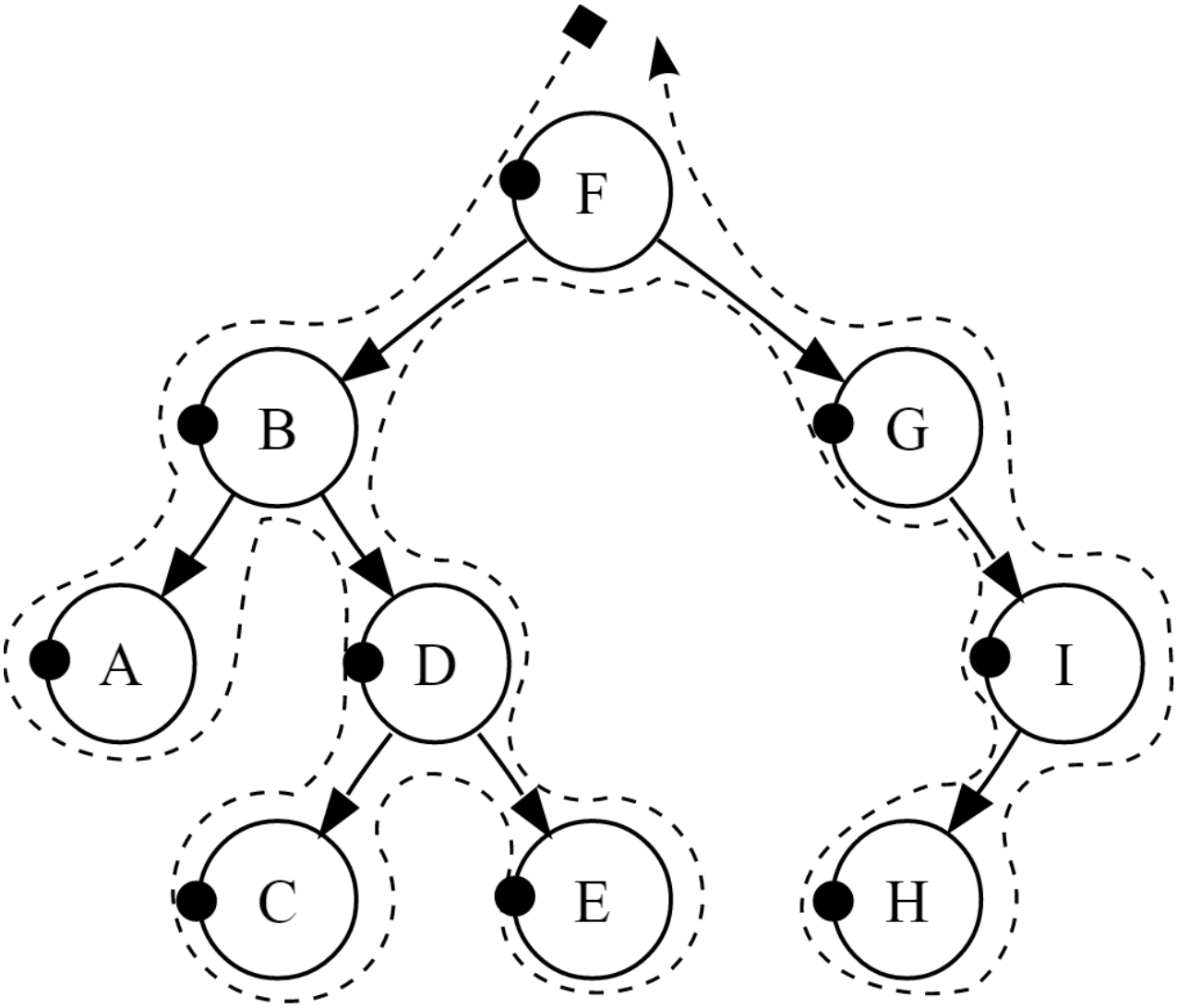
W ramach zadania przyjmujemy, że wszystkie kolekcje zdefiniowane w treści przechowują wyłącznie wartości typu double. Twoje zadanie polega na zaprojektowaniu architektury obiektowej pozwalającej na łatwe przeglądanie elementów kolekcji.

BinaryTree to klasa implementująca drzewo binarne (bez równoważenia). Takie drzewa można przeglądać na wiele różnych sposobów. W ramach tego zadania przyjmujemy jako obowiązującą i domyślną strategię przeglądania left-to-right depth-first. pre-order travelsal. Strategia ta zakłada, że elementy zwracane są w kolejności przechodzenia przez nie. Strategie można opisać procedurą:

1. Zwróć wartość aktualnego węzła;
2. Wykonaj procedurę rekurencyjnie dla lewego poddrzewa
3. Wykonaj procedurę rekurencyjnie dla prawego poddrzewa

Przykład:



Dla powyższego drzewa prawidłowa kolejność to: F, B, A, D, C, E, G, I, H.

Kod zawiera metody statyczne inicjujące kolekcje, które używane będą do testów. W zadaniu wykorzystywane są trzy kolekcje:

- binaryTree1, binaryTree2 – drzewa binarne

- FibonacciLazyList - implementacja leniwej kolekcji zawierającej elementy ciągu Fibonacciego. Ponieważ ciąg ten jest nieograniczony na potrzeby implementacji wprowadzony jest limit zawartych wyrazów, w ramach zadania należy przyjąć, że limit ten określa koniec ciągu.

Inicjalizacja kolekcji wykorzystywanych do testów umieszczona jest w funkcji main(), w ramach rozwiązania nie można zmieniać kodu odpowiadającego za inicjalizację kolekcji testowych.

1. Wyświetl ciąg elementów zawartych w binaryTree1 oraz binaryTree2 zgodnie z zasadami przeglądania przedstawionymi wcześniej. Przykładowe wyjście:
   7
   5
   4
   3
   6
   8
   9

   17
   16
   14
   13
   15
   18
   19
   ---
2. Wyświetl wszystkie elementy FibonacciLazyList o parzystych indeksach.
3. Wyświetl wszystkie dwuelementowe wariacje elementów zbioru składającego się z sumy zbiorów binaryTree1, binaryTree2 oraz FibonacciLazyList
   Przykładowe wyjście:
   (...)
   7 7
   7 5
   7 4
   7 3
   (…)
4. Wyświetl różnice powyższych elementów
   Przykładowe wyjście (dla przykładowych kombinacji z pktu 6-ego):
   (...)
   0
   2
   3
   4
   (…)
5. Dla chętnych (nie podlega ocenie): FibonacciLazyList zawiera błąd w implementacji i w niektórych przypadkach zwraca niepoprawne wartości ciągu. Kiedy i jak to poprawić w możliwie najprostszy sposób?

Uwaga: w ramach rozwiązania przyjmujemy, że nigdy nie może dojść do sytuacji, że kolekcja jest modyfikowana w trakcie przeglądania.