Daniel Górski
Korporacyjna Java
Wykład 8: Interfejsy funkcyjne

Interfejs funkcyjny

- Interfejs z jedną metodą abstrakcyjną
- Zasada jednej odpowiedzialności
- Może być zdefiniowany w linii poprzez wyrażenie lambda

Consumer

- public interface Consumer<T>
- void accept(T t);
- default Consumer<T> andThen(Consumer<? super T> after)

Consumer...

- IntConsumer
- DoubleConsumer
- LongConsumer

BiConsumer

- public interface BiConsumer<T,U>
- void accept(T t, U u)
- default BiConsumer<T,U> andThen(BiConsumer<? super T,? super U> after)

BiConsumer...

- ObjDoubleConsumer<T>
- ObjIntConsumer<T>
- ObjLongConsumer<T>

Supplier

- public interface Supplier<T>
- T get()

Supplier...

- BooleanSupplier
- DoubleSupplier
- IntSupplier
- LongSupplier

Predicate

- public interface Predicate<T>
- boolean test(T t)
- default Predicate<T> and(Predicate<? super T> other)
- default Predicate<T> negate()
- default Predicate<T> or(Predicate<? super T> other)

Predicate...

- DoublePredicate
- IntPredicate
- LongPredicate

Function

- public interface Function<T,R>
- R apply(T t)
- default <V> Function<T,V> andThen(Function<? super R,? extends V> after)
- default <V> Function<V,R> compose(Function<? super V,? extends T> before)
- static <T> Function<T,T>identity()

Function...

- DoubleFunction<R>
- IntFunction<R>
- LongFunction<R>
- ToDoubleFunction<T>
- ToIntFunction<T>
- ToLongFunction<T>

Function...

- DoubleToIntFunction
- DoubleToLongFunction
- IntToDoubleFunction
- IntToLongFunction
- LongToDoubleFunction
- LongToIntFunction

BiFunction

- public interface BiFunction<T,U,R>
- R apply(T t, U u)
- default <V> BiFunction<T,U,V> andThen(Function<? super R,? extends V> after)

Operators

- public interface BinaryOperator<T> extends BiFunction<T,T,T>
- public interface UnaryOperator<T> extends Function<T,T>

Operators...

- DoubleBinaryOperator
- DoubleUnaryOperator
- IntBinaryOperator
- IntUnaryOperator
- LongBinaryOperator
- LongUnaryOperator

Comparator

- public interface Comparator<T>
- int compare(T o1, T o2)
- static <T> Comparator<T> nullsFirst(Comparator<? super T> comparator)
- static <T> Comparator<T> nullsLast(Comparator<? super T> comparator)
- default Comparator<T> reversed()
- default Comparator<T> thenComparing(Comparator<? super T> other)
- ...

Runnable

- public interface Runnable
- void run()

Callable

- public interface Callable<V>
- V call() throws Exception

Pracownia: co wygląda dobrze

- Segregacja klas w pakietach
- Testy, dużo testów unitowych
- Dużo małych pull-request'ów
- Rozdzielenie odpowiedzialności
  - Widziałem, że dla niektórych zaczyna już to być projekt "korporacyjny"
- Rozpoczęcie zarządzania treścią

Pracownia: co wygląda na ciężkie w utrzymaniu / niełądnie stylowo

- Wszystkie klasy w jednym pakiecie
- Duże klasy realizujące różne rzeczy
  - Zwłaszcza pomieszanie logiki wysokopoziomowej ze szczegółową źle wygląda i jest na dłuższą metę trudne w utrzymaniu
- Mało testów
  - Wykrywanie regresji jest problematyczne
- Dużo String hardcode w kodzie
  - Zapowiadam, że wkrótce pojawi się wymaganie: 2 wersje językowe (wymaganie będzie sprecyzowane, ale czas wejścia to ~3 tygodnie)

Pracownia: co zamiast String hardcode

- Najprostsza wersja wystarczająca i prosta w utrzymaniu
  - Tworzymy klasę "Menagera treści"
  - Ma on metodę przyjmującą np enum i zwracającą komunikat
- Inicjalnie trzeba w to włożyć nieco pracy, ale później dodawanie kolejnych komunikatów będzie szło sprawnie
- Dodanie nowej wersji językowej będzie wtedy proste od strony technicznej: główny koszt to samo tłumaczenie

Pracownia

- Program ma umożliwić rozwiązywanie podstawowych zadań geometrycznych dla trójkąta dowolnego
- Wszystkie figury mają od tej pory cechę obwód
  - Nie zmieniają się sposoby tworzenia figur (obwód nie jest parametrem poza tworzeniem koła)
- Mamy umożliwiać 2 sposoby sortowania:
  - Po polu powierzchni
  - Po obwodzie
- Mamy umożliwiać zmianę porządku sortowania:
  - Rosnąco
  - Malejąco

Trójkąt dowolny

- Charakterystyka figury:
  - Długość boku a
  - Długość boku b
  - Długość boku c
  - Pole powierzchni
- Możliwe wejście:
  - 1 rodzaj wejścia: długości 3 boków
    - Możliwe, że to właściwy moment na wprowadzenie struktury klas trójkątów...