Daniel Górski
Korporacyjna Java
Wykład 9: Cykl życia obiektów

Cykl życia obiektu

- Narodziny obiektu: new
  - Klasa może mieć jeden lub wiele konstruktorów
    - Jeśli nie ma żadnego to ma i tak jeden domyślny bezparametrowy
    - Jeśli ma już co najmniej jeden z parametrem to nie ma domyślnego bezparametrowego
  - Klasa może mieć tylko konstruktory prywatne
    - Sama zarządza swoim powstaniem poprzez metody statyczne: wzorzec Singleton
    - Nie musi tego umożliwiać i teoretycznie nie można stworzyć jej instancji

Konstruktor

- Pierwszą linią konstruktora jest wywołanie konstruktora z nadklasy
  - Jeśli klasa nie dziedziczy po żadnej klasie to dziedziczy po klasie Object
  - Konstruktory nadklasy wywołujemy poprzez super() lub super(param1...)
  - Dla ułatwienia konstruktor klasy Object jest dodawany automatycznie, ale możemy go dodać samodzielnie

Koniec życia obiektu

- Jeśli do obiektu nie ma żadnej żywej "silnej" referencji to GC może zechcieć usunąć dany obiekt z pamięci
- Wywołuje wtedy metodę finalize() na obiekcie
  - Od Java 9 jest ona oznaczona jako @Deprecated
  - Od Java 18 zmieniło się działanie w tym obszarze i nie jest możliwe "wskrzeszenie obiektu"
- Także jedyną odpowiedzią na pytanie jak obiekt kończy życie w Javie już wkrótce będzie: jeśli staje się nieużywany to kiedyś zniknie

Weak Reference

- Garbage Collector może usunąć obiekt do którego są tylko referencje Weak
- Generalnie z perspektywy GC referencje Weak nie są istotne
- WeakHashMap -> kolekcja w pakiecie java.util
  - Używana jako cache
  - Nie ma pewności, że obiekt kiedyś dodany nadal tam jest

Soft Reference

- Garbage Collector może usunąć obiekt do którego są tylko referencje Soft, ale tylko gdy mu naprawdę brakuje pamięci
- GC powinien usunąć obiekty adresowane tylko przez referencje Soft zanim rzuci OutOfMemoryError
- W standardowej Javie nie ma gotowej kolekcji Soft
  - Apache Commons
  - Google Collections

Garbage Collector

- Ma szybko dawać nową pamięć dla aplikacji
- Ma szybko wykrywać pamięć jaką przydzielił, a jest już nieużywana
- Miary działania:
  - Przepustowość
  - Opóźnienie
  - Jak dużo pamięci potrzebuje
    - Pamięć wykorzystywana do zarządzania nie jest dostępna dla aplikacji

Rodzaje GC

- Parallel Collector
  - Domyślny do Java 8
  - Stop-the-world
- G1: Garbage First
  - Domyślny od Java 9
  - Dobrze zbalansowany
- Serial
  - Jednowątkowy, do użycia tylko przy małej pamięci
- ZGC, Shenandoah GC
  - Celem jest jak najmniejsze opóźnienie

Struktura pamięci w Javie

- Stack
  - +Native method stack
- Heap
  - Eden
  - Survivor (S1, S2)
  - Tenured (old generation)
- Metaspace
- Cache

Mutable / Immutable objects

- Każdy obiekt ma swój stan
  - W zdegenerowanym przypadku jego stanem może być tylko typ klasy
- Często są możliwe zmiany tego stanu
- Okazuje się, że dużo podstawowych obiektów w Javie nie ma możliwości zmiany stanu
  - String
  - Integer, Long, Double...
  - BigDecimal, BigInteger

Mutable / Immutable objects

- Obiekty, które nie mogą zmienić swojego stanu nazywa obiektami Immutable
- Dlaczego takie obiekty są szczególne?
  - Są w pełni bezpieczne przy przetwarzaniu wielowątkowym
  - Garbage Collector dobrze nimi zarządza

Jak wymusić aby obiekty danej klasy były obiektami Immutable

- Wszystkie zmienne wewnętrzne muszą być final
  - Można teoretycznie zarządzać tym jedynie z poziomu metod i widoczności zmiennych
- Wszystkie zmienne wewnętrzne muszą być również Immutable

Czy kolekcja może być Immutable?

- Musi być kolekcją typów Immutable
- Kolekcja mie może być modyfikowalna
  - Przed Java 9 mieliśmy jedynie niemodyfikowalne widoki kolekcji
    - Mając tylko widok kolekcji nie mogliśmy jej zmienić
    - Zmiany w oryginalnej kolekcji powodowały zmiany w widoku
  - Java 9:
    - List.of, Set.of, Map.of
  - Java 10:
    - List.copyOf, Set.copyOf, Map.copyOf

Pracownia: co wygląda dobrze

- Dużo małych pull-request'ów
  - Przypomina to standardowy styl pracy korporacyjnej
  - Oczywiście zdarzają się wyjątki...
- Segregacja w pakietach
- Rozdzielenie logiki wysokopoziomowej od niskopoziomowej

Pracownia: co wygląda na ciężkie w utrzymaniu

- Mało dużych pull-request'ów
- Powtarzany kod: DRY
- Zbyt szczegółowy kod na wysokim poziomie
  - Powoduje to konieczność wprowadzenia wielu wzajemnie zależnych warunków

Pracownia

- Program ma umożliwić rozwiązywanie podstawowych zadań geometrycznych dla trójkąta prostokątnego
- Nowa akcja: podwojenie figury
  - Wybieramy figurę z już wprowadzonych i tworzymy nową z dwukrotnie większym polem powierzchni
- Nowy atrybut figur: data utworzenia
  - Dodajemy również możliwość sortowania rosnąco i malejąco po dacie utworzenia

Pracownia...

- Oficjalna zapowiedź dwóch dużych wymagań jakie będą miały się znaleźć w finalnej wersji
  - Wykrywanie dubli: nie można dodawać dubli figur (w sensie geometrycznym)
    - Nie można dodać istniejącego już koła opisanego
    - Nie można dodać istniejącej figury przy podwajaniu
    - Nie można wprowadzić np rombu, który jest równoważny z istniejącym kwadratem
    - W pewnych sytuacjach mogą powstać problemy "zaokrągleniowe" i jest to akceptowalne
  - Obsługa dwóch wersji językowych
    - Nie ma to być dwujęzyczność, tylko konfigurowalna jednojęzyczność

Trójkąt prostokątny

- Charakterystyka figury:
  - Długość przyprostokątnej 1
  - Długość przyprostokątnej 2
  - Długość przeciwprostokątnej
  - Pole powierzchni
- Możliwe wejście: (dowolna dwójka)
  - Długość przyprostokątnej 1
  - Długość przyprostokątnej 2
  - Długość przeciwprostokątnej
  - Pole powierzchni