**DESIGN DECISIONS**

A brief summary of the various classes used in the software and their responsibilities:

|  |  |
| --- | --- |
| Class | Responsibility |
| AbstractCityFactory | An abstract factory for generating City instances. |
| AbstractRouteFactory | An abstract factory for generating Route instances. |
| ActionConnectOperation | The strategy which defines the algorithm for connecting cities on mouse press / drag / release actions in the drawable area. It extends ActionStrategy. |
| ActionCreateOperation | The strategy which defines the algorithm for creating cities on mouse press action in the drawable area. It extends ActionStrategy. |
| ActionMoveOperation | The strategy which defines the algorithm for moving cities on mouse press / drag / release actions in the drawable area. It extends ActionStrategy. |
| ActionStrategy | An abstract strategy for implementing the strategy to handle the mouse press / drag / release actions in the drawable area. |
| CircleDecorator | The concrete decorator in the decorator pattern, for decorating the city shape with a circle inside the central rectangle. It extends ShapeDecorator. |
| City | The data structure for city information. |
| CityCenter | The concrete component in the decorator pattern applied to decorating the city shape. It implements the ShapeInterface interface. |
| CityCircleDecorationHandler | The concrete handler in the Chain of Responsibility pattern, for applying circle decoration to a city. It extends CityDecorationHandler. |
| CityDecorationHandler | The abstract handler in the Chain of Responsibility pattern, for applying decorations to a city. |
| CityFactory | A concrete factory for generating City instances. It extends AbstractCityFactory. |
| CityRepository | The repository for City. |
| CitySquareDecorationHandler | The concrete handler in the Chain of Responsibility pattern, for applying square decorations to a city. It extends CityDecorationHandler. |
| ConnectionClustering | The strategy which defines the Clustering algorithm for connecting cities in the drawable area. It extends ActionStrategy. |
| ConnectionContext | The context class which controls the algorithm for connecting cities. |
| ConnectionStrategy | An abstract strategy for implementing the strategy to connect the cities in the drawable area. |
| ConnectionTSPNearestNeighbour | The strategy which defines the TSP Nearest Neighbour algorithm for connecting cities in the drawable area. It extends ActionStrategy. |
| ConnectionTSPPro | The strategy which defines the TSP Pro algorithm for connecting cities in the drawable area. It extends ActionStrategy. |
| Container | The container interface in the iterator pattern. |
| EditCityDialog | A dialog to let the user edit the city properties such as label, size, color and shape decorations. |
| Iterator | The iterator interface in the iterator pattern. |
| Logger | The logger class. |
| MainFrame | The main class for the application which contains the main() method. |
| ObjectIterator | A generic concrete iterator which implements Iterator. |
| Route | The data structure for route information between two cities. |
| RouteFactory | A concrete factory for generating Route instances. It extends AbstractRouteFactory. |
| RouteRepository | The repository for Route. |
| ShapeDecorator | The abstract decorator in the decorator pattern, for decorating the city shape with squares around the sides of the central rectangle. It implements ShapeInterface. |
| ShapeInterface | The component interface in the decorator pattern, for decorating the city shape |
| SquareDecorator | The concrete decorator in the decorator pattern, for decorating the city shape. It extends ShapeDecorator. |
| TextAreaOutputStream | It provides an output stream to write data to a JTextArea. It is used by the Logger for redirecting the output to the GUI. |
| WorkSpacePanel | The drawable area |

**Design Decision #1**: Implement Factory and Singleton Patterns for generating City and Route objects respectively

**Design Decision #2**: Implement Strategy Pattern for the various Actions i.e., Create, Move, and Connect

**Design Decision #3**: Implement Strategy Pattern for the various algorithms for connecting cities i.e., TSP – Nearest Neighbour, TSP -Pro, Clustering, and User Connect

**Design Decision #4**: Implement Observer Pattern for running the connection algorithm in background

**Design Decision #5**: Implement Decorator Pattern for modifying the city shape

**Design Decision #6**: Implement Chain of Responsibility Pattern for applying the decorations to the city shape

**Design Decision #7**: Implement Singleton Pattern for city and route repository

**Design Decision #8**: Implement Iterator Pattern for city and route list contained within their respective repositories

**Design Decision #9**: Implement Singleton Pattern for logger

**Design Decision #10**: Implement Observer Pattern for drawing the cities and routes on the drawable area