**Zad 1.**

Stwórz klasę, która będzie pełniła rolę wrappera (storage-a) konfiguracji połączeniowej z API. Klasa ta ma się nazywać MyRequest oraz przechowywać takie atrybuty jak:

* URL address
* Method (np. “GET”)
* API Token

Obiekt tej klasy ma posiadać gettery oraz settery do odpowiednich pól. Zadbaj o poprawną enkapsulację.

Następnie stwórz klasę Sender, która będzie zawierała takie metody statyczne jak:

* sendReq(request, destination), gdzie request to odpowiednio skonfigurowany obiekt wyżej utworzonej klasy; destination to obiekt niżej utworzonej klasy, do której będziemy kierowali request

Następnie stwórz klasę ApiService, która będzie symulowała działanie API serwisu. Klasa ta ma zawierać:

* statyczne listy:

countries = [“Poland”, “Japan”, “Madagascar”, “Mali”, “Nepal”]

continents = [“European”, “Asia”, “Australia”, “Africa”, “Asia”]

* statyczne pola:

key (pole to będzie weryfikowane z polem API Token z request)

* metody: getCountries, getContinents z parametrem request

Metody te będą odbierały wysyłany przez Sender obiekt typu Request, wyciągały z niego API Token, porównywały z polem statycznym - key. Jeżeli wartości te będą identyczne, to sprawdzamy również, czy methodType również pobrany z request, ma wartość równą GET. Jeżeli wszystkie te warunki zostaną spełnione, to wówczas metody mają zwracać odpowiednie listy (countries/continents). W przeciwnym razie zwracana lista ma być pusta.

**Zad 2.**

Stwórz klasę WaterVassel, która będzie zawierała takie pola jak: id, volume (objętość naczynia), water\_volume (objętość jaką zajmuje woda w obiekcie takie klasy).

Następnie utwórz listę 5 obiektów WaterVassel o losowej pojemności z przedziału 50 - 200 i water\_volume równym 0.

Zdefiniuj zmienną water\_canister = 1000, która będzie odzwierciedlała całkowitą do rozlania ilość wody między naczyniami. Proces rozlewania wody ma następować po równo do każdego z naczyń, dopóki nie zostaną one w pełni napełnione lub dopóki water\_canister wyniesie 0.

**Zad 3**

Operacje na frontendzie to nie tylko dbanie o odpowiedni wygląd strony, ale też często walidacja danych przed przekazaniem ich na backend.  
  
Stwórz klasę User, która będzie zawierała takie atrybuty jak (pamiętaj o enkapsulacji):

* name
* surname
* email
* gender
* password
* date\_joined (datę twórz z wbudowanej biblioteki Date)

Podczas ustawiania wartości tych pól, ma za każdym razem następować walidacja, czy zmienne te są poprawne. To, jakie warunki ma walidować system, zależy od kreatywności programisty. Przykładem może być sprawdzenia, czy hasło zawiera co najmniej 8 znaków (w tym co najmniej jedną wielką literę, jedną cyfrę i jeden znak specjalny). Do walidacji możesz wykorzystać ReGexa.

**Zad 4**

Wyobraź sobie, że poniższa lista reprezentuje nierelacyjną bazę danych użytkowników.

| const users = [  new User(‘ab12ex’, ‘Alex’, ‘[alex@alex.com](mailto:alex@alex.com)’, ‘123123’),  new User(‘fg12cy’, ‘Asab’, ‘[asab@asab.com](mailto:asab@asab.com)’, ‘123456’),  new User(‘zwf8md’, ‘Brook’, ‘[brook@brook.com](mailto:brook@brook.com)’, ‘13111’),  new User(‘eefamr’, ‘Martha’, ‘[martha@martha.com](mailto:martha@martha.com)’, ‘123222’),  new User(‘ghderc’, ‘Thomas’, ‘[thomas@thomas.com](mailto:thomas@thomas.com)’, ‘123333’)  ];  const products = [  {  \_id: 'eedfcf',  name: 'mobile phone',  description: 'Huawei Honor',  price: 200,  ratings: [  { userId: 'fg12cy', rate: 5 },  { userId: 'zwf8md', rate: 4.5 }  ],  likes: []  },  {  \_id: 'aegfal',  name: 'Laptop',  description: 'MacPro: System Darwin',  price: 2500,  ratings: [],  likes: ['fg12cy']  },  {  \_id: 'hedfcg',  name: 'TV',  description: 'Smart TV:Procaster',  price: 400,  ratings: [{ userId: 'fg12cy', rate: 5 }],  likes: ['fg12cy']  }  ] |
| --- |

Stwórz klasę User, która będzie zawierała takie pola i metody jak:

* #id, username, email, #password, #createdAt, #isLoggedIn
* singIn - umożliwiać ona będzie logowanie się użytkownika na podstawie informacji zawartych w bazie (wyżej utworzonej liście users)
* countLikes - metoda ta będzie przeszukiwać listę products i wyświetlać nazwy produktów, które polubił dany użytkownik (zauważ, że informacje o polubieniach przechowujemy wewnątrz pola likes z rekordu każdego produktu)
* rateProduct - która przyjmować będzie id produktu wraz z oceną, którą użytkownik chce przypisać