HomeBase

StaticMapApplication v1.0

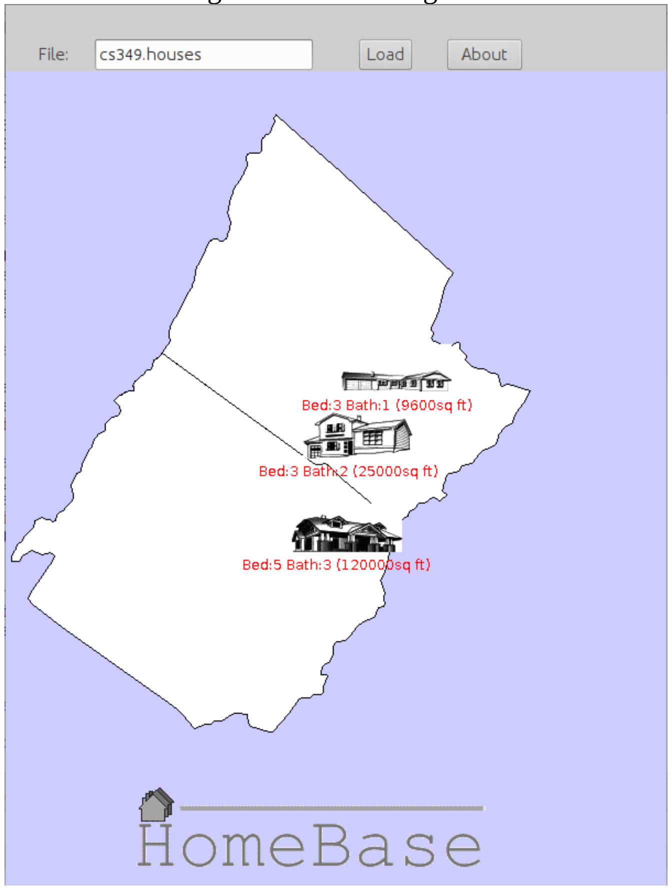
Overview

`StaticMapApplication` is a simple GUI-based application that allows users to view either houses or apartments on a map.

Interaction Design

The system must satisfy the following interaction design requirements.

ID1. The GUI must look something like the following:



ID2. For both apartments and houses, an icon must be displayed that illustrates the associated style.

ID3. The icon for each station must be centered at that property's location.

ID4. For apartments, a line of text must appear centered below the icon that is a terse representation of the apartment.

ID5. For houses, a line of text must appear centered below the icon that is a terse representation of the house.

ID6. The color of the text must alternate between red and blue, each time a file is loaded.

ID7. The apartments/houses must be cleared each time a file is loaded (before the new apartments/houses are displayed).

ID8. The geographic boundary of the area of interest must be black.

ID9. The interior of the area of interest must be white.

ID10. The exterior of the area of interest must be light blue.

ID12. At start-up, the user must be able to indicate that the display should be "watermarked" (by making command-line parameter 0 non-null) using the the HomeBase logo.

ID13. At start-up, the user must be able to indicate that the watermark must be a "grayed-out" version of the HomeBase logo (by making command-line parameter 1 non-null).

ID14. The watermark must be centered horizontally and must be at the bottom of the map.

ID15. The watermark must be rendered "underneath" any other content (i.e., it must be covered by any content that happens to be rendered in the same part of the window).

Deployment

This application must be deployed as a single `.jar` file named `StaticMapApplication.jar`. It must contain all of the .map, .png, and .loc files in the `resources` package. Note that the `.jar` file must not contain any other resources. In particular, it must not contain a grayed-out version of the HomeBase logo. The grayed-out version must be constructed at run-time.

Note that, for testing purposes, the current geographic boundary files and the Zip code locations are for the counties of Rockingham and Augusta in Virginia. Ultimately, they will be replaced by files containing information for the entire world. That is why they are resources that included in the `.jar` file rather than being read from the file system.

Format of Geographic Boundary Files

Geographic boundary files files (`.map` files) contain the vertices of polygons. Each record is comma-delimited and "new line" terminated. Each record contains three fields.

Field 1

An integer that indicates the type of "drawing operation" to be performed.

Value	Meaning
4	Move to
5	Line to

Field 2

A floating point number containing the horizontal position of the vertex (in screen coordinates).

Field 3

A floating point number containing the vertical position of the vertex (in screen coordinates).

Format of Centroid Files

Centroid files (`.loc` files) contain the locations of centroids of polygons. Each record is comma-delimited and "new line" terminated. Each record contains four fields.

Field 1

A textual identifier (e.g., a Zip code).

Field 2

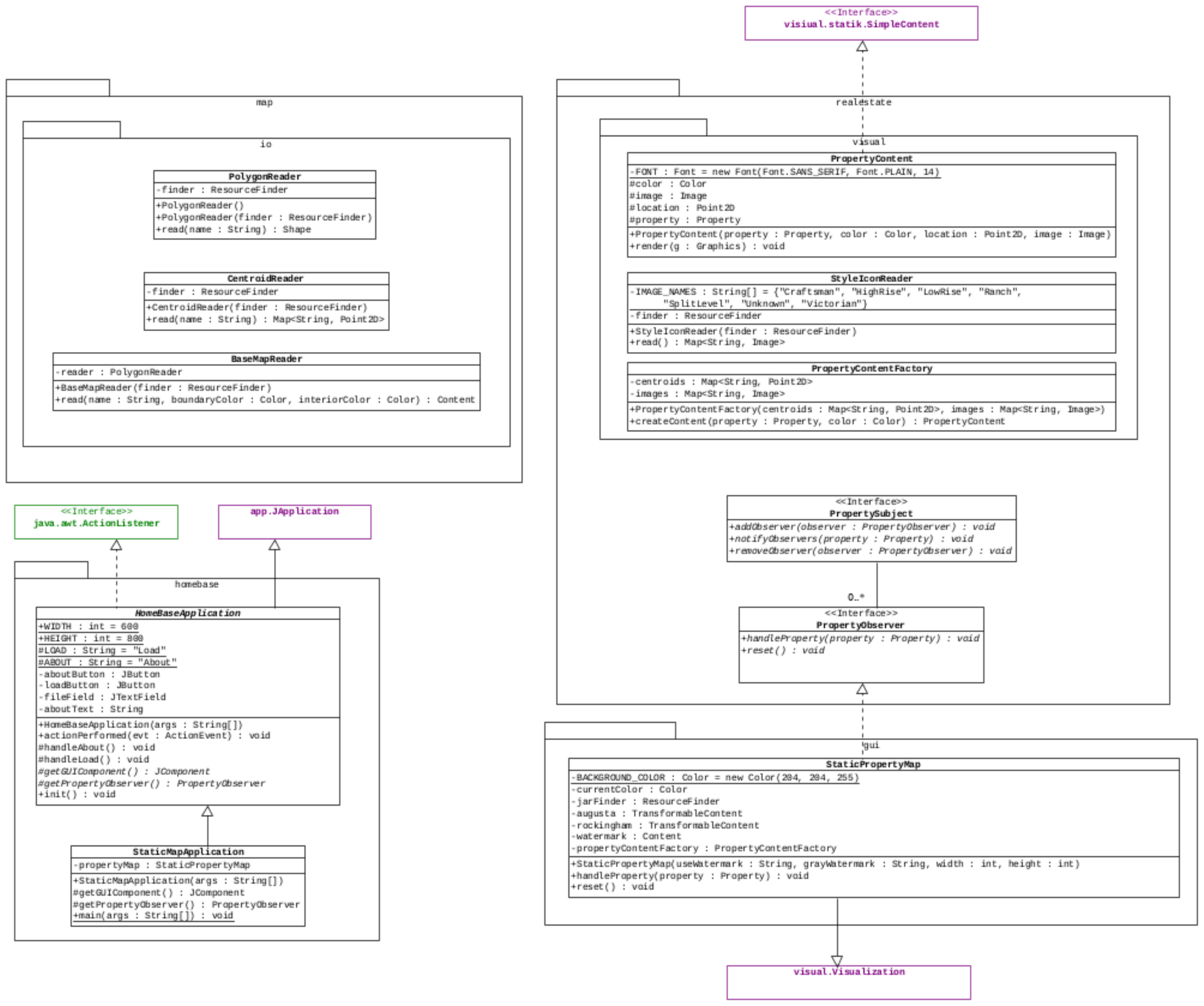
An integer containing the horizontal position of the point (in screen coordinates).

Field 3

An integer containing the vertical position of the point (in screen coordinates).

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).



<<Interface>>
visiual.statik.SimpleContent
map
io
PolygonReader
-finder : ResourceFinder
+PolygonReader()
+PolygonReader(finder : ResourceFinder)
+read(name : String) : Shape
CentroidReader
-finder : ResourceFinder
+CentroidReader(finder : ResourceFinder)
+read(name : String) : Map<String, Point2D>
BaseMapReader
-reader : PolygonReader
+BaseMapReader(finder : ResourceFinder)
+read(name : String, boundaryColor : Color, interiorColor : Color) : Content
realestate
visual
PropertyContent
-FONT : Font = new Font(Font.SANS_SERIF, Font.PLAIN, 14)
#color : Color
#image : Image
#location : Point2D
#property : Property
+PropertyContent(property : Property, color : Color, location : Point2D, image : Image)
+render(g : Graphics) : void
StyleIconReader
-IMAGE_NAMES : String[] = {"Craftsman", "HighRise", "LowRise", "Ranch", "SplitLevel", "Unknown", "Victorian"}
-finder : ResourceFinder
+StyleIconReader(finder : ResourceFinder)
+read() : Map<String, Image>
PropertyContentFactory
-centroids : Map<String, Point2D>
-images : Map<String, Image>
+PropertyContentFactory(centroids : Map<String, Point2D>, images : Map<String, Image>)
+createContent(property : Property, color : Color) : PropertyContent
<<Interface>>
PropertySubject
+addObserver(observer : PropertyObserver) : void
+notifyObservers(property : Property) : void
+removeObserver(observer : PropertyObserver) : void
0..*
<<Interface>>
PropertyObserver
+handleProperty(property : Property) : void
+reset() : void
<<Interface>>
java.awt.ActionListener
app.JApplication
homebase
HomeBaseApplication
+WIDTH : int = 600
+HEIGHT : int = 800
#LOAD : String = "Load"
#ABOUT : String = "About"
-aboutButton : JButton
-loadButton : JButton
-fileField : JTextField
-aboutText : String
+HomeBaseApplication(args : String[])
+actionPerformed(evt : ActionEvent) : void
#handleAbout() : void
#handleLoad() : void
#getGUIComponent() : JComponent
#getPropertyObserver() : PropertyObserver
+init() : void
StaticMapApplication
-propertyMap : StaticPropertyMap
+StaticMapApplication(args : String[])
#getGUIComponent() : JComponent
#getPropertyObserver() : PropertyObserver
+main(args : String[]) : void
gui
StaticPropertyMap
-BACKGROUND_COLOR : Color = new Color(204, 204, 255)
-currentColor : Color
-jarFinder : ResourceFinder
-augusta : TransformableContent
-rockingham : TransformableContent
-watermark : Content
-propertyContentFactory : PropertyContentFactory
+StaticPropertyMap(useWatermark : String, grayWatermark : String, width : int, height : int)
+handleProperty(property : Property) : void
+reset() : void
visual.Visualization

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.

In addition to the specifications that are contained in this class diagram, the implementation must comply with the following specifications.

The `PolygonReader` Class

A `PolygonReader` object can be used to read polygon objects from a stream.

Methods

```
    Shape read(final String name)
```

Must read a polygon from either the resource (if the `finder` attribute is non-`null`) or the file (if the `finder` attribute is `null`) with the given `name`.

The `BaseMapReader` Class

A `BaseMapReader` object can be used to read polygon objects from a stream.

Methods

```
    Content read(final String name,
                 final Color boundary, final Color interior)
```

Must use the `reader` attribute to read a polygon and construct a `Content` object with the appropriate properties.

The `CentroidReader` Class

A `CentroidReader` object can be used to read information about centroids.

Methods

```
    java.util.Map<String, Point2D> read(final String name)
```

Must use the `finder` attribute to read information about a collection of centroids from a geographic location file.

The first field of the centroid file will be the code/identifier for the centroid. This code/identifier must be used as the key, and the horizontal and vertical positions must be used as the value (i.e., the `Point2D`).

The `PropertyContent` Class

A `visual.statik.SimpleContent` object that presents the information in a `Property` object.

The `StyleIconReader` Class

A `StyleIconReader` object can be used to read all of the icons used to present different property styles.

Methods

```
    Map<String, Image> read() throws IOException
```

Must use the `finder` attribute to read information about a collection of icons. The key in the `Map` must be the property style (e.g., `"Craftsman", "HighRise", "LowRise", "Ranch", "SplitLevel", "Unknown", "Victorian"`).