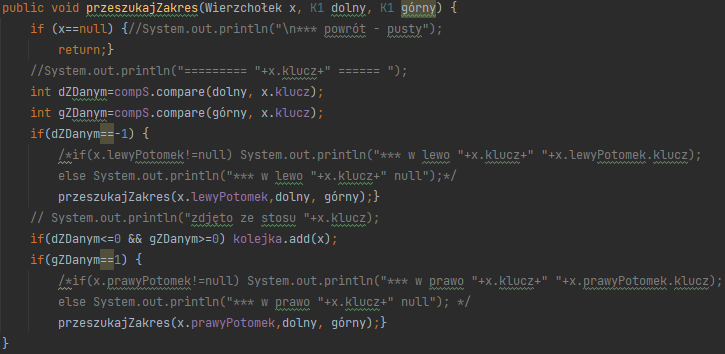
**Sprawozdanie II**

Bartosz Parszczyński

**Metoda przeszukajZakres()**



Na początku jest sprawdzana wartość zmiennej x jeśli jest ona pusta, metoda kończy swoje działanie i nie zwraca żadnej wartości.

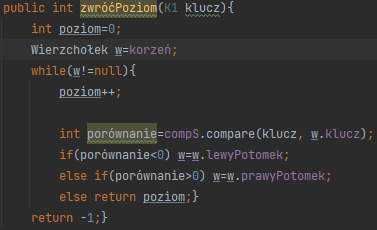
W innym razie wprowadzane są dwie zmienne typu int **dZDanym** i **gZDanym**. Pierwszej z nich jest przypisywany wynik metody *compS.compare* dla kluczy **dolny** i **x**, a drugiej też wynik *compS.compare* tylko z kluczami **górny** i **x**.

Następnie sprawdzane jest czy zmienna **dZDanym** jest równe -1, jeśli tak to wywołana jest metoda **przeszukajZakres()** dla argumentów **x.lewyPotomek**, **dolny** i **górny**.

Kolejno sprawdzany jest warunek czy **dZDanym** jest mniejszy bądź równy 0 i **gZDanym** jest większy bądź równy 0, to dodawany jest **x** do kolekcji o nazwie kolejka.

Na koniec jest sprawdzany ostatni warunek czy **gZDanym** jest równy 1, jeśli tak to wywołana zostanie metoda **przeszukajZakres()** dla argumentów **x.prawyPotomek**, **dolny** i **górny**.

**Metoda zwróćPoziom()**



Metoda sprawdza, na jakim poziomie drzewa znajduje się podany klucz.

Na początku jest deklarowana zmienna typu int o nazwie **poziom** i nadana jest mu wartość 0 oraz Wierzchołek **w** i przypisywany mu jest korzeń.

Później dopóki **w** jest różny od pustej wartości wykonywane jest:

- zwiększenie zmiennej **poziom** o 1,

- zadeklarowanie zmiennej typu int o nazwie **porównanie** i nadanie mu wyniku metody **compS.compare** dla **klucz** i **w.klucz,**

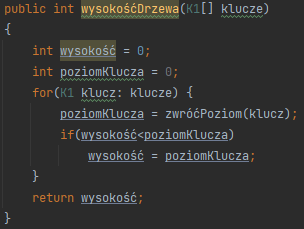
- Sprawdzany jest warunek czy zmienna **porównanie** jest mniejsza od 0, jeśli tak to dla **w** przypisywany jest **lewyPotomek,**

- w innym przypadku sprawdzany jest warunek czy zmienna **porównanie** jest większa od 0, jeśli tak to dla **w** jest przypisywany jest **prawyPotomek,**

**-** jeśli jednak ten warunek też nie jest spełniony to zwracana jest wartość **poziom.**

W sytuacji kiedy **w** jest pusty (null) zwracana jest wartość -1.

**Metoda wysokośćDrzewa()**



Metoda **wysokośćDrzewa()** zwraca wysokość drzewa z wprowadzonej listy **klucze**.

Na początku tworzy dwie zmienne **wysokość** i **poziomKlucza** i nadaje im wartość 0.

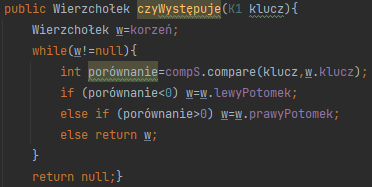
Następnie pętla dla każdego **klucz** w liście **klucze** wykonuje:

- zmiennej **poziomKlucza** przypisuje wynik metody **zwróćPoziom** dla aktualnego **klucza**,

- potem sprawdzany jest warunek czy **wysokość** jest mniejszy od **poziomKlucza** jeśli tak to przypisuje zmiennej **wysokość** wartość **poziomKlucza**.

Po skończonej pętli zwracana jest **wysokość**.

**Metoda czyWystępuje()**



Sprawdza czy występuje podany **klucz** w drzewie binarnym.

Ustalany jest Wierzchołek **w** któremu przypisujemy korzeń.

Następnie do póki **w** jest różny od wartości null to wykonuje:

- deklaruje zmienną typu int o nazwie **porównanie** i przypisuje jej wynik metody compS.compare dla **klucz** i **w.klucz**,

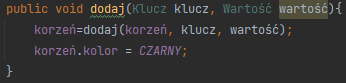
- sprawdzany jest warunek czy **porównanie** jest mniejsze od 0, jeśli tak to dla **w** jest przypisywany **lewyPotomek**,

- w innym przypadku sprawdzany jest warunek czy **porównanie** jest większe od 0, jeśli tak to dla **w** jest przypisywany **prawyPotomek**,

- jest i ten warunek nie jest prawdziwy to zwracana jest wartość **w**.

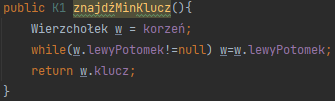
Na koniec metody jest zwracana wartość null, jeśli nie zostanie znaleziony podany **klucz**.

**Metoda dodaj()**



Zmiennej **korzeń** przypisuje wynik metody dodaj z 3 argumentami **korzeń**, **klucz** i **wartość**. A dla argumentu **kolor** nadaje wartość **CZARNY**

**Metoda znajdźMinKlucz()**



Ma na celu znalezienie minimalniej wartości klucza dla całego drzewa.

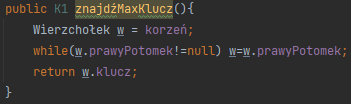
Najpierw jest deklarowany Wierzchołek **w**,któremu przypisywany jest korzeń.

Następnie dopóki **w.lewyPotomek** jest różny od wartości null to wykonywane jest:

- dla **w** jest przypisywany **lewyPotomek**.

Na koniec zwracany jest **w.klucz**.

**Metoda znajdźMaxKlucz()**



Ma na celu znalezienie maksymalnej wartości klucza dla całego drzewa.

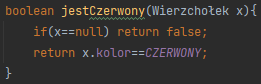
Najpierw jest deklarowany Wierzchołek **w**,któremu przypisywany jest korzeń.

Następnie dopóki **w.prawyPotomek** jest różny od wartości null to wykonywane jest:

- dla **w** jest przypisywany **prawyPotomek**.

Na koniec zwracany jest **w.klucz**.

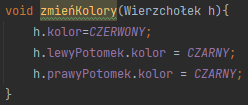
**Metoda jestCzerwony()**



Sprawdza warunek czy Wierzchołek **x** ma wartość null jeśli tak to metoda zwraca wartość false.

W innym przypadku sprawdza czy dla **x.kolor** jego wartość wynosi **CZERWONY** jeśli tak to zwraca wartość prawda jeśli nie to fałsz.

**Metoda zmieńKolory()**



Dla wierzchołka **h** przypisuje atrybutowi **kolor** wartość **CZERWONY**. Kolejno dla **lewyPotomek** i dla **prawyPotomek** przypisuje atrybutom **kolor** wartość **CZARNY**.