Name: _Laleh Rostami Hosoori_

HW6 – Design Decisions

Class: Ground

```
[SyncPattern("HostOnly")]
virtual public void SetFood(int row, int column, Food value)
```

Your Choice for a SyncPattern: HostOnly

Justification: Since the amount of food is a kind of data known only by the ground itself, other objects must not be allowed to set it and its location on the ground. Only Ant objects need to check if there is any food at a location or not and consume it.

```
[SyncPattern("RPC")]
virtual public bool ContainsFood(int row, int column)
```

Your Choice for a SyncPattern: ARPC

Justification: Access to the food is mutually exclusive and the function gets a lock to access food array (only one ant can get food at a time); therefore, RPC is the best choice. The proxy object doesn't have to wait to receive responses; this future makes ARPC the most proper synchronization pattern for this method.

```
[SyncPattern("RPC")]
virtual public Food PeekAtFood(int row, int column)
```

Your Choice for a SyncPattern: Pull

Justification: Because it doesn't need to get the most up-to-date data. The recent value is cached in the proxy object and it is updated frequently, so it's accurate enough for this method.

```
[SyncPattern("RPC")]
virtual public Food PickupFood(int row, int column, int amount)
```

Your Choice for a SyncPattern: RPC

Justification: Because access to the food is mutually exclusive and the function gets a lock to access food array (only one ant can get food at a time); therefore, RPC is the best choice.

Class: Colony

```
[SyncPattern("RPC")]
[ColorFormatter]
virtual public Color ColonyColor { get; set; }
```

Your Choice for a SyncPattern: Push with a specified time frequency

Justification: At each period it checks if the color of a colony was changed, it sends the new color to all other instances. It's one-way of push property's state. I suggest setting a time frequency because the color can change very fast so that human eyes are unable to detect it.

```
[SyncPattern("RPC")]
virtual public List<Ant> Ants { get; set; }
```

Your Choice for a SyncPattern: Push

Justification: Because it sends the new state of Ants one-way whenever it changes. In my opinion, there's no need to set a time frequency since the list changes occasionally.

```
[SyncPattern("RPC")]
[PheromoneLayerFormatter]
public PheromoneLayer PheromoneLayer { get; set; }
```

Your Choice for a SyncPattern: Push with a specified time frequency

Justification: At each period it checks if it was changed, it sends the value to all other instances. It's one-way of push property's state. I suggest setting a time frequency because I think it may change very fast so that if new values push with a time frequency, network can become very crowded.

```
[SyncPattern("RPC")]
virtual public Nest Home { get; set; }
```

Your Choice for a SyncPattern: Constant

Justification: It never changes during the lifetime of a colony.

```
[SyncPattern("RPC")]
public int InitialAntCount { get; set; }
```

Your Choice for a SyncPattern: Constant

Justification: Because it's the initial value!

Class: SimulationSetting

```
[SyncPattern("RPC")]
virtual public int MovementInterval
```

Your Choice for a SyncPattern: Push

Justification: Because it sends the new state of properties one-way whenever it changes. In my opinion, there's no need to set a time frequency since properties change occasionally.

Note: All of the other remotely accessible methods in the SimulationSettings class are basically the same and, would, therefore use the same SyncPattern,