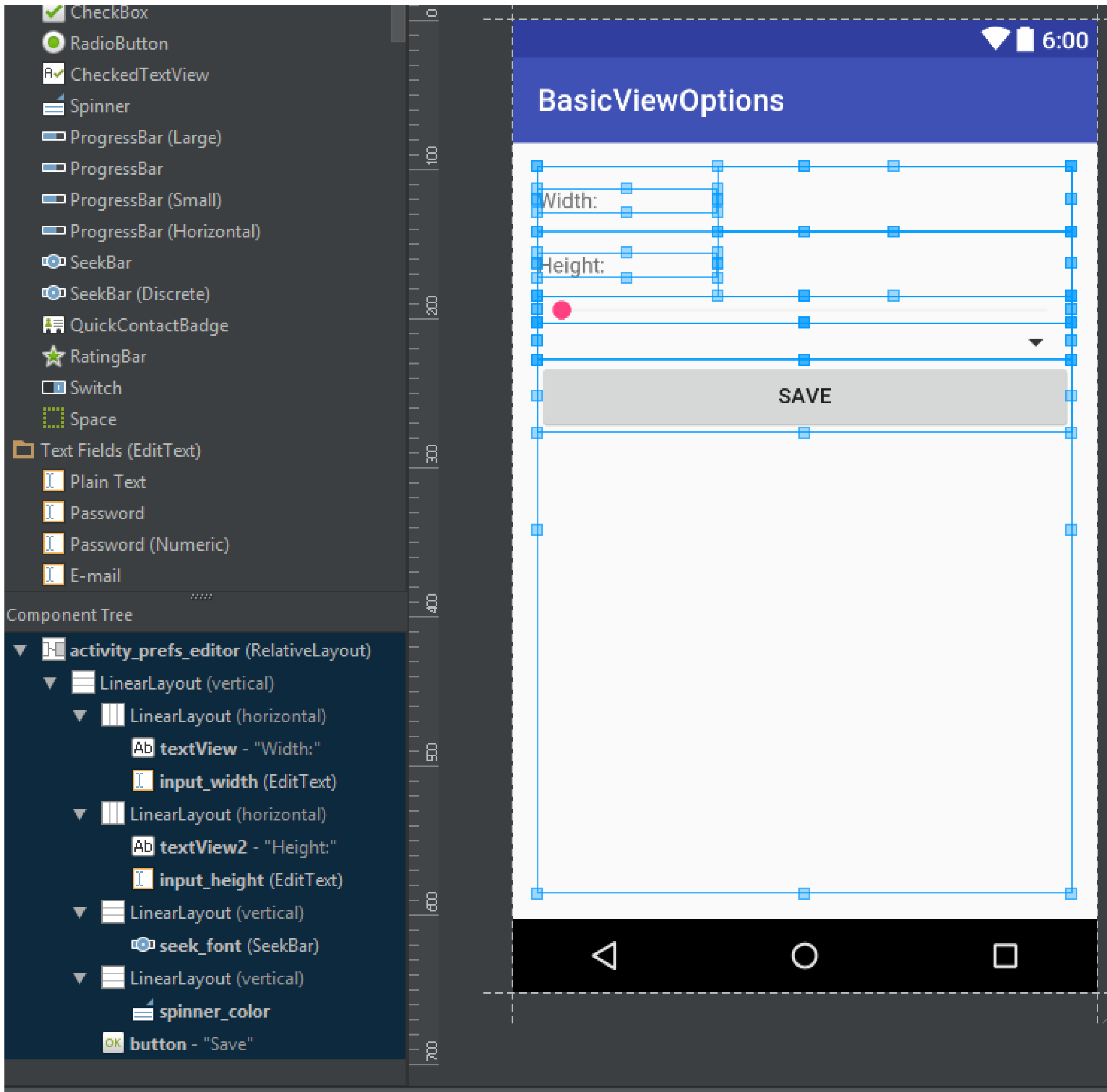
```
private void getStylePreferences() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    int width, height, fontSize, colorInt;
    String color;

    width = preferences.getInt(KEY_WIDTH, -1);
    height = preferences.getInt(KEY_HEIGHT, -1);
    fontSize = preferences.getInt(KEY_FONT_SIZE, -1);
    color = preferences.getString(KEY_BTN_COLOR, "");

    // do something with those values

}
```

Used:

```
@Override
protected void onResume() {
    getStylePreferences();
    super.onResume();
}
```

BasicViewOptions

Ok. Let's go to the PrefsEditorActivity:



CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar (Large)
ProgressBar
ProgressBar (Small)
ProgressBar (Horizontal)
SeekBar
SeekBar (Discrete)
QuickContactBadge
RatingBar
Switch
Space
Text Fields (EditText)
Plain Text
Password
Password (Numeric)
E-mail

Component Tree

```
activity_prefs_editor (RelativeLayout)
  LinearLayout (vertical)
    LinearLayout (horizontal)
      textView - "Width:"
      input_width (EditText)
    LinearLayout (horizontal)
      textView2 - "Height:"
      input_height (EditText)
    LinearLayout (vertical)
      seek_font (SeekBar)
    LinearLayout (vertical)
      spinner_color
    button - "Save"
```

BasicViewOptions

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_prefs_editor"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.amen.basicviewoptions.PrefsEditorActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true">

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:text="Width:"
                android:layout_width="200px"
                android:layout_height="wrap_content"
                android:id="@+id/textView"
                android:layout_weight="1"
                android:minWidth="150px"
                android:textSize="14sp" />

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="number"
                android:ems="10"
                android:id="@+id/input_width"
                android:layout_weight="1"
                android:gravity="right" />

        </LinearLayout>
```

BasicViewOptions

```
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <TextView
                android:text="Height:"
                android:layout_width="200px"
                android:layout_height="wrap_content"
                android:id="@+id/textView2"
                android:minWidth="150px"
                android:layout_weight="1" />
            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:inputType="number"
                android:ems="10"
                android:id="@+id/input_height"
                android:layout_weight="1"
                android:gravity="right" />
        </LinearLayout>
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <SeekBar
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/seek_font" />
        </LinearLayout>
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <Spinner
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/spinner_color" />
        </LinearLayout>
        <Button
            android:text="Save"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/button" />
    </LinearLayout>
</RelativeLayout>
```

BasicViewOptions

Bind variables:

```
private EditText inputWidth, inputHeight;
private Spinner colorsSpinner;
private SeekBar seekBarFont;
private Button saveButton;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_prefs_editor);


    inputHeight = (EditText) findViewById(R.id.input_height);
    inputWidth = (EditText) findViewById(R.id.input_width);
    colorsSpinner = (Spinner) findViewById(R.id.spinner_color);
    saveButton = (Button) findViewById(R.id.button);
    seekBarFont = (SeekBar) findViewById(R.id.seek_font);

    //...
}
```

BasicViewOptions

Now, go to the resources and define String array:

```xml
<resources>
    <string name="app_name">BasicViewOptions</string>
    <string name="action_settings">Settings</string>

    <string-array name="colors">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
        <item>Black</item>
        <item>White</item>
    </string-array>
</resources>
```

Bind those values to the spinner:

```java
ArrayAdapter<String> adapter = new ArrayAdapter<>(
        this,
        android.R.layout.simple_spinner_item,
        getResources().getStringArray(R.array.colors));
colorsSpinner.setAdapter(adapter);
```

BasicViewOptions

Create save preferences method:

```
private void savePreferences() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();

    String widthString = inputWidth.getText().toString();
    String heightString = inputHeight.getText().toString();
    int widthInteger = -1;
    int heightInteger = -1;

    try {
        if (!widthString.isEmpty())
            widthInteger = Integer.parseInt(widthString);
        if (!heightString.isEmpty())
            heightInteger = Integer.parseInt(heightString);
    } catch (NumberFormatException nfe) {
        Toast.makeText(
                getApplicationContext(),
                "Wrong value!",
                Toast.LENGTH_SHORT).show();
        return;
    }

    editor.putInt(KEY_WIDTH, widthInteger);
    editor.putInt(KEY_HEIGHT, heightInteger);
    editor.putInt(KEY_FONT_SIZE, seekBarFont.getProgress());
    editor.putString(KEY_BTN_COLOR, colorsSpinner.getSelectedItem().toString());

    editor.apply();
}
```

This time I added more code. I assigned our variables default value (-1) and if their fields are empty i leave those fields empty. In the next slide i will go back to the MainActivity and add additional check in getStylePreferences method to check if those values are -1, if so, i will not set button styles.

BasicViewOptions

Edit getStylePreferences from MainActivity. But first we need to define method which will set button style. I’ll call it setButtonStyle:

```java
private void setButtonStyle(Button btnToSet, int width, int height, int fontSize, int color) {
    btnToSet.setWidth(width);
    btnToSet.setHeight(height);
    btnToSet.setTextSize((float) fontSize);
    btnToSet.setBackgroundColor(color);
}
```

Now:

```java
private void getStylePreferences() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    int width, height, fontSize, colorInt;
    String color;

    width = preferences.getInt(KEY_WIDTH, -1);
    height = preferences.getInt(KEY_HEIGHT, -1);
    fontSize = preferences.getInt(KEY_FONT_SIZE, -1);
    color = preferences.getString(KEY_BTN_COLOR, "");

    colorInt = parseColor(color);
    if (width == -1 || height == -1) {
        return;
    }

    setButtonStyle(btn1, width, height, fontSize, colorInt);
    setButtonStyle(btn2, width, height, fontSize, colorInt);
    setButtonStyle(btn3, width, height, fontSize, colorInt);
    setButtonStyle(btn4, width, height, fontSize, colorInt);
    setButtonStyle(btn5, width, height, fontSize, colorInt);

}
```

BasicViewOptions

One method is still not yet defined. It’s parseColor. Because spinner contains values and holds only Strings, we need to parse them to Color values. We’ll do this with switch:

```
private int parseColor(String color) {
    switch (color) {
        case "Red":
            return Color.RED;
        case "Blue":
            return Color.BLUE;
        case "Black":
            return Color.BLACK;
        case "Green":
            return Color.GREEN;
        case "White":
            return Color.WHITE;
        default:
            return Color.BLACK;
    }
}
```

BasicViewOptions

Ok, now let’s set saveButton onClickListener:

```
saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        savePreferences();
        finish();
    }
});
```

And add additional line for onCreate:

```
loadPreferences();
```

What will it do?

BasicViewOptions

It will load values from shared preferences to the controls on PrefsEditActivity:

```
private void loadPreferences(){
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);

    int width = preferences.getInt(KEY_WIDTH, -1);
    int height = preferences.getInt(KEY_HEIGHT, -1);
    String color = preferences.getString(KEY_BTN_COLOR, "");
    int fontSize= preferences.getInt(KEY_FONT_SIZE, -1);

    inputHeight.setText(String.valueOf(width));
    inputWidth.setText(String.valueOf(height));
    seekBarFont.setProgress(fontSize);

    int zaznaczonyKolor = adapter.getPosition(color);
    colorsSpinner.setSelection(zaznaczonyKolor);
}
```

Last two lines are additional modifications on adapter. Previously defined adapter needs to be moved and defined as Activity field. Than using adapter we can get selected colors position, and having position allows us to setSelection to that id.

BasicViewOptions

```java
public class MainActivity extends AppCompatActivity {
    public static final String KEY_WIDTH = "view.width";
    public static final String KEY_HEIGHT = "view.height";
    public static final String KEY_FONT_SIZE = "view.font_size";
    public static final String KEY_BTN_COLOR = "view.btn_color";

    public static final String PREFS_NAME = "view.prefs";
    private Button btn1, btn2, btn3, btn4, btn5;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        btn1 = (Button) findViewById(R.id.button6);
        btn2 = (Button) findViewById(R.id.button2);
        btn3 = (Button) findViewById(R.id.button3);
        btn4 = (Button) findViewById(R.id.button4);
        btn5 = (Button) findViewById(R.id.button5);

    }
    private void setButtonStyle(Button btnToSet, int width, int height, int fontSize, int color) {
        btnToSet.setWidth(width);
        btnToSet.setHeight(height);
        btnToSet.setTextSize((float) fontSize);
        btnToSet.setBackgroundColor(color);
    }
     private int parseColor(String color) {
        switch (color) {
            case "Red":
                return Color.RED;
            case "Blue":
                return Color.BLUE;
            case "Black":
                return Color.BLACK;
            case "Green":
                return Color.GREEN;
            case "White":
                return Color.WHITE;
            default:
                return Color.BLACK;
        }
    }
    @Override
    protected void onResume() {
        getStylePreferences();
        super.onResume();
    }
```

BasicViewOptions

```
private void getStylePreferences() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    int width, height, fontSize, colorInt;
    String color;

    width = preferences.getInt(KEY_WIDTH, -1);
    height = preferences.getInt(KEY_HEIGHT, -1);
    fontSize = preferences.getInt(KEY_FONT_SIZE, -1);
    color = preferences.getString(KEY_BTN_COLOR, "");

    colorInt = parseColor(color);
    if (width == -1 || height == -1) {
        return;
    }

    setButtonStyle(btn1, width, height, fontSize, colorInt);
    setButtonStyle(btn2, width, height, fontSize, colorInt);
    setButtonStyle(btn3, width, height, fontSize, colorInt);
    setButtonStyle(btn4, width, height, fontSize, colorInt);
    setButtonStyle(btn5, width, height, fontSize, colorInt);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

BasicViewOptions

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_clear) {

            // obsługa czyszczenia
            clearPreferences();
            getStylePreferences();
            return true;
        } else if (id == R.id.action_edit) {

            // obsługa edycji
            openPreferencesEditor();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    private void openPreferencesEditor() {
        Intent intent = new Intent(this, PrefsEditorActivity.class);
        startActivity(intent);
    }

    private void clearPreferences() {
        SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();

        editor.clear();

        editor.commit();
    }
}
```

```java
public class PrefsEditorActivity extends AppCompatActivity {
    ArrayAdapter<String> adapter;

    private EditText inputWidth, inputHeight;

    private Spinner colorsSpinner;
    private SeekBar seekBarFont;

    private Button saveButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prefs_editor);


        inputHeight = (EditText) findViewById(R.id.input_height);
        inputWidth = (EditText) findViewById(R.id.input_width);

        colorsSpinner = (Spinner) findViewById(R.id.spinner_color);

        saveButton = (Button) findViewById(R.id.button);

        seekBarFont = (SeekBar) findViewById(R.id.seek_font);

        adapter = new ArrayAdapter<>(
                this,
                android.R.layout.simple_spinner_item,
                getResources().getStringArray(R.array.colors));
        colorsSpinner.setAdapter(adapter);

        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                savePreferences();
                finish();
            }
        });

        loadPreferences();
    }
    private void loadPreferences(){
        SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);

        int width = preferences.getInt(KEY_WIDTH, -1);
        int height = preferences.getInt(KEY_HEIGHT, -1);
        String color = preferences.getString(KEY_BTN_COLOR, "");
        int fontSize= preferences.getInt(KEY_FONT_SIZE, -1);

        inputHeight.setText(String.valueOf(width));
        inputWidth.setText(String.valueOf(height));
        seekBarFont.setProgress(fontSize);

        int zaznaczonyKolor = adapter.getPosition(color);

        colorsSpinner.setSelection(zaznaczonyKolor);
    }

    private void savePreferences() {
        SharedPreferences preferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();

        String widthString = inputWidth.getText().toString();
        String heightString = inputHeight.getText().toString();
        int widthInteger = -1;
        int heightInteger = -1;

        try {
            if (!widthString.isEmpty())
                widthInteger = Integer.parseInt(widthString);
            if (!heightString.isEmpty())
                heightInteger = Integer.parseInt(heightString);
        } catch (NumberFormatException nfe) {
            Toast.makeText(
                    getApplicationContext(),
                    "Wrong value!",
                    Toast.LENGTH_SHORT).show();
            return;
        }

        editor.putInt(KEY_WIDTH, widthInteger);
        editor.putInt(KEY_HEIGHT, heightInteger);
        editor.putInt(KEY_FONT_SIZE, seekBarFont.getProgress());
        editor.putString(KEY_BTN_COLOR, colorsSpinner.getSelectedItem().toString());

        editor.apply();
    }
}
```

The end