

Nazwa
kwalifikacji:

Projektowanie, programowanie i testowanie aplikacji

Symbol
kwalifikacji:

INF.04

Numer zadania:

02

Kod arkusza:

INF.04-02-26.06-SG

Wersja arkusza:

SG

Lp.	Elementy podlegające ocenie/kryteria oceny
R.1	Rezultat 1: Implementacja, kompilacja, uruchomienie programu
	<i>Uwaga: kryteria należy odnieść do aplikacji konsolowej, jeżeli ta nie istnieje lub zawiera tylko kod wygenerowany, zastosować do aplikacji mobilnej. Kryteria dotyczą wyłącznie samodzielnie napisanego kodu. Wystarczy, że sprawdzaną cechę zastosowano dla większości (90%) przypadków w kodzie</i>
R.1.1	Kod źródłowy zapisany w sposób czytelny: instrukcje w osobnych liniach, stosowane spacje pomiędzy operatorami, konsekwentnie stosowana wybrana konwencja dla nawiasów klamrowych
R.1.2	Kod zapisany z wcięciami dla zagnieżdżeń bloków
R.1.3	Polskie lub angielskie, znaczące nazewnictwo metod
R.1.4	Polskie lub angielskie, znaczące nazewnictwo pól. Wyjątkami od reguły są zmienne bufor, tmp, iteratory pętli itp. Kryterium <u>nie jest</u> spełnione tylko wtedy, gdy nazwy zmiennych nic nie znaczą, np. x, fun, foo, tab, tablica
R.1.5	Typy zmiennych pasują do problemu np. typ tekstowy dla tytułu i treści notatki, bool dla określenia priorytetu notatki, typ daty dla przypisania aktualnej daty dodania notatki <i>(w przypadku języka Python typ wynika z przypisanych danych)</i>
R.1.6	Podjęta próba uruchomienia kodu, na zrzucie ekranowym widoczny jest uruchomiony program lub błędy jego kompilacji / interpretacji
R.1.7	Komunikacja z użytkownikiem prowadzona jest w sposób zrozumiały. Jeżeli kod nie uruchamia się z powodu błędów kompilacji - sprawdzić w kodzie aplikacji
R.2	Rezultat 2: Aplikacja konsolowa
	<i>Uwaga: kryteria 2.1 ÷ 2.6 należy sprawdzić w kodzie programu, sprawdzane elementy muszą być zapisane zgodnie ze składnią. Gdy aplikacja nie uruchamia się, a zdający zapisał zrzuty ekranu z uruchomienia aplikacji należy sprawdzić powód braku kompilacji. Jeśli występują błędy w plikach źródłowych zdającego, kryteria 2.8, 2.9 nie są spełnione. Jeżeli błędy występują w innych plikach ocenić na podstawie kodu i zrzutu ekranu. W języku Python self nie jest wliczany do liczby parametrów</i>
R.2.1	Program składa się z programu głównego oraz definicji klasy <i>Notatka (Note)</i> , w której zdefiniowane przynajmniej jedno pole
R.2.2	Klasa zawiera cztery pola o zasięgu public: dwa tekstowe, logiczny, oraz do zapisu daty, w zależności od języka programowania: DateTime, Date lub inne w tym prosty o ile jest dobrze obsługiwany <i>(w języku Python typy wynikają z przypisanych im danych)</i>
R.2.3	Klasa zawiera dwa konstruktory (w języku Python jeden konstruktor z dwoma argumentami domyślnymi) o argumentach: 1) tytuł, treść i priorytet 2) tytuł
R.2.4	Konstruktory przypisują wartości argumentów do odpowiednich pól, polu daty aktualną datę oraz w przypadku konstruktora jednoargumentowego jawnie przypisane są wartości do treści: "" i priorytetu: false
R.2.5	Zdefiniowana metoda bezargumentowa, która nie zwraca wartości i ma zakres public

R.2.6	Metoda wyświetla na ekranie w osobnych liniach: - linię poziomą, - tytuł notatki, poprzedzony słowem WAŻNA tylko wtedy, gdy priorytet notatki jest równy true, - treść notatki, - datę notatki, - linię poziomą
R.2.7	Program uruchamia się w konsoli, udokumentowane zrzutem ekranu
R.2.8	W programie głównym jest tworzony obiekt notatki konstruktorem trójargumentowym oraz jednoargumentowym. Dana do konstruktora jednoargumentowego jest wprowadzona z klawiatury (w Python w przypadku konstruktora jednoargumentowego dla parametrów treść oraz priorytet wartości ustawione na domyślne)
R.2.9	Program wypisuje na ekranie dane obu notatek korzystając z metody wypisującej
R.3	Rezultat 3: Aplikacja mobilna
	<i>Uwaga: nie należy brać pod uwagę różnic pomiędzy emulacjami. Kryteria 3.1 ÷ 3.6 sprawdzić w kodzie źródłowym, sprawdzane elementy muszą być zapisane zgodnie ze składnią.</i> <i>Sprawdzenie uruchomienia i działania aplikacji z kryteriów 3.7, 3.8 dopuszcza się według zrzutów ekranu i kodu źródłowego. Działanie aplikacji przedstawione na zrzutach musi być poparte kodem źródłowym i kompilować się bez błędów.</i> <i>Dla Android Studio dopuszcza się także rozwiązanie w języku Kotlin.</i> <i>W przypadku zastosowania Jetpack Compose kryteria 3.1 ÷ 3.4 nie są spełnione.</i> <i>W przypadku emulacji aplikacji .NET MAUI na emulatorze który nie jest mobilny (np. Windows Machine) kryterium 3.7 nie jest spełnione</i>
R.3.1	Interfejs użytkownika zdefiniowany zgodnie z wybraną technologią (np. XML, XAML lub w przypadku Swift w kodzie źródłowym) oraz zastosowano rozkład liniowy (LinearLayout / StackLayout lub inny o tej idei)
R.3.2	Zastosowana kontrolka widoku listy (np. ListView)
R.3.3	Zastosowane kontrolki: cztery tekstowe ("Notatki", "Dodanie notatki", "Tytuł notatki:", "Treść notatki:"), dwie edycyjne oraz przycisku o treści DODAJ
R.3.4	Nadane kolory: tła rozkładu lub strony AliceBlue (#F0F8FF), czcionki przycisku i napisów "Notatki", "Dodanie notatki": MediumBlue (#0000CD). Powiększona czcionka napisów "Notatki" i "Dodanie notatki"
R.3.5	Jako źródło danych do widoku listy zdefiniowana jest dowolna kolekcja, tak aby jej zmiana powodowała zmianę wyświetlenia listy na ekranie. Lista przechowuje elementy notatki. Np. ArrayList<Notatka>, ObservableCollection<Notatka> (W Xamarin i .NET Maui kolekcja musi być Observable; w Android Studio stosowane jest notifyDataSetChanged() w połączeniu z ArrayAdapterem do ListView)
R.3.6	Zdefiniowana funkcja powiązana ze zdarzeniem kliknięcia przycisku. Funkcja dodaje do kolekcji notatkę, której tytuł i treść wpisane są w pola edycyjne
R.3.7	Aplikacja kompiluje się i uruchamia w emulatorze aplikacji mobilnej, co udokumentowane jest zrzutem ekranu. Jej układ jest zgodny z ilustracją 2 lub 4 w arkuszu egzaminacyjnym. W stanie początkowym jest wyświetlana jedna notatka z dowolnym tytułem i treścią (w uruchomionej aplikacji lub na zrzucie i <u>obowiązkowo</u> w kodzie)
R.3.8	Po wpisaniu tekstu do obu pól edycyjnych i wciśnięciu przycisku jest widoczny dodany nowy element listy (w uruchomionej aplikacji lub na zrzucie i <u>obowiązkowo</u> w kodzie)
R.4	Rezultat 4: Dokumentacja aplikacji

Uwaga: nagłówek z kryteriów 4.1 ÷ 4.5 musi być zgodny ze stanem faktycznym z kodu źródłowego, nawet jeżeli w kodzie są błędy logiczne (liczba pól, typy). Komentarz należy sprawdzić tylko dla jednej metody (konstruktora nie należy traktować jako metody) Zrzuty ekranu z kryteriów 4.6 i 4.7 muszą zawierać cały obszar ekranu z widocznym paskiem zadań.

Dokumentacja z kryterium 4.8 zapisana jest w pliku egzamin

R.4.1	Komentarz jest zgodny z Listingiem 1 arkusza egzaminacyjnego. Komentarz może być wieloliniowy lub kilka jednoliniowych, lub Docstrings (potrójny cudzysłów)
R.4.2	W komentarzu umieszczona nazwa metody oraz opisane jej działanie
R.4.3	W komentarzu umieszczone nazwa i krótki opis wszystkich parametrów lub "brak", gdy nie ma parametrów
R.4.4	W komentarzu umieszczona nazwa typu zwracanego i opis co jest zwracane lub "brak"
R.4.5	W komentarzu umieszczony numer zdającego
R.4.6	Zapisany przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji konsolowej
R.4.7	Zapisany przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji mobilnej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.8	Dokumentacja zawiera: nazwę systemu operacyjnego, nazwy środowisk programistycznych, emulatora, nazwy języków programowania
R.4.9	Przygotowana jest dokumentacja w postaci płyty: oba projekty są spakowane oraz zmodyfikowane pliki są umieszczone na zewnątrz archiwów