On a first pass through the edges, mark all the vertices which are ends of selected edges.

On a second pass through the edges, select all edges incident to marked vertices.
On a first pass through the edges count the number of selected edges incident to each vertex
Use an accumulator array to store these values

On a second pass through the edges clear each selected edge incident to a vertex with only one incident selected edge
We need the MeshGraph class to classify the edges.
We need the Partition class to create a partition of the corners.

```java
p = new Partition(nC);
For each face f1
  For each corner c0 of f1
    Let c1 be the corner of f1 next to c0
    Let v0 and v1 be the vertices corresponding to the corners c0 and c1
    Let e be the edge joining v0 and v1
      If (e is a regular edge)
        Let c2 be the corner twin of c0
        Let f2 be the face containing c2
        Let c3 be the corner next to c2
        p.join(c0,c3)
        p.join(c1,c2)

Let nP the number of parts
```

Each part is a subset of corners supported by a common vertex.

To classify the vertices we need to count the number of parts supported by the same vertex:

Use an accumulator array with one entry per vertex initialized to zero
For each part
  Find the supporting vertex
  Increment the array entry corresponding to the vertex
The vertex classification is stored in the accumulator array

To convert a mesh to manifold, we need to generate an output mesh: first the vertices, then the faces

For each part (all corners point to the same input vertex)
  Get the vertex coordinates from the input mesh and push them onto the back of the output coord array
  Store the mapping from corner to part number (output vertex index) in a look-up-table

For each face
  Replace each corner value with the value stored in the look-up table for that corner