Programowanie I

* algorytmy i struktury danych
* złożoność obliczeniowa
* sortowanie

Mateusz „Divo” Górny
szkolenia@divo.net.pl

Struktura danych

Struktura danych (ang. data structure) - sposób uporządkowania informacji w komputerze. Na strukturach danych operują algorytmy.

Podczas implementacji programu programista często staje przed wyborem między różnymi strukturami danych, aby uzyskać pożądany efekt. Odpowiedni wybór może zmniejszyć złożoność obliczeniową.
Wyróżniamy kilka struktur danych:

Lista

Jedną z najprostszych, dynamicznych struktur danych wykorzystywanych w programowaniu są listy. Podobnie jak tablice pozwalają przechowywać różnego rodzaju obiekty, jednak to co je wyróżnia to ich zmienny rozmiar.

W JAVIE mamy dwa rodzaje list:

* **ArrayList** - implementacja tablicowa

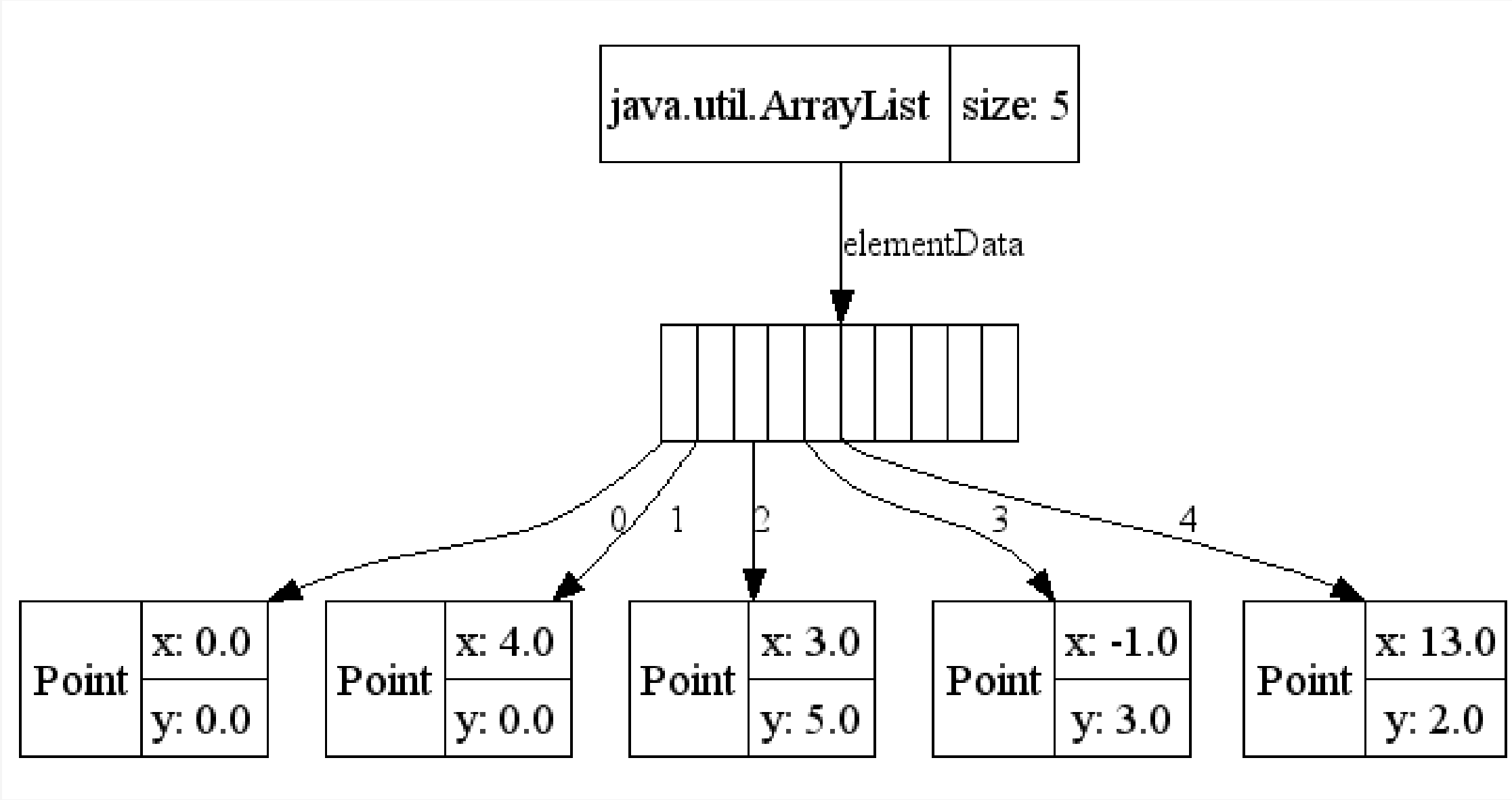
* **LinkedList** - implementacja wiązana

Lista

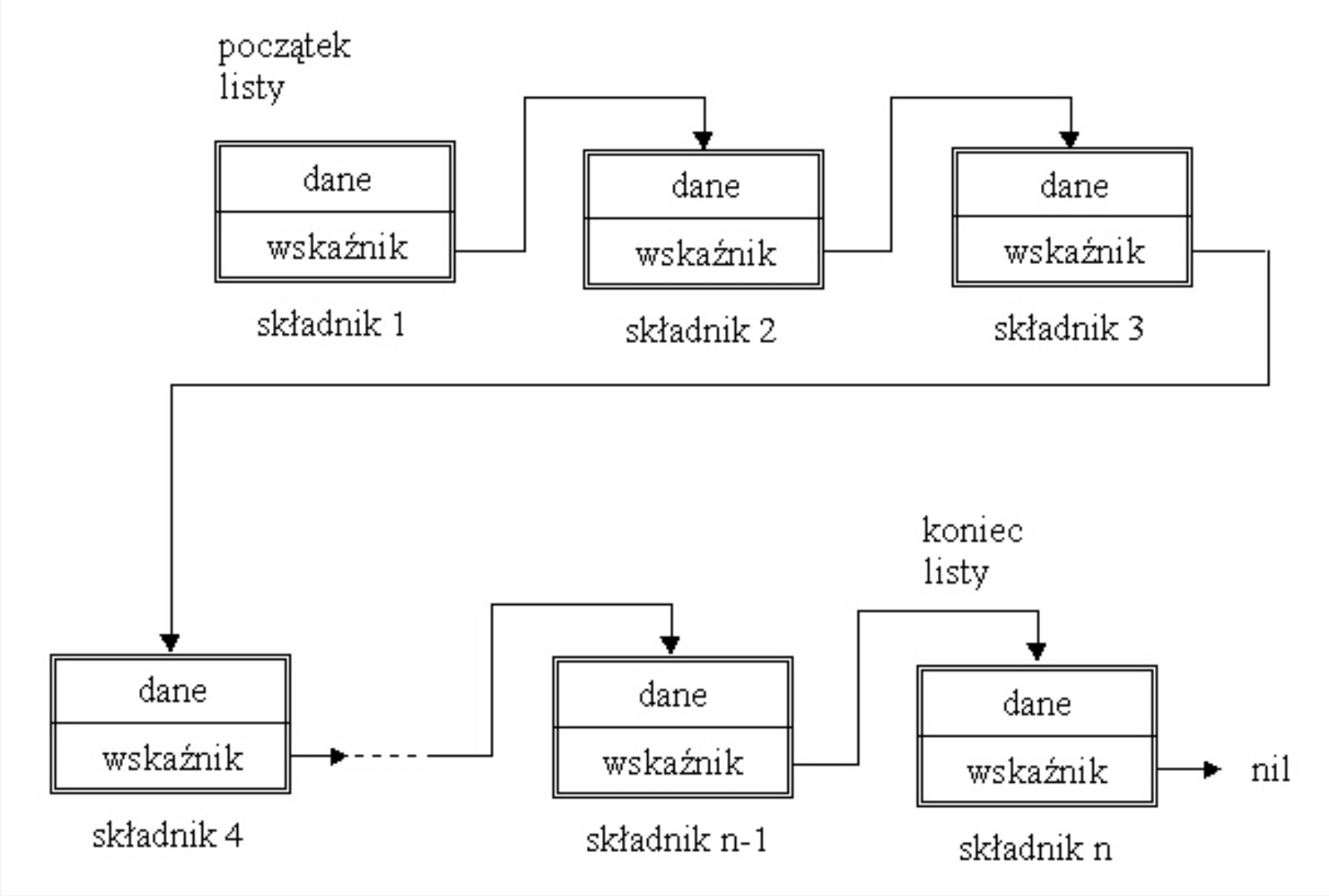
Najprościej mówiąc klasę ArrayList wykorzystujemy w przypadku, gdy najważniejszy jest czas dostępu do danych, natomiast LinkedList, gdy wiemy, że często będą wykonywane operacje usuwania, dodawania itp elementów gdzieś w środku struktury, a sprawą podrzędną są operacje wyszukiwania i szybkości dostępu do elementów listy.

DO PRZYGOTOWANIA : implementacja listy jednokierunkowej połączonej oraz listy tablicowej.

Lista


java.util.ArrayList
size: 5
elementData
0
1
2
3
4
Point
x: 0.0
y: 0.0
Point
x: 4.0
y: 0.0
Point
x: 3.0
y: 5.0
Point
x: -1.0
y: 3.0
Point
x: 13.0
y: 2.0

Lista

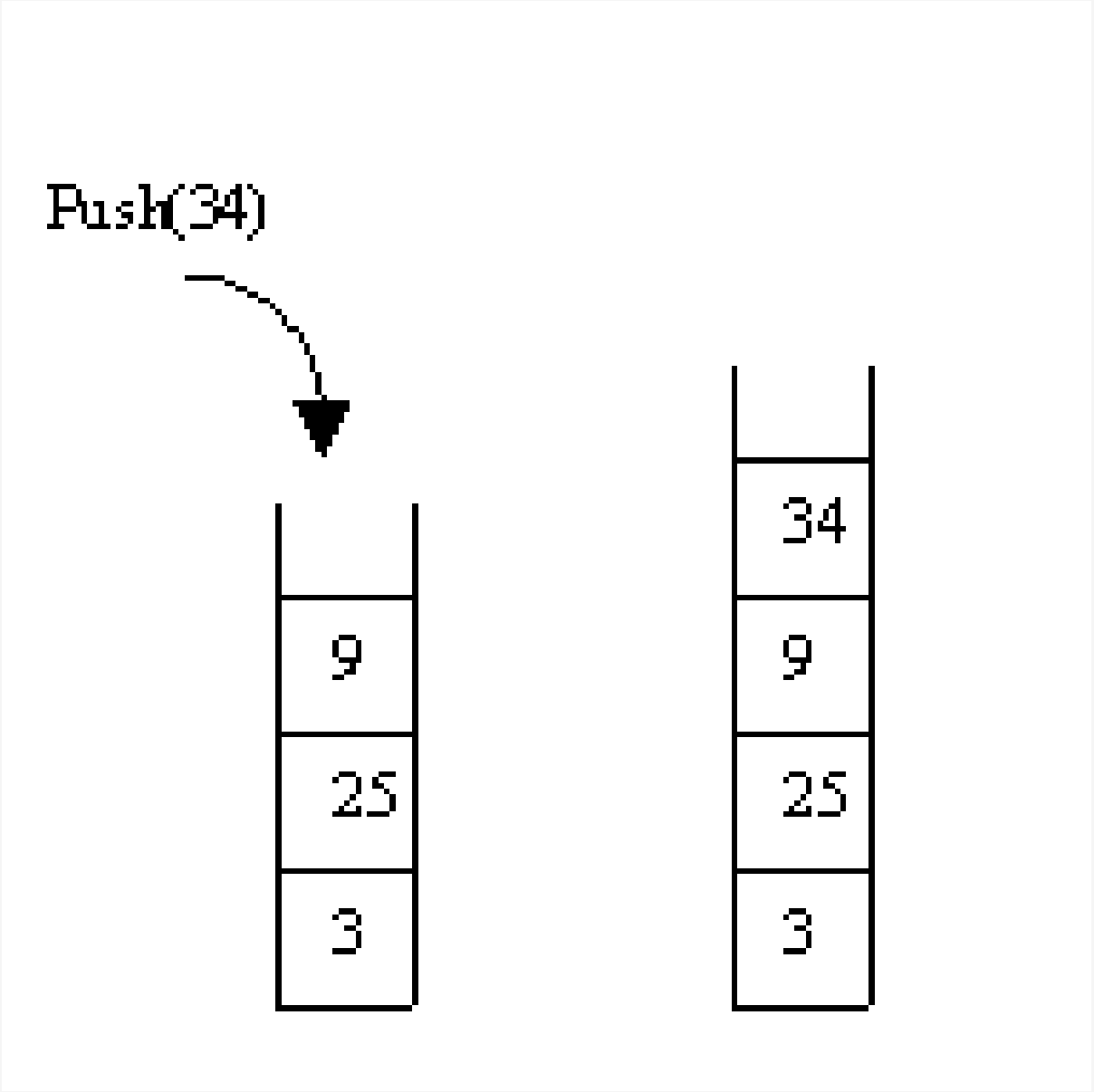

początek
listy
dane
wskaźnik
składnik 1
dane
wskaźnik
składnik 2
dane
wskaźnik
składnik 3
dane
wskaźnik
składnik 4
dane
wskaźnik
składnik n-1
koniec
listy
dane
wskaźnik
składnik n
nil

Stos

Stos - jest to liniowa struktura danych, wykorzystywana często w informatyce. Głównym założeniem stosu, jest strategia LIFO - Last In First Out. W wolnym tłumaczeniu polega to na tym, aby ostatnio położony element, był z niego w pierwszej kolejności zdejmowany.

Aby wyobrazić sobie jak powinien funkcjonować stos, można przywołać w głowie obraz kilku (nastu, dziesięciu, set) dużych i ciężkich arkuszy stali, leżących jeden na drugim. Bez ciężkiego sprzętu (a zakładamy, że takim nie dysponujemy), aby dostać się do arkuszy położonych na dnie stosu, musimy z niego usunąć te leżące na górze.

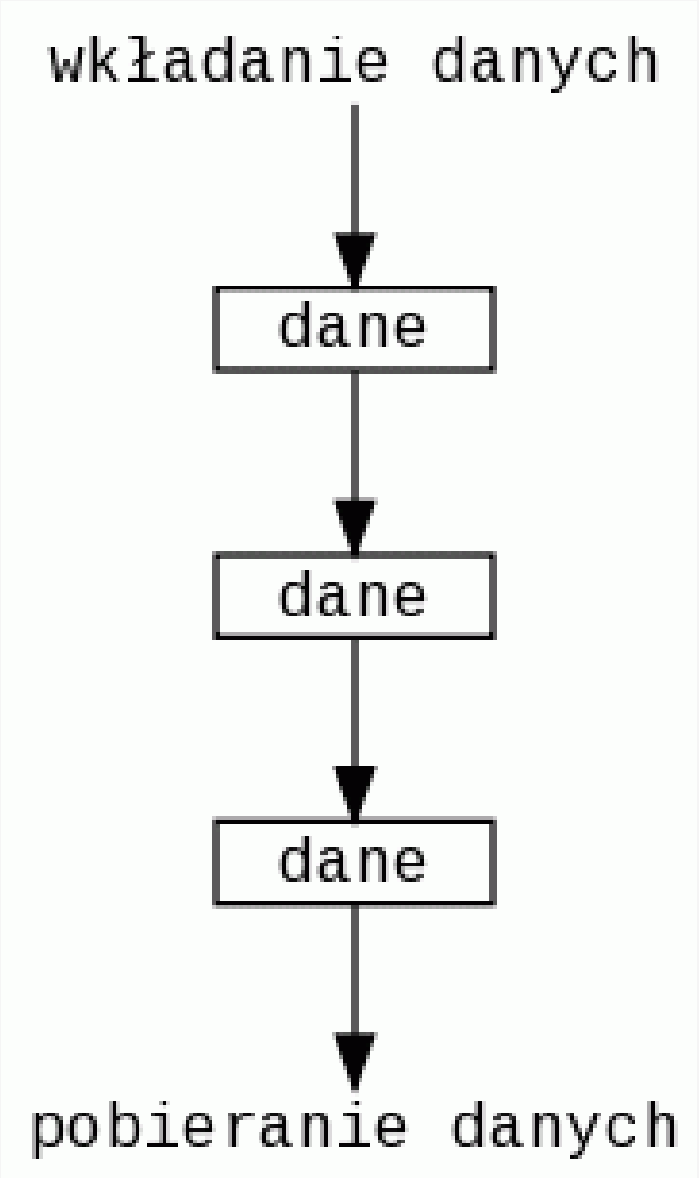
Stos



Kolejka

Kolejka jest strukturą liniowo uporządkowanych danych w której dołączać nowe dane można jedynie na koniec kolejki a usuwać z początku. Procedura usunięcia danych z końca kolejki jest taka sama, jak w przypadku stosu, z tą różnicą, że usuwamy dane od początku a nie od końca.
Pierwszy element (a dokładniej wskaźnik do jego miejsca w pamięci) musi zostać zapamiętany, by możliwe było usuwanie pierwszego elementu w czasie stałym O(1). Gdybyśmy tego nie zrobili, aby dotrzeć do pierwszego elementu należałoby przejść wszystkie od elementu aktualnego (czyli ostatniego), co wymaga czasu O(n).
Działanie na kolejce jest intuicyjnie jasne, gdy skojarzymy ją z kolejką ludzi np. w sklepie. Każdy nowy klient staje na jej końcu, obsługa odbywa się jedynie na początku.

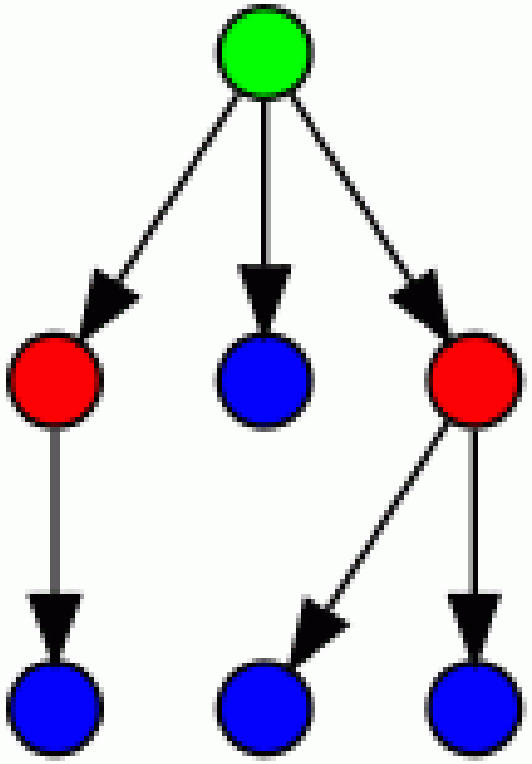
Kolejka

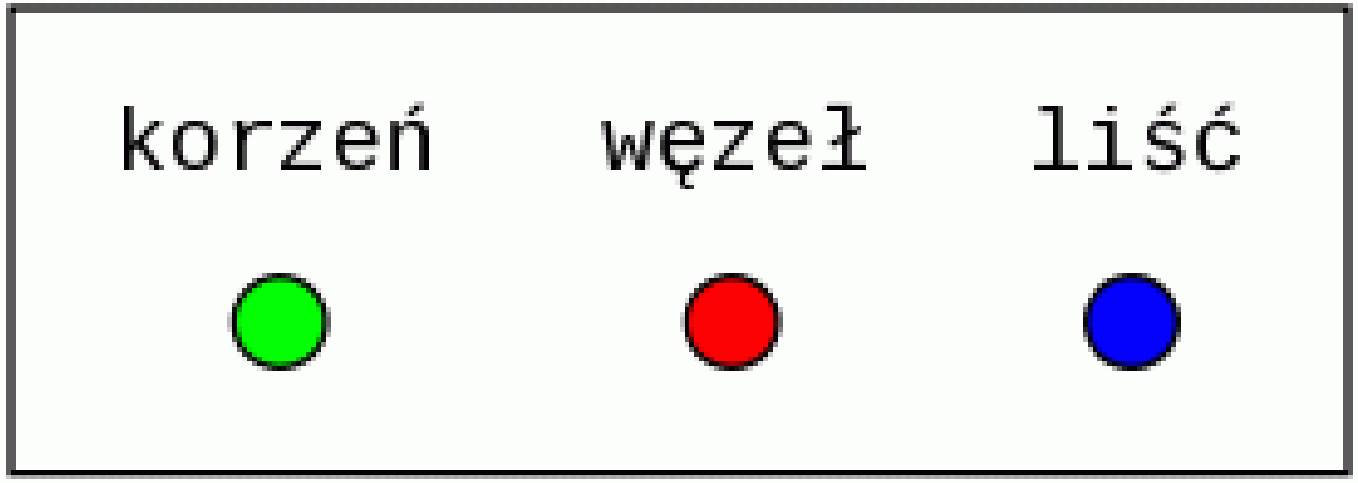


Drzewo

Drzewo jest bardziej skomplikowaną strukturą niż poprzednie. Dla każdego drzewa wyróżniony jest jeden, charakterystyczny element- korzeń. Korzeń jest jedynym elementem drzewa, który nie posiada elementów poprzednich. Dla każdego innego elementu określony jest dokładnie jeden element poprzedni. Dla każdego elementu oprócz ostatnich, tzw. liści istnieje co najmniej 1 element następny. Jeżeli liczba następnych elementów wynosi nie więcej niż 2 to drzewo nazywamy binarnym, jeżeli natomiast liczba elementów wynosi dokładnie 2 to drzewo nazywamy pełnym drzewem binarnym. Drzewo można zdefiniować, jako acykliczny graf.

Drzewo





Mapa

Mapa - obiekt mapujący wartości parametrów na podstawie klucza. W danej mapie może istnieć tylko jeden unikatowy klucz. Szeroko stosowany w komunikacji między aplikacjami, dzięki możliwości przypisanie wartości do stałych, ustalonych parametrów.

Mapę można sobie wyobrazić jako kolekcje przechowującą pary klucz-parameter. Na podstawie klucza wyciągamy wartość powiązaną z kluczem.

Mapa

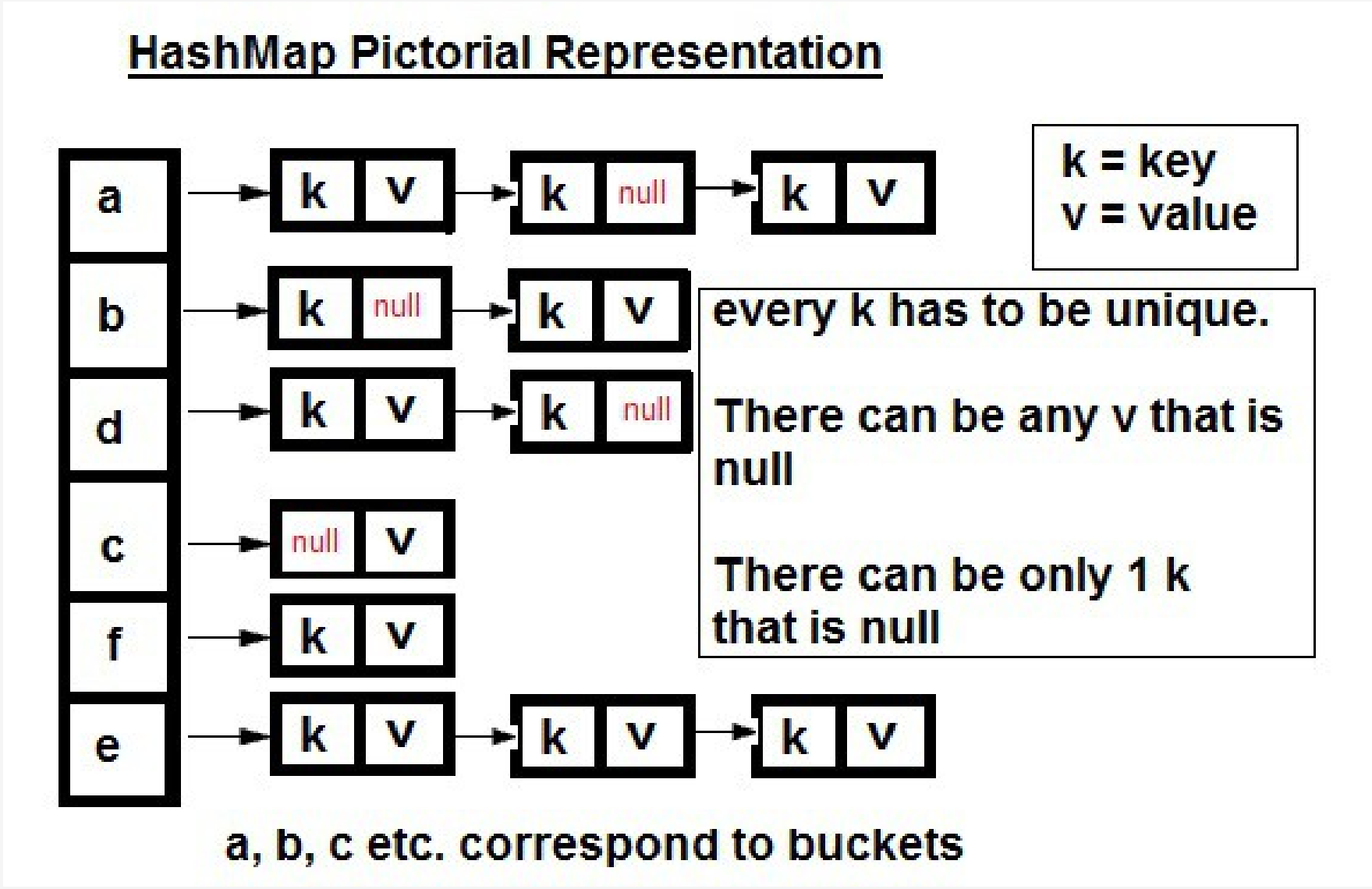
Przykład:

```
Map<String, Integer> Oceny = new HashMap<String, Integer>();
Oceny.put("Matematyka", 5);
Oceny.put("Polski", 2);
```

Właśnie zparametryzowaliśmy sobie oceny. Możemy teraz odwołać się do wartości z mapy za pomocą klucza, który stanowi w naszym przypadku String określający nazwę przedmiotu.

Mapę można zaimplementować implementując interfejs "Map" stosując 2 dowolne kolekcje. Mogą to być tablice, listy czy nawet inne mapy. Osobiście zalecam stosowanie tych już zaimplementowanych w bibliotekach Javy, jak np. HashMap.

Mapa


HashMap Pictorial Representation
a
b
d
c
f
e
k
v
null
k = key
v = value
every k has to be unique.
There can be any v that is null
There can be only 1 k that is null
a, b, c etc. correspond to buckets