Project: Concurrency and Parallelism

Final Report

Project team:

Anuvabh Dutt

Anastasia Kuznetsova

Hackob Melkonyan

4th May, 2015

University of Nice Sophia Antipolis

**Project goal:**

Build a distributed client-server system, with multithreaded dynamically allocated clients, in order to handle the clustering algorithm described in "AntClass: discovery of clusters in numeric data by a hybridization of an ant colony with the Kmeans algorithm" by N. Monmarche, M. Slimane, G. Venturini; January 1999.

**Project description**

The software system consists of 3 main parts, Server, Client, and a shared library project being referenced by the Server and Client projects.

The communication between the Server and Client applications is based on standard Java RMI API.

All the heavy-computations are done on client side, and the update functions are invoked to change the states of the original objects on the server side.

Client

Server

Shared library of classes

Figure 1. Graphical representation of the project Client-Server architecture.

**Clustering algorithm implementation**

We have implemented the board structure according to the

Board architecture:

Board

Cell

Cell Entity

Empty Cell Entity

Ant Cell Entity

Heap Cell Entity

Heap Objects of different types

Figure 2. Graphical representation of the Board structure.

As can be seen on the figure 2, we have board object which consists of m\*m cells objects, where m – board size in rows and columns.

Each cell represents one of three types of Cell Entity objects: Empty Cell, an Ant or a Heap.

Heap represents one or more Heap Object of predefined different types.

Graphical representation of a Board.



Figure 3. Graphical representation of an empty cell, an Ant in the cell, a Heap, containing three heap objects of two different types.

Properties of the board objects:

Empty Cell:

* Can be filled with dropped Heap by an Ant
* Can be filled with an Ant for the move step

Heap:

* Can be modified by an Ant:
  + If there was one heap object in the heap:
    - Ant takes the object – heap is destroyed
    - Ant is carrying an object – put an object in the cell – heap grows
  + If there were 2 and more heap objects in the heap:
    - Ant takes the object – heap is becoming smaller
    - Ant puts the object – heap grows

Ant:

Ant objects have several methods:

1. Move – Randomly assigned movement of an Ant, on x, y or both axis’s of the board, but only in empty cells.
2. Look Around - Ant explores 8 cells around it in the following order: cell 1 = in front of an ant, cell 2 = to the left, cell 3 = to the right, etc. The location of the first heap found around an ant is returned as a result.
3. Process pick up algorithm – If there was found a heap near the ant and an ant is not carrying any objects, pick up algorithm is called to take one of the objects of a heap. Before taking an object ant locks the whole board to avoid inconsistency by checking if the heap still exists on the same location and prevent other ants from taking an object from the same heap.
4. Process drop algorithm – If there was found a heap and ant is carrying an object – it locks the board to avoid clashes with other ants, drops an object on the heap and continues to move randomly.

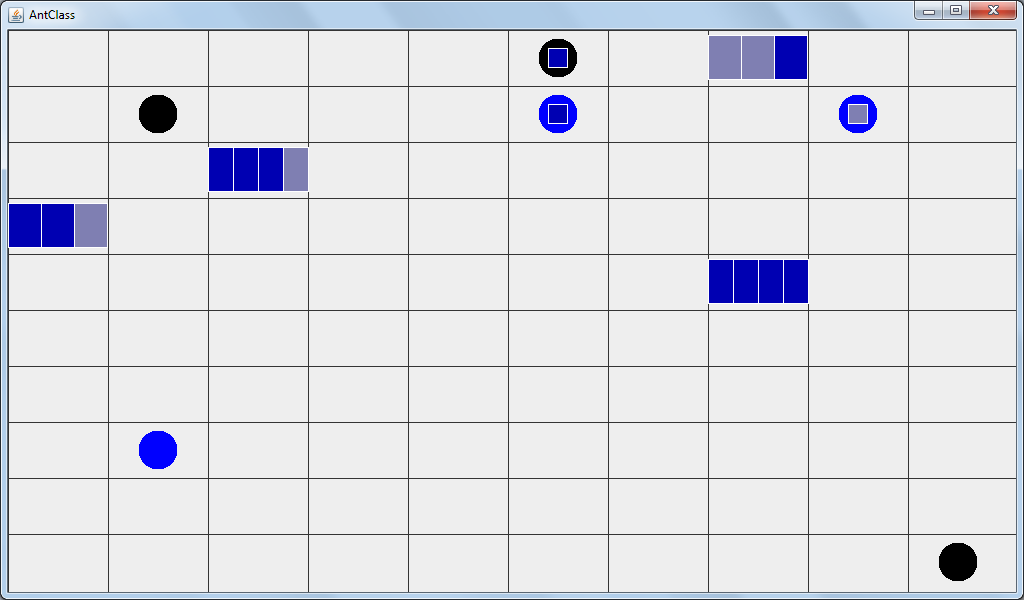


Figure 4. Full board view during runtime. Black ants belong to a client 1, blue ants belong to a client 2. Ants with a rectangle on top are considered to carry the heap object of a type indicated by the color of rectangle.

General structure of the project: To be changed

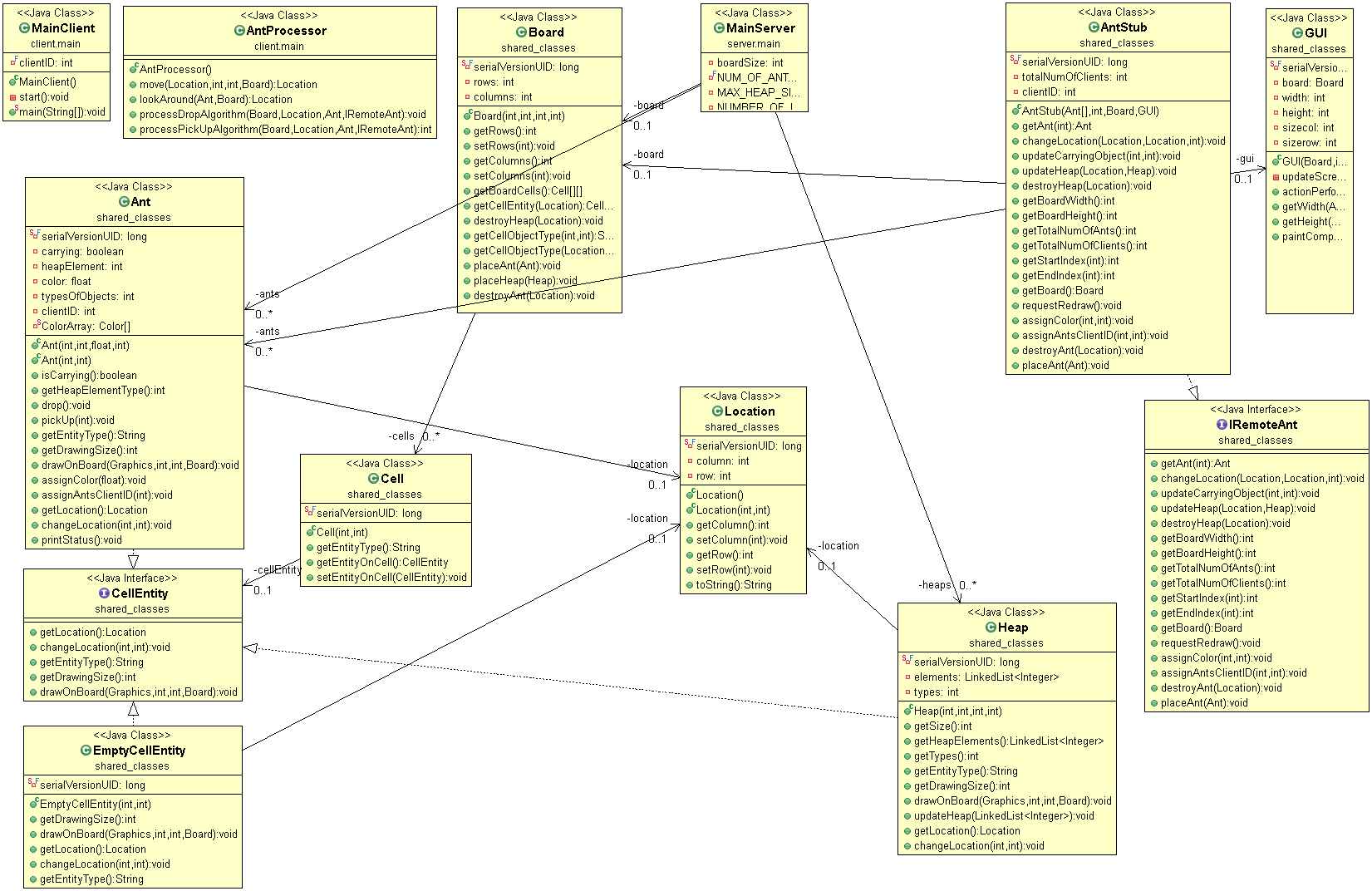


Figure 4. Class diagram of a project.