Nazwa kwalifikacji:	**Projektowanie, programowanie i testowanie aplikacji**
Oznaczenie kwalifikacji:	**INF.04**
Numer zadania:	**02**
Kod arkusza:	**INF.04-02-24.01-SG**
Wersja arkusza:	**SG**

Lp.	Elementy podlegające ocenie/kryteria oceny
R.1	**Rezultat 1: Implementacja, kompilacja, uruchomienie programu**
	Uwaga: kryteria należy odnieść do aplikacji konsolowej, jeżeli ta nie istnieje, zastosować kryteria 1.1 ÷ 1.6 do aplikacji mobilnej. Kryteria dotyczą wyłącznie kodu napisanego samodzielnie *Wystarczy, że sprawdzaną cechę zastosowano dla większości przypadków w kodzie*
R.1.1	Kod źródłowy zapisano w sposób czytelny: instrukcje w osobnych liniach, stosowane spacje pomiędzy operatorami, konsekwentnie stosowana wybrana konwencja dla nawiasów klamrowych
R.1.2	Kod zapisano z wcięciami dla zagłębień bloków
R.1.3	Użyto znaczące nazewnictwo metod / funkcji
R.1.4	Użyto polskie lub angielskie, znaczące nazewnictwo zmiennych oraz klasy. Wyjątkami od reguły są zmienne bufor, tmp, iteratory pętli itp. Kryterium **nie jest** spełnione tylko wtedy, gdy nazwy zmiennych nic nie znaczą, np. x, fun, foo, tab, tablica
R.1.5	Zastosowano typy zmiennych pasujące do problemu (np. dowolny typ numeryczny dla licznika; dowolny typ napisowy)
R.1.6	Podjęto próbę skompilowania kodu lub uruchomienia w interpreterze, co udokumentowano zrzutem ekranowym przedstawiającym uruchomiony program lub jego kompilację
R.1.7	Program nawiązuje zrozumiałą komunikację z użytkownikiem: monit o wprowadzenie danych, wyprowadzanie wyników opatrzone komentarzem. Jeżeli kod nie uruchamia się z powodu błędów kompilacji - sprawdzić w kodzie aplikacji
R.2	**Rezultat 2: Aplikacja konsolowa**
	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 2.7 ÷ 2.9 nie są spełnione. Jeżeli błędy występują w innych plikach ocenić na podstawie kodu i zrzutu ekranu.* *W kryteriach 2.3 ÷ 2.6 dopuszcza się funkcje zamiast metod (podejście strukturalne)*
R.2.1	Program składa się z programu głównego oraz definicji klasy, w której znajduje się przynajmniej jedna metoda zgodna z treścią zadania (może być niedokończona, lub z błędami)
R.2.2	Zdefiniowane w klasie metody są statyczne o zakresie public
R.2.3	Metoda zliczająca samogłoski ma zmienną łańcuchową jako parametr oraz zwraca wartość typu całkowitego. Zastosowano instrukcję return dla każdej ścieżki decyzyjnej
R.2.4	Metoda zliczająca samogłoski jest zabezpieczona przed wartościami argumentu null lub pusty napis - w obu przypadkach zwraca 0. Metoda nie powoduje błędów odwołania się do nieistniejącego indeksu
R.2.5	Zmienna łańcuchowa jest parametrem metody usuwającej duplikaty. Metoda zwraca łańcuch. Zastosowano instrukcję return dla każdej ścieżki decyzyjnej
R.2.6	Metoda usuwająca duplikaty jest zabezpieczona przed wartościami argumentu null lub pusty napis - w obu przypadkach zwraca pusty napis. Metoda nie powoduje błędów odwołania się do nieistniejącego indeksu
R.2.7	Program uruchamia się w konsoli, co udokumentowano zrzutem ekranu
R.2.8	Program liczy samogłoski "aąeęiouóyAĄEĘIOUÓY" w łańcuchu i wyświetla ich liczbę

R.2.9	Program usuwa duplikaty znaków występujące obok siebie i wypisuje łańcuch bez nich. Sprawdzić dla wielokrotnych spacji, liter, znaków specjalnych. Sprawdzić, gdy powtórzenie jest na początku łańcucha oraz na końcu łańcucha
R.3	**Rezultat 3: Aplikacja mobilna**
	Uwaga: należy uwzględnić różnice pomiędzy emulacjami, nie należy brać pod uwagę cech takich jak marginesy, wielkości bloków itp. Kryteria 3.1 ÷ 3.5 sprawdzić w kodzie źródłowym, 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 3.6 ÷ 3.10 nie są spełnione. Jeżeli błędy występują w innych plikach lub bibliotekach sprawdzić w kodzie oraz na zrzucie ekranu*
R.3.1	Zastosowano język znaczników XML/XAML lub inny do opisu interfejsu użytkownika oraz kod zawiera przynajmniej jeden element / kontrolkę interfejsu graficznego
R.3.2	Zastosowano rozkład liniowy wertykalny (LinearLayout / StackLayout lub inny o tej idei) z zagłębionym rozkładem liniowym horyzontalnym dla kontrolek dotyczących wieku (napisy i suwak). Dla rozkładu głównego lub dla strony zastosowano kolor tła LightGreen (#90EE90)
R.3.3	Zastosowano przynajmniej dwa pola tekstowe, dla pola "Wizyta u weterynarza" zastosowano czcionkę większą niż pozostałych elementów, kolor tła SeaGreen (#2E8B57) i padding 10 oraz zastosowano dwa pola edycyjne, przynajmniej jedno ma podpowiedź "Imię i nazwisko właściciela...” lub "Cel wizyty"
R.3.4	Zastosowano suwak (Slider lub SeekBar) o wartości początkowej 0 i i wartości maksymalnej 20 oraz kontrolkę do wpisania czasu (TimePicker lub EditText z android:inputType="time") o wartości początkowej 16:00 lub 4:00PM
R.3.5	Zdefiniowano zmienną dowolnej kolekcji o typie tekstowym powiązaną z kontrolką listy, np. String[], List<string> z elementami: „Pies”, „Kot”, „Świnka morska”
R.3.6	Pole listy wypełniono elementami: „Pies”, „Kot”, „Świnka morska” (sprawdzić w uruchomionej aplikacji lub na zrzucie i **obowiązkowo** w kodzie)
R.3.7	Zdefiniowano funkcję powiązaną ze zdarzeniem kliknięcia przycisku. Funkcja wypisuje pod przyciskiem lub w oknie dane wpisane do kontrolek: właściciel, gatunek, wiek zwierzęcia, cel wizyty i czas. Dane są oddzielone przecinkiem (sprawdzić w uruchomionej aplikacji lub na zrzucie i **obowiązkowo** w kodzie)
R.3.8	Po wybraniu elementu z listy zmienia się wartość maksymalna suwaka na: 18 dla psa, 20 dla kota, 9 dla świnki morskiej (sprawdzić w uruchomionej aplikacji lub na zrzucie i **obowiązkowo** w kodzie)
R.3.9	W momencie przesuwania suwaka wartość wieku z suwaka wyświetla się w polu tekstowym oznaczającym wiek zwierzęcia, zapis w postaci liczby całkowitej (sprawdzić w uruchomionej aplikacji lub na zrzucie i **obowiązkowo** w kodzie)
R.3.10	Aplikacja kompiluje się i uruchamia w emulatorze, co udokumentowano zrzutem ekranu. Jej układ jest zgodny z obrazem 1a lub 1b w arkuszu egzaminacyjnym
R.4	**Rezultat 4: Dokumentacja aplikacji**
	Uwaga: nagłówek oceniany w kryteriach 4.1 ÷ 4.5 musi być zgodny ze stanem faktycznym z kodu źródłowego, nawet jeżeli w kodzie są błędy logiczne (liczba pól, typy). *Zrzuty ekranu oceniane w kryteriach 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	Dla klasy z aplikacji konsolowej zapisano nagłówek w postaci komentarza zgodny z Listingiem 1 z arkusza egzaminacyjnego (nie liczymy gwiazdek), komentarz może być wieloliniowy lub kilka jednoliniowych lub Docstrings (potrójny cudzysłów) - w tym przypadku opis znajduje się pod słowem class (na początku definicji klasy)
R.4.2	W komentarzu ujęto nazwę i opis działania klasy
R.4.3	W komentarzu ujęto nazwy tylu metod ile znajduje się w klasie
R.4.4	Dla metod, które ujęto w komentarzu zapisano opis
R.4.5	W komentarzu ujęto numer zdającego

R.4.6	Zapisano przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji konsolowej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.7	Zapisano 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, emulatora, nazwy języków programowania