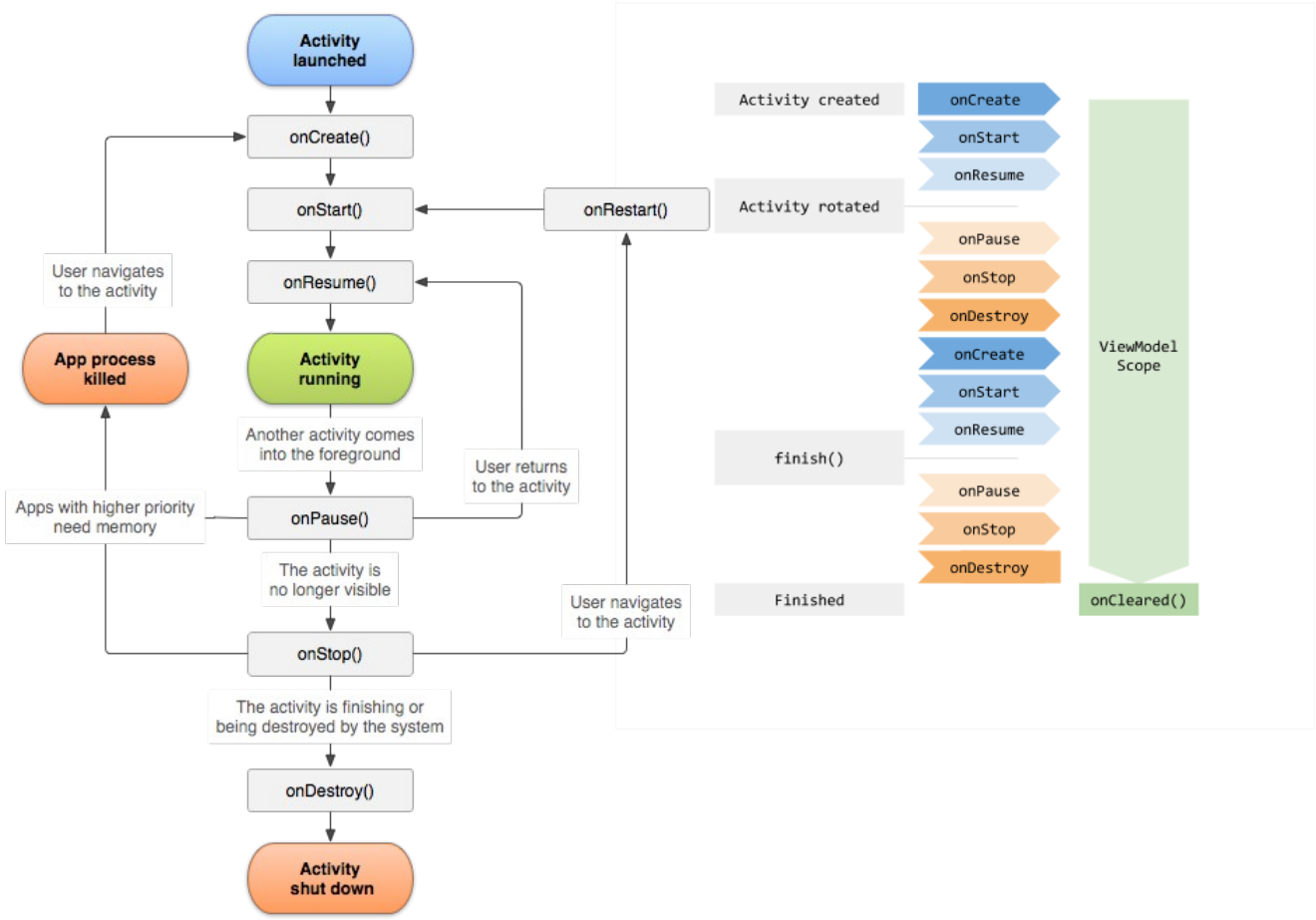
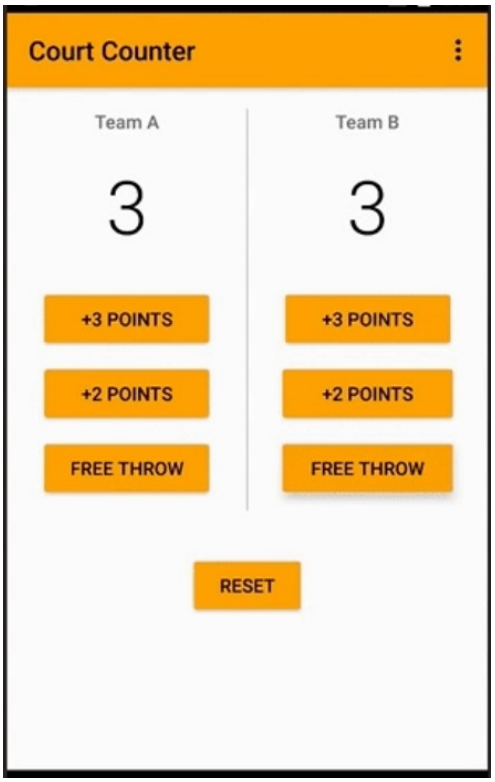
ViewModel.

Cykl życia aplikacji oraz zasięg ViewModel. Jak widać ViewModel jest niszczony dopiero wtedy, kiedy cała aplikacja jest niszczona!



Wykonamy następującą aplikacje: (wygląd po prawej)



Aplikacja w standardowym wykonaniu straci dane jeśli nastąpi zmiana konfiguracji aplikacji czyli np.: obrócimy ekran! Co spowoduje utworzenie na nowo całego wyglądu aplikacji.

Unikniemy tego jeśli wprowadzimy architekturę tworzenia aplikacji MVVM czyli **model → view → viewmodel**

Tym samym oddzielamy WIDOK od DANYCH.

Prosty ViewModel:

```
public class ScoreViewModel extends ViewModel {
  // Tracks the score for Team A
  public int scoreTeamA = 0;
  // Tracks the score for Team B
  public int scoreTeamB = 0;
}
```

PAMIĘTAJ: ViewModel nie może mieć w sobie referencji (połączeń) do aktywności, fragmentów lub kontekstu aplikacji.

W głównej klasie programu w metodzie onCreate() tworzymy nasz ViewModel jak pokazano poniżej. mViewModel – zmienna stworzona globalnie

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  myViewModel = new ViewModelProvider(this).get(MyViewModel.class);
  // Other setup code below...
}
```

Użycie ViewModel w programie:

poleTekstowe.setText(””+mViewModel. ScoreTeamA); //w onCreate() lub w onResume()

Zwiększenie wartości:

mViewModel. ScoreTeamA ++; //lub mViewModel. ScoreTeamA += 1;

Stwórz program przedstawiony na rysunku na stronie 1.

Na celującą:

Zrób listę rozwijalną do której będziesz dodawać:

TeamA +3 //jeśli nacisnąłeś przycisk dla TeamA

TeamB +1 // jeśli free throw został naciśnięty dla TeamB

Oczywiście po obróceniu ekranu dalej dane mają być widoczne.