

# Laboratorium 7

Jan Seredyński

14 maja 2015

## 1 Wstęp

Zadaniem laboratorium był refaktoring dotychczas stworzonych klas do sortowania tak aby spełniały zasada otwarte-zamknięte. Należało również zastosować wzorzec projektowy Obserwator.

## 2 Moja implementacja zasady otwarte-zamknięte

Stworzyłem interfejsy/klasy abstrakcyjne bazowe dla obiektów typu lista, element listy, soter, benchmark oraz obserwator. Na podstawie nich można tworzyć dalej odpowiednie klasy. Przykład Klasa MyList(linkedlist) implementuje interfejs List, który narzuca jej stworzenie metod np push\_back, pop\_ack itp.

## 3 Wzorzec projektowy - Obserwator

Stworzyłem klasę abstrakcyjną Observer oraz Observable które są odzwierciedleniem danego wzorca. Odpowiednio po tych klasach dziedziczą klasy MyBenchmark jako obserwator oraz jeden z trzech rodzajów sortera jako obiekt obserwowany.

## 4 Wzorzec projektowy - Budowniczy

Dzięki zastosowaniu wzorca Budowniczego do wszystkich\* sortowań można użyć różnego rodzaju list. W moim programie mam stworzoną LinkedList ale gdyby stworzyć inny rodzaj listy, który spełniałby mój interfejs List to równie dobrze można go wykorzystać w każdej implementacji.

\*Wyjątkiem jest sortowanie przez scalanie które w mojej implementacji działa tylko na LinkedLiscie (MyList)

## 5 Wzorzec projektowy - Fabryka abstrakcyjna

Przy stosowaniu wzorca - budowniczego napotkałem się na problem z kompatybilnością obiektów. Jako obiekty które zwracałem po każdej wywołanej metodzie z różnych sortowań były wskaźniki/referencje na klasę abstrakcyjną List. Co powodowało problem, gdy chciałem te otrzymane wartości wykorzystać do dalszych działań w klasach niżej. Jako rozwiązanie tego problemu zastosowałem fabrykę abstrakcyjną.

## 6 Podsumowanie

Zastosowanie zasady otwarte-zamknięte jest bardzo wydajnym sposobem na utrzymaniu kodu w porządku. Każda nowa funkcjonalność jest w oddzielnej klasie, która swoje podstawowe własności

dziedziczy po klasie abstrakcyjnej albo interfejsie. Dzięki temu kolejne cechy programu można łatwo dodawać nie tracąc wcześniejszych własności klas.