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

Oznaczenie kwalifikacji: **INF.04**

Numer zadania: **02**

Kod arkusza:

Wersja arkusza: **INF.04-02-22.06-SG**

SG

Lp.	Elementy podlegające ocenie/kryteria oceny
R.1	**Rezultat 1: Implementacja, kompilacja, uruchomienie programu**
	Uwaga: Wystarczy, że sprawdzaną cechę zastosowano dla większości przypadków w kodzie. Kryteria należy odnieść do aplikacji konsolowej, jeżeli ta nie istnieje, zastosować 1.1 ÷ 1.5 i 1.7 do aplikacji web
R.1.1	Kod źródłowy zapisano w sposób czytelny: instrukcje w osobnych liniach, stosowane spacje pomiędzy operatorami, konsekwentnie stosowana wybrana konwencja dla nawiasów klamrowych instrukcji blokowej
R.1.2	Kod zapisano z wcięciami dla zagłębień bloków
R.1.3	Użyto znaczące nazewnictwo funkcji / metod
R.1.4	Użyto znaczące nazewnictwo zmiennych / pól. Wyjątkami od reguły są zmienne bufor, tmp, iteratory pętli itp. Kryterium **nie jest** spełnione tylko wtedy, gdy nazwy zmiennych nic nie znaczą, np. x, a, tab, tablica, foo
R.1.5	Zastosowano typy pól pasujące do problemu. W przypadku Python zastosowano jawną konwersję do typu int dla wczytywanych liczb
R.1.6	Program podejmuje komunikację z użytkownikiem, np. monit o wprowadzenie danych jest znaczący (w przypadku aplikacji web podpisy kontrolek, komunikaty w konsoli)
R.1.7	Podjęto próbę skompilowania kodu, co udokumentowano obrazem przedstawiającym wykonywany program lub jego kompilację
R.2	**Rezultat 2: Aplikacja konsolowa**
	Uwaga: kryteria 2.1 ÷ 2.8 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 kryterium 2.9 nie jest spełnione. Jeżeli błędy występują w innych plikach ocenić na podstawie kodu i zrzutu ekranu*
R.2.1	W programie zdefiniowano klasę *Osoba*
R.2.2	Zdefiniowano pola o zakresie private reprezentujące: id dowolnego typu liczbowego całkowitego, imię dowolnego typu tekstowego (w Python zgodnie z konwencją zastosowano jeden lub dwa podkreślniki w nazwie, np. __imie)
R.2.3	Zdefiniowano pole statyczne o zakresie public. Każde powołanie obiektu inkrementuje pole (w konstruktorze). W programie głównym przed powołaniem obiektów odwołano się do pola za pomocą nazwy klasy
R.2.4	Zdefiniowano konstruktor bezparametrowy, w którym są przypisywane wartości pól: 0 dla identyfikatora, "" lub null dla imienia (w Python konstruktor z parametrami, których wartości domyślne to: 0 i "" lub null)
R.2.5	Zdefiniowano konstruktor z dwoma parametrami typu liczbowego i napisowego, wartości parametrów są przypisane do identyfikatora i imienia nowego obiektu
R.2.6	Zdefiniowano konstruktor kopiujący, którego argumentem jest obiekt klasy *Osoba* , w konstruktorze następuje przepisanie wartości pól obiektu do nowo utworzonej instancji (w Python metoda kopiująca z parametrem typu *Osoba*)

R.2.7	Zdefiniowano metodę z argumentem typu napisowego, która wypisuje lub zwraca napis w postaci: „Cześć <argument>, mam na imię <imie>”, gdzie pole <argument> jest przekazane jako parametr wejściowy metody, a <imie> jest wartością pola *imie* obiektu klasy *Osoba*
R.2.8	Zapisano warunek w metodzie wypisującej imię - gdy pole z imieniem osoby jest puste wypisywany jest komunikat „Brak danych”
R.2.9	Program kompiluje się i uruchamia w konsoli, co udokumentowano zrzutem ekranu
R.3	**Rezultat 3: Aplikacja web**
	Uwaga: Kryteria 3.1 ÷ 3.7 sprawdzić w kodzie źródłowym, sprawdzane elementy muszą być zapisane zgodnie ze składnią. W przykładach kodu nazwą tablicy jest "kursy" należy to zaadaptować do nazw zmiennych zdefiniowanych przez zdającego. Aplikacja jest zapisana w bibliotece React.js lub frameworku Angular w innym wypadku rezultat nie jest sprawdzany 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 3.8, 3.9 nie są spełnione. Jeżeli błędy występują w innych plikach lub bibliotekach sprawdzić w kodzie oraz na zrzucie ekranu
R.3.1	Zapisano w aplikacji jeden komponent, który zawiera zadeklarowaną tablicę kursów, np. `kursy = ["Programowanie w C#", "Angular dla początkujących", "Kurs Django"];`

R.3.2	Zdefiniowano formularz, tak że jego pola poprzedzono etykietą powiązaną z polem, zastosowano klasy bootstrap form-group oraz form-control np.
	`<div class="form-group">`
	`  <label for="imieNazw">Imię i nazwisko: </label>`
	`  <input id="imieNazw" type="text" class="form-control" ...>`
	`</div>`
	Uwaga: w React.js atrybut className zamiast class oraz htmlFor zamiast for

R.3.3	Zastosowano znaczące nazwy kontrolek dla atrybutu for i id. Kryterium należy uznać za spełnione zawsze wtedy, gdy nazwa nie jest przepisana z pomocy (exampleInputEmail1) i gdy nie jest postaci input1, x, itp.
R.3.4	Przypisano dla przycisku klasę btn btn-primary
R.3.5	W nagłówku drugiego stopnia napisano liczbę kursów stosując atrybut długości tabeli (kursy.length, lub this.kursy.length) oraz wypisano w liście przynajmniej jedną nazwę kursu odwołując się do dowolnego indeksu tabeli z nazwami kursów
R.3.6	W liście wypisano wszystkie elementy tablicy z nazwami kursów niezależnie od wymiaru tablicy, np.: Angular: `<li *ngFor= "let kurs of kursy">{{ kurs }}</li>` React: `{this.state.kursy.map(kurs => <li key={kurs}>{kurs}</li>)}`

R.3.7	W kodzie zdefiniowano funkcję realizującą zatwierdzenie formularza zdarzeniem submit (onSubmit, ngSubmit), dostęp do wartości zapisanych w polach następuje poprzez mechanizmy charakterystyczne dla środowiska np. (f, i - dowolna nazwa)
	Angular: `<form #f="ngForm" (ngSubmit) = "onSubmit(f)"> ...`
	`  <input ... ngModel name="i">`
	React: `i = React.createRef(); ...`
	`<form onSubmit={this.hndlSubmit}>...<input ... ref={this.i}/>`

R.3.8	Aplikacja jest interpretowana bez błędów i uruchamia się w przeglądarce, co udokumentowano zrzutem ekranu
R.3.9	Po wybraniu przycisku „Zapisz do kursu” w konsoli przeglądarki zostaje wypisane: - wartość wpisana w pierwsze pole formularza - gdy numer wpisany w formularzu przyjmuje wartości 1, 2, 3 - nazwa kursu pod tym indeksem, w przeciwnym wypadku komunikat "Nieprawidłowy numer kursu"
R.4	**Rezultat 4: Testy aplikacji**

	Uwaga: kryteria 4.1 i 4.4 sprawdzić w kodzie źródłowym klasy. 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	W programie głównym aplikacji konsolowej zapisano testy sprawdzające działanie klasy *Osoba* (przynajmniej jedna instrukcja zgodna ze składnią operująca na klasie lub obiekcie klasy)
R.4.2	Na początku działania aplikacji jest wyświetlany stan pola z liczbą instancji, który jest równy 0
R.4.3	Na końcu działania aplikacji jest wyświetlany stan pola statycznego klasy, który jest równy liczbie powołanych obiektów
R.4.4	Powołane zostały trzy obiekty, każdy innym konstruktorem: bezparametrowym, dwuparametrowym, kopiującym (w Python bezparametrowym i metodą kopiującą). Dane dla konstruktora dwuparametrowego zostały pobrane z klawiatury w programie głównym
R.4.5	Dla obiektów których pole imię zostało ustawione w konstruktorze został wyświetlony komunikat: "Cześć Jan, mam na imię <imie> "
R.4.6	Dla obiektu utworzonego konstruktorem bezparametrowym został wyświetlony komunikat "Brak danych"
R.4.7	Zapisano przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji konsolowej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.8	Zapisano przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji web, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.9	Dokumentacja zawiera: nazwę systemu operacyjnego, nazwy środowisk, nazwy języków programowania, frameworka Angular lub bilbioteki React