Programowanie obiektowe - cz.3

Spis treści

Immutability 1
Garbage collection 1

Zapiski uczestnika Bootcampu Zajavka.pl w 12 tygodni by Bartek Borowczyk aka Samuraj Programowania. Dopiski na zielono od Karola Rogowskiego.

Immutability

Immutable variables oznacza zmienne **niemutowalne**. Stosując podejście zmiennych niemutowalnych, dążymy do tego, żeby w żaden sposób nie dało się zmodyfikować stanu obiektu, na którym wykonujemy operacje. Jeżeli chcemy coś zmienić, to musimy stworzyć nowy obiekt na podstawie tego poprzedniego ze zmienioną wartością jakiegoś pola.

Garbage collection

Na pewno do tego czasu udało Ci się już zauważyć, że tworzymy obiekty i dalej nie zawracamy sobie głowy ich usuwaniem. Za każdym razem, gdy tworzymy obiekt, rezerwujemy pamięć operacyjną, w której są przechowywane informacje, których potrzebujemy w programie. Potem gdy używany fragment pamięci nie jest już nam potrzebny, należałoby go posprzątać, ale tutaj tego nie robimy. Dlaczego? Istnieją języki programowania, w których pamięcią zarządza się ręcznie. Java robi to za nas - ułatwiając programistom pracę.

Garbage collection (GC) to proces, który uruchamia się w trakcie działania programu, który służy do usuwania nieużywanych obiektów z pamięci. Java jest w stanie wykryć, które obiekty stworzone w pamięci nie są więcej używane i w trakcie uruchomienia procesu **GC** zwolnić miejsce w pamięci, które jest zarezerwowane przez te obiekty. Java w praktyce jest w stanie oznaczyć wszystkie obiekty, które są ciągle w użyciu przez program i jednocześnie wie, że te, które nie zostały oznaczone, można usunąć.

Oprócz tego, że automatyzacja tego procesu ułatwia nam pracę, dochodzi za tym kolejny aspekt - wycieki pamięci. Zarządzając pamięcią ręcznie, często zdarza się zapomnieć, że dany obiekt można usunąć, co przekłada się na to, że program zajmuje więcej pamięci, niż potrzebuje aż do zakończenia jego działania. Może się przez to również zdarzyć, że przez nieprawidłowe zarządzanie pamięcią, program przestanie działać i nie będzie w stanie dokończyć swojej pracy, bo skończyła się pamięć operacyjna na komputerze, gdyż niewłaściwie nią zarządzaliśmy.

Za procesem **GC** stoi dużo teorii, gdyż sam proces oznaczania, które obiekty są wciąż żywe, a które można usunąć, jak również samo usuwanie jest czasowo kosztownym procesem. Zostało wymyślonych dużo rozwiązań jak taki proces optymalizować, aby w jak najmniejszym stopniu był on odczuwany przez użytkownika aplikacji.

Najprostszym sposobem na to, aby obiekt nadawał się do posprzątania przez **GC**, jest sprawienie, by nie

był on wskazywane przez żadną referencję.

Na tym etapie nauki skupiamy się na ogólnym wyjaśnieniu koncepcji **GC**, nie wchodzimy mocno w szczegóły, bo jest to jeszcze zbyt skomplikowane na ten moment.