

Nazwa kwalifikacji: **Projektowanie, programowanie i testowanie aplikacji**

Symbol kwalifikacji: **INF.04**

Numer zadania: **02**

Wersja arkusza: **SG**

Wypełnia zdający

Numer PESEL zdającego*

--	--	--	--	--	--	--	--	--	--	--	--	--

Numer stanowiska

--	--	--

Miejsce na naklejkę z numerem
PESEL i z kodem ośrodka

Czas trwania egzaminu: **180** minut

INF.04-02-26.06-SG

EGZAMIN ZAWODOWY

Rok 2026

CZĘŚĆ PRAKTYCZNA

**PODSTAWA PROGRAMOWA
2019**

Instrukcja dla zdającego

1. Na pierwszej stronie arkusza egzaminacyjnego wpisz w oznaczonym miejscu swój numer PESEL*, numer stanowiska i naklej naklejkę** z numerem PESEL i z kodem ośrodka.
2. Sprawdź, czy arkusz egzaminacyjny zawiera 7 stron i nie zawiera błędów. Ewentualny brak stron lub inne usterki zgłoś przez podniesienie ręki przewodniczącemu zespołu nadzorującego.
3. Zapoznaj się z treścią zadania oraz stanowiskiem egzaminacyjnym. Masz na to 10 minut. Czas ten nie jest wliczany do czasu trwania egzaminu.
4. Czas rozpoczęcia i zakończenia pracy zapisze w widocznym miejscu przewodniczący zespołu nadzorującego.
5. Wykonaj samodzielnie zadanie egzaminacyjne. Przestrzegaj zasad bezpieczeństwa i organizacji pracy.
6. Po zakończeniu wykonania zadania pozostaw arkusz egzaminacyjny z rezultatami wykonania zadania na swoim stanowisku lub w miejscu wskazanym przez przewodniczącego zespołu nadzorującego.
7. Po uzyskaniu zgody zespołu nadzorującego możesz opuścić salę/miejsce przeprowadzania egzaminu.

Powodzenia!

* w przypadku braku numeru PESEL – seria i numer paszportu lub innego dokumentu potwierdzającego tożsamość

** w przypadku otrzymania naklejki

Zadanie egzaminacyjne

UWAGA: numerem zdającego jest w zadaniu nazywany numer, którym został podpisany arkusz egzaminacyjny (PESEL lub, w przypadku jego braku, numer paszportu).

Wykonaj aplikacje konsolową i mobilną oraz ich dokumentację. Aplikacje wykonaj według wskazań podanych w części I i II zadania, a dokumentację zgodnie z opisem w części III. Wykorzystaj konto **Egzamin** bez hasła.

Utwórz folder i nazwij go numerem zdającego. W folderze utwórz trzy podfoldery o nazwach: *konsolowa*, *mobilna*, *dokumentacja*. Po wykonaniu zadania umieścisz w nich rezultaty swojej pracy.

Część I. Aplikacja konsolowa

Napisz program implementujący klasę *Notatka* służącą programowi do notatek użytkownika.

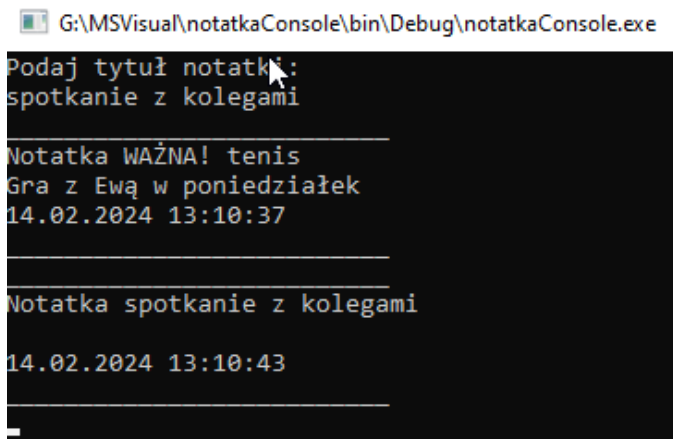
Założenia do programu:

- Program wykonywany w konsoli
- Zastosowany obiektowy język programowania zgodny z zainstalowanym na stanowisku egzaminacyjnym: C++ lub C#, lub Java, lub Python
- Zastosowane znaczące nazewnictwo, polskie albo angielskie
- Kod zapisany czytelnie, z zachowaniem zasad czystego formatowania
- Komunikacja z użytkownikiem jest prowadzona w sposób zrozumiały (ilustracja 1)

- Zawiera klasę *Notatka* (*Note*) oraz sprawdzenie działania klasy w programie głównym
- Ogólnodostępne pola klasy, zawierające:
 - Tytuł notatki, typu napisowego
 - Treść notatki, typu napisowego
 - Priorytet notatki, typu logicznego, notatka jest ważna lub nie jest ważna
 - Datę dodania notatki typu przechowującego datę
- Konstruktor trójargumentowy, którego argumentami są: tytuł, treść i priorytet, mający za zadanie:
 - Przypisać wartości argumentów do odpowiednich pól klasy
 - Przypisać dacie dodania notatki aktualną datę
- Konstruktor jednoargumentowy, którego argumentem jest tytuł, mający za zadanie:
 - Przypisać wartość argumentu do pola tytułu
 - Przypisać treści pusty napis
 - Priorytet notatki ustawić na wartość oznaczającą, że notatka nie jest ważna
 - Przypisać dacie dodania notatki aktualną datę

Uwaga: dopuszcza się zastosowanie typu dla daty i czasu do pola przechowującego datę. Dopuszcza się zastosowanie jednego konstruktora z argumentami domyślnymi, jeśli język programowania nie przewiduje przeciążania konstruktorów.

- Ogólnodostępna metoda wypisująca notatkę, bezargumentowa i niezwracająca wartości. Wyświetla na ekranie w osobnych liniach:
 - Linie poziomą utworzoną z kilkunastu podkreślników
 - Jeżeli notatka jest ważna, tekst: „Notatka WAŻNA! <tytuł>”
 - W przeciwnym razie, tekst: „Notatka <tytuł>” gdzie <tytuł> jest wartością odpowiedniego pola klasy
 - Treść notatki
 - Datę dodania notatki
 - Linie poziomą utworzoną z kilkunastu podkreślników



```
G:\MSVisual\notatkaConsole\bin\Debug\notatkaConsole.exe
Podaj tytuł notatki:
spotkanie z kolegami

Notatka WAŻNA! tenis
Gra z Ewą w poniedziałek
14.02.2024 13:10:37

Notatka spotkanie z kolegami
14.02.2024 13:10:43
```

Ilustracja 1. Działanie programu głównego

Na ilustracji 1 przedstawiono działanie programu. Pierwsza notatka została utworzona konstruktorem trójargumentowym, druga jednoargumentowym. Obie notatki powstały dnia 14.02.2024 r.

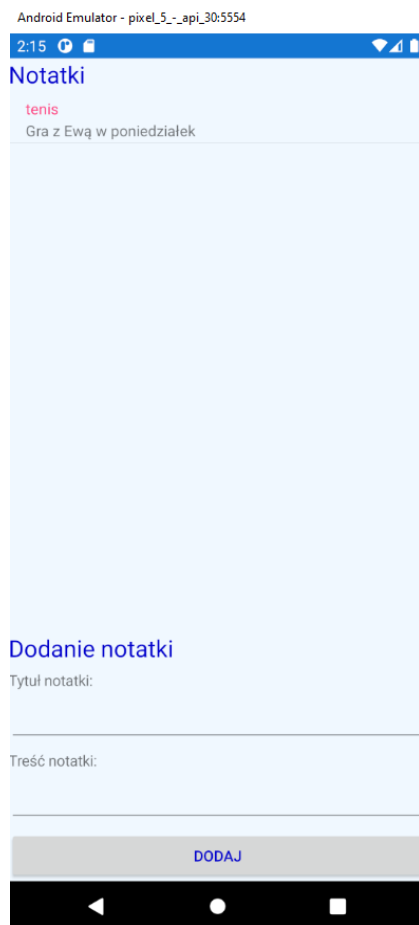
- Program główny ma za zadanie:
 - Utworzyć obiekt klasy *Notatka* za pomocą konstruktora trójargumentowego (podając trzy argumenty). Notatka jest ważna, wartości argumentów konstruktora dowolne – statycznie wpisane w kodzie
 - Utworzyć obiekt klasy *Notatka* za pomocą konstruktora jednoargumentowego (podając jeden argument). Tytuł notatki został pobrany z klawiatury
 - Wyświetlić dane obu obiektów za pomocą metody do tego przeznaczonej (ilustracja 1)
- Dokumentację aplikacji należy utworzyć zgodnie z opisem w części III treści zadania.

Kod aplikacji przygotuj do nagrania na płytę. W folderze *konsołowa* należy umieścić archiwum całego projektu i skopiowany z projektu plik z kodem źródłowym programu oraz plik wykonywalny, jeżeli istnieje.

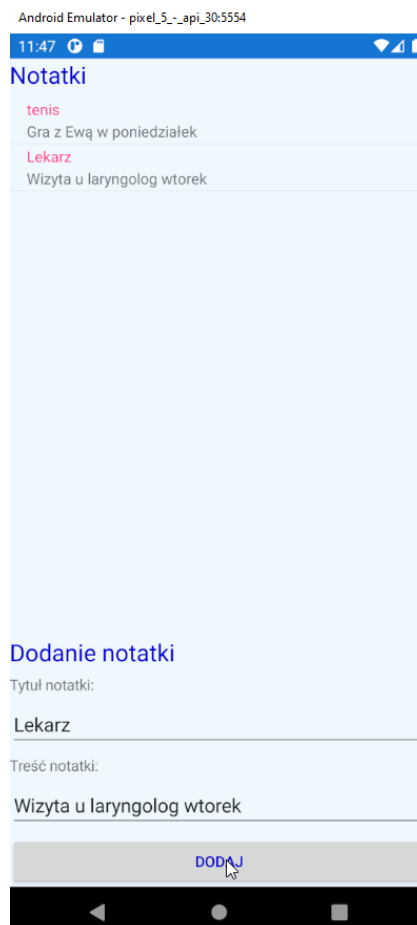
Część II. Aplikacja mobilna

Za pomocą środowiska programistycznego dostępnego na stanowisku egzaminacyjnym wykonaj aplikację mobilną służącą do zapisywania notatek użytkownika. Do wykonania zadania można zastosować klasę *Notatka* z aplikacji konsolowej.

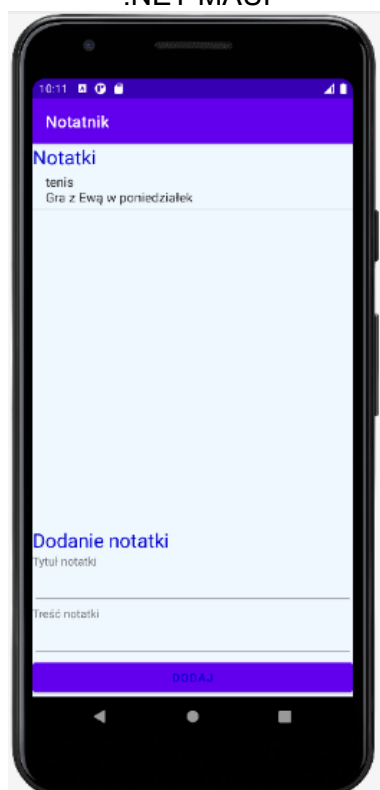
Stany aplikacji zostały przedstawione na ilustracjach 2, 3, 4 i 5. W zależności od zastosowanego środowiska programistycznego oraz emulowanego systemu wynik końcowy może nieznacznie się różnić od przedstawionego.



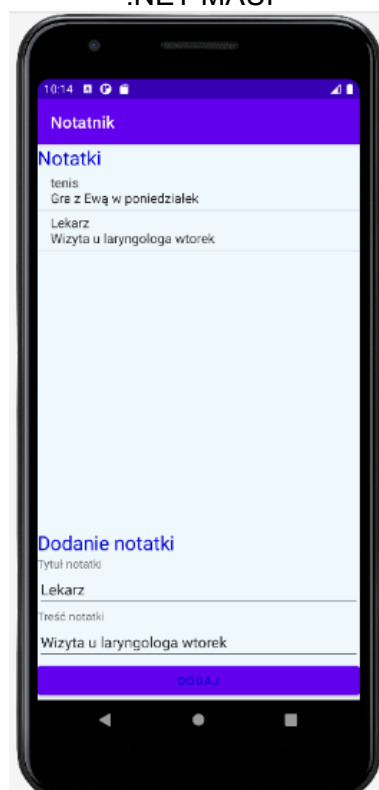
Ilustracja 2. Stan początkowy
.NET MAUI



Ilustracja 3. Dodanie notatki
.NET MAUI



Ilustracja 4. Stan początkowy
Android Studio



Ilustracja 5. Dodanie notatki
Android Studio

Założenia aplikacji

- Interfejs użytkownika zapisany za pomocą języka znaczników wspieranego w danym środowisku (np. XAML, XML)
- Zastosowany liniowy typ rozkładu z kolorem tła AliceBlue (#F0F8FF). Rozkład zawiera:
 - Napis o treści „Notatki”, rozmiar czcionki większy od pozostałych napisów, kolor tekstu MediumBlue (#0000CD)
 - Lista (widok listy), której źródłem jest kolekcja przechowująca notatki jako elementy (kolekcja musi być typu, który jest możliwy do zastosowania z widokiem listy. Dodanie elementów do kolekcji jest widoczne w widoku listy)
 - Napis o treści „Dodanie notatki”, rozmiar czcionki większy od domyślnego rozmiaru, kolor tekstu MediumBlue (#0000CD)
 - Napis o treści „Tytuł notatki: ”
 - Pole do wprowadzania tekstu
 - Napis o treści „Treść notatki: ”
 - Pole do wprowadzania tekstu
 - Przycisk o treści „DODAJ” o kolorze czcionki MediumBlue (#0000CD)

Stan początkowy:

- Wyświetlona jest jedna notatka z dowolnym tytułem i treścią (ilustracja 2 lub 4)

Działanie aplikacji po wybraniu przycisku (ilustracja 3 lub 5):

- Tworzony jest nowy element notatki z wartościami tytułu i treści pobranymi z pól edycyjnych, notatka nie jest ważna
- Element jest dodawany do kolekcji oraz jest widoczny w widoku listy

Kod aplikacji powinien być zapisany czytelnie, z zachowaniem zasad czystego formatowania, należy stosować znaczące nazewnictwo, polskie albo angielskie.

Podejmij próbę kompilacji i emulacji aplikacji. Informacje dotyczące dokumentacji i zrzutów ekranowych umieszczono w części III zadania.

Kod aplikacji przygotuj do nagrania na płytę. W folderze *mobilna* należy umieścić archiwum całego folderu projektu oraz skopiowane pliki z kodem źródłowym, które były modyfikowane.

Część III. Dokumentacja aplikacji

Wykonaj dokumentację do aplikacji utworzonych na egzaminie.

W kodzie źródłowym aplikacji konsolowej za pomocą komentarza utwórz jeden nagłówek do metody według wzoru z listingu 1. Komentarz powinien znaleźć się nad lub pod nazwą funkcji. W miejscu nawiasów <> należy podać odpowiednie opisy. W miejscu autor należy podać numer zdającego.

UWAGA: Dokumentację metody należy umieścić w komentarzu (wieloliniowym lub kilku jednoliniowych). Znajdujący się w listingu 1 wzór dokumentacji jest bez znaków początku i końca komentarza, gdyż te są różne dla różnych języków programowania.

Listing 1. Wzór dokumentacji metody

```
nazwa:          <nazwa metody>
opis:           <opis metody>
parametry:      <opis parametru1, lub „brak”>
                 <opis parametru2>
zwracany typ:   <nazwa typu i opis co jest zwracane lub „brak”>
autor:          <numer zdającego>
```

Wykonaj zrzuty ekranu dokumentujące uruchomienie aplikacji utworzonych podczas egzaminu. Zrzuty powinny obejmować cały obszar ekranu monitora z widocznym paskiem zadań. Jeżeli aplikacja uruchamia się, na zrzucie należy umieścić okno z wynikiem działania programu oraz otwarte środowisko programistyczne z projektem lub okno terminala z kompilacją projektu. Jeżeli aplikacja nie uruchamia się z powodu błędów kompilacji, należy na zrzucie umieścić okno ze spisem błędów i widocznym otwartym środowiskiem programistycznym. Wykonać należy tyle zrzutów, ile interakcji podejmuje aplikacja. Wymagane zrzuty ekranu:

- Aplikacja konsolowa – dowolna liczba zrzutów nazwanych *konsola1*, *konsola2* ...
- Aplikacja mobilna – dowolna liczba zrzutów nazwanych *mobilna1*, *mobilna2* ... (np. stan początkowy, po dodaniu elementu)

Utwórz plik tekstowy i nazwij go *egzamin*. Dokument powinien zawierać informacje o narzędziach wykorzystanych na egzaminie:

- Nazwę systemu operacyjnego
- Nazwy środowisk programistycznych
- Nazwę emulatora dla aplikacji mobilnej
- Nazwy języków programowania

Zrzuty ekranu i dokument umieść w folderze *dokumentacja*.

UWAGA: Nagraj płytę z rezultatami pracy. W folderze z numerem zdającego powinny się znajdować trzy foldery: *dokumentacja*, *konsolowa* i *mobilna*.

W folderze *dokumentacja*: pliki ze zrzutami oraz plik *egzamin*.

W folderze *konsolowa*: spakowany cały projekt aplikacji konsolowej, modyfikowane pliki źródłowe, plik wykonywalny, jeśli istnieje.

W folderze *mobilna*: spakowany cały projekt aplikacji mobilnej, modyfikowane pliki źródłowe.

Opisz nagrany płytę numerem zdającego i zapakuj w pudełko. Pozostaw ją na stanowisku wraz z arkuszem egzaminacyjnym.

Czas przeznaczony na wykonanie zadania wynosi 180 minut.

Ocenie podlegać będą 4 rezultaty:

- implementacja, kompilacja, uruchomienie programu,
- aplikacja konsolowa,
- aplikacja mobilna,
- dokumentacja aplikacji.

Automatyczne odświeżanie widoku listy dla Xamarin z <https://docs.microsoft.com>

If you want the ListView to automatically update as items are added, removed and changed in the underlying list, you'll need to use an ObservableCollection. ObservableCollection is defined in System.Collections.ObjectModel and is just like List, except that it can notify ListView of any changes:

```
ObservableCollection<Type> CollectionName = new ObservableCollection<Type>();  
// add elements to collection  
// assign collection to ListView ItemsSource property  
  
//to add element to ListView just add it to collection  
CollectionName.Add(collectionElement);
```

Zastosowanie adaptera do odświeżania widoku listy w Android Studio z <https://developer.android.com>

In order to display items in the list, call `setAdapter(android.widget.ListAdapter)` to associate an adapter with the list.

`ArrayAdapter<T>` implements `ListAdapter`, returns a view for each object in a collection of data objects you provide. By default, the array adapter creates a view by calling `Object#toString()` on each data object in the collection you provide, and places the result in a `TextView`. You may also customize what type of view is used for the data object in the collection. To customize what type of view is used for the data object, override `getView(int, android.view.View, android.view.ViewGroup)` and inflate a view resource.

```
public ArrayAdapter (Context context,int resource, T[] objects)  
You can use embedded resources like android.R.layout.simple_list_item_1
```

```
notifyDataSetChanged();
```

Notifies the attached observers that the underlying data has been changed and any View reflecting the data set should refresh itself.

Example for class Notatka:

```
notes.add(new Notatka("...", "...", true));  
ListView notesList = findViewById(R.id.notes);  
ArrayAdapter<Notatka> arrayAdapter =  
    new ArrayAdapter<>(this,  
        android.R.layout.simple_list_item_1, notes);  
notesList.setAdapter(arrayAdapter);  
...  
//in AddElement Button listener  
notes.add(new Notatka(...));  
arrayAdapter.notifyDataSetChanged();
```

Wypełnia zdający w przypadku nagranej płyty

Do arkusza egzaminacyjnego dołączam płytę CD/DVD z rezultatami wykonania zadania egzaminacyjnego opisaną przeze mnie numerem PESEL

--	--	--	--	--	--	--	--	--	--

 i oświadczam, że jakość nagrania została przeze mnie sprawdzona.

Wypełnia zdający w przypadku nienagranej (pustej) płyty

Poświadczam numerem PESEL

--	--	--	--	--	--	--	--	--	--

, że do arkusza nie dołączam płyty CD/DVD z rezultatami wykonania zadania egzaminacyjnego.

Wypełnia Przewodniczący ZN

- ☐ Potwierdzam, że do arkusza egzaminacyjnego dołączona jest płyta CD/DVD, opisana numerem PESEL zdającego.
- ☐ Potwierdzam, że do arkusza egzaminacyjnego nie została dołączona płyta CD/DVD.

Czytelny podpis Przewodniczącego ZN