Logowanie do serwisu | Użytkownik: AA Aplaceholder | Kod: | Login...

Java

Strona z zadaniami + interfejs systemu wysyłania rozwiązań i pytań

Zadania z terminem oddawania rozwiązań ustalonym na 15 lutego 2019 są zadaniami dla sesji poprawkowej.

Uwagi dotyczące wysyłania rozwiązań dostępne są przez ten link. Proszę się z nimi zapoznać!

Instrukcja uruchamiania testu:

W jednym katalogu umieszczamy własne rozwiązanie, pobrane pliki testu i plik junit-platform-console-standalone-1.3.1.jar.
Kompilujemy własne rozwiązanie
Kompilujemy test za pomocą polecenia:

```
javac -cp junit-platform-console-standalone-1.3.1.jar:. PMO_Test.java
```

Uruchamiamy test za pomocą polecenia:

```
java -jar junit-platform-console-standalone-1.3.1.jar -cp . -c PMO_Test
```

Wysyłka kodu uwierzytelniającego

Aby uzyskac **kod do autoryzacji** prosze wybrac wlasne imie i nazwisko z listy i nacisnac przycisk "Wyslij>>>"

AA Aplaceholder | Wyslij>>>

Zadanie 03 termin VI. Do zdobycia maksymalnie: 1pkt.

Traktat morski

Podpisany został traktat morski. Na mocy tego traktatu nie można posiadać zbyt silnej floty. Ustalone zostały limity jednostek różnego rozmiaru. Potrzebne jest narzędzie, które wymusi egzekwowanie postanowień traktatu i nie dopuści do wytworzenia zbyt dużej liczby okrętów.

Klasa `Ship`

W tym zadaniu pojawia się klasa `Ship`. Reprezentuje ona okręt o danym rozmiarze, przechowuje jego stan (sprawny/wrak) i, co ciekawe, sama zarządza liczbą własnych instancji.

Limity liczby okrętów ustalane są za pośrednictwem klasy `ShipSizeLimit`. Rozmiar okrętu to liczba całkowita z zakresu od 1 do getNumberOfSizes() włącznie. Obiekt klasy `ShipSizeLimit` dostarcza wiadomości o limicie jednostek określonego rozmiaru.

Ograniczenie ilości okrętów.

Ograniczeniu podlegają wyłącznie okręty sprawne. Te, które są w stanie "wrak" nie są uwzględnianie w obliczeniach. Ograniczenie wprowadzana jest za pomocą metody `getShip` to ona ma udostępniać każdy obiekt

reprezentujący okręt lub zwracać null jeśli udostępnienie kolejnego okrętu miałoby doprowadzić do przekroczenia limitu.

Podpowiedzi:

Aby użytkownik sam nie mógł tworzyć obiektów klasy `Ship` klasa ta **musi** posiadać prywatny konstruktor. Tylko wtedy o liczbie utworzonych okrętów zadecyduje metoda `getShip`.

Aby sprawdzić stan udostępnionych wcześniej przez metodę `getShip` okrętów, należy referencję do nich zapamiętać.

Ograniczenia/umowy

Możemy się umówić, że wszystkie limity będą nieujemne.

Jeśli limity są nieustalone lub, gdy okrętu o podanym rozmiarze nie uwzględniono w limitach, to metoda `getShip` zwraca null. Np. nałożone zostało ograniczenie dla okrętów o rozmiarach od 1 do 3 włączenie; użytkownik, który prosi o okręt o rozmiarze -1 lub 4 zawsze otrzymuje null.

Rozwiązując zadanie wolno zmodyfikować wyłącznie klasę `Ship`.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-11-16 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 9, a liczba odpowiedzi to: 9 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 04 termin VI. Do zdobycia maksymalnie: 1pkt.

Idea zadania

W tym zadaniu rozwiążemy problem ochrony obiektu przed niechciamymi modyfikacjami jego stanu. Ochrona będzie wielopoziomowa, a każdy z poziomów używać będzie hasła. Po nałożeniu ochrony stan obiektu może być tylko odczytywany. Zmiany staną się na powrót możliwe dopiero po odblokowaniu dostępu. Aby całkowicie usunąć ochronę przez zmianami stanu obiektu, należy podac prawidłowo wszystkie hasła i zrobić to we właściwej kolejności.

Dostarczanie rozwiązania:

Proszę zaimplementować w klasie `BetterPoint` funkcjonalności zadeklarowane w abstrakcyjnej klasie `AbstractBetterPoint`.

Uwagi:

W klasie `BetterPoint` może (ale nie musi) pojawić się metoda main. Ja jej używać nie będę.

Ochrona ma działać na poziomie obiektu. Jedne obiekt klasy `BetterPoint` może zablokować zmiany stanu, a inny, w tym samym czasie, może na takie zmiany się zgadzać.

Należy uwzględnić fakt, że hasła mogę się powtarzać!

Klasa `BetterPoint` musi posiadać konstruktor bezparametrowy - to ten konstruktor zostanie użyty do utworzenia

obietu tej klasy.

W przypadku dostarczenia do metody `lock` zamiast hasła wartości `null` - należy uznać, że hasło nie zostało dostarczone i poziom ochrony obiektu nie ulega zmienie.

W przypadku pojawienia się `null` w przypadku metody `unlock` poziom ochrony także nie ulega zmianie.

Gwarancja

Metoda setDimensions zostanie na pewno wykonana zaraz po utworzeniu obiektu i będzie to zarazem pierwsze jak i ostatnie jej użycie dla każdego obiektu.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy testu

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-11-20 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 5, a liczba odpowiedzi to: 5 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 05 termin VI. Do zdobycia maksymalnie: 1pkt.

Tym razem zadanie polegać będzie na opracowaniu systemu odnajdującego połączenia autobusowe. System ma uwzględniać możliwość pokonania zadanej trasy z określoną liczbą przesiadek.

W zadaniu tym poznajemy siłę interfejsów - dzięki nim wszystkie szczegóły komunikacji pomiędzy obiektami mogą być ustalone, choć nie powstała żadna konkretna klasa.

Rozróżnianie przystanków i autobusów

Aby umożliwić rozróżnianie przystanków i autobusów ustalamy, że

- Przystanki rozpoznawane są poprzez unikalne nazwy.
- Autobusy maja przyznane unikalne numery identyfikacyjne.

Wyszukiwanie połączeń

W przypadku poszukiwania połączenia z przesiadkami nie należy uwzględniać pętli - czyli przesiadka nie może doprowadzić, do dotarcia do przystanku, który już wcześniej został odwiedzony.

Ponadto, nie wolno zaproponować przesiadki powodującej powrót do użytego wcześniej autobusu (tak np. nie wolno: Autobus nr 135, Autobus nr 200 i ponownie Autobus nr 135). Innymi słowy, nie należy generować sztucznych rozwiązań, które w rzeczywistości nie byłyby użyteczne dla pasażera.

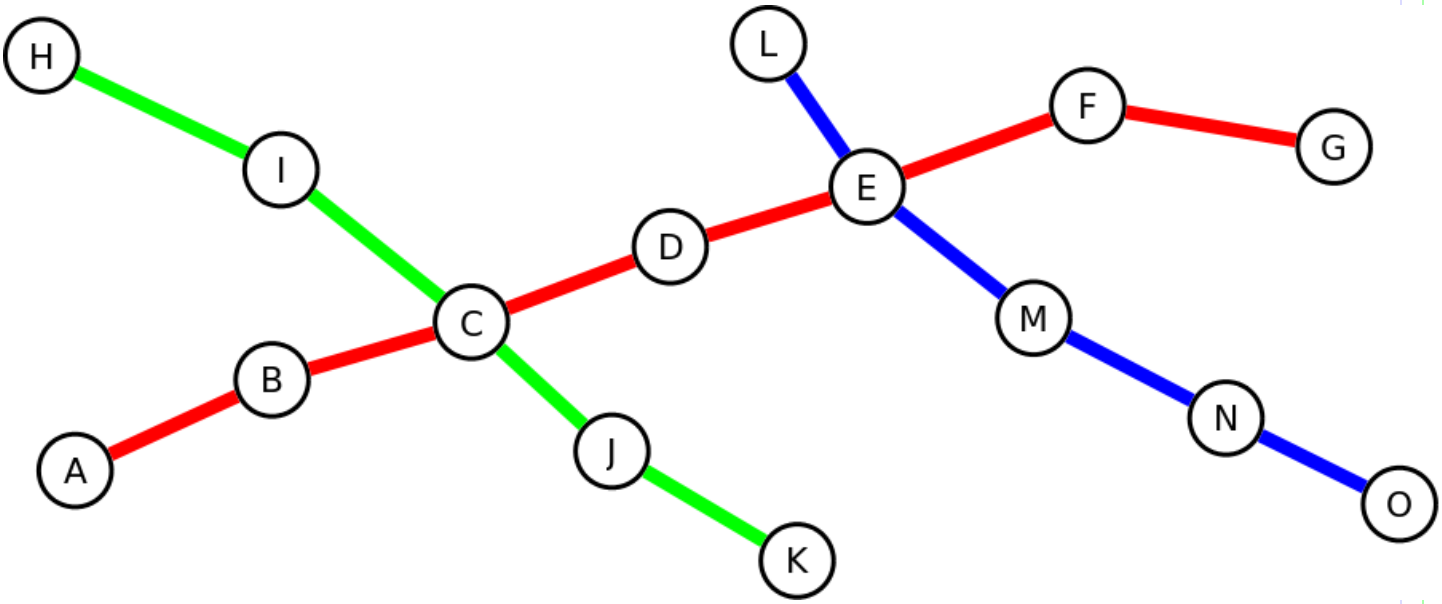
Zakładamy, że trasa przemierzana w obie strony dotyczy tego samego zbioru przystanków - zmienia się tylko ich kolejność. Czyli jeśli np. przemierzamy przystanki A-B-C-D, to w drugą stronę D-C-B-A.

Ustalenia

Dla ułatwnienia zakładamy, że w argumentach wywołania metod nie pojawi się null. Można jeszcze założyć, że nie będzie linii, która obsługiwana jest przez więcej niż jeden autobus. Same trasy będą utworzone poprawnie i pętli w nich nie będzie.

Uwaga: może się tak zdarzyć, że nie będzie istniało żadne rozwiązanie - np. trasa będzie wymagała użycia innej niż dozwolona liczby przesiadek albo dany przystanek nie będzie obsługiwała żadna linia itd.

Przykład:



Mamy 3 line autobusowe:

- Linia A-B-C-D-E-F-G - autobus nr 100
- Linia H-I-C-J-K - autobus nr 200
- Linia L-E-M-N-O - autobus nr 300

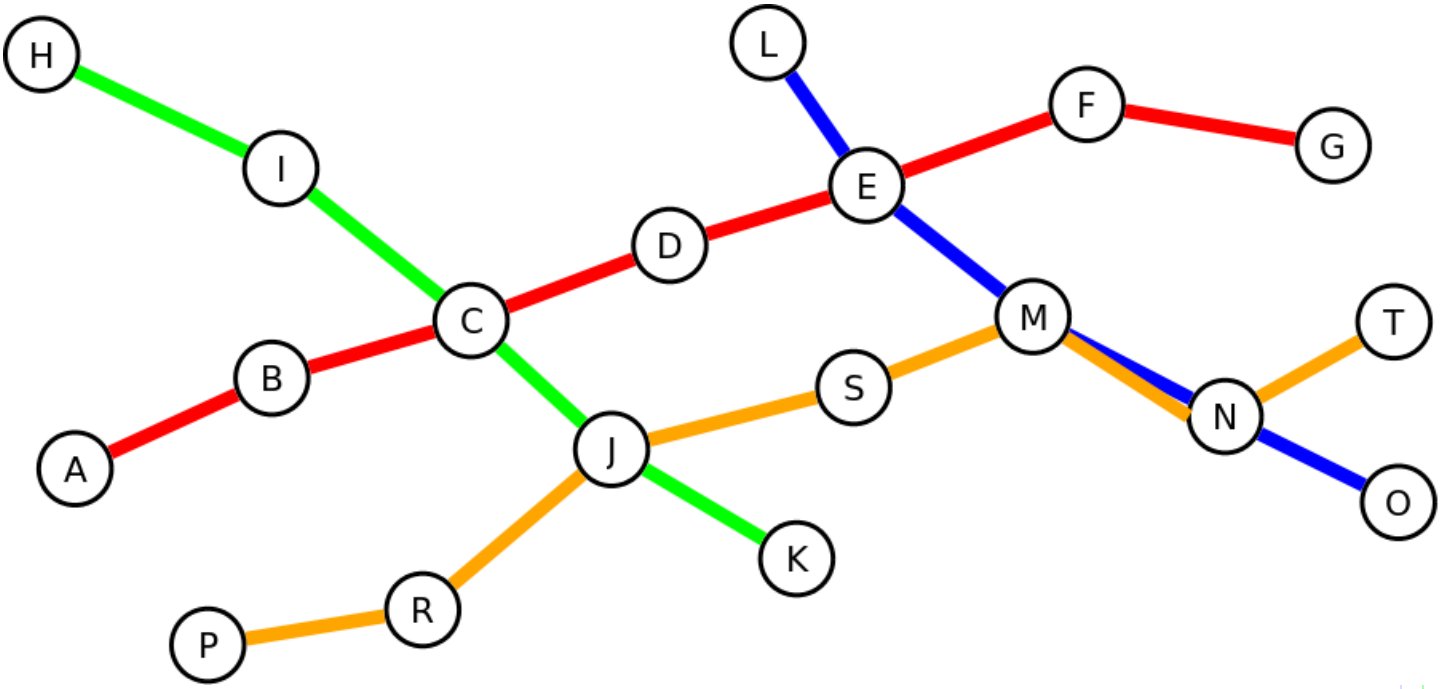
Dla połączenia z przystanku I do N i 2 przesiadek istnieje jedno rozwiązanie. Trasa ma 6 przystanków:

- Przystanek nr 0 - I - autobus 200
- Przystanek nr 1 - C - autobus 100
- Przystanek nr 2 - D - autobus 100
- Przystanek nr 3 - E - autobus 300
- Przystanek nr 4 - M - autobus 300
- Przystanek nr 5 - N - autobus 300

Pytania.

Pojawiły się pytania o możliwość wystąpienia pętli. W obrębie jednej linii pętli nigdy nie będzie, ale różne linie mogą doprowadzić do istnienia układu przystanków, które łączy kilka tras.

Jeden autobus będzie obsługiwał jedną linię. Jedna linia będzie obsługiwana przez jeden autobus, choć mogą pojawić się przystanki (M i N), pomiędzy którymi można przemieścić się za pomocą różnych autobusów.



Generalnie, całość pomyślana jest jako dająca się rozwiązać metodą "brutalnej siły" tj. wychodząc z pierwszego przystanku i idąc we wszystkich możliwych kierunkach. Można także zrobić prostą rzecz: tabelę możliwych zmian linii autobusowej.

```
zielona -> czerwona(C), pomarańczowa(J)
czerwona -> zielona(C), niebieska(E)
pomarańczowa -> zielona(J), niebieska(M,N)
niebieska -> czerwona(E), pomarańczowa(M,N)
```

Jeśli szukam trasy od I do O to:

```
I jest na trasie zielonej
O jest na niebieskiej

Czyli
- bez przesiadki się nie uda
- tabelka powyżej pokazuje, że brak bezpośredniego przejścia z trasy
  zielonej na niebieską -> z jedną przesiadką się nie uda
- idąc od zielonej mam linię czerwoną i pomarańczową
  z czerwonej mam niebieską
  z pomarańczowej mam niebieską
  czyli przy 2 przesiadkach się uda. Wystarczy sprawdzić, na których
  przystankach można się przesiąść... i wygenerować rozwiązania...
```

Rozwiązanie

Rozwiązaniem ma być klasa `PathFinder`. Klasa ma implementować interfejs `PathFinderInterface`. Musi on posiadać konstruktor bezparametrowy - to ten konstruktor zostanie użyty do utworzenia obiektu tej klasy.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy testu

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-12-10 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 8, a liczba odpowiedzi to: 8 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 06 termin VI. Do zdobycia maksymalnie: 1pkt.

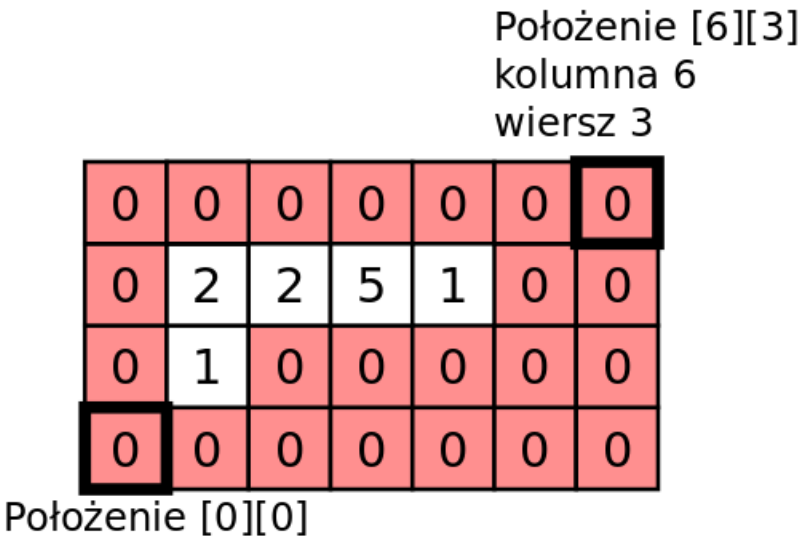
Zadanie polega na wyznaczeniu optymalnej trasy prowadzącej przez bardzo prosty plan miasta.

Mapa miasta

Plan miasta to w tym zadaniu zwykła, dwuwymiarowa tablica liczb całkowitych. Pozycje w tablicy, do których wpisano liczbę niewiększą od 0 (mniejszą lub równą 0) są nieprzejezdne. Pozycje zawierające liczby dodatnie reprezentują ulice i informują o czasie potrzebnym na przebycie danej komórki tablicy (danego fragmentu ulicy).

Ze względu na sposób reprezentacji mapy, zakręty są możliwe tylko pod kątem prostym. Dodatkowo, dla uproszczenia, drogi będą mieć szerokość jednej komórki.

Umawiamy się, że mapa zorientowana jest tak, że pierwsza współrzędna tablicy to numerKolumny, druga współrzędna to numerWiersza. Położenie o współrzędnych [0][0] znajduje się w lewym, dolnym rogu mapy.



Pierwszy i ostatni wiersz oraz pierwsza i ostatnia kolumna mapy wypełniona będzie zerami.

Rozwiązanie zadania

Zadanie polega na implementacji interfejsu `PathFinderInterface` za pomocą **typu wyliczeniowego** o nazwie `PathFinderEnum`.

Typ wyliczeniowy **musi** posiadać dwie stałe: **LEFT_HAND_TRAFFIC** oraz **RIGHT_HAND_TRAFFIC** - odpowiadające za wskazanie ruchu lewo- i prawo-stronnego.

Podstawowa struktura kodu wygląda następująco:

```
public enum PathFinderEnum implements PathFinderInterface {
    LEFT_HAND_TRAFFIC,
    RIGHT_HAND_TRAFFIC;
}
```

Wynik pracy metod ma być uzależniony od typu ruchu oraz danych wpisanych w tablicy reprezentującej mapę miasta. W przypadku gdy dostępne są alternatywne trasy dla metod `getEasiestRoute`, `getShortestRoute` oraz `getFastestRoute`, należy jako rozwiązanie podać tą, w której na skrzyżowaniach wykonywany jest wybór kierunku wg. poniższych preferencji.

Priorytetyzacja kierunku ruchu

Dla `LEFT_HAND_TRAFFIC` priorytety kierunków na skrzyżowaniu są następujące:

- Na wprost
- Skręt w lewo
- Skręt w prawo.

Dla `RIGHT_HAND_TRAFFIC` priorytety kierunków na skrzyżowaniu są następujące:

- Na wprost
- Skręt w prawo.
- Skręt w lewo

Aby nie było wątpliwości, na znajdującym się poniżej obrazku pokazane zostały dwa fragmenty mapy miasta - po stronie lewej jest tylko jedna ścieżka - nie ma tam żadnego skrzyżowania, nie ma zatem żadnego wyboru trasy, nie ma więc znaczenia jak układają się zakręty. Po stronie prawej można odszukać dwa skrzyżowania, czyli takie miejsca na mapie, gdzie droga się rozwidla i możliwa jest więcej niż jedna kontynuacja trasy.

0	0	0	0	0	0	0
0	0	0	1	10	1	0
0	1	1	2	0	1	0
0	1	0	0	5	1	0
0	2	0	0	1	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	1
0	1	0	0	0	0
0	1	0	0	0	0
0	1	1	1	1	0
0	0	0	0	0	0

UWAGA: w podawanych powyżej prorytetach chodzi o względny kierunek ruchu czyli jadąc "na północ" kierunek "na wprost" to kontynuacja ruchu "na północ".

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-12-20 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 4, a liczba odpowiedzi to: 4 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 07 termin VI. Do zdobycia maksymalnie: 1pkt.

Zadanie polega na utworzeniu klasy odpowiedzialnej za budowę figury składającej się z wielu punktów.

W tym zadaniu pojawia się nowsza wersja klasy Point. W stosunku do poprzedniej wersji została ona wzbogacona w metody equals oraz hashCode. Obecnie używana klasa Point używa zmiennych typu całkowitego, dzięki temu nie będzie problemu z jednoznacznym ustaleniem, które punkty się ze sobą pokrywają. Ponadto, liczba wymiarów została ustalona i nie będzie mogła ulec zmianie.

Budowa figury sprowadza się do realizacji prostych działań: dodawania punktów i ich usuwania. Aby można było prowadzić/usunąć punkty w określonej kolejności dodane zostały metody pozwalające na podanie punktu odniesienia, po którym lub przed którym ma nastąpić zlecona akcja.

Dodatkowo, w zadaniu wprowadzony został mechanizm undo/redo. Polecenia dodające/usuwające punkty należy zapamiętać, aby gdy pojawi się wywołanie metody undo, możliwe było przywrócenie poprzedniego stanu listy

punktów. Zakładamy, że nie ma ograniczenia w ilości poleceń, które można anulować. Oznacza to, że polecenia undo można wywołać dowolną liczbę razy i w szczególności doprowadzić do stanu początkowego, w którym żadnego punktu jeszcze nie było.

Na działanie metody undo nie mają wpływu operacje odczytu czy operacje, które nie udało się zrealizować. Nie zmieniają one bowiem stanu obiektu.

Dla uproszczenia zakładamy, że metoda redo służy do ewentualnego przywrócenia tylko tej zmiany usuniętej przez undo, która została wykonana bezpośrednio przed użyciem redo. Wykonanie redo kilka razy pod rząd jest zabronione i nie będzie realizowane.

Oczywiście para undo+redo prowadzi do tego samego stanu obiektu, który był przed wykonaniem tej pary poleceń.

W przypadku wykonania zbyt dużej liczby poleceń undo i przywrócenia początkowego stanu obiektu (brak punktów), dalsze wykonania undo już niczego w stanie obiektu nie zmieniają.

Na zakończenie mały przykład. Niech obiekt reprezentujący naszą figurę przechodzi kolejno przez stany S1, S2, S3 i S4. Jeśli wykonamy undo(), undo() to obiekt powinien znajdować się w stanie S2 (czyli usunięto operacje powodujące przejście ze stanu S2 do S3 i dalej z S3 do S4). Jeśli wykonam undo(), undo() i redo(), to obiekt powinien być w stanie S3. Para undo() + redo() powoduje, że obiekt nadal jest w stanie S4.

Rozwiązanie

Rozwiązaniem zadania ma być klasa `GeometricShape`. Klasa ma implementować interfejs `GeometricShapeInterface`. Klasa musi posiadać konstruktor bezparametrowy.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-12-17 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 9, a liczba odpowiedzi to: 9 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 08 termin VI. Do zdobycia maksymalnie: 1pkt.

To zadanie jest odmianą poprzedniego zadania, do którego wprowadzamy możliwość obsługi błędów za pomocą wyjątków. Dodatkowo, w jednej z metod pojawi się użycie klasy `Optional` wprowadzonej w Java 8.

W tym zadaniu ukryty jest pewien ciekawy problem: klasa Point pozwala na ustalenie liczby wymiarów w momencie konstrukcji obiektu, ale nie udostępnia żadnej metody, która udostępnia tą informację na zewnątrz... Ale od czego są wyjątki - za ich pomocą możliwe jest wydobycie ten informacji. UWAGA: użycie refleksji jest zabronione!

Rozwiązanie

Rozwiązaniem zadania ma być klasa `GeometricShape`. Klasa ma implementować interfejs `GeometricShapeInterface`. Klasa musi posiadać konstruktor bezparametrowy.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

- Kod źródłowy

Punktacja: do 1pkt. | Zadawanie pytań do: 2018-12-31 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 18, a liczba odpowiedzi to: 18 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 09 termin VI. Do zdobycia maksymalnie: 1pkt.

W tym zadaniu w celu przyspieszenia pewnych prostych obliczeń należy zastosować wątki.

W zadaniu tym pojawia się klasa generująca obiekty zgodne z typem `PointInterface`. Interfejs `PointInterface` to bardzo uproszczona wersja interfejsu obsługi punktu - pozwala ona wyłącznie na pobranie położenia punktu. Dla uproszczenia zakładamy, że w tym zadaniu wszystkie punkty będą umieszczone w dwuwymiarowej przestrzeni (czyli na płaszczyźnie). Dodatkowo, wszystkie punkty będą leżeć w obszarze ograniczonym punktami o współrzędnych (0,0) i (PointInterface.MAX_POSITION,PointInterface.MAX_POSITION) włącznie, czyli np. położenie (4,PointInterface.MAX_POSITION) może zostać wygenerowane.

Należy napisać program, który pozwoli na **współbieżne** wyliczenie histogramu rozkładu punktów oraz położenia środka geometrycznego.

Rozkład punktów to nic innego jak liczba punktów, które mają daną współrzędną. Środek geometryczny to środek masy, gdy masa wszystkich punktów jest taka sama Link do wikipedii

W pseudokodzie i **sekwencyjnie** rzecz wygląda tak:

```
utwórz tablicę histogram
utwórz wektor o rozmiarze 2
liczbaPunktów = 0
etykieta:
    punkt = generator.getPoint()
    liczbaPunktów++
    położenie = punkt.getPosition()
    histogram[ położenie[0] ][ położenie[1] ]++
    wektor[ 0 ] += położenie[ 0 ];
    wektor[ 1 ] += położenie[ 1 ];
    goto etykieta
```

Położenie środka geometrycznego to (wektor[0]/liczbaPunktów, wektor[1]/liczbaPunktów).

Wymagane cechy programu to:

- możliwość wstrzymywania obliczeń na żądanie i ich dalszej kontynuacji
- współbieżne prowadzenie obliczeń za pomocą wskazanej ilości wątków
- wynik, pomimo współbieżnego prowadzenia obliczeń musi być dokładnie taki sam, jak w przypadku obliczeń wykonanych sekwencyjnie (tj. z użyciem jednego wątku)
- w czasie, gdy obliczenia są wstrzymane, program nie może obciążać procesora.
- wykonanie metody getPoint *po zakończeniu* metody suspend traktowane jest jako błąd krytyczny i nie może się zdarzyć.
- metody setPointGenerator, setNumberOfThreads, start, suspendCalculations, continueCalculations oraz metody get nie mogą zablokować pracy wątku użytkownika - mają one wykonać swoją pracę i się zakończyć.

Pobieranie wyniku działania programu.

Wynik obliczeń będzie pobierany z programu pomiędzy wcześniejszym wykonaniem metody suspendCalculations a uruchomieniem metody continueCalculations. Czyli tak:

```
suspendCalculations
getGeometricCenter
getHistogram
continueCalculations
```

Uruchomienie obliczeń

I jeszcze na koniec gwarantowana procedura uruchomienia obliczeń, czyli w jakiej kolejności wykonane zostaną metody uruchamiające obliczenia:

```
setNumberOfThreads
setPointGenerator
start
```

Rozwiązanie

Rozwiązaniem zadania ma być klasa `ParallelCalculations`. Klasa ma implementować interfejs `ParallelCalculationsInterface`. Klasa musi posiadać konstruktor bezparametrowy.

Linki

- Dokumentacja klas
- Kod źródłowy

Test

Typowe problemy z programami:

- **Nie liczą równolegle - obliczenia nie są prowadzone równolegle za pomocą wielu wątków lecz sekwencyjnie poprzez metody getHistogram i getGeometricCenter. Te metody nie mają prowadzić obliczeń! Mają zwracać wynik (w getGeometricCenter można wynik przeskalować na podstawie sumy odebranych punktów, ale nic więcej). Liczyć mają wątki.**
- **Nawet danych z generatora nie pobierają równolegle - nie sposób wykryć aby wątki cokolwiek równolegle robiły. Dane z generatora pobiera w danej chwili tylko jeden wątek - oznacza to, że nie ma żadnych szans na przyspieszenie obliczeń.**

Rozwiązanie zadania nie polega tylko na równoległym pobieraniu danych z generatora, lecz na równoległym liczeniu!

Test do zadania został poprawiony - ten nowy powinien lepiej wykrywać błędy w liczeniu histogramu.

- Kod źródłowy Plik PMO_Test.java.old zawiera stary kod testu, PMO_Test.java zawiera kod poprawiony.

Test cd.

W nadesłanych rozwiązaniach pojawia się wiele różnego rodzaju błędów. Aby je wykryć test został zmodyfikowany. Link poniżej.

- Kod źródłowy Nowsza wersja testu do zadania 9

Typowe usterki to:

- Używanie interrupt w trakcie pracy mojego kodu. Wywołania metody generującej punkty są celowo spowolnione za pomocą sleep - tak ma być! Proszę nie wpływać na wykonywanie mojego kodu. Nowa wersja testu wykrywa coś takiego.
- Używanie metod uznawanych za niebezpieczne i złe (suspend/resume). Prowadzą one (czasami) do uzyskiwania błędnych wyników. Programy, które używały suspend, ale testy zaliczyły otrzymały punkty, choć jest to proszenie się o kłopoty. Dlaczego to takie złe pokażę na ćwiczeniach.
- Używanie tego samego, współdzielonego pola przez wiele wątków i jego modyfikacja. Przypominam: współbieżnie wolno tylko dane odczytywać, nie wolno doprowadzić do sytuacji, gdy ta sama wielkość jest jednocześnie używana przez wiele wątków a co najmniej jeden ją modyfikuje. Po to jest synchronized aby tego czegoś używać.
- Używanie synchronized ale z różnymi obiektami używanymi jako zatrzask (to coś co pozwala tylko jednemu z wątków wejść do danego bloku kodu). Jeśli każdy wątek używa innego obiektu, to każdy z nich robi co chce.

```
synchronized( zatrzask ) {
```

```
  to coś strasznie ważnego jest zmieniane/czytane
}
zatrzask - MUSI być wspólny, inaczej w tym bloku kodu każdy z wątków
może sobie poczynać wedle własnej woli. Przypominam problem z zakupami
i wspólnym kontem! Ktoś chce mieć debet?
```

- Używanie pól bez synchronized i volatile - o tym więcej na ćwiczeniach.
- Część programów wykonuje wątkami pojedynczy odczyt z generatora i... wątki umierają. W efekcie z generatora pobierane jest tylko tyle punktów ile jest wątków. Cała praca to raptem 4/5 odczytanych z generatora punktów.
- Zgodność kodu - używane przeze mnie oprogramowanie zasygnalizowało zgodność kodu w przypadku kilku prac, ale ponieważ i tak są one do poprawy więc tylko zwracam na tą rzecz uwagę. Szkoda tracić punkty...

Co dalej?

- Zadanie będzie mieć pięć terminów. Termin III i IV za 1.5pkt. Termin V za 1.0pkt.
- W teście w terminie III ulegnie zmianie linia numer 93 na sleep(100) - da to więcej czasu na zakończenie pracy wątków. Limity czasu wykonania metod suspendCalculations, continueCalculations, setNumberOfThreads wzrosną do 50msec. Limity czasu na wykonanie metod getHistogram oraz getGeometricCenter zostaną kosmetycznie zmienione z 7msec na 10msec. Program ma przede wszystkim prawidłowo liczyć z użyciem wielu wątków.
- Projekty wgrane w dniu 11 stycznia zostaną wstępnie sprawdzone przed zakończeniem terminu III. Jeśli projekt przejdzie test, to wynik zostanie umieszczony w USOS jako ostateczna punktacja za to zadanie.
- Terminy IV i V tego zadania nie będą liczyć się jako "koła ratunkowe".

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-01-10 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 4, a liczba odpowiedzi to: 4 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 10 termin VI. Do zdobycia maksymalnie: 1pkt.

Termin II dla tego zadania będzie w styczniu i punktacja nadal będzie wynosić do 2pkt.

W tym zadaniu należy stworzyć program, który odczyta dane ze skompresowanego pliku i wyświetli na ich podstawie animację.

Kodowanie danych

Dane o położeniu cząstek zakodowane są w bardzo prosty sposób. Pierwszy wiersz zawiera informację o liczbie cząstek. Drugi wiersz to informacja o liczbie wymiarów - na pewno wpisane będzie tam 2. Kolejne wiersze (ich liczba nie jest podana) zawierają informacje o położeniach cząstek na kolejnych klatkach. Pierwsza liczba to numer klatki, kolejne to pozycje x,y kolejnych cząstek. W katalogu z danymi jest plik small.txt - za jego pomocą można łatwo zrozumieć ideę kodowania danych.

Dane znajdować się będą w skompresowanym pliku. Jego dekompresja ma być realizowana na bieżąco (w locie).

Opuszczanie klatek

Aby przyspieszyć wyświetlaną animację (skrócić czas) program ma być wyposażony w dwa przyciski kontrolujące liczbę klatek, które są omijane. Parametr speedUP o wartości 5 oznacza, że prezentowana jest co piąta klatka animacji. Parametr może przyjąć wartość 0 - oznacza to wstrzymanie animacji. Dalsze wciskanie przycisku zmniejszającego liczbę opuszczanych klatek nie może doprowadzić do uzyskania ujemnej wartości speedUP.

Aktualna wartość parametru speedUP ma być wyświetlana w oknie animacji.

Początkowa wartość parametru speedUP to 1, czyli program ma wyświetlać wszystkie klatki animacji, które zapisane są w pliku.

Uruchomienie programu.

Po uruchomieniu programu powinno zostać wyświetlone okno pozwalające na wybór pliku. Plik będzie mieć rozszerzenie txt.gz - czyli będzie to skompresowany za pomocą programu gzip plik tekstowy i taki rodzaj plików musi dać się wybrać. Plik niekoniecznie będzie znajdować się w katalogu, w którym program zostanie uruchomiony. Przydatny link: filechooser

Koniec danych

Gdy dane zapisane w pliku skończą się animację należy rozpocząć ponownie od klatki numer 0.

Orientacja obrazu i skalowanie

Umawiamy się, że położenie 0,0 znajduje się lewym dolnym rogu obrazka.

Wyświetlane muszą być wszystkie punkty - nie mogą one "uciekać" poza ekran. Skalowanie także musi działać poprawnie - rozmiar okna musi dać się zmieniać, a wyświetlane obrazy muszą zajmować całą dostępną przestrzeń (oczywiście, z uwzględnieniem przycisków).

Rozwiązanie

Rozwiązaniem ma być program, który uruchomi klasa o nazwie `Start`. Klasę tą należy dostarczyć. W pliku z rozwiązaniem mogą znajdować się inne klasy, ale tylko klasa `Start` może być publiczna.

Linki

- Dane
- Tak ma działać animacja (plik został już wybrany)

Test

- Do danych dodany został plik zawierający 10 punktów ułożonych w strzałkę. Ma ona wskazywać prawy-górny róg obrazka.
- Strzałka ma wypełniać całą dostępną przestrzeń okienka i prawidłowo reagować na jego rozmiar.

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-01-08 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 3, a liczba odpowiedzi to: 3 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 12 termin VI. Do zdobycia maksymalnie: 1pkt.

Ostatnie zadanie związane jest z użyciem mechanizmów refleksji.

Zadanie polega na stworzeniu klasy `Starter` implementującej interfejs `java.util.function.Consumer<String>`.

Zadaniem metody `accept`, do której przekazana zostanie nazwa pewnej klasy, jest:

- utworzenie obiektu tej klasy za pomocą konstruktora domyślnego
- odnalezienie metod publicznych i uruchomienie tych, które:
  - nie przyjmują żadnego parametru lub pojedynczy parametr typu String
  - oznaczone zostały za pomocą adnotacji `@MethodToStart`.
  - nie zostały oznaczone za pomocą adnotacji `@MethodDisabled`

Adnotacja `@MethodToStart` posiada pole `value` typu int. Wskazuje ono ile razy ma zostać uruchomiona metoda, która jest oznaczona tą adnotacją.

Jeśli metoda przeznaczona do uruchomienia została dodatkowo oznaczona za pomocą adnotacji `@StringParameter` to oznacza, że jeśli wymaga przekazania argumentu typu String, to należy uruchomić ją przekazując do niej wartość wpisaną w polu `value` tej adnotacji.

Uwaga: możliwa jest sytuacja gdy adnotacja `@StringParameter` się pojawi jednak metoda będzie bezparametrowa albo nie przeznaczona do uruchomienia.

Linki

- Kod źródłowy

Test

- Kod źródłowy

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-01-12 | Nadsyłanie rozwiązań do: 2019-02-15

Do zadania zgłoszono pytań: 2, a liczba odpowiedzi to: 2 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 13 termin VII. Do zdobycia maksymalnie: 1pkt.

Niniejszy projekt zainspirowany jest rzeczywistym problemem: zbieramy dane, w których pojawiają się dane osobowe, czyli dane, które podlegają szczególnej ochronie prawnej. Cóż, trzeba chronić, to chronimy. Tylko, że pewnego pięknego dnia chcemy zebrane dane przekazać do przetworzenia przez firmę zewnętrzną. Jak to zrobić, aby danych osobowych nie ujawnić, a jednocześnie umożliwić ich badanie?

I tu z pomocą przychodzi Państwa program. Dokonuje on anonimizacji - zamienia dane osobowe (u nas po prostu numery telefonów) na losowe ciągi znaków. Ale zmienia tak, aby nie zgubić tego co ważne, czyli:

- ten sam numer telefonu jest zawsze wymieniony na ten sam zestaw znaków
- każdy nowy, inny niż wcześniejsze, numer telefonu jest zamieniony na nowy, unikalny ciąg znaków (ciągi znaków przypisane różnym numerom nie mogą być takie same).

Na przykład:

```
12 123456 -> abba
12 214364 -> abbb
12 123456 -> abba
12 654321 -> babb
12 654399 -> bbbb
12 123456 -> abba
12 999999 -> aaaa
12 654321 -> babb
21 654321 -> baaa
itd.
```

A teraz o kwestiach technicznych. Klasa będąca rozwiązaniem zadania musi nazywać się `Anonymizer` i implementować interfejs `AnonymizerInterface`. Ponadto, do generowania ciągów znaków, które mają posłużyć do zamiany numeru telefonu, koniecznie musi używać metody `generate` klasy `Helper` - metoda ta generuje losowe ciągi znaków, które mogą się jednak powtarzać i muszą Państwo sobie z tym problemem poradzić! Z generatora wolno pobrać tylko tyle ciągów ile jest niezbędnych do rozwiązania zadania.

Oczywiście, w trakcie testów dane zostaną tak dobrane, aby ilość unikalnych ciągów, które `Helper` potrafi wygenerować była wystarczająca.

Dodatkowo do zadania dostarczam klasę `Contact` zgodną z `PhoneInterface`, może się ona Państwu przydać w trakcie testowania kodu.

Linki:

- Dokumentacja: -tutaj-
- Kod źródłowy: -tutaj-

Test:

- Kod źródłowy: -tutaj-

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-02-09 | Nadsyłanie rozwiązań do: 2019-02-19

Do zadania zgłoszono pytań: 0, a liczba odpowiedzi to: 0 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 14 termin VII. Do zdobycia maksymalnie: 1pkt.

W zadaniu tym trzeba będzie zmierzyć się z implementacją uniwersalnego mechanizmu do przechowywania różnych różności (obiektów).

Użytkownik powierzać będzie obiekty do przechowania. Będą one różnych typów. Obiekty będą rozróżniane za pomocą klucza typu `String`. System ma pozwolić na pobieranie zapisanych obiektów w formie referencji do Object, ale i nie tylko! Np. użytkownik zapisuje wartość `1.0f` typu `Float`. Może ją pobrać jako `Float` i `String`. A ponadto, po włączeniu wymuszonej konwersji można ją pobrać jako np. `Integer` czy `Long`. Podobnie użytkownik może zapisać daną typu `String` a oczekiwać pobrania jej jako wartości typu np. `Float`, ale będzie to możliwe dopiero po włączeniu wymuszonej konwersji.

Dla uproszczenia zakładamy, że użytkownik nigdzie nie prześle `null`.

Proszę uwzględnić możliwość pojawienia się danych typu:

- Byte
- Short
- Integer
- Long
- Float
- Double
- Boolean - uwaga: true kodowane przy konwersji jako wartość 1, false kodowane przy konwersji jako 0.0
- String
- Inne - typy wprost niekonwertowalne do typów numerycznych, ale możliwość konwersji to String za pomocą `toString` i dalej **potencjalnie** do oczekiwanego typu numerycznego.

Proszę o dostarczenie rozwiązania w postaci klasy o nazwie `UniversalStorage`, która będzie implementować `UniversalStorageInterface`.

Linki:

- Dokumentacja: -tutaj-
- Kod źródłowy: -tutaj-

Test:

- Kod źródłowy: -tutaj-

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-02-09 | Nadsyłanie rozwiązań do: 2019-02-19

Do zadania zgłoszono pytań: 0, a liczba odpowiedzi to: 0 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.

Zadanie 15 termin VII. Do zdobycia maksymalnie: 1pkt.

Proszę zapoznać się z zasadami gry w Otello/Reversi. Są one proste i krótkie. Pomocne linki to:

- Reversi
- Otellomania.pl
- Opis reversi po angielsku

Celem jest napisanie kodu, który przeanalizuje otrzymany stan planszy do gry w Otello i przekaże swojemu użytkownikowi tablicę możliwych (legalnych) kontynuacji gry, czyli pozycji na planszy, w których gracz może legalnie umieścić swój pionek. Informacje maja zaczynać się od propozycji najlepszych (najwięcej zdobytych pionów przeciwnika) po punktowo najsłabsze.

Rozwiązanie ma się znajdować w klasie o nazwie `BetterOtelloHelper` i implementować interfejs `BetterOtelloHelperInterface`. Klasa ma posiadać konstruktor bezparametrowy, który zostanie użyty w testach do stworzenia obiektu dostarczonej klasy.

Uwaga: testy będą obejmować również takie stany gry, w których jeden z graczy może wykonać kilka posunięć pod rząd (blokuje ruch przeciwnika).

Linki:

- Dokumentacja tutaj
- Kod źródłowy interfejsu BetterOtelloHelperInterface

Test:

- Kod źródłowy tutaj

Punktacja: do 1pkt. | Zadawanie pytań do: 2019-02-09 | Nadsyłanie rozwiązań do: 2019-02-19

Do zadania zgłoszono pytań: 0, a liczba odpowiedzi to: 0 | LINK do bazy pytań/odpowiedzi.

Brak możliwości dostarczania rozwiązań dla tego zadania.