Projektowanie obiektowe

Dr Karol Przystalski

Poznajmy się

2017 - doktorat uzyskany w PAN oraz UJ

od 2010 - CTO @ **Codete**

2007 - 2009 - Software Engineer @ **IBM**

Praca naukowa

Multispectral skin patterns analysis using fractal methods}, K. Przystalski and M. J.Ogorzalek. Expert Systems with Applications, 2017

https://www.sciencedirect.com/science/article/pii/S0957417417304803

karol.przystalski@uj.edu.pl

kprzystalski@gmail.com

Zaliczenie

Średnia ocena ze wszystkich projektów.

Warunki:

- Termin na każdy projekt: max. 2 tygodnie po zajęciach
- Każdy projekt zaliczony na minimum 3.0

Egzamin: test wyboru, 30 pytań. Aby zaliczyć trzeba poprawnie odpowiedzieć na 20 pytań.

Terminy do ustalenia

Paradygmaty programowania

ang. Programming paradigms

Programowanie imperatywne

Komputer pracuje w sposób imperatywny, czyli wykonywania instrukcji jedna po drugiej jednocześnie zmieniając stan programu.

Języki, które realizują paradygmat programowania imperatywny: COBOL, PHP, Ruby, Java.

Programowanie strukturalne

Wprowadzony został przez Edsgera Wybe Dijkstra w 1968 roku. Zastąpił on skoki typu `goto` instrukcjami typu `if-then`, `while`, itp.

Języki, które realizują paradygmat programowania strukturalnego: ALGOL, Pascal, Ada.

Programowanie obiektowe

Programowanie obiektowe jest jeszcze starsze od strukturalnego, ponieważ jego historia sięga 1966, kiedy Ole Johan Dahl oraz Kristen Nygaard zaproponowali, aby przenieść wywołanie funkcji przenieść do stosu, tak aby zmienne w niej istniejące były dostępne również poza funkcją. I tak powstał konstruktor.

Języki, które realizują paradygmat programowania strukturalnego: Smalltalk, Java.

Programowanie funkcjonalne

Jeszcze starszym paradygmatem jest znany od 1936 roku wraz z wprowadzeniem rachunku lambda (ang. l-calculus). Pierwszy raz został jednak zastosowany dopiero w 1958 w języku LISP.

Języki funkcjonalne nie mają możliwości przypisania (ang. assignment), ponieważ język funkcjonalny z definicji jest niezmienny (immutable).

Języki, które realizują paradygmat programowania strukturalnego: JavaScript, Erlang, LISP, Haskell.

Pozostałe paradygmaty

- Proceduralne: Pascal
- Wizualne: Scratch
- Uogólnione: Java
- Logiczne: Prolog
- Aspektowe: AspectJ
- Deklaratywny: SQL
- Modularne: Fortran90

Zarządzanie pamięcią

ang. Memory management

Heap vs stack

Stack działa na zasadzie LIFO i alokuje pamięć per wątek. Zarządzany przez system operacyjny.

Heap jest wykorzystywany do alokacji pamięci kiedy tylko chcemy, ale jest też wolniejszy. Zarządzany z poziomu aplikacji.

Obie pamięci wykorzystują pamięć RAM.

Manualne zarządzanie pamięcią

Przykładem języka, gdzie zarządzanie pamięcią odbywa się w sposób manualny jest C++.

Służą do tego m.in. funkcje `malloc` i `free`.

Wykorzystujemy do tego również `new` oraz `delete`.

W innych językach też mamy takie funkcje: Java (`new`), Python (`malloc`).

ARC

Jest wykorzystywany m.in. w Swift'cie. Implementuje metody `retain` oraz `release`. Gdy liczba referencji zejdzie do 0, obiekt jest usuwany (`release`).

Więcej o ARC:
https://docs.swift.org/swift-book/LanguageGuide/AutomaticReferenceCounting.html

Słabe referencje (weak references)

Aby zapobiec zapętleniu się referencji wprowadzono weak references, np. w Swift'cie czy Pythonie. Dzięki temu inaczej liczone są referencje np. w ARC i zapobiega to wyciekowi pamięci.

Python: https://docs.python.org/3/library/weakref.html

Swift: https://docs.swift.org/swift-book/LanguageGuide/AutomaticReferenceCounting.html

Garbage collector

Języki, które go stosują to m.in.: Java oraz inne na JVM, JavaScript, C#, Go, Ruby.

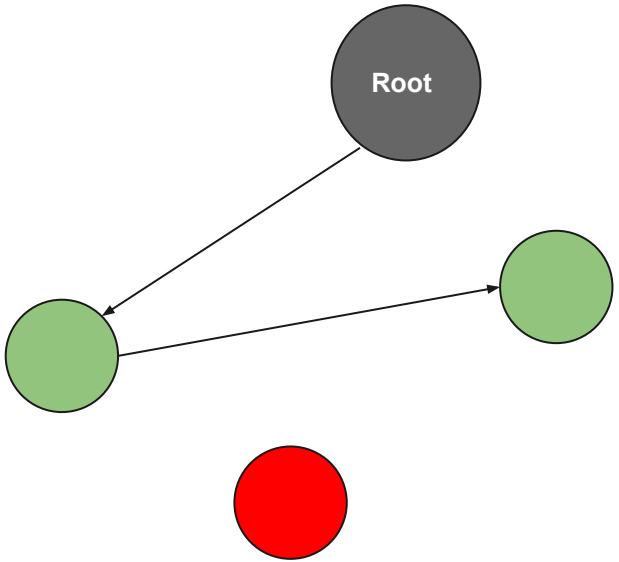
Istnieje wiele implementacji GC:

- Tracing GC - składa się z dwóch etapów: wykrywania „żyjących” obiektów oraz zwalnianie pamięci,
- RC GC - podobne do ARC; posiada osobny proces GC, który czyści pamięć

Garbage collector

Mark - zaznacz (zielone)

Sweep - usuń (czerwone)



Java Garbage collector

Istnieje wiele GC w Javie 11 (`-XX:`):

- Serial Collector - jeden wątek, dobry dla małych aplikacji
- Parallel Collector - wiele wątków; średnie aplikacje
- Garbage-First Collector - równoległe wątki; wykorzystywany na sprzęcie z wieloma procesorami i dużą ilością pamięci
- Z Garbage Collector - dostępny od Javy 11 zoptymalizowany na ograniczenie opóźnień po stronie aplikacji.

Zarządzanie pamięcią w różnych językach

- Python - wykorzystuje połączenie RC z GC.
- Java - GC
- JavaScript - GC
- Haskell - GC
- Go - GC via Go scheduler
- Ruby - GC
- Swift - ARC
- Lisp - GC
- COBOL - manualne
- Lua - GC