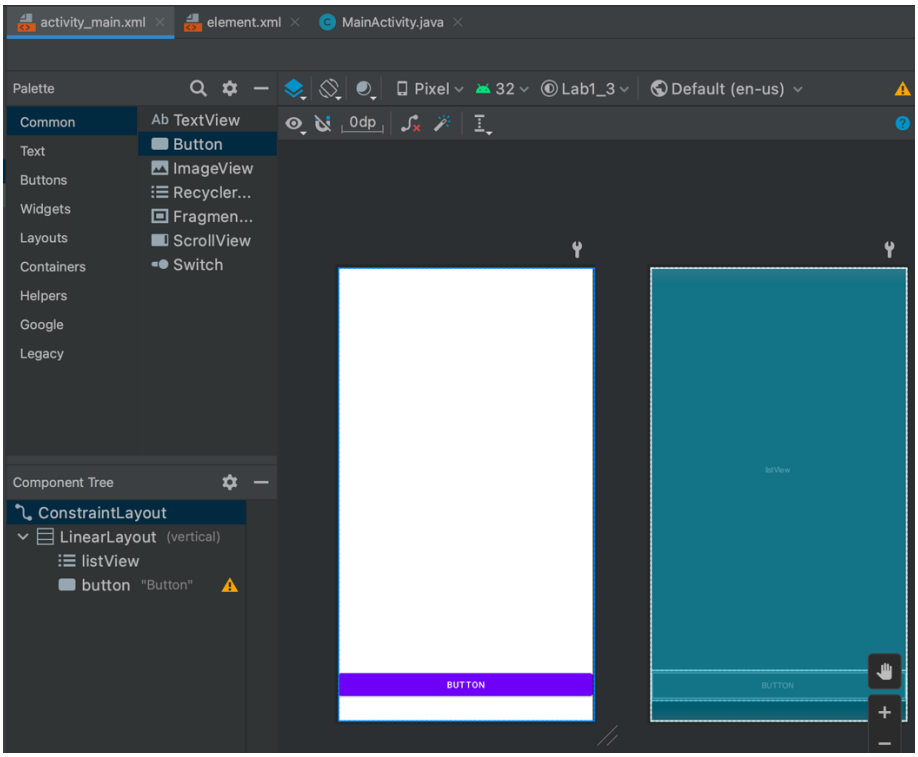
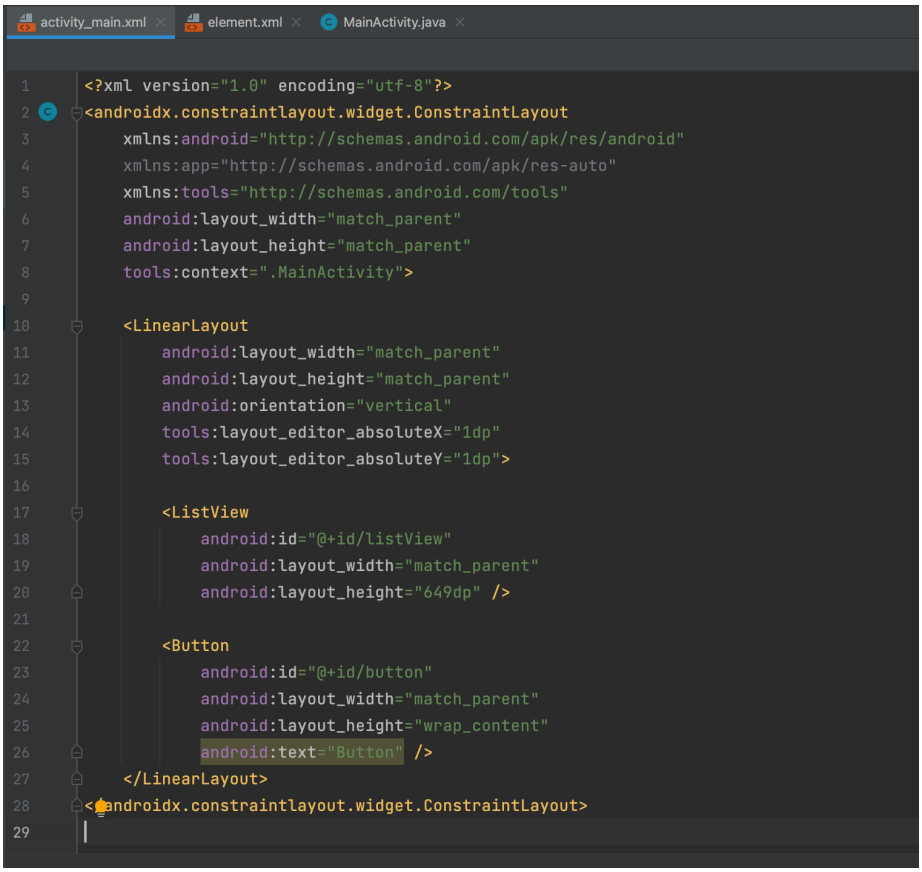
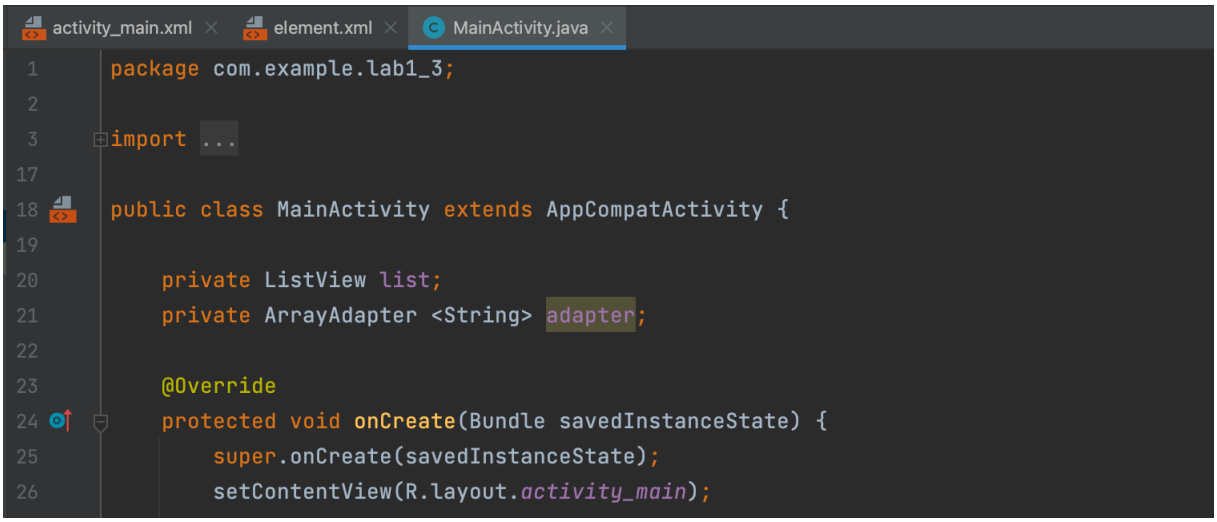
ListView, komunikaty Toast, AlertDialog i inne.

Zaczniemy od ListView, czyli zwyczajnej, przewijalnej listy elementów, w naszym wypadku będzie to textView. Zacznijmy od stworzenia interfejsu, LinearLayout, ListView oraz Button:





Żeby użyć ListView trzeba przebić się przez kilka kontenerów i adapterów, tworzymy obiekt klasy ListView (20) oraz ArrayAdapter (21), ponieważ będę chciał zwykłą tablicę Stringów umieścić w widoku ListView, czego nie mogę zrobić bezpośrednio, ponieważ Listview przyjmuje jako argument do wyświetlenia właśnie ArrayAdapter.



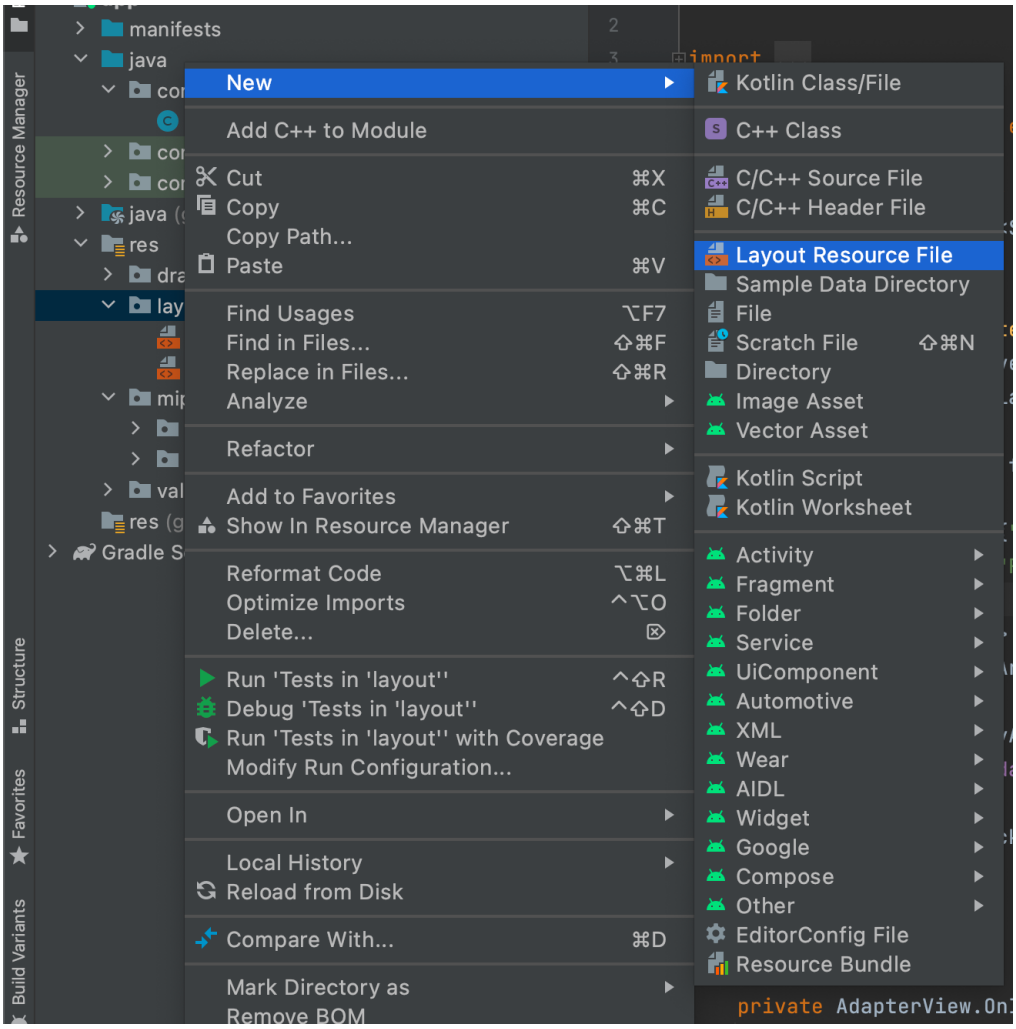
Następnie łącze nasz obiekt z elementem xml (28) oraz tworze nową tablicę, którą będę chciał wyświetlać (30).
(Linijki 33 i 34 można pominąć. Chcę tu tylko pokazać, że możemy użyć zwykłej tablicy jak również ArrayListy. Jeśli pominiemy ten krok, to w linii 36 jako ostatni argument podajemy tablicę „marki" zamiast listy „samochody")

Tworzymy nowy adapter, pierwszy argument to kontekst (this), layout elementu do wyświetlenia oraz tablicę (listę) do wyświetlenia.

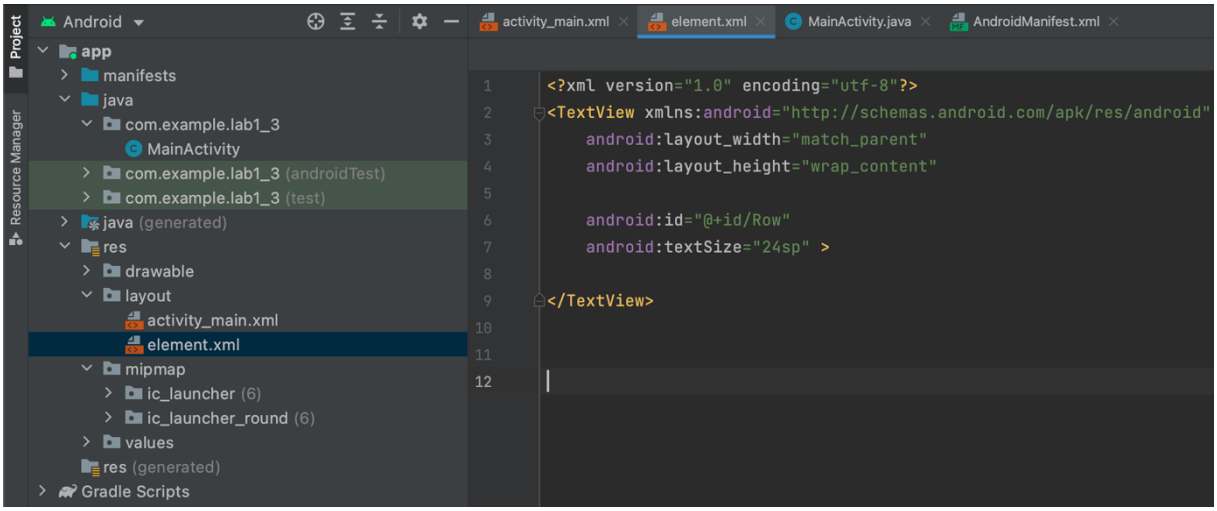
Context to jeden z najważniejszych obiektów w Androidzie - oznacza on kontekst obecnego stanu aplikacji i odpowiada za:

Zdobywanie informacji dotyczących aktywności oraz aplikacji
Zdobywanie dostępu do bazy danych, zasobów dotyczących jakiejś aplikacji (potrzebny tekst, drawable itd.) oraz do klas, systemów plików i współdzielonych preferencji
Up-calls dla operacji na poziomie aplikacji, takich jak uruchamianie aktywności, broadcasting i otrzymywanie intencji (intents).

Na koniec dodajemy do naszego widoku ListView stworzony wcześniej adapter (37). Oczywiście mamy błąd spowodowany brakiem czegoś takiego jak „element”. Element to nic innego niż zwykły plik .xml, który odpowiada za wygląd pojedynczego elementu wyświetlanej listy. Klikamy PPM na folder layouts -> New -> Layout Resource File.



W moim przypadku wyrzuciłem wszystko z nowo utworzonego pliku i wpisałem tam tylko TextView:



Po tej operacji można uruchomić aplikację, lista powinna się wyświetlać i być sprawna.

Jako kolejny krok spróbuję coś zrobić z tą listą. Zacznijmy od rzeczy prostej, czyli „wyciągnięcia” tego co zostało kliknięte na liście. W tym celu dodajemy nasłuchiwanie elementu (39):

Argument klikniecie jest po prostu nowy obiekt napisany jak niżej.

Obiekt klikniecie to Obiekt realizujący OnItemClickListenera (https://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener) do zmiennej typu String pobieram element spod indeksu, który jest zwracany po odczytaniu kliknięcia na element listy (jest nim zmienna i) (47) następnie wyświetlam komunikat Toast z kliknięty elementem listy. (https://developer.android.com/guide/topics/ui/notifiers/toasts)

To dobry moment, żeby uruchomić aplikacje i ją przetestować.

Dodatkowo pokazuję jak skopiować dany element do schowka systemowego (można wkleić go w dowolne miejsce do pisania, np. w wiadomość sms albo wyszukiwarkę) używam ClipboardMenager.
https://developer.android.com/reference/android/content/ClipboardManager

```
44      private AdapterView.OnItemClickListener klikniecie = new AdapterView.OnItemClickListener() {
45          @Override
46          public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
47              String itemValue = (String) list.getItemAtPosition(i);
48
49              Toast.makeText(getApplicationContext(),  text: "Skopiowano: " + itemValue, Toast.LENGTH_SHORT).show();
50
51              ClipboardManager kopiuj = (ClipboardManager) getSystemService(CLIPBOARD_SERVICE);
52              ClipData clip = ClipData.newPlainText( label: "Skopiowana wartość", itemValue);
53              kopiuj.setPrimaryClip(clip);
54          }
55      };
56  }
```

Zróbmy jeszcze użytek z przycisku, o którym pewnie każdy zdążył już zapomnieć (w interfejsie jest do niczego nie podpięty przycisk.) użyhemy go jako przycisk wychodzenia z aplikacji, wychodzenia takiego, że po jego naciśnięciu aplikacja jest usuwana z listy ostatnio otwieranych tasków.

```
57      public void close (View view){
58
59      }
```

Przygotujmy sobie zwykłą metodę, do której podepniemy sobie przycisk za pomocą android:onClick. Następnie wypełniam metodę: (pod screenem opis)

```
57      };
58
59      public void close (View view){
60          AlertDialog.Builder komunikat= new AlertDialog.Builder( context: this);
61          komunikat.setMessage("Jesteś pewien, że chcesz wyjść?");
62
63          komunikat.setPositiveButton( text: "Tak", new DialogInterface.OnClickListener() {
64              @Override
65              public void onClick(DialogInterface dialogInterface, int i) {
66                  //w tym miejscu umieszczamy kod, który ma się wykonać
67                  //po naciśnięciu przycisku "tak"
68                  finish();
69              }
70          });
71
72          komunikat.setNegativeButton( text: "Nie", new DialogInterface.OnClickListener() {
73              @Override
74              public void onClick(DialogInterface dialogInterface, int i) {
75                  Toast.makeText(getApplicationContext(),  text: "Super, że zostałeś!", Toast.LENGTH_SHORT).show();
76              }
77          });
78
79          AlertDialog alertDialog= komunikat.create();
80          alertDialog.show();
81      }
82  }
83
```

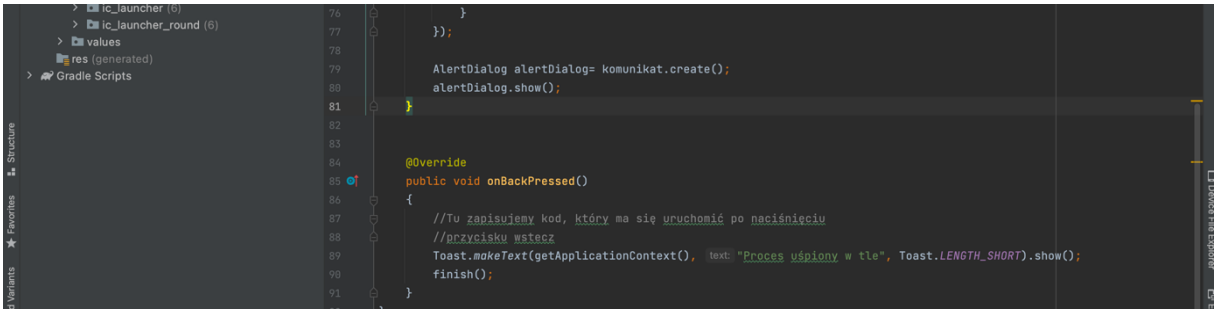
Korzystam z okien dialogowych wbudowanych w system Android, w tym przypadku z AlerDialog (https://developer.android.com/guide/topics/ui/dialogs). W oknie dialogowym możemy mieć przycisk potwierdzający (komunikat.setPositiveButton, linia 63) przycisk odrzucająca (komunikat.setNegativeButton, linia 72). Opcjonalnie istnieje jeszcze cośtakiego

jak przycisk neutralny, np. „Anuluj”. Finish() tylko wychodzi z aplikacji, nie wyrzucając jej z pamięci. Można powiedzieć, że zostaje ona uśpiona. Uruchom aplikację w takim stanie, naciśnij przycisk wychodzenia, następnie zwróć uwagę, że nasza aplikacja nadal jest na liście aplikacji w tle. Zamień teraz finish() na:

```
63      komunikat.setPositiveButton( text: "Tak", new DialogInterface.OnClickListener() {
64          @Override
65          public void onClick(DialogInterface dialogInterface, int i) {
66              //w tym miejscu umieszczamy kod, który ma się wykonać
67              //po naciśnięciu przycisku "tak"
68              finishAndRemoveTask();
69          }
70      });
```

Przetestuj w ten sam sposób, nasza aplikacja powinna być wyrzucona z pamięci.

Żeby dopiąć całość, trzeba opanować systemowy przycisk wstecz, skoro mamy autorski przycisk wychodzenia z aplikacji. Wystarczy przesłonić odpowiednią metodę:



Zadania:

1. Stwórz aplikację, która po naciśnięciu przycisku wyświetla listę 10 losowych liczb z zakresu od 0 do 1000000. Jeśli dojdziesz do tego momentu, zapytaj prowadzącego jak odświeżyć listę. Po kliknięciu na listę wyświetli się komunikat w postaci okna dialogowego o treści: „Czy skopiować liczbę do schowka?” z trzema opacjami:
   - kopiuj – w tej opcji kopiujemy do schowka listę
   - losuj nową – losuje nowa liczbę i zastępuje poprzednią na tej samej pozycji
   - nie – po prostu robi nic ☺
   Dodaj przycisk „losuj kolejne 10”, który DODAJE do listy kolejne 10 losowych liczb.

2. Stwórz aplikację, która wyświetla listę miast. Po naciśnięciu na element listy otwiera się nowa aktywność z informacją o danym mieście. W aplikacji wykorzystaj tylko dwie aktywności oraz plik strings.xml.