Daniel Górski
Korporacyjna Java
Wykład 10: Przetwarzanie asynchroniczne

Runnable

- Reprezentuje zadanie
- Interfejs funkcyjny:
  - void run()

Thread

- Klasa reprezentująca wątek
- public class Thread implements Runnable
  - Kilkadziesiąt metod
  - Ponad 2000 linii kodu
- Różne konstruktory, między innymi:
  - public Thread(Runnable target)
  - public Thread(Runnable target, String name)

Thread...

- Wywołanie synchroniczne:
  - public void run()
- Wywołanie asynchroniczne:
  - public synchronized void start()

Thread...

- Oczekiwanie na zakończenie:
  - public final void join() throws InterruptedException
  - public final synchronized void join(final long millis) throws InterruptedException

Thread...

- Co się dzieje w przypadku gdy stworzony wątek w programie kończy się błędem
  - Przy wywołaniu synchronicznym rzucany jest wyjątek w programie
  - Przy wywołaniu asynchronicznym z perspektywy programu wywołującego nie dzieje się nic

Callable

- Reprezentuje zadanie, które zwraca wynik i może rzucić wyjątek
- Interfejs funkcyjny
- public interface Callable<V>
  - V call() throws Exception
  - Wywołanie zwraca wynik więc jest wywołaniem synchronicznym

ExecutorService

- public interface ExecutorService extends Executor
  - Future<?> submit(Runnable task);
  - Future<T> submit(Runnable task, T result);
  - Future<T> submit(Callable<T> task);

ExecutorService...

- void shutdown();
- List<Runnable> shutdownNow();
- boolean isShutdown();
- boolean isTerminated();

Future

- Interfejs który reprezentuje wynik asynchronicznej operacji
- public interface Future<V>
  - V get() throws InterruptedException, ExecutionException;
  - V get(long timeout, TimeUnit unit) throws InterruptedException, ExecutionException, TimeoutException;

Future...

- boolean isDone();
- boolean cancel(boolean mayInterruptIfRunning);
- boolean isCancelled();

CompletableFuture

- Klasa umożliwiająca na różne działania na obliczeniach synchronicznych i asynchronicznych
- public class CompletableFuture<T> implements Future<T>, CompletionStage<T>
  - public static <U> CompletableFuture<U> supplyAsync(Supplier<U> supplier)
  - public static <U> CompletableFuture<U> supplyAsync(Supplier<U> supplier, Executor executor)

CompletableFuture...

- public static CompletableFuture<Void> allOf(CompletableFuture<?>... cfs)
- public static CompletableFuture<Object> anyOf(CompletableFuture<?>... cfs)

CompletableFuture...

- public <U> CompletableFuture<U> thenCompose(Function<? super T, ? extends CompletionStage<U>> fn)
- public <U> CompletableFuture<U> thenApply(Function<? super T,? extends U> fn)

InterruptedException

- Jest sygnałem dla wątku żeby kończyć pracę
- Może być złapany i zignorowany
- Poprawnym zachowaniem w większości sytuacji powinno być jego złapanie i szybkie uporządkowane zwinięcie zadania
- Metody na puli wątków: shutdownNow i na typie Future: cancel(true) są jedynie sugestiami aby procesy kończyły pracę

Pracownia: co wygląda dobrze

- Używanie zewnętrznych bibliotek
  - Po co wymyślać koło na nowo
  - Oczywiście bez przesady, bo rośnie złożoność "zależnościowa" projektu
- Realizacja wymagań z tygodnia w kilku pull-request'ach
  - Jeśli zmiany są w 2 – 3 plikach lub (dublowanie figur) w kilkunastu plikach, ale krótkie i powtarzalne to też się szybko przegląda
- Czytelna, dosyć rozbudowana struktura pakietów

Pracownia: co wygląda na ciężkie w utrzymaniu / niełądnie stylowo

- Niepełne testy: np tylko części figur
  - Utrudnia do wykrycie regresji
  - Dobrze, że się pojawiają, będzie to brane pod uwagę w ocenie końcowej
- Konwencje nazewnicze
  - Literówki
    - Zwłaszcza te, które np Intellij podkreśla

Pracownia

- Program ma umożliwić rozwiązywanie podstawowych zadań geometrycznych dla trapezu równoramiennego
- Wyświetlamy liczby do dwóch miejscu po przecinku (w zaokrągleniu)
  - Dotyczy to tylko i wyłącznie wyświetlania
  - Wprowadzać możemy liczby z większą liczbą miejsc po przecinku
  - Zaokrąglenie nie wpływa na przeliczenia
  - Możliwe, że w przyszłości będziemy chcieli zmienić format wyświetlania liczb

Pracownia

- Program ma umożliwić asynchroniczne zapisanie wszystkich przechowywanych figur do pliku
  - Opis każdej figury w jednej linii !

Trapez równoramienny

- Charakterystyka figury:
  - Podstawa 1
  - Podstawa 2
  - Ramię
  - Wysokość
  - Pole powierzchni

Trapez równoramienny...

- Możliwe wejście: (dowolna trójka)
  - Podstawa 1
  - Podstawa 2
  - Ramię
  - Wysokość
  - Pole powierzchni
- https://calcoolator.pl/trapez_rownoramienny_przeka