|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich  Wydział Telekomunikacji, Informatyki i Elektrotechniki  al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz | | | |  | |
| Przedmiot | **Programowanie urządzeń mobilnych** | | | Kierunek/Tryb |  |
| Nr laboratorium | 5 | Data wykonania |  | Grupa |  |
| Ocena |  | Data oddania |  | Imię Nazwisko |  |
| Nazwa ćwiczenia | Interfejs gry, tworzenie i obsługa interfejsu, intencje | | |

**Cel ćwiczenia laboratoryjnego**

Celem ćwiczenia jest zapoznanie studentów z tworzeniem prostej gry mobilnej na platformie Android przy użyciu interfejsu użytkownika. Studenci mają nauczyć się:

1. Projektować interfejs użytkownika, który obejmuje zarządzanie grą.
2. Programować obsługę zdarzeń związaną z rozgrywką i obsługą interfejsu.
3. Wykonywać proste operacje na danych wejściowych.
4. Wyświetlać wynik, obrazy i komunikaty na ekranie w czytelny sposób.
5. Obsługiwać odtwarzanie dźwięków w aplikacji.

**Zadanie**

Studenci zostaną poproszeni o utworzenie aplikacji, którą będzie obsługiwała grę. Program ma wyświetlać i obsługiwać wszystkie interfejsy i ekrany gry.

**Opcjonalne rozszerzenia**

* Możesz dodać bardziej rozbudowany interfejs użytkownika, który pozwoli graczowi wybierać słowa lub odczytywać je z pliku.
* Możesz rozszerzyć listę słów, które gracz może odgadywać, i losowo wybierać jedno z tych słów.
* Możesz dodać zapis wyników gry i wyników graczy.

**Oczekiwane wyniki**

Po zakończeniu tego ćwiczenia studenci powinni być w stanie:

1. Tworzyć interfejsy użytkownika w aplikacji mobilnej.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Przeprowadzać logikę gry piętnastka.
4. Wyświetlać interfejs i obsługiwać grę.
5. Zrozumieć i obsługiwać błędy w grze.
6. Testować i debugować grę.
7. Projektować interaktywne rozwiązania gier.
8. Rozwiązywać problemy związane z logiką gry.
9. Organizować kod i dokumentować go.
10. Tworzyć aplikacje mobilne w Android Studio.

**Zadanie**

Na podstawie przedstawionego poniżej szkieletu aplikacji implementującego interfejs gry uzupełnić ekran rozgrywki grą w „piętnastkę" (lub inaczej "Gra w 15"). Jest to logiczna łamigłówka polegająca na układaniu kawałków obrazka w odpowiedniej kolejności, aby uzyskać poprawny obrazek. Gra ta jest często rozgrywana na planszy o wymiarach 4x4 (czyli 15 kawałków plus jedno puste pole), stąd jej nazwa.

**Zasady gry w piętnastkę**

Celem gry jest ułożenie kawałków obrazka w odpowiedniej kolejności, aby uzyskać poprawny obrazek (zazwyczaj numerowany od 1 do 15 lub zawierający obrazek). Plansza do gry składa się z 4 rzędów i 4 kolumn (łącznie 16 pól). Na planszy znajdują się kawałki obrazka o numerach od 1 do 15 oraz jedno puste pole. Gracz może przemieszczać kawałki obrazka na planszy, przesuwając je na puste pole. Można to zrobić w jednym z czterech kierunków: w górę, w dół, w lewo lub w prawo. Kawałek jest przesuwany na puste pole, które staje się teraz zajęte przez ten kawałek. Gracz wygrywa, gdy uda mu się ułożyć kawałki w odpowiedniej kolejności, na przykład od 1 do 15, tworząc pełny obrazek. Gra w piętnastkę jest znana z tego, że może być trudna, ponieważ przemieszanie kawałków na planszy wymaga rozwiązywania układanki logicznej. Niektóre konfiguracje planszy mogą być nierozwiązywalne. Implementację gry i rozgrywki należy uzupełnić o licznik czasu, licznik ruchów, warunek zwycięstwa lub porażki, liczbę przesuwanych puzzli w trakcie mieszania, dźwięki przesuwania i mieszania, dźwięki menu.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**Sprawozdanie**

Sprawozdanie z ćwiczenia w ramach nauki programowania aplikacji mobilnych powinno zawierać istotne informacje i dokumentację dotyczącą zadania oraz jego realizacji. Oto kilka kluczowych elementów, które powinny być uwzględnione w sprawozdaniu:

**1. Tytuł i informacje ogólne**

* Tytuł ćwiczenia.
* Imię i nazwisko studenta.
* Data realizacji ćwiczenia.

**2. Cel ćwiczenia**

* Krótka informacja o celu i znaczeniu ćwiczenia, jak również o tym, czego studenci mieli się nauczyć.

**3. Opis projektu**

* Opis funkcjonalności gry w ramach projektu.
* Charakteryzacja interfejsu użytkownika, zawierająca informacje o wykorzystanych widokach i elementach interfejsu.

**4. Implementacja**

* Opis procesu tworzenia projektu, wraz z krokami realizacji gry.
* Omówienie wykorzystanych narzędzi i technologii, w tym Android Studio, język Java/Kotlin.
* Przedstawienie kodu źródłowego aplikacji, zarówno XML (layout) jak i kodu Java/Kotlin.

**5. Funkcje kluczowe**

* Omówienie kluczowych funkcji aplikacji, takich jak obsługa puzzli, zarządzanie stanem gry oraz sposób prezentacji wyników.

**6. Testowanie**

* Opis testowania gry, włączając w to przykłady testów przeprowadzonych w trakcie implementacji.
* Przykładowe przypadki testowe i raport z wynikami testów.

**7. Wyniki**

* Przedstawienie wyników działania aplikacji, w tym zrzuty ekranu demonstrujące działanie gry.

**8. Podsumowanie**

* Krótka ocena projektu i osiągnięć w kontekście zrealizowanego zadania.
* Wnioski wynikające z ćwiczenia, jakie umiejętności i doświadczenie zdobyli studenci.

**9. Trudności i błędy**

* Informacje na temat ewentualnych problemów napotkanych podczas implementacji gry i jak zostały one rozwiązane.

**10. Źródła i odniesienia**

* Jeśli korzystano z materiałów lub źródeł zewnętrznych, uwzględnij je w tekście.

**11. Dodatkowe materiały**

* Ewentualne dodatkowe materiały, takie jak kody źródłowe, zrzuty ekranu lub inne dokumentacje, które uzupełniają sprawozdanie.

**Zalecenia ogólne**

* Sprawozdanie powinno być czytelne i przejrzyste, z odpowiednimi nagłówkami i numeracją stron.
* Projektowanie i implementacja powinny być opisane w sposób logiczny i zrozumiały.
* Sprawozdanie powinno być dostatecznie szczegółowe, aby inny programista mógł zrozumieć projekt i ewentualnie go udoskonalić lub wykorzystać w przyszłości.

**Przykład do wykorzystania w ćwiczeniu i opis propozycji rozwiązania zadania**

Oto przykład prostego projektu aplikacji Android z ekranami menu, opcji, rozgrywką, informacjami o autorach, ekranem zakończenia gry i listą wyników. W tym przykładzie używamy Android Studio, języka Java i bibliotek Android.

**Utworzenie nowego projektu w Android Studio**

Uruchom Android Studio.

Wybierz opcję "New Project".

Wybierz szablon "Empty Activity" i kliknij "Next".

Skonfiguruj nazwę i lokalizację projektu oraz inne parametry według własnych potrzeb.

Kliknij "Finish", aby utworzyć projekt.

**Dodanie ekranów**

W projekcie Android Studio dodawanie ekranów polega na tworzeniu layoutów XML dla każdego z ekranów, które chcesz wyświetlać w aplikacji. Każdy ekran będzie miał swój własny plik XML, który definiuje wygląd tego ekranu.

W projekcie utwórz layouty XML dla każdego z ekranów, czyli pliki .xml w katalogu **res/layout**. Wykorzystaj narzędzia do projektowania interfejsu Android Studio do tworzenia tych plików. W Android Studio przejdź do folderu **res/layout**. Poniżej opiszę, jak utworzyć pliki layoutów XML dla poszczególnych ekranów.

**Menu** - W Android Studio przejdź do folderu res/layout. Kliknij prawym przyciskiem myszy na folderze layout, wybierz "**New**" > "**Layout Resource File**".

Wpisz nazwę pliku, na przykład "activity\_menu", i kliknij "OK".

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

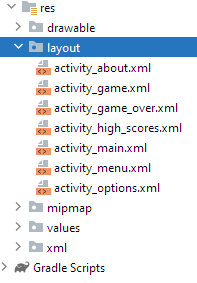
**Opcje** - Teraz utwórz ekran opcji. Postępuj podobnie jak w przypadku ekranu menu, tworząc nowy plik XML o nazwie "activity\_options" i definiując interfejs użytkownika, który umożliwia użytkownikowi dostosowanie ustawień.

**Gra** - Rozgrywka: Dla ekranu rozgrywki, utwórz plik XML o nazwie "activity\_game". Tutaj możesz definiować interfejs gry, na przykład planszę gry, elementy sterujące itp.

Informacje o Autorach: Twórz ekran "Informacje o Autorach" w pliku XML o nazwie "activity\_about". Dodaj tu informacje na temat autorów aplikacji lub inne treści informacyjne.

**Gra Zakończenie -** Jeśli potrzebujesz ekranu zakończenia gry, utwórz plik XML o nazwie "activity\_game\_over". Możesz tu wyświetlić wynik gracza lub informacje o zakończeniu gry.

**Gra Lista Wyników -** Jeśli planujesz ekran z listą wyników, stwórz plik XML o nazwie "activity\_high\_scores". Tutaj możesz wyświetlić wyniki graczy w formie listy.



**activity\_menu.xml** - Ekran menu.

**activity\_options.xml** - Ekran opcji.

**activity\_game.xml** - Ekran rozgrywki.

**activity\_about.xml** - Ekran informacji o autorach.

**activity\_game\_over.xml** - Ekran zakończenia gry.

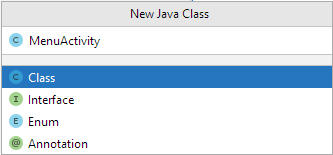
**activity\_high\_scores.xml** - Ekran listy wyników.

W każdym z tych plików XML możesz używać różnych elementów interfejsu użytkownika, takich jak TextView, Button, ImageView, ListView itp., aby dostosować wygląd i zachowanie każdego ekranu. Następnie będziesz mógł podać zawartość tych ekranów i zdefiniować ich zachowanie w kodzie aplikacji.

**Utworzenie klas aktywności**

Następny krok polega na utworzeniu klas aktywności (Activity) dla każdego z ekranów w Twojej aplikacji Android. Każda klasa aktywności będzie zarządzać jednym z ekranów, co pozwoli na kontrolowanie interakcji użytkownika i logiki na danym ekranie.

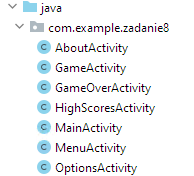
W katalogu java Twojego projektu utwórz nowe klasy Java dla każdej z aktywności, które chcesz dodać do swojej aplikacji. Na przykład, utwórz klasy o nazwach MenuActivity, OptionsActivity, GameActivity, AboutActivity, GameOverActivity i HighScoresActivity.



Każda z tych klas powinna dziedziczyć po AppCompatActivity, co jest standardowe w aplikacjach Android:

public class MenuActivity extends AppCompatActivity {  
}

Powtórz ten krok dla pozostałych ekranów, tworząc osobne klasy aktywności dla każdego z nich.



**Zdefiniowanie układu XML**

W każdej z klas aktywności będziesz potrzebować pliku XML do definiowania układu interfejsu użytkownika na tym ekranie. Dla przykładu, w **MenuActivity** utworzłeś wcześniej plik **activity\_menu.xml** w katalogu **res/layout** i zdefiniujesz w nim układ ekranu menu.

Teraz we wszystkich utworzonych klasach aktywności użyj metody

**setContentView(R.layout.nazwa\_pliku\_xml)** do skojarzenia klasy aktywności z odpowiednim plikiem XML. Na przykład:

import android.os.Bundle;  
import androidx.appcompat.app.AppCompatActivity;  
public class MenuActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_menu*);  
 }  
}

**Uaktualnienie AndroidManifest.xml**

Plik manifestu (AndroidManifest.xml) jest kluczowym plikiem w projekcie Android, który definiuje różne informacje o aplikacji, w tym konfiguracje, uprawnienia i deklaracje komponentów, takie jak aktywności. Tak, aby Twoje aktywności były widoczne i dostępne w aplikacji, musisz dodać ich deklaracje do pliku manifestu. Oto jak to zrobić:

W Android Studio znajdziesz plik manifestu w folderze **app/src/main**. Otwórz go, aby edytować jego zawartość.

W sekcji **<application>**, możesz dodać deklaracje aktywności za pomocą elementu **<activity>**. Każda aktywność powinna mieć swoją własną deklarację.

<application>  
 <activity android:name=".MenuActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity android:name=".OptionsActivity" />  
 <activity android:name=".GameActivity" />  
 *<!-- Dodaj deklaracje dla pozostałych aktywności -->*</application>

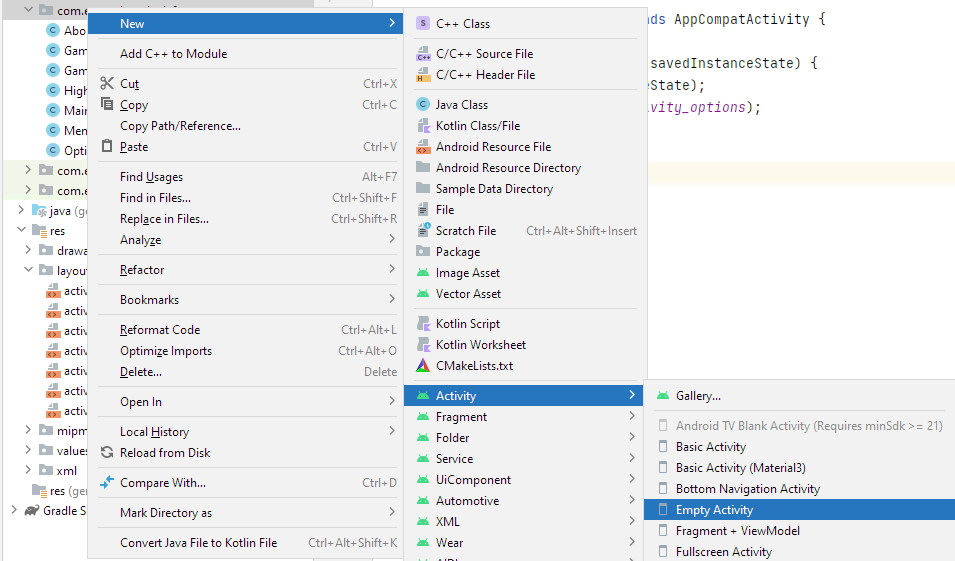
W przykładzie powyżej deklarujemy trzy aktywności: MenuActivity, OptionsActivity i GameActivity. Aktywność oznaczona jako LAUNCHER będzie punktem startowym aplikacji.

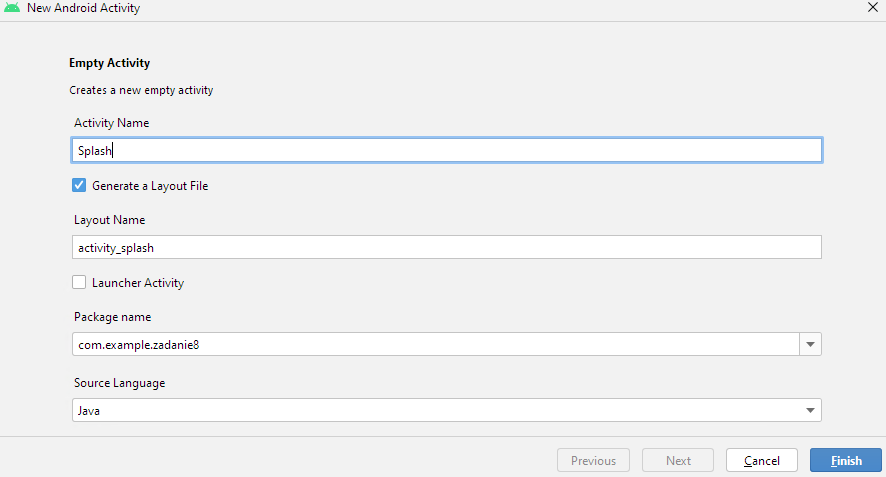
Upewnij się, że zapisujesz zmiany w pliku manifestu po dodaniu deklaracji aktywności.

Po zmianach w pliku manifestu może być konieczne przebudowanie projektu. Możesz to zrobić, wybierając opcję "Build" > "Rebuild Project" w Android Studio.

Teraz Twoje aktywności są zadeklarowane w pliku manifestu i są dostępne w ramach Twojej aplikacji. Plik manifestu pełni kluczową rolę w definiowaniu, jak aplikacja ma się zachowywać, w tym w jaki sposób aktywności są uruchamiane i zarządzane.

Innym sposobem utworzenia klas aktywności jest skorzystanie z kreatora aktywności:





**Implementacja logiki ekranów**

W każdej z klas aktywności implementuj logikę odpowiednią dla danego ekranu. Na przykład, w **GameActivity** możesz dodawać logikę gry, w **OptionsActivity** obsługiwać ustawienia, a w **AboutActivity** prezentować informacje o autorach.

W odpowiednich klasach aktywności obsługuj interakcje użytkownika, przyciski i wszystkie elementy interfejsu użytkownika, które mają wpływ na zachowanie ekranów.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**Nawigacja między ekranami**

Aby umożliwić nawigację między ekranami, użyj intencji (Intent) do rozpoczęcia nowych aktywności z innych ekranów. Przykład nawigacji z ekranu menu do ekranu opcji:

public void goToOptions(View view) {  
 Intent intent = new Intent(this, OptionsActivity.class);  
 startActivity(intent);}

**Przygotowanie pliku XML**

W pliku XML dla danego ekranu, w którym chcesz dodać przycisk nawigacyjny, umieść odpowiedni widok (np. Button), który będzie służył jako przycisk nawigacyjny.

**Przypisanie identyfikatora do przycisku**

Nadaj przyciskowi identyfikator za pomocą atrybutu android:id w pliku XML. To pozwoli na identyfikację przycisku w kodzie Java. Przykład dla przycisku **Ekran Gry** w aktywności **activity\_menu**:

<Button  
 android:id="@+id/buttonGame"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Ekran Gry"  
 app:layout\_constraintBottom\_toTopOf="@+id/buttonOptions"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />

**Obsługa przycisku nawigacyjnego w kodzie**

W klasie aktywności, która odpowiada za dany ekran, znajdź przycisk nawigacyjny za pomocą jego identyfikatora i dodaj obsługę jego kliknięcia. Przykład implementacji logiki w dla przycisku **Ekran Gry** w aktywności **MenuActivity**:

Button gameButton = findViewById(R.id.*buttonGame*);  
gameButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MenuActivity.this,GameActivity.class);  
 startActivity(intent);  
 }  
});

Dodatkowo w każdym widoku należy dodać przycisk powrotu do menu

<Button  
 android:id="@+id/backButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Powrót do menu"  
 tools:layout\_editor\_absoluteX="2dp" />

W każdej aktywności należy dodać obsługę przycisku powrotu do menu

Button backButton = findViewById(R.id.*backButton*);  
backButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(AboutActivity.this, MenuActivity.class);  
 startActivity(intent);  
  
 }  
});

Można jeszcze dodać przycisk Exit

Button exitButton = findViewById(R.id.*buttonExit*);  
exitButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
});

Oraz splash screen, który przez pewnien czas wyświetli np. logo i przejdzie do menu.

public class SplashActivity extends AppCompatActivity {  
 private static final long *SPLASH\_DISPLAY\_LENGTH* = 3000;  
 private static final long *ANIMATION\_DURATION* = 2000; *// Czas animacji w milisekundach* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_splash*);  
 ImageView imageView=findViewById(R.id.*imageView*);  
  
 *// Tworzenie animacji Alpha* Animation alphaAnimation = new AlphaAnimation(0.0f, 1.0f);  
 alphaAnimation.setDuration(*ANIMATION\_DURATION*);  
  
 *// Dodawanie Listenera do animacji, aby obsłużyć zakończenie* alphaAnimation.setAnimationListener(new Animation.AnimationListener() {  
 @Override  
 public void onAnimationStart(Animation animation) {  
 *// Początek animacji* imageView.setImageResource(R.drawable.*amiga*);  
 }  
  
 @Override  
 public void onAnimationEnd(Animation animation) {  
 *// Koniec animacji* }  
  
 @Override  
 public void onAnimationRepeat(Animation animation) {  
 *// Powtórzenie animacji (nie jest używane)* }  
 });  
 imageView.startAnimation(alphaAnimation);  
  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Intent menuIntent = new Intent(SplashActivity.this, MenuActivity.class);  
 startActivity(menuIntent);  
 finish();  
 }  
 }, *SPLASH\_DISPLAY\_LENGTH*);  
 }  
}

Omówiony wyżej projekt obsługi gry dołączony jest do instrukcji ćwiczenia (Zadanie8). Przykładowa implementacja gry w Piętnastkę została dołączona do ćwiczenia (Zadanie9). Dołączona gra nie obsługuje warunku zwycięstwa i wymaga implementacji tej funkcjonalności. Należy połączyć szkielet obsługi gry (Zadanie8) oraz grę w piętnastkę (Zadanie9). Dodatkowo należy utworzyć plansze tytułową, menu, ekrany informujące o wygranej oraz ekran wyjścia. Należy również przewidzieć możliwość samodzielnej konfiguracji grafiki (zamiast obrazków cyfr). Należy dodać efekty dźwiękowe, np. towarzyszące przesuwanym puzzlom, efekty menu itp.