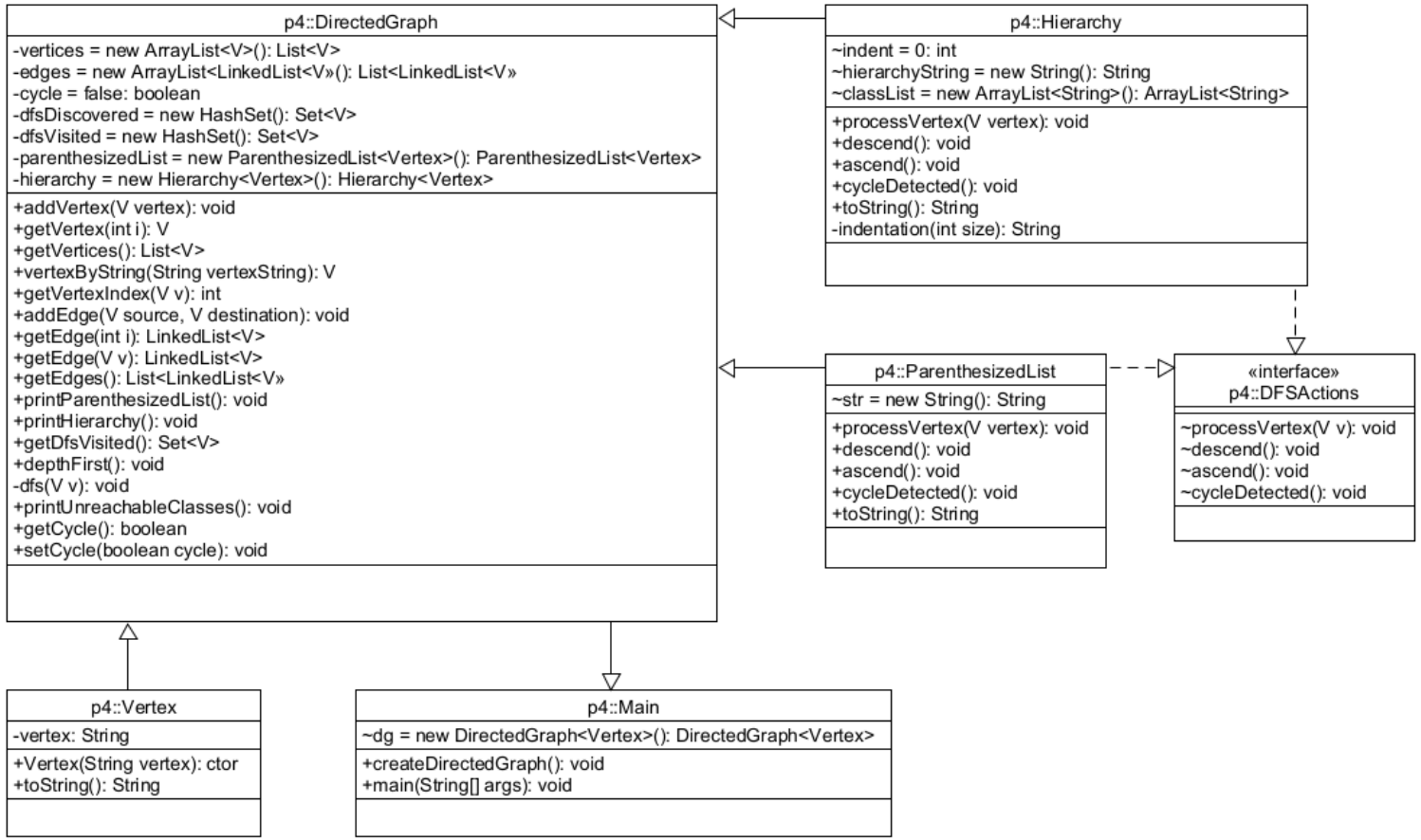
CMIS 350 6382

Project 4

Allen Taylor

2/18/2022

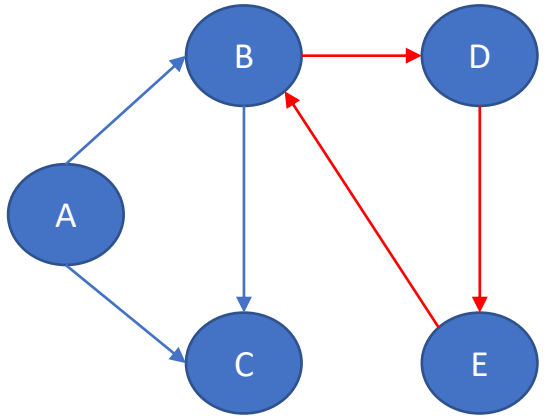
UML Diagram:


p4::DirectedGraph
-vertices = new ArrayList<V>(): List<V>
-edges = new ArrayList<LinkedList<V»(): List<LinkedList<V»
-cycle = false: boolean
-dfsDiscovered = new HashSet(): Set<V>
-dfsVisited = new HashSet(): Set<V>
-parenthesizedList = new ParenthesizedList<Vertex>(): ParenthesizedList<Vertex>
-hierarchy = new Hierarchy<Vertex>(): Hierarchy<Vertex>
+addVertex(V vertex): void
+getVertex(int i): V
+getVertices(): List<V>
+vertexByString(String vertexString): V
+getVertexIndex(V v): int
+addEdge(V source, V destination): void
+getEdge(int i): LinkedList<V>
+getEdge(V v): LinkedList<V>
+getEdges(): List<LinkedList<V»
+printParenthesizedList(): void
+printHierarchy(): void
+getDfsVisited(): Set<V>
+depthFirst(): void
-dfs(V v): void
+printUnreachableClasses(): void
+getCycle(): boolean
+setCycle(boolean cycle): void
p4::Hierarchy
~indent = 0: int
~hierarchyString = new String(): String
~classList = new ArrayList<String>(): ArrayList<String>
+processVertex(V vertex): void
+descend(): void
+ascend(): void
+cycleDetected(): void
+toString(): String
-indentation(int size): String
p4::ParenthesizedList
~str = new String(): String
+processVertex(V vertex): void
+descend(): void
+ascend(): void
+cycleDetected(): void
+toString(): String
«interface»
p4::DFSActions
~processVertex(V v): void
~descend(): void
~ascend(): void
~cycleDetected(): void
p4::Vertex
-vertex: String
+Vertex(String vertex): ctor
+toString(): String
p4::Main
~dg = new DirectedGraph<Vertex>(): DirectedGraph<Vertex>
+createDirectedGraph(): void
+main(String[] args): void

Test Case 1: Circular Dependencies

Input:

```
ClassA ClassB ClassC
ClassB ClassC ClassD
ClassD ClassE
ClassE ClassB
```



B->D->E->B (creates a cycle)

Test Case 1:
Circular Dependencies

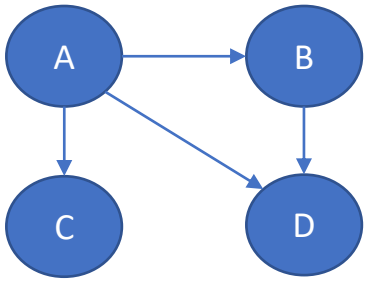
Output:

```
Parenthesized List Representation:
( ClassA ( ClassB ( ClassC ClassD ( ClassE * ) ) ClassC ) )

Hierarchy Representation:
ClassA
    ClassB
        ClassC
        ClassD
            ClassE *
    ClassC


The following classes are unreachable:
None
```

Test Case 2: No Circular Dependencies

Input:

```
ClassA ClassB ClassC ClassD
ClassB ClassD
```



Test Case 2:

No Circular Dependencies

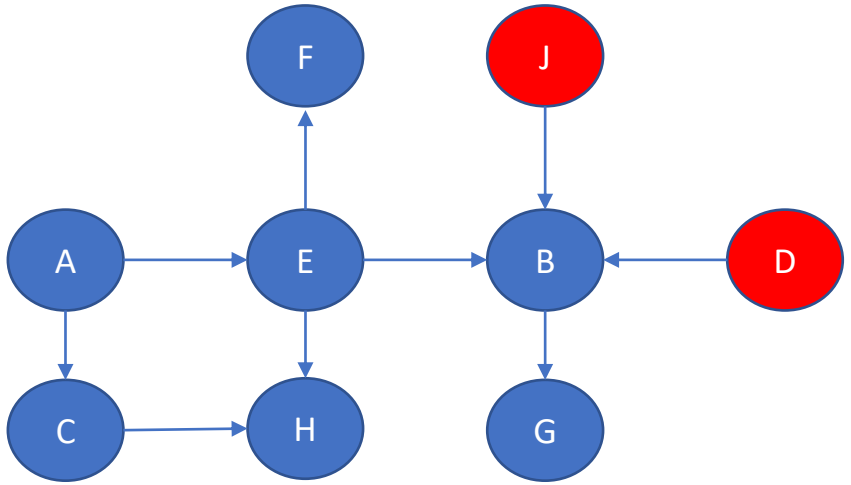
Output:

```
Parenthesized List Representation:
( ClassA ( ClassB ( ClassD ) ClassC ClassD ) )

Hierarchy Representation:
ClassA
    ClassB
        ClassD
    ClassC
    ClassD


The following classes are unreachable:
None
```

Test Case 3: Unreachable Classes

Input:

```
ClassA ClassC ClassE
ClassB ClassG
ClassC ClassH
ClassD ClassB
ClassE ClassB ClassF ClassH
ClassJ ClassB
```



Test Case 3:
Unreachable Classes

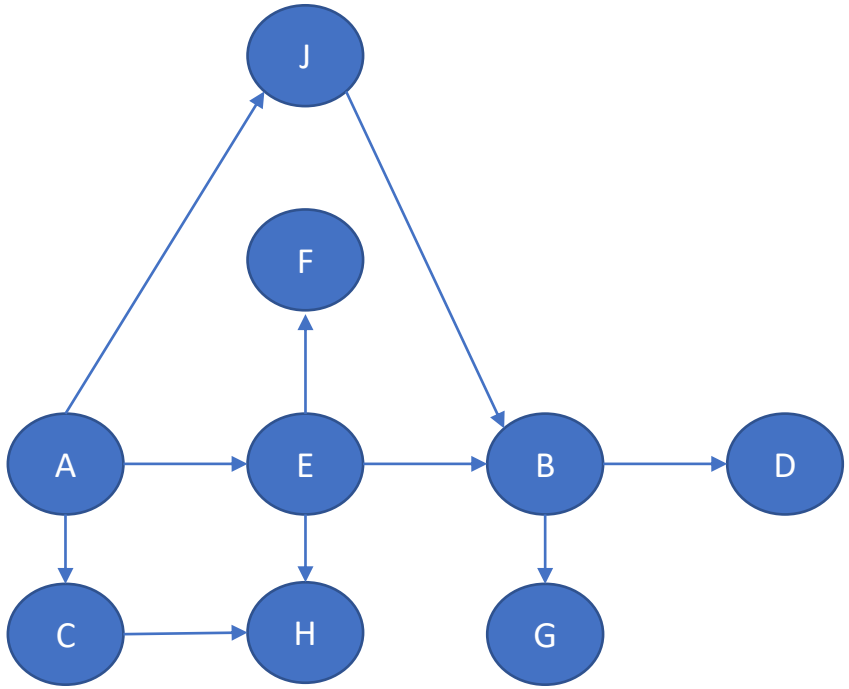
Output:

```
Parenthesized List Representation:
( ClassA ( ClassC ( ClassH ) ClassE ( ClassB ( ClassG ) ClassF ClassH ) ) )

Hierarchy Representation:
ClassA
    ClassC
        ClassH
    ClassE
        ClassB
            ClassG
        ClassF
        ClassH


The following classes are unreachable:
ClassD
ClassJ
```

Test Case 4: No Unreachable Classes

Input:

```
ClassA ClassC ClassE ClassJ
ClassB ClassD ClassG
ClassC ClassH
ClassE ClassB ClassF ClassH
ClassJ ClassB
```



Test Case 4:
No Unreachable Classes

Output:

```
Parenthesized List Representation:
( ClassA ( ClassC ( ClassH ) ClassE ( ClassB ( ClassD ClassG ) ClassF ClassH ) ClassJ ( ClassB ( ClassD ClassG ) ) ) )

Hierarchy Representation:
ClassA
    ClassC
        ClassH
    ClassE
        ClassB
            ClassD
            ClassG
        ClassF
        ClassH
    ClassJ
        ClassB
            ClassD
            ClassG


The following classes are unreachable:
None
```

Lessons Learned:

A lesson I learned when coding this project is that writing pseudo code and drawing diagrams can help a programmer *visualize* their code or algorithms.

For example, I made this diagram while trying to understand how code in the Hierarchy class should flow. I wrote what I wanted to happen at each step of the depth first search, in order to create the Hierarchy.

By creating a diagram of the pseudo code, I was able to produce the solution for generating the Hierarchy string representation.