Unit Test Report

```
public GraphNode() : base() { }
public GraphNode(T value) : base(value) { }
public GraphNode(T value, NodeList<T> neighbors) : base(value, neighbors) { }
```

EC1: Nothing is provided.
EC2: Value is provided.
EC3: Value and neighbors are provided and T is of the same type of variable.
EC4: Value and neighbors are provided and T is of differing types of variables.

TC1: GraphNode<GameObject> newNode = new GraphNode();
Expected Output = newNode is a new GraphNode without a value and NodeList.
ECs covered=**EC1**
TC2: GraphNode<GameObject> newNode = new GraphNode<GameObject>(GameObject.CreatePrimitive(PrimitiveType.Sphere));
Expected Output = newNode is a new GraphNode with a value of GameObject.CreatePrimitive(PrimitiveType.Sphere).
ECs covered=**EC2**
TC3: GraphNode<GameObject> newNode = new GraphNode<GameObject>(GameObject.CreatePrimitive(PrimitiveType.Sphere), new NodeList());
Expected Output = newNode is a new GraphNode with a value of GameObject.CreatePrimitive(PrimitiveType.Sphere) and an empty NodeList<GameObjects> called neighbors.
ECs covered = **EC3**
TC4: GraphNode<Int> intNode = new GraphNode(2, new NodeList<string>());
Expected Output = We shouldn't be able to compile.
ECs covered = **E4**

```
public Graph() : this(null) {}
public Graph(NodeList<T> nodeSet)
{
   if (nodeSet == null)
      this.nodeSet = new NodeList<T>();
   else
      this.nodeSet = nodeSet;
}
```

EC1: There is no NodeList<T>
EC2: There is a NodeList<T> provided.
EC3: nodeSet is null

TC1: Graph <GameObject> newGraph = new Graph();
Expected Output = newGraph is a new Graph without a NodeList
ECs covered = **EC1**
TC2: Graph <GameObject> newGraph = new Graph(new NodeList<GameObject>())
Expected Output = newGraph is a new Graph with a new NodeList<T> provided.
ECs covered = **EC2, EC3**

```
public void AddNode(T value)
{
	nodeSet.Add(new GraphNode<T>(value));
}
```

EC1: Nothing is provided.
EC2: Value is provided and T is of the same type as the graph it is being added to.
EC2: Value is provided and T is of a different type than the graph it is being added to.

TC1: Graph graph = new Graph();
graph.AddNode();
Expected Output = Should not allow to compile.
ECs covered = **EC1**
TC2: Graph graph = new Graph(new nodeList<GameObject>());
graph.AddNode(new GameObject());
Expected Output = graph is a new Graph and it will have a new GameObject in its nodeList
ECs covered = **EC2**
TC3: Graph graph = new Graph(new nodeList<Int>());
graph.AddNode(new GameObject());
Expected Output = Should not allow to compile.
ECs covered = **EC3**

```
scaleGraph(float scale)
 {
	 if(scale < .025f)
	{
		scale = .025f;
	}

	if(scale > .1f)
	{
```

```
                scale = .1f;
        }

        graphParent.transform.localScale = new Vector3(scale, scale, scale);
}
```

EC1: Scale is between .025f and .1f
EC2: Scale is greater than .1f
EC3: Scale is less than .025f

TC1: Scale = .024f
EC covered: EC1
TC2: Scale = .09f
EC covered: EC2
TC3: Scale = .11f
EC covered: EC3