Experiment No. 8

Aim: To create Drawing application in Android.

Requirements: Compatible version of Android Studio, Gradle and Java.

Theory:

A ShapeDrawable object can be a good option when you want to dynamically draw a two-dimensional graphic. You can programmatically draw primitive shapes on a ShapeDrawable object and apply the styles that your app needs.

ShapeDrawable is a subclass of Drawable. For this reason, you can use a ShapeDrawable wherever a Drawable is expected. For example, you can use a ShapeDrawable object to set the background of a view by passing it to the setBackgroundDrawable() method of the view. You can also draw your shape as its own custom view and add it to a layout in your app.

Because ShapeDrawable has its own draw() method, you can create a subclass of View that draws the ShapeDrawable object during the onDraw() event.

If you want to use the custom view in the XML layout instead, then the CustomDrawableView class must override the View(Context, AttributeSet) constructor, which is called when the class is inflated from XML. The following example shows how to declare the CustomDrawableView in the XML layout:

```
<com.example.shapedrawable.CustomDrawableView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
```

The ShapeDrawable class, like many other drawable types in the android.graphics.drawable package, allows you to define various properties of the object by using public methods. Some example properties you might want to adjust include alpha transparency, color filter, dither, opacity, and color.

Code:

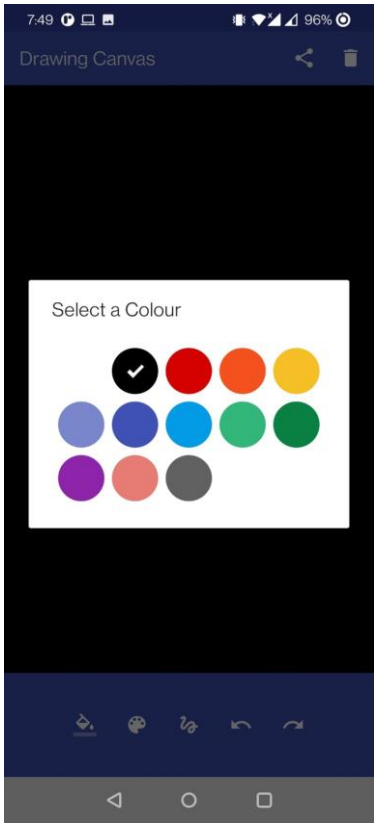
```
private void initDrawingView()
	{
		mCurrentBackgroundColor = ContextCompat.getColor(this,
android.R.color.white);
		mCurrentColor = ContextCompat.getColor(this, android.R.color.black);
		mCurrentStroke = 10;

		mDrawingView.setBackgroundColor(mCurrentBackgroundColor);
		mDrawingView.setPaintColor(mCurrentColor);
		mDrawingView.setPaintStrokeWidth(mCurrentStroke);
	}

	private void startFillBackgroundDialog()
	{
		int[] colors = getResources().getIntArray(R.array.palette);

		ColorPickerDialog dialog =
ColorPickerDialog.newInstance(R.string.color_picker_default_title,
				colors,
				mCurrentBackgroundColor,
				5,
				ColorPickerDialog.SIZE_SMALL);

		dialog.setOnColorSelectedListener(new
ColorPickerSwatch.OnColorSelectedListener()
		{

			@Override
			public void onColorSelected(int color)
			{
				mCurrentBackgroundColor = color;

	mDrawingView.setBackgroundColor(mCurrentBackgroundColor);
			}

		});

		dialog.show(getFragmentManager(), "ColorPickerDialog");
	}
```

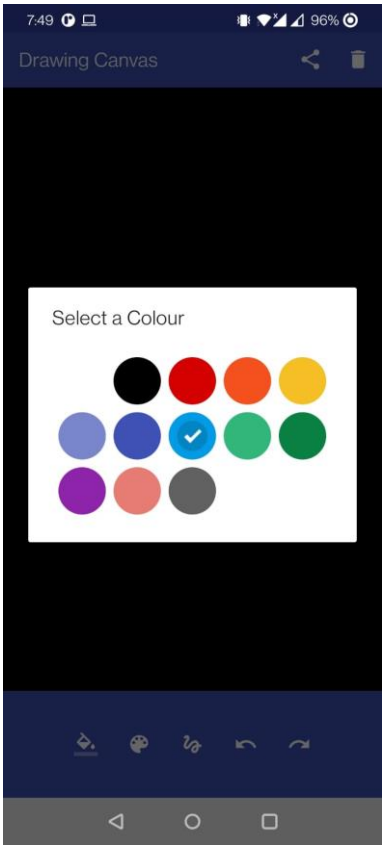
```
	private void startColorPickerDialog()
	{
		int[] colors = getResources().getIntArray(R.array.palette);

		ColorPickerDialog dialog =
ColorPickerDialog.newInstance(R.string.color_picker_default_title,
				colors,
				mCurrentColor,
				5,
				ColorPickerDialog.SIZE_SMALL);

		dialog.setOnColorSelectedListener(new
ColorPickerSwatch.OnColorSelectedListener()
		{

			@Override
			public void onColorSelected(int color)
			{
				mCurrentColor = color;
				mDrawingView.setPaintColor(mCurrentColor);
			}

		});

		dialog.show(getFragmentManager(), "ColorPickerDialog");
	}

	private void startStrokeSelectorDialog()
	{
		StrokeSelectorDialog dialog =
StrokeSelectorDialog.newInstance(mCurrentStroke, MAX_STROKE_WIDTH);

		dialog.setOnStrokeSelectedListener(new
StrokeSelectorDialog.OnStrokeSelectedListener()
		{
			@Override
			public void onStrokeSelected(int stroke)
			{
				mCurrentStroke = stroke;
				mDrawingView.setPaintStrokeWidth(mCurrentStroke);
			}
		});

		dialog.show(getSupportFragmentManager(), "StrokeSelectorDialog");
	}

	private void startShareDialog(Uri uri)
	{
		Intent intent = new Intent();
```

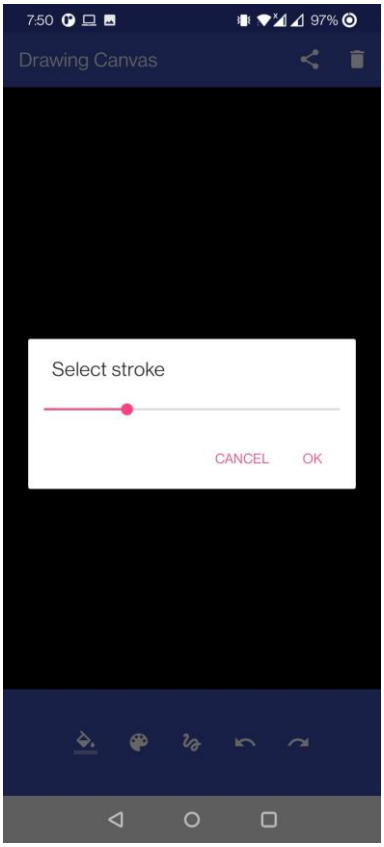
```
		intent.setAction(Intent.ACTION_SEND);
		intent.setType("image/*");

		intent.putExtra(android.content.Intent.EXTRA_SUBJECT, "");
		intent.putExtra(android.content.Intent.EXTRA_TEXT, "");
		intent.putExtra(Intent.EXTRA_STREAM, uri);
		intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
		startActivity(Intent.createChooser(intent, "Share Image"));
	}

	private void requestPermissionsAndSaveBitmap()
	{
		if (PermissionManager.checkWriteStoragePermissions(this))
		{
			Uri uri = FileManager.saveBitmap(this, mDrawingView.getBitmap());
			startShareDialog(uri);
		}
	}

	@Override
	public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults)
	{
		super.onRequestPermissionsResult(requestCode, permissions, grantResults);
		switch (requestCode)
		{
			case PermissionManager.REQUEST_WRITE_STORAGE:
			{
				if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
				{
					Uri uri = FileManager.saveBitmap(this,
mDrawingView.getBitmap());
					startShareDialog(uri);
				} else
				{
					Toast.makeText(this, "The app was not allowed to write
to your storage. Hence, it cannot function properly. Please consider granting it this
permission", Toast.LENGTH_LONG).show();
				}
			}
		}
	}
```

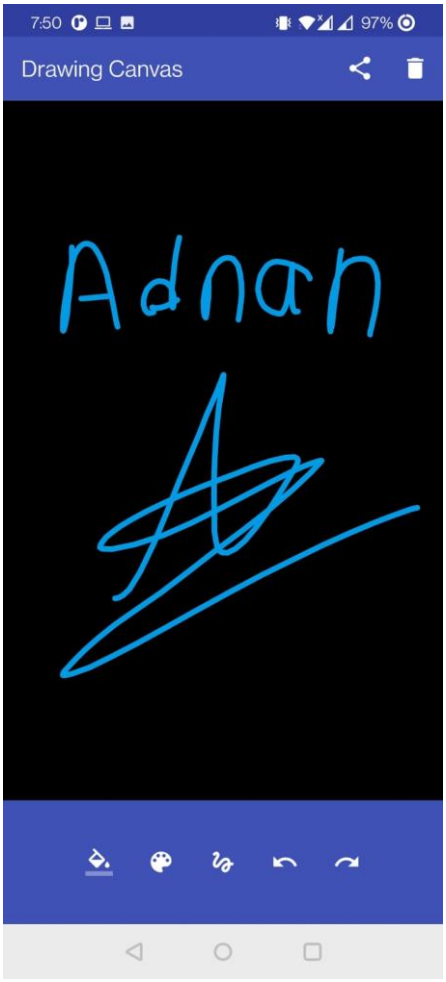
Output:


7:49
96%
Drawing Canvas


7:49
96%
Drawing Canvas
Select a Colour


7:49
96%
Drawing Canvas
Select a Colour


7:50
97%
Drawing Canvas
Select stroke
CANCEL
OK



Conclusion: We have successfully implemented Drawing App in Android.