B-OOP 2025: Zadanie č. 3

Aktalizované - 27.03.2025

Stiahnite si zdrojové kódy ku zadaniu 3. Nájdete v nich priečinky src a test. Majú nasledujúcu štruktúru:

z3.zip

src

Main.java handler

 TaskHandler.java tasks

TaskIdGenerator.java AbstractTask.java SimpleTask.java RecurringTask.java

test

handler

 TaskHandlerTest.java tasks

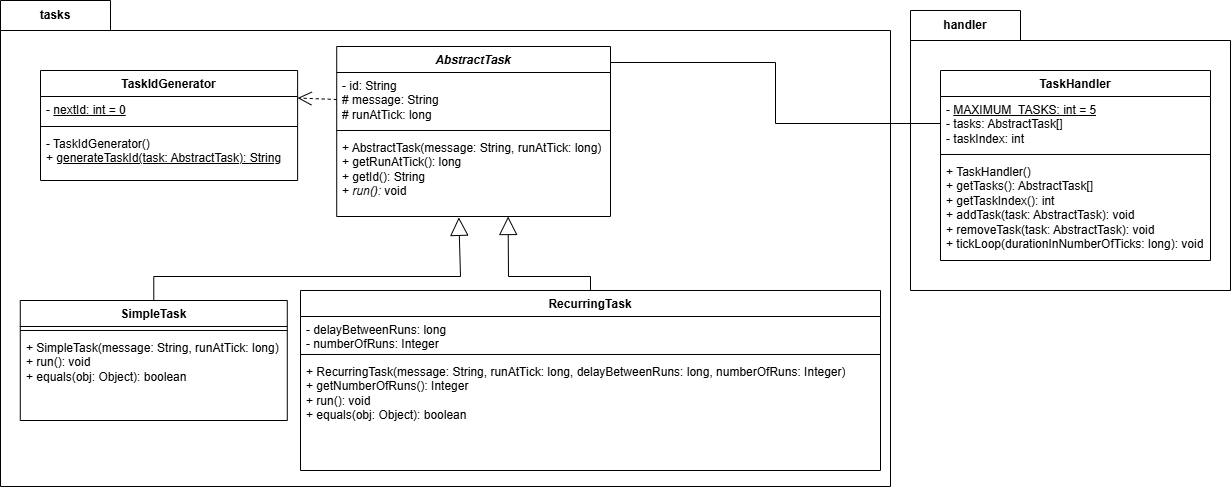
 RecurringTaskTest.java

Vašou úlohou je implementovať triedy:

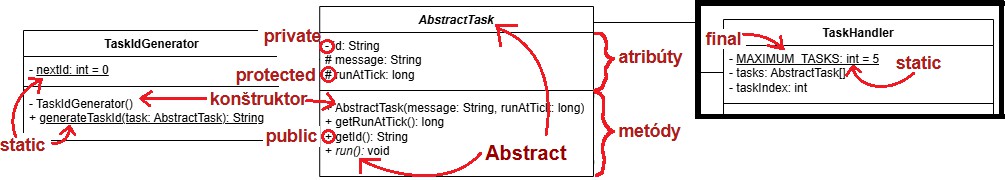
* TaskHandler.java,
* AbstractTask.java,
* SimpleTask.java,
* RecurringTask.java.

Trieda TaskIdGenerator.java už je implementovaná. Nemeňte ju!

V implementácii Vám pomôže UML diagram tried, ktorý vidíte na obrázku. S UML diagramom tried ste sa už stretli na seminároch, no ak potrebujete osviežiť pamäť, [na tejto stránke](https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/) nájdete relatívne pekný prehľad elementov. Pri implementácii musíte rešpektovať názvy metód a atribútov tried, viditeľnosť jednotlivých prvkov a vzťahy medzi triedami, ktoré sú vytýčené v tomto diagrame.



Pre tých z vás, ktorí preferujú prístup typu “rýchlokurz geniality” je k dispozícii nasledujúca anotovaná verzia časti diagramu:



Vašou úlohou bude implementovať úlohy, ktoré sa majú vykonať v určitom momente, ako aj mechanizmus na uloženie týchto úloh a ich rozbehnutie, keď nastane vhodný čas. V podstate ide o veľmi zjednodušenú verziu programu Windows Task Scheduler, resp. crontab.

Začnime s popisom triedy TaskHandler. Táto trieda ukladá úlohy a spúšťa ich vo vhodný čas. Obsahuje pole tasks, ktoré má kapacitu MAXIMUM\_TASKS. Môžete predpokladať, že MAXIMUM\_TASKS je statická konštanta, ktorá bude mať vždy hodnotu 5.

Metóda addTask(AbstractTask task) slúži na pridanie úlohy task do poľa tasks na prvú voľnú pozíciu. Prvú voľnú pozíciu označuje tasksIndex. Ak už nie je k dispozícii voľné miesto, tak sa úloha nepridá. Pred- stavme si, že sa zavolá metóda addTask trikrát. tasksIndex bude mať hodnotu 3 a v poli tasks ostane priestor na uloženie ďalších dvoch úloh.

Metóda removeTask(AbstractTask task) odstraňuje z poľa tasks úlohu task. Úlohy porovnávajte me- tódou equals, ktorú si sami implementujete v SimpleTask aj RecurringTask. Predstavme si, že v poli tasks máme aktuálne štyri úlohy s identifikátormi SimpleTask0, SimpleTask1, SimpleTask2 a RecurringTask3 v tomto poradí. Premenná tasksIndex má hodnotu 4. Prajeme si odstrániť úlohu SimpleTask2. Keď ju od- stránime, v poli ostanú úlohy s identifikátormi SimpleTask0, SimpleTask1 a RecurringTask3 a premenná tasksIndex má hodnotu 3. Úloha s identifikátorom RecurringTask3 sa v poli posunula o jednu pozíciu nižšie. Inak povedané, odstránili sme úlohu a posunuli sme obsah poľa tak, aby bolo možné opäť pridávať nové úlohy na jeho koniec.

Metóda tickLoop(durationInNumberOfTicks) bude simulovať priebeh času. Túto metódu si vieme pred- staviť ako akýsi metronóm, ktorý periodicky kliká. Ak aktuálne číslo kliku zodpovedá nejakej uloženej úlohe, tak spustí metódu run() tejto úlohy. Inak povedané, metóda tickLoop bude obsahovať premennú typu long, ktorú bude navyšovať v cykle. Táto premenná štartuje na hodnote 0 a navyšuje sa, pokým nedosiahne hodnotu durationInNumberOfTicks. Ak sa v danom momente má spustiť nejaká úloha, tak sa zavolá jej metóda run. Ak TaskHandler po ukončení metódy run vie určiť, že daná úloha už je ukončená, tak ju odstráni (tzn. zavolá sa metóda removeTask). Úloha je ukončená vtedy, keď ide o SimpleTask alebo RecurringTask, ktorý sa už nemá viac opakovať (viď popis úloh nižšie). Na začiatku iterácie musí metóda vypísať aktuálne číslo kliknutia a na konci kliknutia musí vypísať predelovač. Popísaný proces (vrátane formátu výpisov) zachytáva nasledujúci pseudokód:

tickLoop(durationInNumberOfTicks):

for i in {0, 1, ... durationInNumberOfTicks-1}: print("Current tick: " + i)

for task in tasks:

if task is scheduled at tick i: run task

if task is finished

remove task from tasks print("===")

Dodržte predpísané výpisy, budú súčasťou testovania!

**Pozor!** Môže byť lákavé overovať, či je daná úloha ukončená, pomocou operátora instanceof a následným pretypovaním. Toto zachytáva nasledujúci pseudokód:

// deciding if the task is finished after run() if task instanceof SimpleTask:

remove task from tasks

if task instanceof RecurringTask: t = (RecurringTask)task

if t.getNumberOfRuns() == 0: remove t from tasks

Z hľadiska princípov OOP nejde o vhodné riešenie (intuitívnym dôvodom je, že pre každý nový typ úlohy, ktorý by ste v budúcnosti pridali, by ste museli pridať ďalšiu vetvu a váš kód by sa postupne zmenil na špagetu). Preto vaša implementácia nesmie v tomto kontexte využiť operátor instanceof! Vymyslite iný spôsob. Napr., skúste sa zamyslieť, kedy má TaskHandler rozbehnúť úlohu a kedy vie, že ju už určite nerozbehne znova.

Trieda AbstractTask bude uchovávať správu (message) a čas/kliknutie (runAtTick), v ktorom sa má spustiť. Definuje abstraktnú metódu run, ktorú implementujú jej potomkovia. Implementujte parametrický konštruktor, ktorý bude akceptovať String message a long runAtTick. Pomocou nich nastaví korešpondujúce atribúty triedy. Správa (message) nesmie mať hodnotu null, ani nesmie byť prázdna. Číslo runAtTick nesmie byť záporné. Tieto podmienky overte v konštruktore pomocou assert. V konštruktore sa tiež musí nastaviť identifikátor úlohy. Identifikátor získate volaním statickej metódy generateTaskId(task) z triedy TaskIdGenerator. Keď chcete vytvoriť identifikátor pre aktuálnu triedu, volanie bude vyzerať takto:

id = TaskIdGenerator.generateTaskId(this);

Implementujte aj get metódy pre atribúty triedy.

Od triedy AbstractTask budú dediť triedy SimpleTask a RecurringTask. Obe tieto triedy musia imple- mentovať metódu equals. Trieda SimpleTask implementuje metódu run jednoducho tak, že vypíše správu na obrazovku v nasledujúcom formáte:

Task {id}: {message}

Trieda RecurringTask bude mať konštruktor, ktorý bude navyše akceptovať parameter delayBetweenRuns a parameter numberOfRuns. Oba tieto parametre použite na nastavenie príslušných atribútov triedy a definujte k nim get metódy. Parameter delayBetweenRuns musí byť kladný a parameter numberOfRuns musí byť buď kladný alebo musí byť null.

V rámci metódy run bude RecurringTask postupovať nasledujúcim spôsobom:

* Ak je numberOfRuns null, tak to znamená, že táto úloha sa má opakovať donekonečna (respektíve do doby, pokým neskončí cyklus v metóde tickLoop). Metóda run vypíše správu v správnom formáte a potom navýši runAtTick o hodnotu delayBetweenRuns.
* Ak je numberOfRuns kladné číslo, tak to znamená, že sa úloha môže opakovať už len numberOfRuns. Metóda run teda túto hodnotu dekrementuje, vypíše správu v správnom formáte a potom navýši runAtTick o hodnotu delayBetweenRuns.
* Inak metóda run nevykoná nič.

Správny formát výpisu je pre metódu run triedy RecurringTask rovnaký, ako v prípade SimpleTask. Uvažujme nasledujúci pseudokód:

t0 = SimpleTask("uvarii obed", 2)

t1 = SimpleTask("nakúpii potraviny", 1)

t2 = RecurringTask("umyi si zuby", 1, 2, 2) th = TaskHandler()

th.addTask(t0) th.addTask(t1) th.addTask(t2) th.tickLoop(6)

Z tohto pseudokódu očakávame nasledujúci výstup:

Current tick: 0

===

Current tick: 1

Task SimpleTask1: nakúpii potraviny Task RecurringTask2: umyi si zuby

===

Current tick: 2

Task SimpleTask0: uvarii obed

===

Current tick: 3

Task RecurringTask2: umyi si zuby

===

Current tick: 4

===

Current tick: 5

===

V priečinku test nájdete podmnožinu testov, ktoré budú použité na hodnotenie vášho zadania. Môžete si nimi čiastočne otestovať korektnosť vašej implementácie.

**Pozor!** Je dôležité, aby ste dodržali predpísané názvy tried a metód, ktoré vyplývajú z popisu zadania a z priloženého UML diagramu. Nemeňte štruktúru projektu! Ak to považujete za potrebné, môžete si v jednotlivých triedach vytvoriť dodatočné pomocné metódy.

V AIS je vytvorené miesto odovzdania OOP - Zadanie 3 do 2025-04-02T23:59:00+01:00. Do tohto miesta odovzdania nahráte **zip** archív (to znamená nie rar, ani tar...), ani žiadny iný formát než zip. Váš archív bude obsahovať priečinok src s nasledujúcou štruktúrou (za predpokladu, že ste nevytvorili dodatočné triedy):

archiv.zip  src

handler

 TaskHandler.java tasks

AbstractTask.java RecurringTask.java SimpleTask.java TaskIdGenerator.java

Vaše zadanie bude hodnotené pomocou automatizovaných testov, aby sme si overili, či splňuje všetky náleži- tosti. Zadania, ktoré splnia uvedené náležitosti, a prejdú všetkými automatizovanými testami, budú ohodnotené 1 bodom.