|  |  |  |
| --- | --- | --- |
| Aleksandra Duda | Gr. 31 lab 02 | Techniki Obiektowe |
| Nr indeksu: 130510 | Laboratorium nr 2 | 13.10.20 r. |

**Sprawozdanie z labolatorium nr 2**

1. **Cel ćwiczenia**

Implementacja interfejsu IVector oraz klass Vector2d, Vector3d wraz z metodami.

Należało również umożliwić obliczanie iloczynu wektorowego oraz tworzenie wektorów 2D oraz 3D na podstawie współrzędnych biegunowych. W projekcie należało uwzględnić takie wzorce jak dekoder i adapter.

1. **Model UML**
2. **Przebieg ćwiczenia**

Z racji tego, że tworzyłam projekt w Pythonie i w tym języku raczej interfejsów się nie praktykuje to zastąpiłam go klasą abstrakcyjną IVector.



Klasa Vector2D ma implementować IVector – w tym przypadku tobię to przy pomocy dziedziczenia. W klasie Vector2d definiuję funkcje działające na wektorach dwuwymiarowych.

//kod

Następnie stworzyłam klasę Vector3D, która obsługuje wektory trójwymiarowe. Klasa Vector3D dziedziczy po klasie Vector2D i przeciąża metody wcześniej zdefiniowane w Vectorze2D.

//kod

Współrzędne x i y w klasie Vector2D są protected – ten modyfikator pozwala na odziedziczenie wartości x oraz y w klasie Vector3D. Dzięki temu, że korzystamy w Vector3D z wartosci x i y z klasy Vector2D, musimy tylko dodać współrzędną „z” w konstruktorze Vector3D.

1. **Wnioski**

IVector możemy implementować na dwa sposoby – poprzez dziedziczenie lub agregację. Jeśli wybraliśmy dziedziczenie to należy pamiętać, że należy zmienić modyfikatory dostępu klasy bazowej na protected, aby móc korzystać ze zmiennych w klasy bazowej w klasie pochodnej.

Dzięki adapterowi mieliśmy możliwość połączenia działania niekompatybilnych obiektów. Działa jak „przełącznik” pomiędzy klasami.