DR INŻ. PIOTR SIEWNIAK

Moduły lokalne

Zadania #1

Zadanie 1

Zdefiniować moduł `node.js` o nazwie `prostokat` zawierający definicje funkcji pozwalających na obliczenie pola, obwodu i przekątnej prostokąta. Parametrami wspomnianych funkcji powinny być długości boków prostokąta.

Wyeksportować zdefiniowane funkcje jako zasoby publiczne modułu `prostokat`. Eksport funkcji zrealizować przy wykorzystaniu obiektu `exports`.

Napisać prostą aplikację konsolową `node.js` pozwalającą na obliczenie pola i obwodu prostokąta dla zadanych długości boków: 1 oraz 2. W celu wykonania obliczeń wykorzystać zasoby publiczne modułu `prostokat`.

Zadanie 2

Zdefiniować moduł `node.js` o nazwie `prostopadloscian` zawierający funkcje pozwalające na obliczenie objętości, pola powierzchni całkowitej oraz łącznej długości wszystkich krawędzi prostopadłościanu. Parametrami wspomnianych funkcji powinny być długości boków prostopadłościanu.

Wyeksportować zdefiniowane funkcje jako zasoby publiczne modułu `prostopadloscian`. Eksport funkcji zrealizować przy wykorzystaniu właściwości `exports` obiektu `module`.

Napisać prostą aplikację konsolową `node.js` pozwalającą na obliczenie objętości, pola powierzchni całkowitej oraz łącznej długości wszystkich krawędzi prostopadłościanu dla zadanych długości boków: 1, 2 oraz 3.

Uwaga do zadań 1, 2:

Zaprojektować moduły `prostokat` i `prostopadloscian` w trzech wariantach w zależności od rodzaju definicji funkcji. Wykorzystać:

1. deklaracje funkcji (*wariant I*);
2. funkcje anonimowe zdefiniowane za pomocą wyrażeń funkcyjnych (*wariant II*);
3. funkcje strzałkowe (*wariant III*).

Zadanie 3

Treść jak w *zadaniu 1*, ale zasoby modułu `prostokat` zdefiniować zgodnie z zasadami programowania zorientowanego obiektowego. Mianowicie, zamiast niezależnych funkcji zaprojektować i zaimplementować klasę `Prostokat` zawierającą niezbędne komponenty składowe (tj. właściwości i metody). Wspomniana klasa powinna stanowić zasób publicznych modułu `prostokat`.

Wykorzystać komponenty składowe klasy `Prostokat` w testowej aplikacji konsolowej.

Zadanie 4

Treść jak w *zadaniu 2*, ale zasoby modułu `prostopadloscian` zdefiniować zgodnie z zasadami programowania zorientowanego obiektowego. Mianowicie, zamiast niezależnych funkcji zaprojektować i zaimplementować klasę `Prostopadloscian` zawierającą niezbędne komponenty składowe – właściwości i metody.

Wykorzystać elementy członkowskie klasy `Prostopadloscian` w testowej aplikacji konsolowej.

Zadanie 5

Zaprojektować i zaimplementować klasę `Pracownik`, opisującą pracownika szpitala. Uwzględnić co najmniej: imię, nazwisko i staż pracy pracownika oraz metodę zwracającą wymienione dane. Właściwości klasy `Pracownik` zaimplementować jako dane hermetyzowane (ukryte). Definicję klasy `Pracownik` umieścić w module `node.js` o nazwie `pracownik` – jako jej zasób publiczny.

Wykorzystać komponenty składowe klasy `Pracownik` (z modułu `pracownik`) w testowej aplikacji konsolowej.

Zadanie 6

Treść jak w *zadaniu 5*, lecz dodatkowo wykorzystać mechanizm dziedziczenia. W tym celu zaprojektować i zaimplementować klasy: `Lekarz`, `Ordynator` i `Pielegniarka` – które wraz z klasą `Pracownik` wchodzą w skład łańcucha (struktury) dziedziczenia. Wszystkie klasy wchodzące w skład łańcucha dziedziczenia powinny stanowić zasoby publiczne modułu `pracownik`.

Wykorzystać komponenty składowe klas opisanego powyżej łańcucha dziedziczenia (zasobów publicznych modułu `pracownik`) w testowej aplikacji konsolowej.

Zadanie 7

Treść jak w *zadaniu 6*, lecz dodatkowo wykorzystać mechanizm polimorfizmu.

Wykorzystać komponenty składowe klas łańcucha dziedziczenia w testowej aplikacji konsolowej.

Zadanie 8

Treść jak w *zadaniu 7*, lecz zamiast klas (czyli narzędzi ES6) wykorzystać struktury programistyczne pozwalające na implementacje podstawowych cech paradygmatu programowania obiektowego w ES5.

Wykorzystać zasoby publiczne modułu `pracownik` w testowej aplikacji konsolowej.