INF.04 (Projektowanie, programowanie i testowanie aplikacji) (Przykładowe) zadanie praktyczne II

Zadanie zostało stworzone przez Marioneq.

To drugie stworzone przeze mnie zadanie podobne do tych, które pojawiają się na egzaminie praktycznym INF.04. Tym razem I część zadania skupia się na programowaniu obiektowym. W części II do zrobienia jest aplikacja pozwalająca na przeglądanie filmów. Natomiast III część zadania dotyczy testów jednostkowych, które dotychczas w zadaniach egzaminacyjnych tworzonych przez CKE pojawiły się tylko raz.

To zadanie prawdopodobnie trochę jest trudniejsze od pierwszego, ale wydaje mi się, że zdarzają się zadania od CKE, które są podobnej trudności, choć nie gwarantuję tego. Zadanie nie powinno, ale może zawierać błędy.

Materiały wideo użyte w tym zadaniu są pobrane z serwisu Pixabay.

Powodzenia dla rozwiązujących

Zadanie

Zaimplementuj klasę (część I), wykonaj aplikację webową (część II) oraz testy jednostkowe (część III) według opisanych wymagań. Do wykonania aplikacji webowej potrzebne będą materiały z dołączonego archiwum *materialy.7z*, archiwum nie jest zabezpieczone żadnym hasłem.

Część I. Implementacja klasy

Za pomocą narzędzi do tworzenia aplikacji konsolowych, zaprojektuj część logiki systemu automatów paczkowych. Zaimplementuj klasę o nazwie *Skrytka* oraz typ wyliczeniowy *RozmiarSkrytki*.

Założenia dotyczące klasy *Skrytka* i typu *RozmiarSkrytki*:

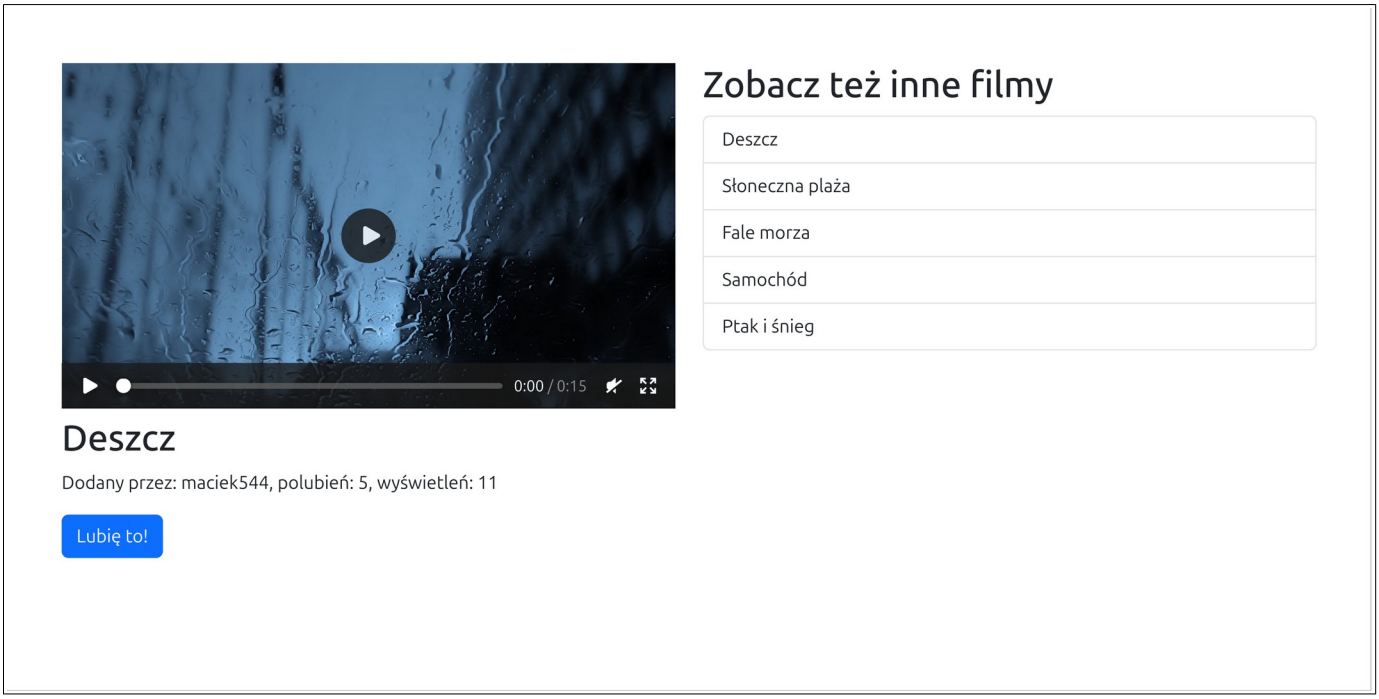
- Typ *RozmiarSkrytki* posiada trzy możliwe wartości *Maly* = 10, *Sredni* = 20, *Duzy* = 30
- Klasa *Skrytka* zawiera:
  - Trzy pola: rozmiar typu *RozmiarSkrytki*, zawartość skrytki typu tekstowego lub pustego oraz kod dostępu typu tekstowego lub pustego. Dostęp do tych pól ma jedynie klasa. W przypadku późniejszego rozszerzania klasy, klasy potomne **nie mają** dostępu do tych pól
  - Konstruktor z jednym parametrem, który przekazuje rozmiar skrytki, reszta pól ma ustawioną wartość pustą
  - Ogólnie dostępne pole statyczne zliczające liczbę instancji klasy, początkowo ma wartość 0
  - Metodę statyczną, która generuje losowy kod składający się z 5 cyfr
  - Metodę, która zwraca informację o tym czy skrytka jest pusta (pole z zawartością skrytki ma wartość pustą)
  - Metodę, która umieszcza paczkę w skrytce: ustawia zawartość na przekazaną jako parametr metody i ustawia kod na losowo wygenerowany przez odpowiedzialną za to metodę tej klasy. W przypadku, gdy w skrytce już się coś znajduje (pole z zawartością skrytki nie jest puste) lub zawartość jest za duża (długość zawartości jest większa niż ustalony rozmiar skrytki), należy zwrócić wartość fałsz i nie ustawiać nowej zawartości i kodu. Jeśli udało się umieścić paczkę w skrytce, należy zwrócić ustawiony kod
  - Metodę, do otwierania skrytki: sprawdza czy podany kod (przekazany jako parametr metody) jest poprawny. Jeśli tak zwraca zawartość skrytki, ustawia kod i zawartość skrytki na pustą. Jeśli jednak kod jest nieprawidłowy należy zwrócić wartość fałsz

Ogólne założenia:

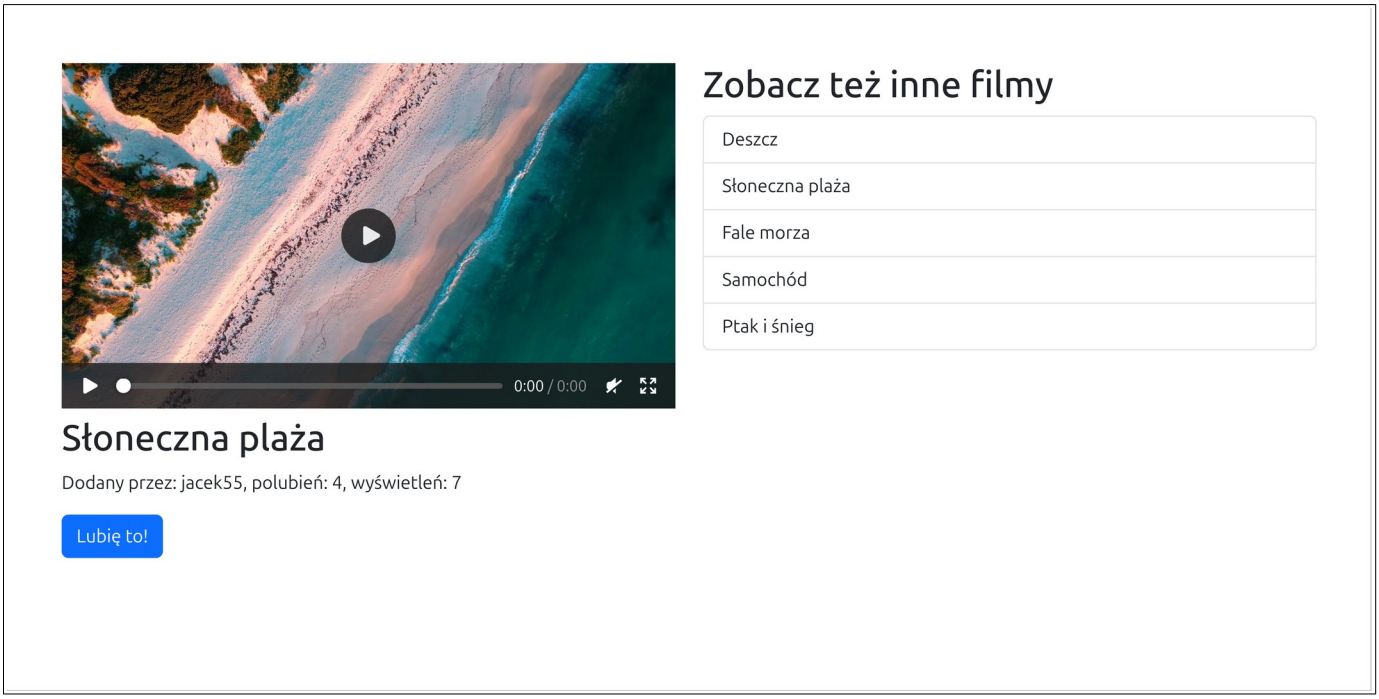
- Zastosowany wybrany obiektowy język programowania spośród: C++, C#, Java, Python
- Kod powinien być zapisany czytelnie, z zachowaniem zasad czystego formatowania kodu
- Do klasy dołączono testy jednostkowe, ich opis znajduje się w części III zadania
- Należy stosować znaczące, angielskie lub polskie nazwenictwo zmiennych i funkcji

Część II. Aplikacja webowa

Z zastosowaniem frameworka Angular lub biblioteki React.js (do wyboru) wykonaj aplikację internetową typu front-end będącą fragmentem aplikacji portalu społecznościowego pozwalającej na oglądanie filmów przesłanych przez użytkowników i dzielenia się swoimi wrażeniami na temat umieszczonych filmów.



Obraz 1. Stan początkowy aplikacji



Obraz 2. Stan po przełączeniu filmu i kliknięciu przycisku „Lubię to!”

Założenia aplikacji:

- Aplikacja składa się z dokładnie jednego komponentu, którego widok początkowy przedstawiono na obrazie 1
- Do utworzenia aplikacji należy wykorzystać materiały wideo i plik tekstowy o nazwie *dane*, które znajdują się w archiwum dołączonym do zadania
- Z pliku *dane.txt*, który zawiera listę filmów z ich danymi, należy skopiować treść jako tablicę. Dane z tej tablicy należy wykorzystać w aplikacji

- Kod aplikacji powinien być zapisany czytelnie, z zachowaniem zasad czystego formatowania kodu, należy stosować znaczące, angielskie lub polskie nazwenictwo zmiennych i funkcji

Zawartość interfejsu aplikacji:

- Blok główny
  - Kolumna lewa
    - Blok wideo z włączonymi kontrolkami przeglądarki (wyświetlany jest aktualnie wybrany film), należy użyć materiałów wideo z archiwum
    - Nagłówek 2 stopnia, którego treścią jest tytuł aktualnie wybranego filmu
    - Akapit o treści „Dodany przez: <autor>, polubień: <polubienia>, wyświetleń: <wyswietlenia>", gdzie fragmenty otoczone nawiasami < i > oznaczają wartości dotyczące aktualne wybranego filmu
    - Akapit zawierający przycisk o treści „Lubię to!"
  - Kolumna prawa
    - Nagłówek 2 stopnia o treści „Zobacz też inne filmy"
    - Lista zawierająca wszystkie filmy, treścią kolejnych elementów listy są tytuły filmów

Działanie aplikacji:

- Domyślnie aktualnie wybranym filmem jest pierwszy znajdujący się w tablicy, zostaje dla niego zwiększona liczba wyświetleń o 1
- Kliknięcie przycisku „Lubię to!" powoduje zwiększenie liczby polubień aktualnie wybranego filmu o 1 (dozwolone jest kilkukrotne zwiększanie liczby polubień przez kilkukrotne wciskanie tego przycisku)
- Kliknięcie jednego z elementów listy znajdującej się w prawej kolumnie powoduje zmienienie aktualnie wybranego filmu na ten, który został kliknięty. Zostaje zwiększona liczba wyświetleń o 1 (dla tego filmu)

Wygląd aplikacji:

- Przycisk i lista są stylowane z użyciem Bootstrap zgodnie (patrz 2. tabela)
- Podział na dwie kolumny jest zrealizowany również z użyciem Bootstrap (patrz 2. tabela)
- Blok wideo ma szerokość równą 100% szerokości kolumny
- Blok główny ma ustawione margiesy wewnętrzne 48px (z każdej strony)

Podejmij próbę uruchomienia aplikacji w przeglądarce. Wykonaj zrzut ekranu dokumentujący uruchomienie aplikacji. Na zrzucie należy umieścić okno przeglądarki internetowej z widoczną otwartą aplikacją.

Część III. Testy jednostkowe

Wykonaj testy jednostkowe dla klasy *Skrytka* utworzonej wcześniej przez siebie w części I zadania. Do napisania testów wykorzystaj wybraną przez siebię bibliotekę do przeprowadzania testów jednostkowych odpowiednią dla języka programowania, w którym została wykonana ta klasa. Podczas pisania testów zastosuj się do ogólnych założeń opisanych w tamtej części zadania. W testach jednostkowych należy sprawdzić przypadki, które zostały opisane w poniższej tabeli.

Nr testu	Cel testu	Kroki testowe	Oczekiwany rezultat
1.	Licznik instancji	Utworzenie dwóch obiektów klasy *Skrytka.*	Pole statyczne zliczające ilość utworzonych instancji ma wartość 2.
2.	Generowanie kodu	Wywołanie metody generującej losowy kod.	Metoda zwraca kod składający się z 5 znaków, które są cyframi.
3.	Poprawne umieszczenie paczki do skrytki	Utworzenie obiektu, próba umieszczenia paczki z zawartością o dopuszczalnej długości.	Metoda zwraca kod (poprawność generowania kodu jest sprawdzana w 2 teście, w tym teście wystarczy sprawdzić czy zwracany jest tekst).
4.	Umieszczenie paczki do zajętej skrytki	Utworzenie obiektu, poprawne umieszczenie paczki, następnie próba kolejnego umieszczenia paczki z zawartością o dopuszczalnej długości.	Metoda zwraca wartość fałsz.
5.	Umieszczenie za dużej paczki do skrytki	Utworzenie obiektu, próba umieszczenia paczki z zawartością dłuższą niż dopuszczalna.	Metoda zwraca wartość fałsz.
6.	Otwieranie skrytki poprawnym kodem	Utworzenie obiektu, poprawne umieszczenie paczki, próba otwarcia skrytki z użyciem poprawnego kodu.	Metoda zwraca zawartość skrytki, jest taka sama jak przy wcześniejszym umieszczaniu do skrytki.
7.	Otwieranie skrytki błędnym kodem	Utworzenie obiektu, poprawne umieszczenie paczki, próba otwarcia paczki z użyciem błędnego kodu.	Metoda zwraca wartość fałsz.

Tabela 1. Scenariusz testów jednostkowych

Po napisaniu testów, uruchom wszystkie z nich. Wykonaj zrzut/zrzuty ekranu dokumentujące uruchomienie utworzonych testów, zapisz je pod nazwami *test1.png*, *test2.png*, itd. Zrzuty powinne obejmować okno środowiska programistycznego/terminala, na którym widać wyniki przeprowadzonych testów. W tej części zadania byłoby oceniane wykonanie testów jednostkowych, **a nie poprawność działania klasy** utworzonej w części I tego zadania (poprawność klasy byłaby oceniana **tylko** w rezultatach dotyczących części I zadania). Jeśli część I została wykonana niepoprawnie, to testy powinne wykazać błędy.

Czas przeznaczony na wykonanie zadania wynosiłby 180 minut.

Angular

To use Bootstrap add to styles.css: `@import "~bootstrap/dist/css/bootstrap.css";`

React.js

To use Bootstrap add: `import 'bootstrap/dist/css/bootstrap.css';`

Buttons

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Bootstrap has a base `.btn` class that sets up basic styles such as padding and content alignment. By default, `.btn` controls have a transparent border and background color, and lack any explicit focus and hover styles.

```
<button type="button" class="btn">Base class</button>
```

Bootstrap includes several button variants, each serving its own semantic purpose, with a few extras thrown in for more control.

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
```

Grid system

Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system, six default responsive tiers, Sass variables and mixins, and dozens of predefined classes.

Basic example

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive. Below is an example and an in-depth explanation for how the grid system comes together.

```
<div class="container text-center">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>
```

List group

List groups are a flexible and powerful component for displaying a series of content. Modify and extend them to support just about any content within.

Basic example

The most basic list group is an unordered list with list items and the proper classes. Build upon it with the

options that follow, or with your own CSS as needed.

```
<ul class="list-group">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

Tabela 2. Wybrane fragmenty dokumentacji Bootstrap