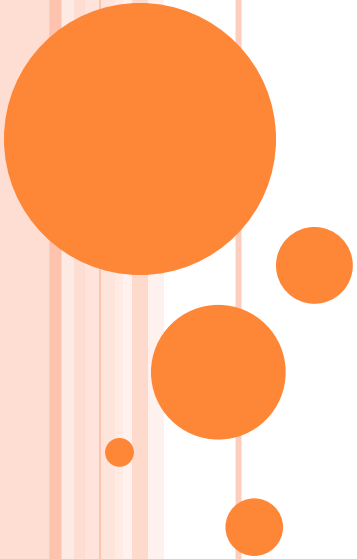
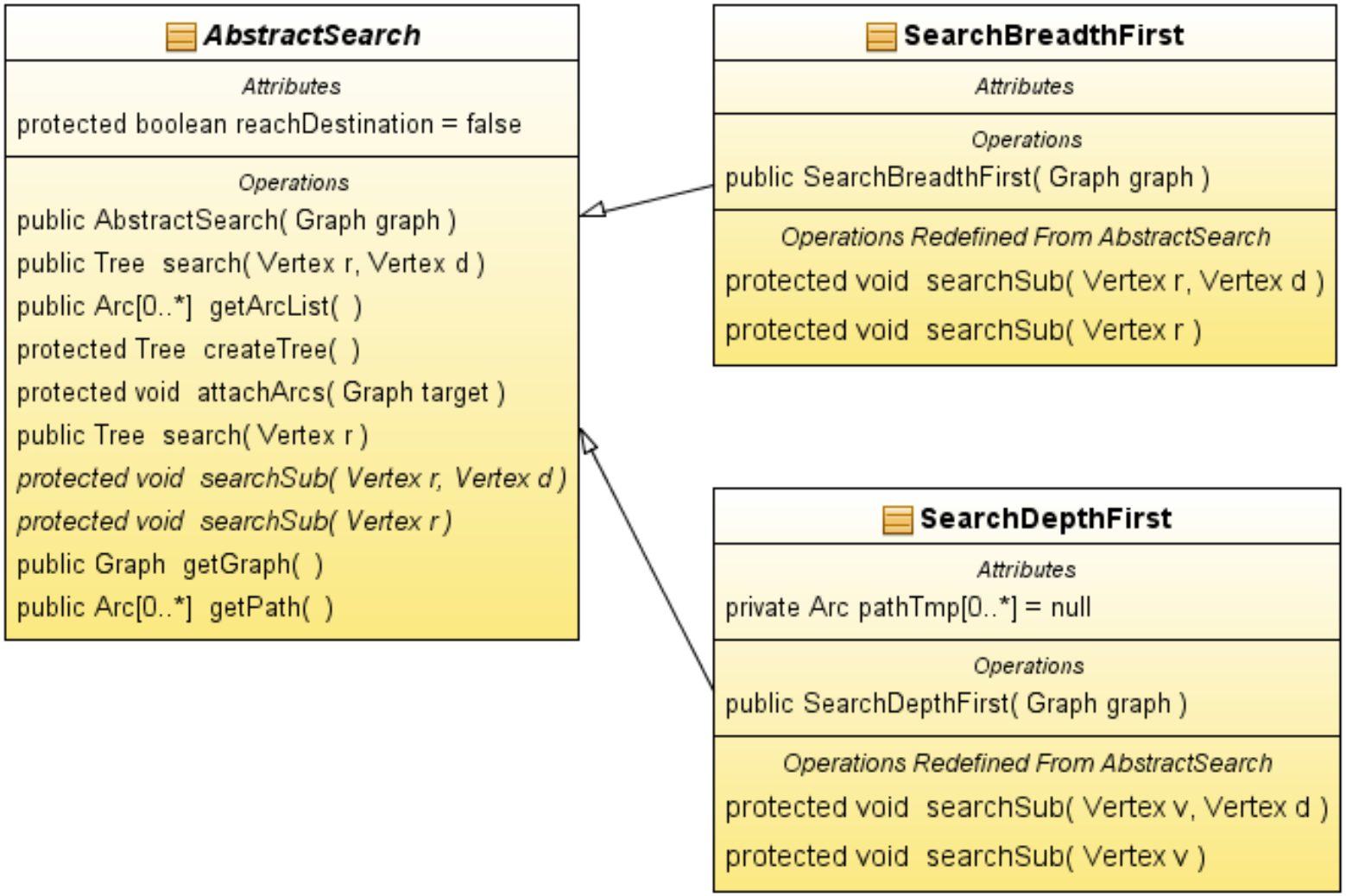


グラフの探索
Javaでの実装


AbstractSearch
Attributes
protected boolean reachDestination = false
Operations
public AbstractSearch(Graph graph)
public Tree search(Vertex r, Vertex d)
public Arc[0..*] getArcList( )
protected Tree createTree( )
protected void attachArcs(Graph target)
public Tree search(Vertex r)
protected void searchSub(Vertex r, Vertex d)
protected void searchSub(Vertex r)
public Graph getGraph( )
public Arc[0..*] getPath( )
SearchBreadthFirst
Attributes
Operations
public SearchBreadthFirst(Graph graph)
Operations Redefined From AbstractSearch
protected void searchSub(Vertex r, Vertex d)
protected void searchSub(Vertex r)
SearchDepthFirst
Attributes
private Arc pathTmp[0..*] = null
Operations
public SearchDepthFirst(Graph graph)
Operations Redefined From AbstractSearch
protected void searchSub(Vertex v, Vertex d)
protected void searchSub(Vertex v)

AbstractSearchクラスのフィールド

```
/** 探索対象となるgraph */
  protected Graph graph;
/** 探索した頂点のリスト */
  protected List<Vertex> listOfVertex = null;
/** 終点を指定した場合、その終点に到達したか否か */
  protected boolean reachDestination = false;
/** 探索に使用した孤 */
  protected List<Arc>  arcList= null;
/** 始点 */
  protected Vertex r = null;
/** 終点 */
  protected Vertex d = null;
/** 始点から終点への道 */
  protected List<Arc> path=null;
```

AbstractSearchクラスの主要メソッド

public AbstractSearch(Graph graph)	コンストラクタ
public Tree search(Vertex r)	探索実行（終点指定なし）
public Tree search(Vertex r, Vertex d)	探索実行（終点指定）
protected Tree createTree()	探索結果から木を得る
protected void attachArcs(Graph target)	使用した弧をtargetに追加

- 以下は、各派生クラスで実装する

protected abstract void searchSub(Vertex r)	探索の実体
protected abstract void searchSub(Vertex r, Vertex d)	
Protected abstract void createPath(List<Arc> aList)	道の生成

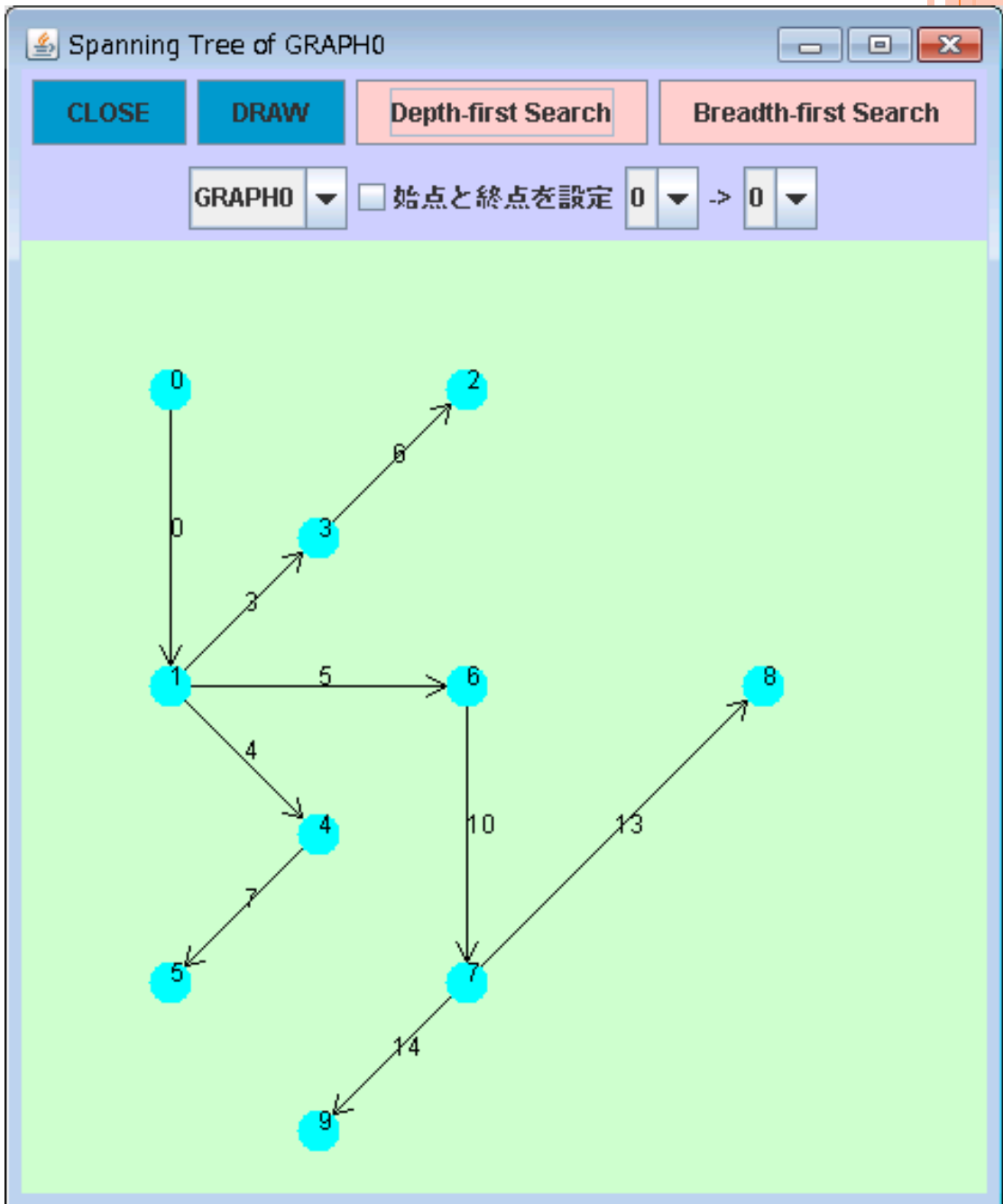
深さ優先探索

```
protected void searchSub(Vertex v) {
        List<Arc> aList = graph.getArcs(v);
        if (aList == null) {
            return;
        }
        for (Arc a : aList) {//頂点v から出ている弧
            //弧の先の頂点
            Vertex to = graph.getTerminal(a, v);
            if (!listOfVertex.contains(to)) {//弧の先の頂点はまだtreeに無い
                //頂点を追加
                listOfVertex.add(to);
                /** 探索に使用した弧を保存 */
                arcList.add(a);
                searchSub(to);
            }
        }
    }
```

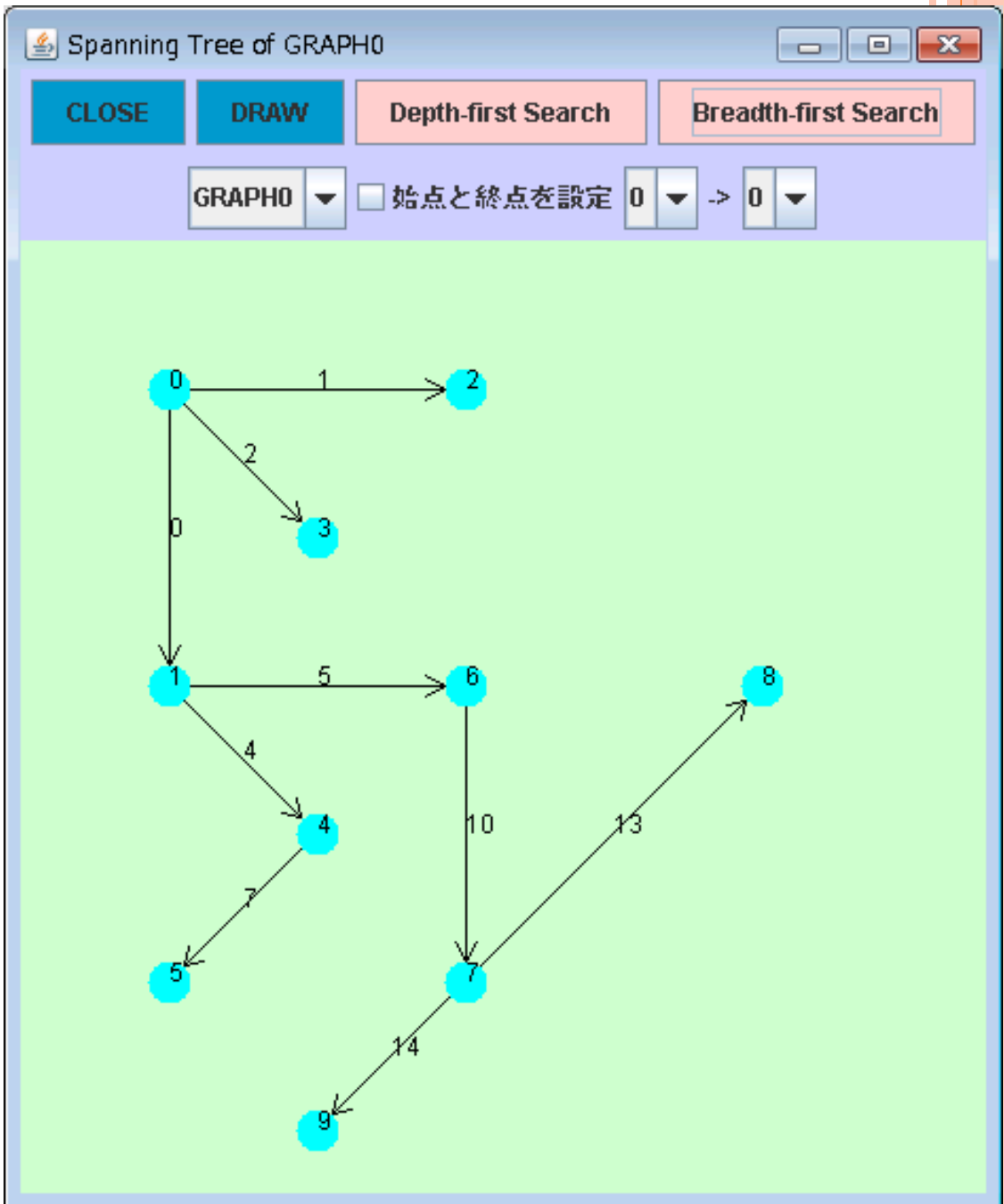
幅優先探索

```
protected void searchSub(Vertex r) {
    //待ち行列の作成
    ConcurrentLinkedQueue<Vertex> queue
                        = new ConcurrentLinkedQueue<>();
    queue.add(r);
    while (!queue.isEmpty()) {
        Vertex v = queue.poll();
        List<Arc> aList = graph.getArcs(v);
        if (aList != null) {
            for (Arc a : aList) {
                checkArc(v, null, a, queue);
            }
        }
    }
}
```

深さ優先探索の実施


Spanning Tree of GRAPH0
CLOSE
DRAW
Depth-first Search
Breadth-first Search
GRAPH0
始点と終点を設定
0
->
0
0
2
3
1
6
8
4
5
7
9
6
0
3
5
4
7
10
13
14

幅優先探索の実施


Spanning Tree of GRAPH0
CLOSE
DRAW
Depth-first Search
Breadth-first Search
GRAPH0
始点と終点を設定
0
->
0
0
1
2
3
4
5
6
7
8
9
1
2
0
5
4
7
10
13
14