**8 JS Zadania Dziedziczenie i prototypy**

**Zad 1.**

Wymień korzyści płynące z wykorzystywania dziedziczenia w swoich programach.

**Zad 2.**

Stwórz klasę Teacher dziedziczącą po Person. W klasie Person mają znajdować się takie pola jak: name oraz surname. W Teacher zaimplementuj metodę *teach*, która otrzymuje stringa *subject* i wydrukuje:

<Teacher’s name and surname> is now teaching <subject>.

**Zad 3.**

Stwórz klasę Account, która będzie przechowywała pola: balance, number (numer konta) oraz metody: deposit(value), withdraw(value) oraz gettery i settery dla wymienionych pól. Implementację metod deposit oraz withdraw pozostawiam Tobie. Pamiętaj o dodaniu odpowiednich walidacji w takich metodach (np. wpłacana wartość nie może być ujemna).

Następnie stwórz dwie klasy dziedziczące po Account: SavingAccount oraz CurrentAccout.

Klasa SavingAccount powinna posiadać również pole: interest i metodę, która będzie odpowiednio zwiększała wartość przechowywanego atrybutu.

CurrentAccount powinien natomiast składać się z atrybutu overdraft\_limit z metodą zwiększającą jego wartość.

Następnie stwórz klasę Bank, która będzie zawierała tablicę wielu obiektów Account (konkretnie CurrentAccout oraz SavingAccount). W banku stwórz metodę update, która będzie iterowała po każdym koncie i dodawała do niego dowolną wielkość depozytu. Dodatkowo, w przypadku obiektu typu SavingAccount, ma być zwiększane pole interest każdego konta o 5, a dla CurrentAccount - overdraft\_limit o 10.  
  
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/instanceof

**Zad 4.**

Podaj przykład praktycznego wykorzystania prototypu w JS.

**Zad 5.**

Stwórz klasę Airplane z polem name oraz flagą isFlying (typ bool), która domyślnie jest ustawiana na false.

Utwórz dwa obiekty Airplane.

Korzystając z prototypu, dodaj do obiektów metodę takeOff, która będzie ustawiała flagę isFlying na true oraz land - setującą isFlying na false. Sprawdź, czy metody dodane dzięki prototypowi są prawidłowo wywoływane i realizują swoje założenia.

**Zad 6. (zewnętrzne źródło)**

- Write a Car constructor that initializes `model` and `milesPerGallon` from arguments.

- All instances built with Car:

+ should initialize with an `tank` at 0

+ should initialize with an `odometer` at 0

- Give cars the ability to get fueled with a `.fill(gallons)` method. Add the gallons to `tank`.

- Give cars ability to `.drive(distance)`. The distance driven:

+ Should cause the `odometer` to go up.

+ Should cause the the `tank` to go down taking `milesPerGallon` into account.

- A car which runs out of `fuel` while driving can't drive any more distance:

+ The `drive` method should return a string "I ran out of fuel at x miles!" x being `odometer`.