Programowanie Współbieżne 2023/2024

Zadanie zaliczeniowe ze współbieżności w języku Java

Systemy składowania danych (ang. *storage systems*) muszą jednocześnie spełniać wiele wymagań dotyczących w szczególności wydajności operacji dostępu do danych, wykorzystania pojemności nośników danych oraz odporności na awarie urządzeń będących tymi nośnikami. W tym celu często przenoszą fragmenty danych pomiędzy poszczególnymi urządzeniami. Twoim zadaniem będzie implementacja w języku Java mechanizmów koordynujących współbieżne operacje takiego przenoszenia zgodnie z poniższymi wymaganiami. Do implementacji rozwiązania należy wykorzystać załączony szablon.

Specyfikacja

W naszym modelu systemu dane są grupowane w komponenty i w takich jednostkach przechowywane na urządzeniach. Zarówno każde urządzenie jak i każdy komponent danych mają przypisany niezmienny i unikalny w ramach systemu identyfikator (obiekt klasy odpowiednio `cp2023.base.DeviceId` oraz `cp2023.base.ComponentId`). Każde urządzenie ma ponadto określoną pojemność, to jest maksymalną liczbę komponentów, które może przechowywać w dowolnym momencie. Przypisaniem komponentów do urządzeń zarządza system (obiekt implementujący interfejs `cp2023.base.StorageSystem`, przedstawiony poniżej):

```
public interface StorageSystem {

    void execute(ComponentTransfer transfer) throws TransferException;

}
```

Dokładniej, każdy istniejący w systemie komponent znajduje się na dokładnie jednym urządzeniu, chyba że użytkownik systemu zlecił transfer tego komponentu na inne urządzenie (wołając metodę `execute` ww. klasy `StorageSystem` i przekazując jej jako parametr obiekt implementujący interfejs `cp2023.base.ComponentTransfer` reprezentujący zlecany transfer).

```
public interface ComponentTransfer {

    public ComponentId getComponentId();

    public DeviceId getSourceDeviceId();

    public DeviceId getDestinationDeviceId();

    public void prepare();

    public void perform();

}
```

Transfer komponentu jest zlecany także, gdy użytkownik chce dodać nowy komponent do systemu (w takim przypadku metoda `getSourceDeviceId` obiektu transferu zwraca wartość `null`) lub usunąć istniejący komponent z systemu (w takim przypadku, symetrycznie, metoda `getDestinationDeviceId` obiektu transferu zwraca wartość `null`). Innymi słowy, pojedynczy transfer reprezentuje jedną z trzech dostępnych operacji na komponencie:

- *dodanie* nowego komponentu na urządzenie systemu (`getSourceDeviceId` obiektu transferu zwraca `null` a `getDestinationDeviceId` nie-`null` oznaczający identyfikator urządzenia, na którym ma znaleźć się dodawany komponent),
- *przeniesienie* istniejącego komponentu pomiędzy urządzeniami systemu (`getSourceDeviceId` oraz `getDestinationDeviceId` obie zwracają nie-`null`-e reprezentujące identyfikatory odpowiednio aktualnego urządzenia, na którym znajduje się komponent, oraz docelowego urządzenia, na którym komponent powinien się znaleźć po transferze),
- *usunięcie* istniejącego komponentu z urządzenia a tym samym – systemu (`getSourceDeviceId` zwraca nie-`null` oznaczający identyfikator urządzenia, na którym znajduje się komponent, a `getDestinationDeviceId` zwraca `null`).

Zlecanie trzech ww. typów operacji przez użytkownika jest poza kontrolą implementowanego rozwiązania. Zadaniem Twojego rozwiązania jest natomiast przeprowadzanie zlecanych transferów w sposób synchroniczny (tj. jeśli zlecany transfer jest poprawny, metoda `execute` wywołana na obiekcie implementującym `StorageSystem` z transferem reprezentowanym jako parametr implementujący interfejs `ComponentTransfer` nie może zakończyć swojego działania dopóki ten transfer się nie zakończy). Jako że wiele różnych operacji może być zlecanych jednocześnie przez użytkownika, implementowany przez Ciebie system musi zapewnić ich koordynację według następujących reguł.

W dowolnym momencie dla danego komponentu może być zgłoszony co najwyżej jeden transfer. Dopóki ten transfer się nie zakończy, komponent nazwiemy *transferowanym* zaś każdy kolejny transfer zgłaszany dla tego komponentu należy traktować jako niepoprawny.

Sam transfer komponentu jest dwuetapowy i może trwać dłuższy czas (zwłaszcza jego drugi etap). Rozpoczęcie transferu polega na jego przygotowaniu (tj. wywołaniu metody `prepare` na obiekcie o interfejsie `ComponentTransfer` reprezentującym transfer). Dopiero po takim przygotowaniu mogą być przesyłane dane stanowiące komponent (co odbywa się poprzez wywołanie metody `perform` dla ww. obiektu). Gdy dane zostaną przesłane (tj. metoda `perform` zakończy swoje działanie), transfer się kończy. Obie ww. metody muszą być wykonane w kontekście wątku zlecającego transfer.

Bezpieczeństwo

Transfer może być poprawny albo niepoprawny. Poniższe wymagania bezpieczeństwa dotyczą transferów poprawnych. Obsługa transferów niepoprawnych jest z kolei opisana w dalszej sekcji.

Jeśli transfer reprezentuje operację usunięcia komponentu, jego rozpoczęcie jest *dozwolone* bez żadnych dodatkowych warunków wstępnych. W przeciwnym przypadku, rozpoczęcie transferu jest dozwolone o ile na urządzeniu docelowym jest miejsce na transferowany komponent, w szczególności miejsce to właśnie jest lub będzie zwalniane, może można je dzięki temu zarezerwować. Dokładniej, rozpoczęcie transferu reprezentującego przesunięcie lub dodanie komponentu *Cx* jest *dozwolone*, jeśli zachodzi któryś z poniższych warunków:

- Na urządzeniu docelowym transferu jest wolne miejsce na komponent, które nie zostało zarezerwowane przez system na inny komponent, który jest/będzie przenoszony/dodawany na to urządzenie.
- Na urządzeniu docelowym znajduje się komponent *Cy* transferowany z tego urządzenia, którego transfer się rozpoczął lub jego rozpoczęcie jest dozwolone, oraz miejsce zwalniane przez ten komponent nie zostało zarezerwowane przez system na inny komponent.
- Komponent *Cx* należy do pewnego zbioru transferowanych komponentów takiego, że urządzeniem docelowym każdego komponentu ze zbioru jest urządzenie, na którym znajduje się dokładnie jeden inny komponent ze zbioru, oraz miejsce żadnego z komponentów ze zbioru nie zostało zarezerwowane na komponent spoza zbioru.

Jeśli transfer komponentu *Cx* jest dozwolony, ale odbywać się ma w miejsce jeszcze zajmowane przez inny transferowany komponent *Cy* (dwa ostatnie przypadki powyżej), to drugi etap transferu komponentu *Cx* (tj. wywołanie funkcji `perform` dla tego transferu) nie może rozpocząć się przed zakończeniem pierwszego etapu transferu komponentu *Cy* (tj. wywołaniem funkcji `prepare` dla tego transferu).

Oczywiście, jeśli transfer komponentu jest niedozwolony, to nie może on być rozpoczęty (tj. ani funkcja `prepare`, ani funkcja `perform` nie mogą zostać wywołane na obiekcie reprezentującym ten transfer).

Twoje rozwiązanie powinno bezwzględnie zapewniać wszystkie powyższe warunki bezpieczeństwa.

Żywotność

Jeśli natomiast chodzi o żywotność, to transfer (zarówno jego faza `prepare` jak i `perform`) powinien się rozpoczynać tak szybko, jak tylko jest on dozwolony i pozostałe wymagania bezpieczeństwa zostaną spełnione. W przypadku, gdy kilka transferów konkuruje o miejsce na urządzeniu, spomiędzy tych z nich, które są dozwolone, Twój algorytm powinien lokalnie priorytetyzować transfery czekające dłużej na to urządzenie. Globalnie może to potencjalnie prowadzić do zagłodzenia pewnych transferów (zachęcamy do wymyślenia scenariusza takiej sytuacji). Rozwiązanie tego problemu jest oczywiście możliwe do implementacji, ale komplikuje kod ponad to, co chcielibyśmy od Państwa wymagać. Nie należy go więc implementować, zwłaszcza że w praktyce użytkownik systemu widząc, że transfer na jakieś urządzenie długo nie może się wykonać, mógłby wytransferować inne komponenty z tego urządzenia.

Obsługa błędów

Wreszcie, zaproponowane rozwiązanie powinno sprawdzać, czy zlecony przez użytkownika transfer jest niepoprawny (co powinno skutkować podniesieniem przez metodę `execute` interfejsu `StorageSystem` odpowiedniego wyjątku dziedziczącego po klasie `cp2023.exceptions.TransferException`). Zgodnie z wcześniejszymi wyjaśnieniami transfer jest niepoprawny, jeśli zachodzi co najmniej jeden z poniższych warunków:

- transfer nie reprezentuje żadnej z trzech dostępnych operacji na komponentach lub nie wskazuje żadnego komponentu (wyjątek `IllegalTransferType`);
- urządzenie wskazane przez transfer jako źródłowe lub docelowe nie istnieje w systemie (wyjątek `DeviceDoesNotExist`);
- komponent o identyfikatorze równym dodawanemu w ramach transferu komponentowi już istnieje w systemie (wyjątek `ComponentAlreadyExists`);
- komponent o identyfikatorze równym usuwanemu lub przesuwanemu w ramach transferu komponentowi nie istnieje w systemie lub znajduje się na innym urządzeniu niż wskazane przez transfer (wyjątek `ComponentDoesNotExist`);
- komponent, którego dotyczy transfer, znajduje się już na urządzeniu wskazanym przez transfer jako docelowe (wyjątek `ComponentDoesNotNeedTransfer`);
- komponent, którego dotyczy transfer, jest jeszcze transferowany (wyjątek `ComponentIsBeingOperatedOn`).

W rozwiązaniu można przyjąć dowolną sensowną kolejność sprawdzania tych warunków.

Wymagania

Twoim zadaniem jest zaimplementowanie systemu według powyższej specyfikacji i dostarczonego szablonu przy wykorzystaniu mechanizmów współbieżności języka Java 17. Twój kod źródłowy powinien być napisany w zgodzie z dobrymi praktykami programistycznymi. Rozwiązania oparte na aktywnym lub półaktywnym (np. `sleep`, `yield` lub inne metody wykorzystujące ograniczenia czasowe) oczekiwaniu nie otrzymają żadnych punktów.

Dla uproszczenia rozwiązań zakładamy, że wątki zgłaszające transfery nie są nigdy przerywane (tj. nigdy nie jest wołana dla nich metoda `interrupt` klasy `Thread`). Reakcją na pojawienie się kontrolowanego wyjątku wynikającego z takiego przerwania (np. `InterruptedException` lub `BrokenBarrierException`) powinno być podniesienie wyjątku niekontrolowanego w następujący sposób: `throw new RuntimeException("panic: unexpected thread interruption");`.

Szczegółowe dalsze wymagania formalne są następujące.

1. Nie możesz w żaden sposób zmieniać zawartości pakietów `cp2023.base`, `cp2023.demo` oraz `cp2023.exceptions`.
2. Klasy implementujące rozwiązanie możesz dodawać jedynie w pakiecie `cp2023.solution`, ale nie możesz tworzyć w tym pakiecie żadnych podpakietów.
3. Twoja implementacja nie może tworzyć żadnych wątków.
4. Twoja implementacja nie powinna wypisywać niczego na standardowe wyjście (`System.out`) i standardowe wyjście diagnostyczne (`System.err`).
5. W klasie `cp2023.solution.StorageSystemFactory` musisz dodać treść metody `newSystem`, która będzie wykorzystywana do instancjonowania zaimplementowanego przez Ciebie systemu. Każde wywołanie tej metody powinno tworzyć nowy obiekt systemu. Wiele obiektów systemu powinno być w stanie działać w tym samym czasie. Nie wolno natomiast w żaden sposób zmieniać sygnatury tej metody ani nazwy klasy czy jej lokalizacji. Jeśli konfiguracja systemu dostarczona jako argumenty tej metody jest niepoprawna (np. jakiś komponent jest przypisany do urządzenia bez podanej pojemności lub liczba komponentów przypisanych do jakiegoś urządzenia przekracza jego pojemność), to metoda powinna podnieść wyjątek `java.lang.IllegalArgumentException` z odpowiednim komunikatem tekstowym.
6. Możesz stworzyć sobie własne pakiety do testów, np. `cp2023.tests`, ale te pakiety będą ignorowane przy testowaniu przez nas, więc w szczególności kod Twojego systemu nie może od nich zależeć.
7. W plikach źródłowych Javy nie możesz używać nieanglojęzycznych znaków (w szczególności polskich znaków).
8. Twoje rozwiązanie powinno składać się z jednego pliku `ab123456.zip`, gdzie `ab123456` należy zastąpić swoim loginem z maszyny `students.mimuw.edu.pl` (będącym zwykle konkatenacją inicjałów i numeru indeksu). Plik ten musi mieć taką samą strukturę, jak szablon, to jest musi zawierać jedynie katalog `cp2023` reprezentujący pakiet o tej samej nazwie, który zawiera katalogi odpowiednich podpakietów, co najmniej `base`, `demo`, `exceptions` i `solution`, które z kolei zawierają odpowiednie pliki źródłowe (`*.java`).
9. Twoje rozwiązanie musi kompilować się na maszynie `students.mimuw.edu.pl` poleceniem `javac cp2023/base/*.java cp2023/exceptions/*.java cp2023/solution/*.java cp2023/demo/*.java`.
10. W Twoim rozwiązaniu musi działać program demonstracyjny, wywoływany poleceniem `java cp2023.demo.TransferBurst`, to jest nie może on zgłaszać żadnych wyjątków.

Rozwiązania niespełniające któregokolwiek z powyższych wymagań nie będą sprawdzane i automatycznie dostaną 0 punktów.

Prosimy o zrozumienie! Będziemy mieli do sprawdzenia nawet ponad 160 rozwiązań. Rozwiązania te będą w pierwszej fazie testowane automatycznie. Gdybyśmy musieli każde rozwiązanie w jakikolwiek sposób poprawiać, aby uruchomienie testów było możliwe, stracilibyśmy niepotrzebnie mnóstwo czasu. Dlatego też zapewnienie zgodności z powyższymi wymaganiami jest po Państwa stronie.

Aby w tym celu dać Państwu więcej informacji co do samej procedury testowania, to dla każdego rozwiązania przebiegać będzie ona z grubsza następująco.

1. Archiwum ZIP z rozwiązaniem zostanie rozpakowane do dedykowanego katalogu głównego na maszynie `students` (lub kompatybilnej jeśli chodzi o wersję Javy).
2. Z katalogu tego zostaną usunięte wszystkie pliki i podkatalogi za wyjątkiem podkatalogu `cp2023/solution` i jego zawartości.
3. Z podkatalogu `cp2023/solution` zostaną usunięte wszystkie pliki (i podkatalogi) za wyjątkiem plików `*.java`.
4. Do katalogu głównego zostaną skopiowane katalogi `cp2023/base`, `cp2023/demo` oraz `cp2023/exceptions` z dostarczanego szablonu, aby pliki z interfejsami oraz aplikacja demonstracyjna były w wersji oryginalnej, oraz katalogi z kodem naszych testów.
5. Wszystko zostanie skompilowane.
6. Uruchomiona zostanie aplikacja demonstracyjna, aby sprawdzić, czy działa.
7. Jeśli rozpakowanie archiwum, kompilacja lub uruchomienie aplikacji demonstracyjnej się nie powiedzie, to rozwiązanie otrzymuje automatycznie 0 punktów. W przeciwnym przypadku, w ramach testowania, uruchamiane będą kolejne aplikacje testowe oraz ewentualne dodatkowe programy (np. weryfikacja anty-plagiatowa).

Wszelkie pytania i uwagi powinny być kierowane do Konrada Iwanickiego poprzez forum Moodle dedykowane zadaniu. Przed wysłaniem pytania, proszę sprawdzić na forum, czy ktoś wcześniej nie zadał podobnego.

Powodzenia!