	<p style="text-align: center;">Metody programowania 2021/2022 Pociągi</p>	<p style="text-align: center;">P_04</p>
---	---	---

Opis

Do pewnej stacji przyjechały pociągi, by przejść od dawna zapowiadaną modernizację. Każdy pociąg ma swoją *nazwę* i *listę wagonów*, zawierającą co najmniej jeden wagon. *Lista pociągów* nie zawiera duplikatów. W trakcie przeglądu technicznego pociągów okazało się, że mogą być konieczne następujące operacje na pociągach:

- a. **New T1 W** – tworzy nowy pociąg o nazwie *T1* z jednym wagonem o nazwie *W* i wstawia go do listy pociągów.
- b. **InsertFirst T1 W** – wstawia wagon o nazwie *W* na początek pociągu o nazwie *T1*
- c. **InsertLast T1 W** – wstawia wagon o nazwie *W* na koniec pociągu o nazwie *T1*
- d. **Display T1** – wypisuje listę wagonów pociągu o nazwie *T1* poczynawszy od pierwszego wagonu
- e. **Trains** – wypisuje aktualną listę pociągów.
- f. **Reverse T1** – odwraca kolejność wagonów w pociągu o nazwie *T1*
- g. **Union T1 T2** – dołącza pociąg o nazwie o nazwie *T2* na koniec pociągu o nazwie *T1* i usuwa pociąg *T2* z listy pociągów
- h. **DelFirst T1 T2** – usuwa pierwszy wagon z pociągu o nazwie *T1* i tworzy z niego nowy pociąg o nazwie *T2* i jeśli to był jedyny wagon w *T1* to *T1* przestaje istnieć (jest usuwany z listy pociągów).
- i. **Dellast T1 T2** – usuwa ostatni wagon z pociągu o nazwie *T1* i tworzy z niego nowy pociąg o nazwie *T2*, przy czym, jeśli to był jedyny wagon w *T1* to *T1* przestaje istnieć (jest usuwany z listy pociągów).

Twoim zadaniem jest przeprowadzić symulację powyższych operacji używając efektywnej implementacji związanej poniższych struktur danych:

- jednostronnej listy dwukierunkowej, cyklicznej, której referencja first wskazuje pierwszy element listy dla reprezentacji listy wagonów danego pociągu.
- listy pojedynczej, której referencja trains wskazuje pierwszy element listy dla reprezentacji listy pociągów.

Wejście

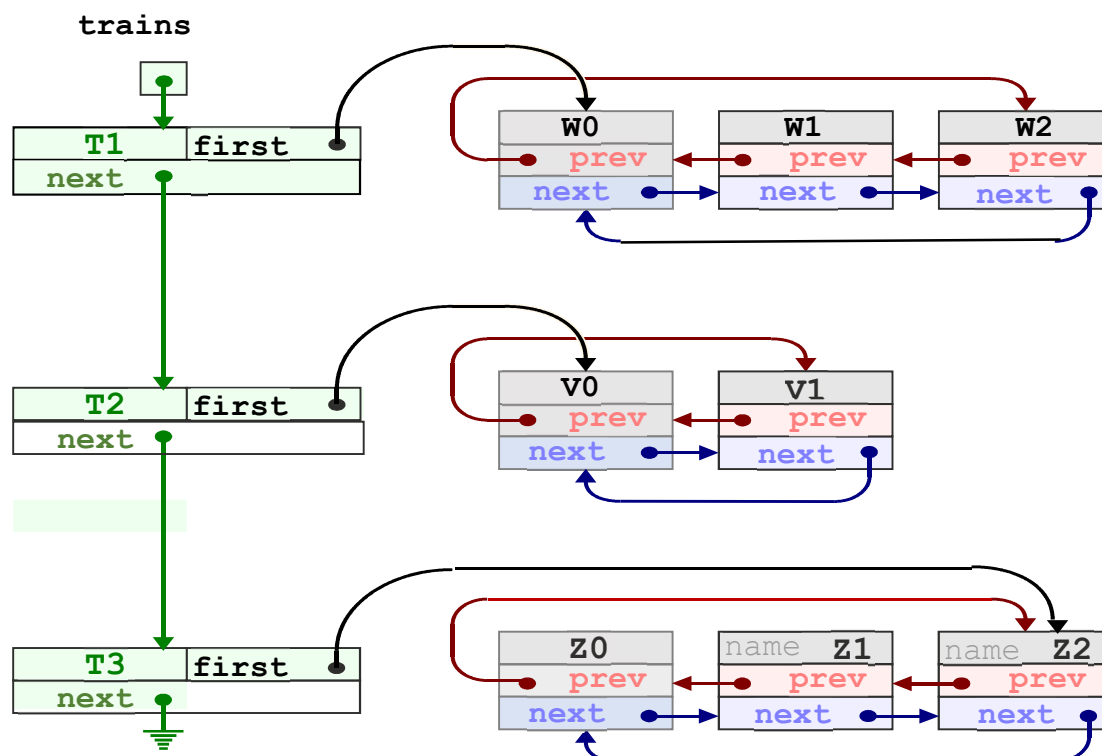
Pierwsza linia wejścia zawiera liczbę całkowitą *z* – liczbę zestawów danych, których opisy występują kolejno po sobie.

Pierwsza linia każdego zestawu zawiera liczbę całkowitą *n* ($1 \leq n \leq 10^6$) będącą liczbą operacji, przy czym każda operacja umieszczona jest w osobnej linii i zawiera od jednego do trzech słów.

Pierwsze słowo jest *nazwą* operacji i jest zawsze zakończone spacją, zaś pozostałe słowa, jeśli występują są jej parametrami, oddzielonymi pojedynczą spacją. Każda operacja kończy się znakiem nowej linii.

Nazwy pociągów i *wagonów* spełniają wymogi identyfikatorów stosowanych w programowaniu w języku Java, zaś *nazwy operacji* są traktowane jako *słowa zastrzeżone*.

Przykładową listę trzech pociągów ilustruje poniższy rysunek:




Wyjście

- W reakcji na operację **Display** wypisz opis pociągów o zadanych nazwach. Opis pociągu rozpoczyna się jego *nazwą*, zakończoną znakiem ':' i spacją, po której występują nazwy wagonów rozdzielanych znakiem spacji w kolejności od pierwszego do ostatniego wagonu.
- W reakcji na operację **Trains** wypisz aktualną listę pociągów biorących udział w modernizacji. Opis listy pociągów rozpoczyna się słowem **Trains**, zakończonym znakiem ':' i spacją, następnie występują nazwy pociągów rozdzielanych znakiem spacji w kolejności od pierwszego do ostatniego pociągu na liście.
- W obu przypadkach wyświetlane listy kończą się znakiem nowej linii.

Wymagania implementacyjne

- Jedynym możliwym importem jest **java.util.Scanner**. W szczególności zabronione są zarówno w całości jak i w jakiegokolwiek części importy **java.util.AbstractList** oraz **java.awt.List**.
- Deklaracje klas *lista pociągów* i *lista wagonów* muszą być zgodne z podanym wyżej opisem, w przeciwnym przypadku program zostanie odrzucony pomimo akceptacji przez BaCę.

	<h1 style="text-align: center;">Metody programowania 2021/2022</h1> <h2 style="text-align: center;">Pociągi</h2>	<h1 style="text-align: center;">P_04</h1>
---	--	---

- Wszystkie wymienione operacje, poza **Display** oraz **Trains** muszą używać jak najmniej pamięci i działać w czasie $O(1)$ nie licząc pomocniczych operacji związanych z wyszukiwaniem zadanego pociągu.
- Wszystkie pomocnicze operacje jak np. wstawianie nowego pociągu, wyszukiwanie zadanego pociągu lub usuwanie pociągu zaimplementuj tak, aby zawierały minimalną liczbę przeglądów list.
- Elementy klasy **listy pociągów** nie mogą mieć dodatkowych pól postaci boolean reserved.
- Program powinien sprawdzać czy wszystkie operacje są sensowne np.
 - a. nie tworzą duplikatów pociągów, w przeciwnym przypadku program wypisuje komunikat: **Train name already exists**
 - b. nie odwołują się do pociągów - nieistniejących na liście, w przeciwnym przypadku wypisuje komunikaty: **Train name does not exist**
- W przypadku operacji **DelFirst(T1, T2)** lub **DellLast(T1, T2)** program działa w tzw. "krótkim obiegu". To znaczy wykonanie **usuń** jest realizowane tylko gdy istnieje pociąg **T1** i nie istnieje **T2**.

Przykład danych

Wejście:	Wynik:
1	T1: W1
13	Train T1 already exists
New T1 W1	T1: W1 W2
Display T1	T1: W1
New T1 W0	T2: W2
InsertLast T1 W2	T3: W1
Display T1	T2: W2
DellLast T1 T2	Train T1 does not exist
Display T1	Trains: T3 T2
Display T2	
DelFirst T1 T3	
Display T3	
Display T2	
Display T1	
Trains	