HomeBase

# DynamicMapApplication v1.0

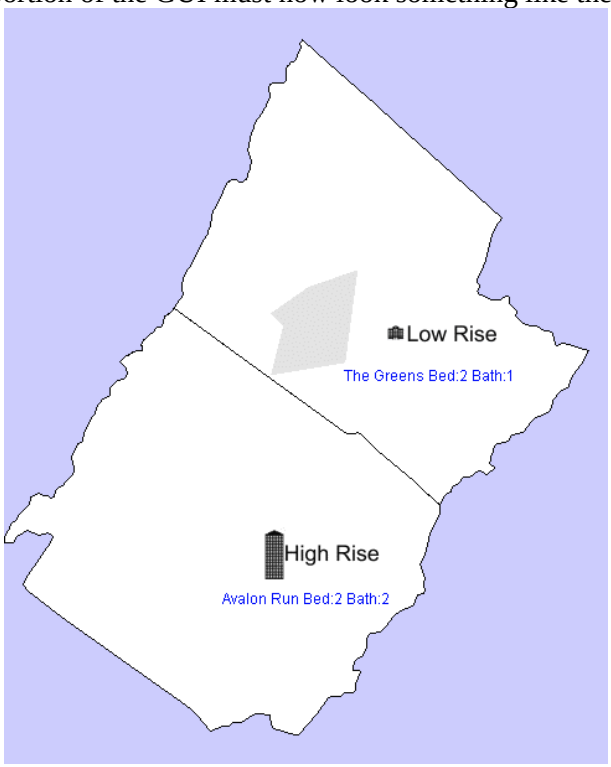
## Overview

`DynamicMapApplication` is a simple GUI-based application that allows users to view an animated visualization of the urbanized area (along with optional property listings) on a map.

## Interaction Design

In addition to the specifications for the `StaticMapApplication`, the `DynamicMapApplication` must satisfy the following interaction design specifications.

**ID1.** The relevant portion of the GUI must now look something like the following:




**ID16.** The system must display an animated "urbanized area" on top of the property listings.

**ID17.** The animated "urbanized area" must be rendered in gray and must be 50% transparent..

**ID18.** The animated "urbanized area" must be created from polygons that are provided for specific key times.

**ID19.** At times between the key times, the shape of the "urbanized area" must be determined using linear interpolation.

**ID20.** The animated "urbanized area" must loop (i.e., must repeat after the last key time).

**ID21.** The amount of time between “ticks” must be 100 milliseconds.

## Data Files

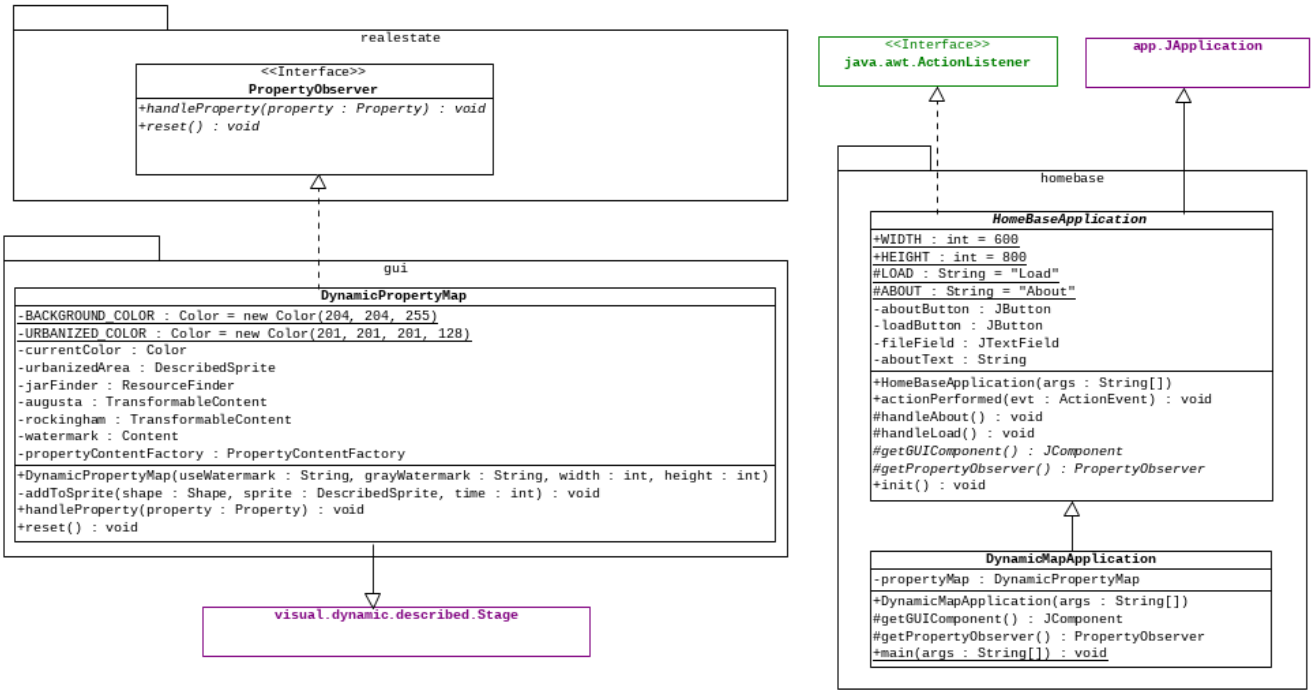
The history of how the urbanized area has evolved consists of three geographic boundary files representing the shape of the urbanized area at times 0, 1500 and 4000. They are named urban-0.map, urban-1500.map, and urban-4000.map. These files must not be included in the .jar file (i.e., they must be read from the current working directory of the local file system). Since this is just a demonstration, the file names should be hard-coded.

## Deployment

This application must be deployed as a single `.jar` file named `DynamicMapApplication.jar`. It must contain the same files in the `resources` package as the `StaticMapApplication` (since it provides all of the functionality as that application).

## Class Diagram

The relationships between the various classes and interfaces of the system are illustrated in the following UML class diagram (which is in addition to the class diagrams created earlier).




Classes/interfaces that are shown in jade green are part of the Java API. Classes/interfaces that are show in magenta are part of the Multimedia API.