The tree class stores the variable for each tree. An array of tree objects is created and sores each trees position and each has an original extent is 0.4. each tree is able to return its x and y position as well as extent. The trees are able to return their canopy’s start and end x and y values. The Start methods return zero if the extent leads to a negative co-ordinate and the End methods take in a height or width value and ensures that the canopy does not exceed these values. The sunexposure method has parameter of type Land.

Land objects store the width and height of the forest as well as two double arrays one storing the initialSun values and the other the shadedSun values after trees have added shade. Both of these are set to the initial sun values as the Trees create shadows the shadedSun array is edited and keeps track of the shade created by each tree before effecting the next Tree. The initialSun is used to reset the shadedSun each year as to continue the randomness of the growth and simulate the growing of a real forest. The shadedSun is also able to return the shade value for trees when they are needing to calculate their average sunexposure. Therefore the shadedSun variable needs to be thread-safe as it is shared across threads and it is vital that one thread isn’t writing to it while another is reading from as this will lead to incorrect calculations. The whole shadedSun array cannot be locked though as this would result in too many overheads and

* A description of each of the classes you added and any modifications you made to the existing classes.
* A description of all the Java concurrency features you used and why they were necessary (e.g. atomic variables, synchronized classes, synchronized collections, barriers etc.).
* You will need to explain how you wrote the code to ensure:
  1. thread safety (for both shared variables and the Swing library). You should

describe when you need to protect data and when you don’t – and explain why.

* 1. Thread synchronization where necessary
  2. liveness
  3. no deadlock.
* An explanation of how you validated your system and checked for errors (esp. race conditions).
* An explanation of how your design conforms to the Model-View-Controller pattern.
* Any additional features/extensions to (or improvements on) the basic simulation that

you think merit extra credit. There are many things that you can do to improve this simulation.