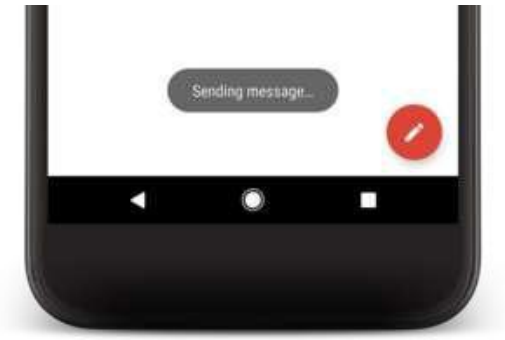
Toasts overview

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

For example, clicking **Send** on an email triggers a "Sending message..." toast, as shown in the following screen capture:



Creating the Toast

First, instantiate a Toast object with one of the makeText() methods. This method takes three parameters: the application Context, the text message, and the duration for the toast. It returns a properly initialized Toast object. You can display the toast notification with show(), as shown in the following example:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text,
duration); toast.show();
```

This example demonstrates everything you need for most toast notifications. You should rarely need anything else. You may, however, want to position the toast differently or even use your own layout instead of a simple text message. The following sections describe how you can do these things.

You can also chain your methods and avoid holding on to the Toast object, like this:

```
Toast.makeText(context, text, duration).show();
```

Positioning your Toast

A standard toast notification appears near the bottom of the screen, centered horizontally. You can change this position with the setGravity(int, int, int) method. This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

For example, if you decide that the toast should appear in the top-left corner, you can set the gravity like this:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Creating Custom Toast View

If a simple text message isn't enough, you can create a customized layout for your toast notification. To create a custom layout, define a View layout, in XML or in your application code, and pass the root View object to the setView(View) method.

- The following snippet contains a customized layout for a toast notification (saved as layout/custom_toast.xml):

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/custom_toast_container"
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:padding="8dp"
```

```
                android:background="#DAAA"
                >
    <ImageView android:src="@drawable/droid"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginRight="8dp"
                />
    <TextView android:id="@+id/text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFF"
                />
</LinearLayout>
```

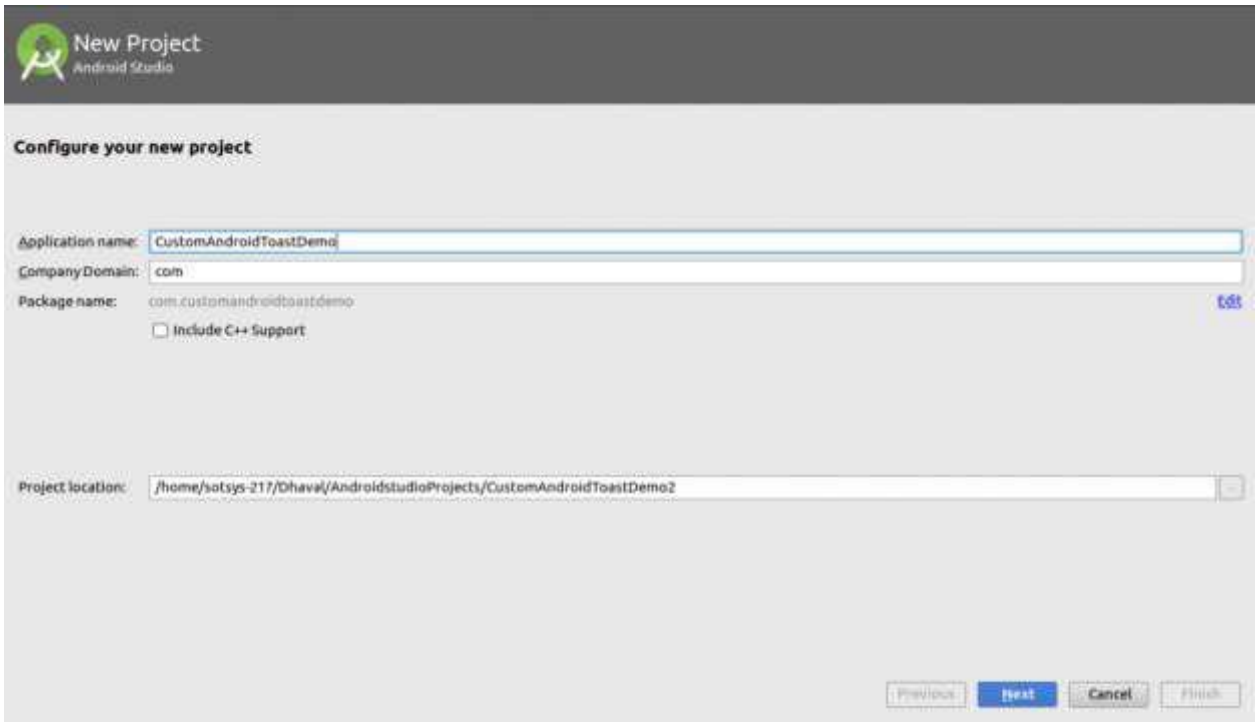
Notice that the ID of the LinearLayout element is "custom_toast_container". You must use this ID and the ID of the XML layout file "custom_toast" to inflate the layout, as shown here:

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast,
                (ViewGroup) findViewById(R.id.custom_toast_container));

TextView text = (TextView) layout.findViewById(R.id.text);
text.setText("This is a custom toast");

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

First, retrieve the LayoutInflater with getLayoutInflater() (or getSystemService()), and then inflate the layout from XML using inflate(int, ViewGroup). The first parameter is the layout resource ID and the second is the root View. You can use this inflated layout to find more View objects in the layout, so now capture and define the content for the ImageView and TextView elements. Finally, create a new Toast with Toast(Context)and set some properties of the toast, such as the gravity and duration. Then call setView(View) and pass it the inflated layout. You can now display the toast with your custom layout by calling show().

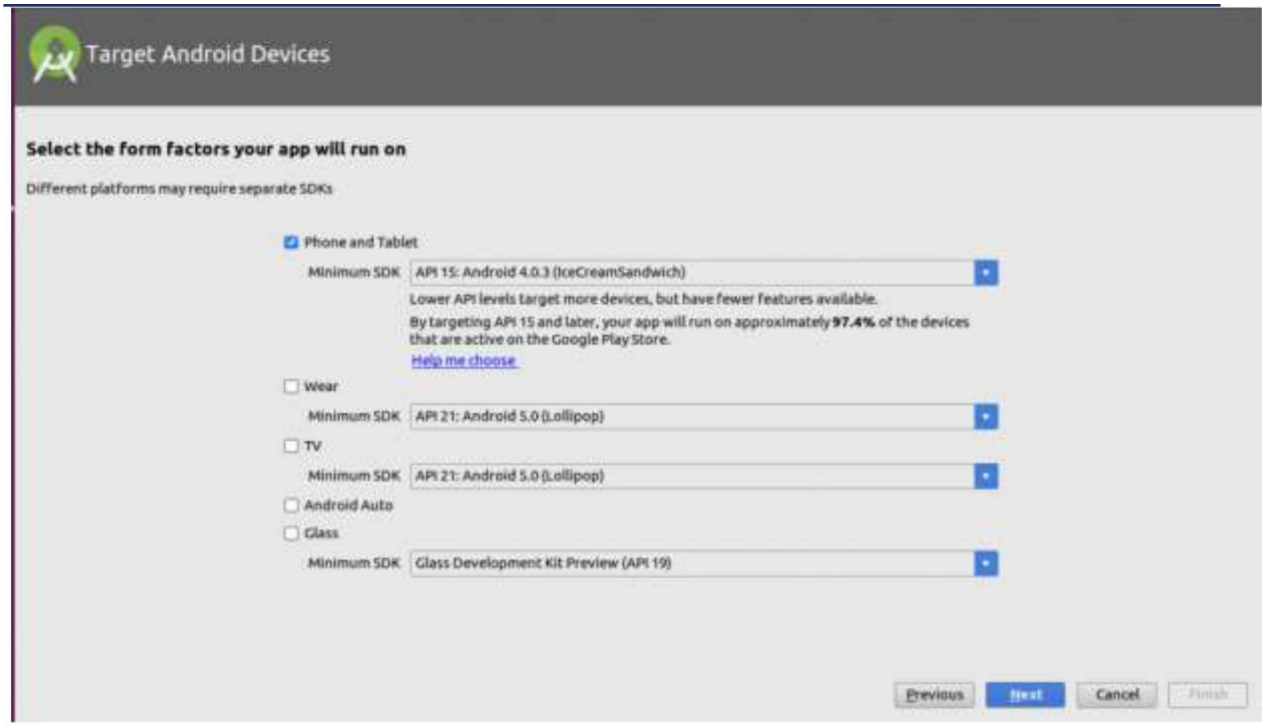
Exercise

Open your Android Studio and create a new project “CustomAndroidToastDemo”.

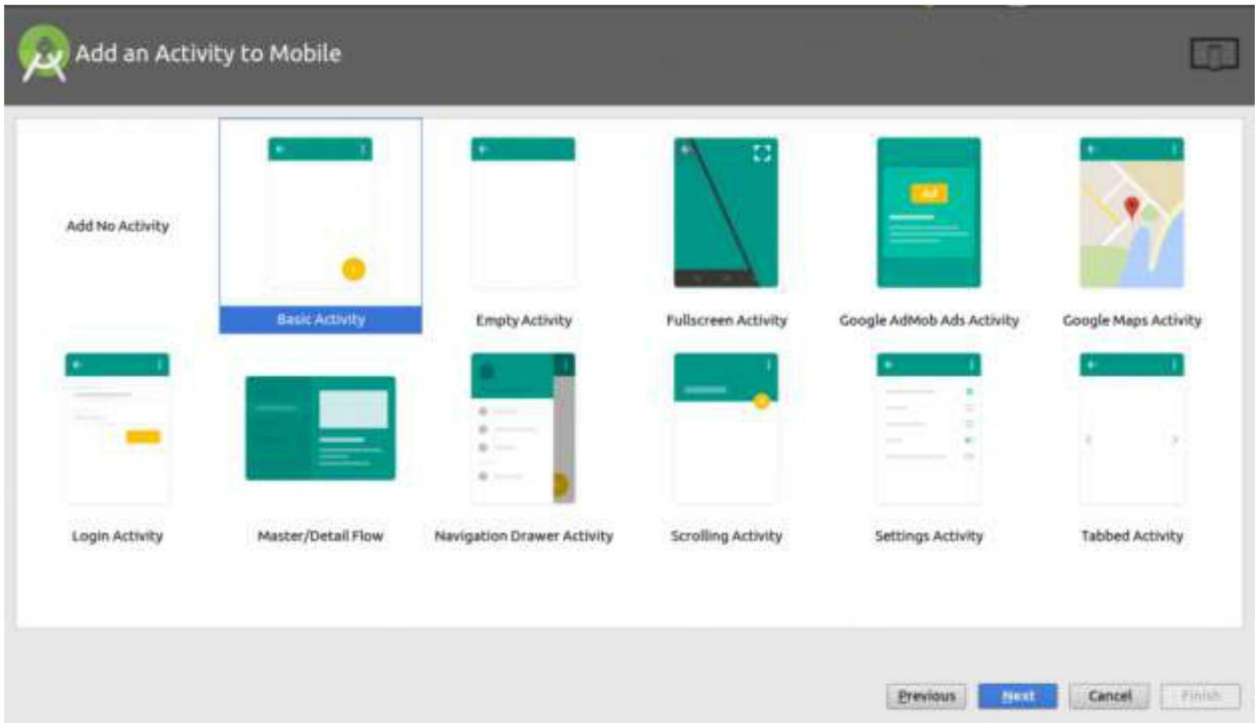


Select your target Android device and click on next.

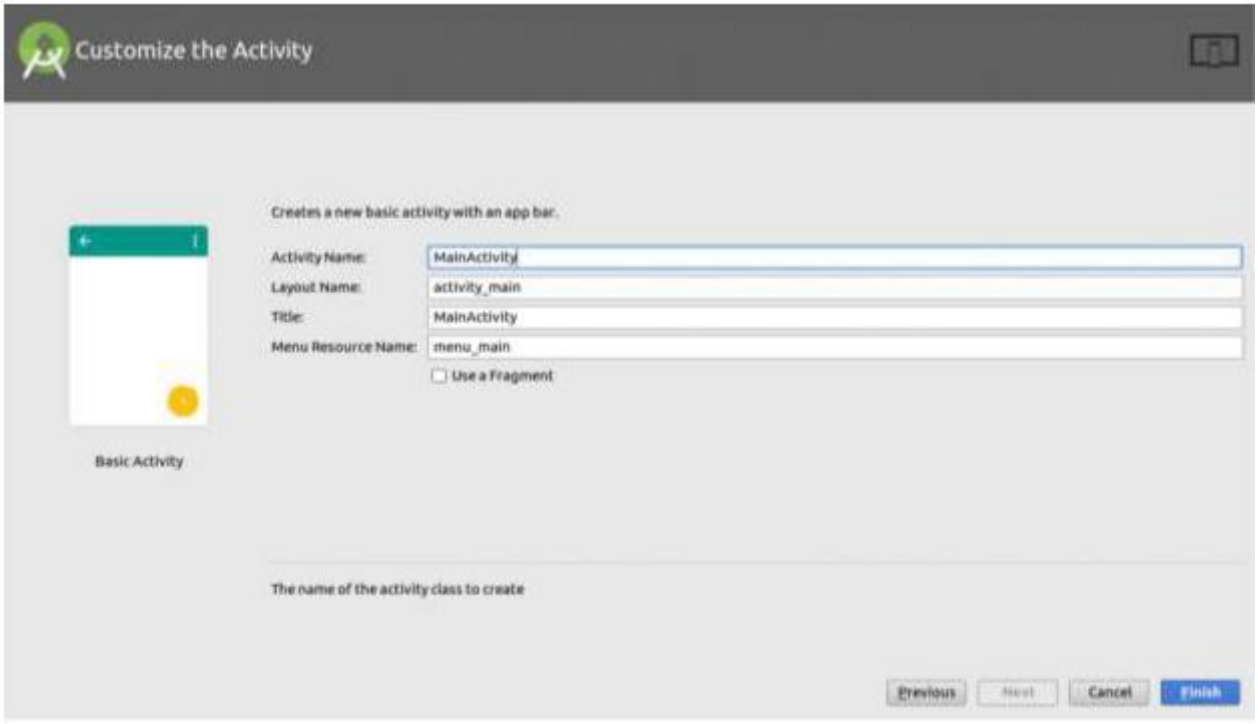
Lab sheet 04 **Semester 2, 2019**



In the next tab, select the Base activity.



Lastly, customize the activity.



Now, create an XML layout file to display custom toast with message and icon.

Content_Custom_Toast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/llCustom"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
```

```
android:background="@color/colorAccent"
android:drawableLeft="@android:drawable/ic_dialog_alert"
android:gravity="center"
android:padding="10dp"
android:drawablePadding="5dp"
android:text="This is custom toast message"
android:textColor="@android:color/white"
android:textSize="16sp" />

</LinearLayout>
```

Modify the XML activity content layout file. Here, we'll add two buttons. One for default layout and second for custom layout.

Content_Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
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.customandroidtoastdemo.MainActivity"
tools:showIn="@layout/activity_main">
<Button
```

```
android:id="@+id/btnDefaultToast"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="@dimen/activity_vertical_margin"
android:text="Show Default Toast"
android:textAllCaps="false" />

<Button
android:id="@+id/btnCustomToast"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="@dimen/activity_vertical_margin"
android:text="Show Custom Toast"
android:textAllCaps="false" />
</LinearLayout>
```

Now, Initialize the button in activity and button click handling.

Declare button variables on top of Activity class

```
private Button btnDefaultToast, btnCustomToast;
```

Initialize both buttons.

```
btnDefaultToast = (Button) findViewById(R.id.btnDefaultToast);
btnCustomToast = (Button) findViewById(R.id.btnCustomToast);
```

Handle button click.

```
@Override
public void onClick(View view) {
if (view.equals(btnDefaultToast)) {
Toast.makeText(MainActivity.this, "This is default toast message", Toast.LENGTH_SH
ORT).show();
```

```
} else if (view.equals(btnCustomToast)) {

LayoutInflater inflater = getLayoutInflater();

View toastLayout = inflater.inflate(R.layout.content_custom_toast, (ViewGroup) fin
dViewById(R.id.llCustom));

Toast toast = new Toast(getApplicationContext());

toast.setDuration(Toast.LENGTH_LONG);

toast.setView(toastLayout); toast.show();


}
}
```

And Done!

Reference

https://developer.android.com/guide/topics/ui/notifiers/toasts#java

https://www.lynda.com/Android-tutorials/Toast-overview/513591/554009-4.html

https://developer.android.com/guide/topics/resources/providing-resources

https://developer.android.com/guide/topics/ui/declaring-layout