# Codebase Documentation

Daniel Swaab 4237455
Sille Kamoen 1534866

June 2, 2014

This document will contain a simple discreption about the codebase of BW4T. We will use som UML-diagrams to make relations clear. Use this when you want to refactor/understand the code. It took us hours to comprehend the code before we could start refactoring, we hope this will allow other people to get an understanding of the code in a shorter time period.

## The begin

The project is split up into 3 sub-projects: *Core, Client and Server*.

- The Server project contains all classes needed to run a BW4T-ServerInstance.
- The Core project contains classes that are shared by both client and server. It can be seen as a shared library.
- The Client project contains all classes to run a BW4T-ClientInstane.

### EIS

A lot of classes make use of the *EIS* library. *EIS* is an Environment Interface Standard. Consider this a black box that you use to implement environments. It contains function/methods/interfaces to run environments.

> Sille Kamoen: "You dont know what it is, but you do know what it does"

***INSERT IMAGE***

## Server

You would think this is a client-server architecture but it's more complicated. The central class is the Singleton *BW4TEnvironment.java*. This is a class that contains the main method to start the server entity. From javadoc:

> The central environment which runs the data model and performs actions received from remote environments through the server. Remote environments also poll percepts from this environment. Remote environments are notified of entity and environment events also using the server.

The environment uses a "BW4T-server" instance to communicate with the connecting clients. The name "server" can be a little confusing because of this, because it's mostly a list of connected clients that can be accessed by the Environment instead of the actual controlling more typical 'server' role. The environment extends an *AbstractEnvironment* from the "EIS" library. This means that most of its functionality comes from the EIS standard, on which we won't go into much detail for now.

Another attribute is the Repast Context. Repast is a comprehensive agent modeling platform. Currently, BW4T makes little use of the advanced functionality and even has some custom classes to make it compatible with GOAL. Examples of these classes are the Stepper and BW4TRunner. Lastly, the server package contains the BW4TLogger, which is a custom logger class (likely to be replaced).

Moving on to the other packages, we find the visualizations package. This big package takes care of the GUI, containing an overview of the rooms, blocks and bots when you launch the server. It uses some parts of Repasts displaying functionality, with a custom ServerMapRenderer class to actually draw the panels. The ServerMapRenderer uses BoundedMovableObjectStyle to know which object to run. The actual classes for these objects from the last big portion of classes in the Server package. Divided over zone, bw4t, robots, doors and blocks, we find the objects that can exist within an environment. BoundedMovableObject is the superclass, and each package represents a child. Zone is a square in the map, which can be a Corridor or a Room. A Room can be a (regular) BlocksRoom or a Dropzone.

# Core

Moving on to the Core project. This project should be treated as a library for classes used by both the Server and the Client classes. It contains a relatively large map package, which consists of various classes used to draw the map on both the Server and the Client. In addition to the map package, the Core Project contains interfaces for the Server and the Client, called BW4TClientActions and BW4TServerActions respectively. These interfaces contain all methods that can be used throughout the entire project. This is why you encounter several objects of the type BW3TServerActions instead of the regular BW4TServer in the client code. The client can access the interface, but not the BW4TServer class itself.

## Client

The Client project is the 3rd sub-project in the BW4T project. The client puts an Agent in the environment by connecting to the Server. It contains a RemoteEnvironment, which is an extension of the EIS RemoteEnvironment class. It communicates directly with the Enviroment using EnvironmentListeners. This does not go through the server! The Server class is only used to register connected clients.

A very large part of the client is once again in the visualizations. The BW4TClientMapRenderer is mostly unexplored, but uses a large number of listeners to collect all data required to draw the map.