```
45            if (events.Count > 0)
46            {
47              intersect = true;
48            }
49          }
50          else // -- remember the index of the original rectangle
51            origID = collectRectangles.IndexOf(curTestRect);
52        }
53
54        // -- decide what to do if there is an intersection
55        if (intersect)
56        {
57          Print("intersect");
58          toRemoveRects.Add(curRect);  // if it can't grow further,
                                            remove element
59          grownRectangles.Add(curRect);// add element to the final list
60        }
61        else
62        {
63          Print("grow on");
64          growCadidates[i] = copyRect; // assign new size for
                                            intersection test
65          collectRectangles[origID] = copyRect; // assign new size for
                                                     imtersection test
66        }
67      }
68
69      // -- remove rectangles which can not grow anymore
70      foreach(Rectangle3d delRect in toRemoveRects)
71      {
72        growCadidates.Remove(delRect);
73      }
74
75      Print("iteration " + counter.ToString());
76      counter++;
77    } while(growCadidates.Count > 0 && counter < maxIterations);
78
79    // -- extrude the rectangles
80    foreach(Rectangle3d curRect in grownRectangles)
81    {
82      // -- convert the recangle to a nurbs and create a planar Brep
           Surface in Rhino
83      NurbsCurve myCurve = curRect.ToNurbsCurve();
84      Rhino.Geometry.Extrusion myExtrusion = Extrusion.Create(myCurve,
           rnd.Next(5, 20), true);
85      collectBreps.Add(myExtrusion);
86    }
```