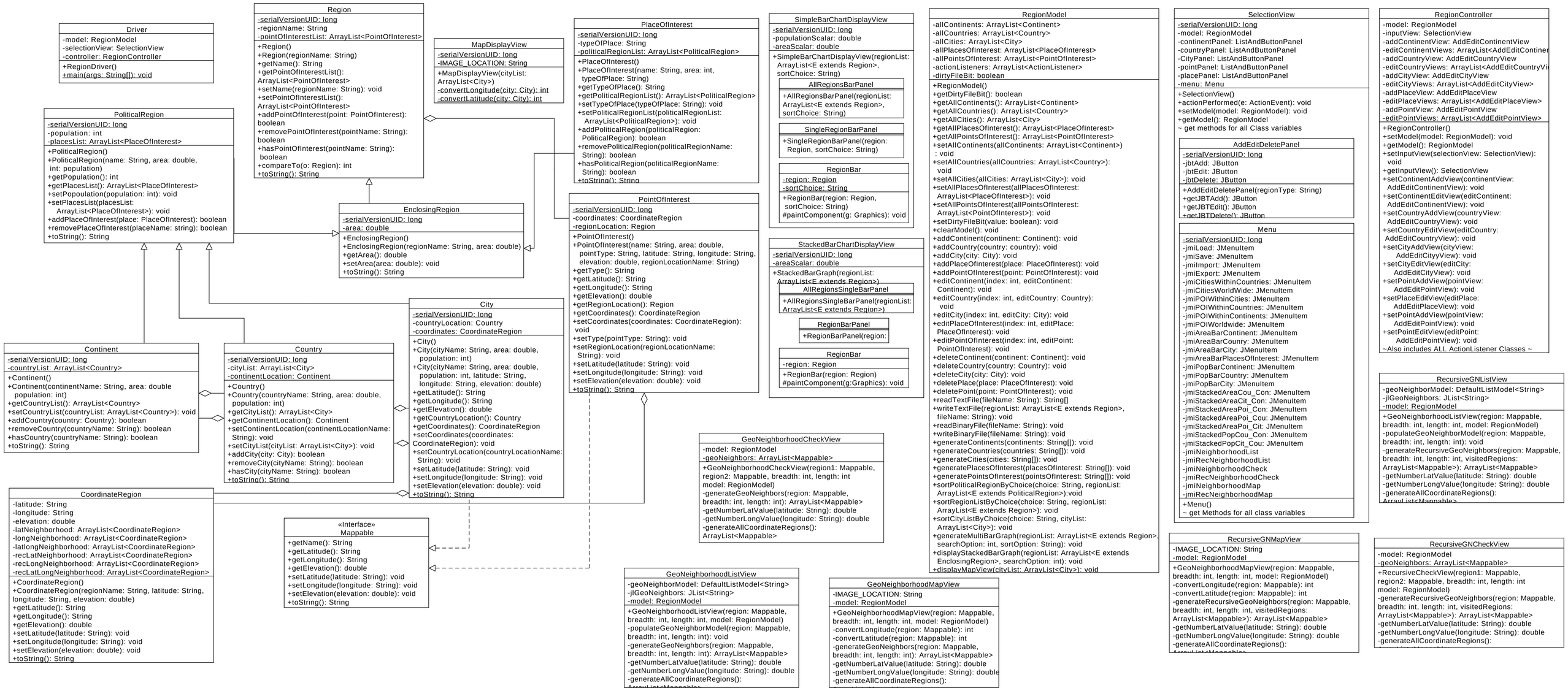
Driver
-model: RegionModel
-selectionView: SelectionView
-controller: RegionController
+RegionDriver()
+main(args: String[]): void
Region
-serialVersionUID: long
-regionName: String
-pointOfInterestList: ArrayList<PointOfInterest>
+Region()
+Region(regionName: String)
+getName(): String
+getPointOfInterestList(): ArrayList<PointOfInterest>
+setName(regionName: String): void
+setPointOfInterestList(): ArrayList<PointOfInterest>
+addPointOfInterest(point: PointOfInterest): boolean
+removePointOfInterest(pointName: String): boolean
+hasPointOfInterest(pointName: String): boolean
+compareTo(o: Region): int
+toString(): String
MapDisplayView
-serialVersionUID: long
-IMAGE_LOCATION: String
+MapDisplayView(cityList: ArrayList<City>)
-convertLongitude(city: City): int
-convertLatitude(city: City): int
PlaceOfInterest
-serialVersionUID: long
-typeOfPlace: String
-politicalRegionList: ArrayList<PoliticalRegion>
+PlaceOfInterest()
+PlaceOfInterest(name: String, area: int, typeOfPlace: String)
+getTypeOfPlace(): String
+getPoliticalRegionList(): ArrayList<PoliticalRegion>
+setTypeOfPlace(typeOfPlace: String): void
+setPoliticalRegionList(politicalRegionList: ArrayList<PoliticalRegion>): void
+addPoliticalRegion(politicalRegion: PoliticalRegion): boolean
+removePoliticalRegion(politicalRegionName: String): boolean
+hasPoliticalRegion(politicalRegionName: String): boolean
+toString(): String
SimpleBarChartDisplayView
-serialVersionUID: long
-populationScalar: double
-areaScalar: double
+SimpleBarChartDisplayView(regionList: ArrayList<E extends Region>, sortChoice: String)
AllRegionsBarPanel
+AllRegionsBarPanel(regionList: ArrayList<E extends Region>, sortChoice: String)
SingleRegionBarPanel
+SingleRegionBarPanel(region: Region, sortChoice: String)
RegionBar
-region: Region
-sortChoice: String
+RegionBar(region: Region, sortChoice: String)
#paintComponent(g: Graphics): void
RegionModel
-allContinents: ArrayList<Continent>
-allCountries: ArrayList<Country>
-allCities: ArrayList<City>
-allPlacesOfInterest: ArrayList<PlaceOfInterest>
-allPointsOfInterest: ArrayList<PointOfInterest>
-actionListeners: ArrayList<ActionListener>
-dirtyFileBit: boolean
+RegionModel()
+getDirtyFileBit(): boolean
+getAllContinents(): ArrayList<Continent>
+getAllCountries(): ArrayList<Country>
+getAllCities(): ArrayList<City>
+getAllPlacesOfInterest(): ArrayList<PlaceOfInterest>
+getAllPointsOfInterest(): ArrayList<PointOfInterest>
+setAllContinents(allContinents: ArrayList<Continent>) : void
+setAllCountries(allCountries: ArrayList<Country>): void
+setAllCities(allCities: ArrayList<City>): void
+setAllPlacesOfInterest(allPlacesOfInterest: ArrayList<PlaceOfInterest>): void
+setAllPointsOfInterest(allPointsOfInterest: ArrayList<PointOfInterest>): void
+setDirtyFileBit(value: boolean): void
+clearModel(): void
+addContinent(continent: Continent): void
+addCountry(country: country): void
+addCity(city: City): void
+addPlaceOfInterest(place: PlaceOfInterest): void
+addPointOfInterest(point: PointOfInterest): void
+editContinent(index: int, editContinent: Continent): void
+editCountry(index: int, editCountry: Country): void
+editCity(index: int, editCity: City): void
+editPlaceOfInterest(index: int, editPlace: PlaceOfInterest): void
+editPointOfInterest(index: int, editPoint: PointOfInterest): void
+deleteContinent(continent: Continent): void
+deleteCountry(country: Country): void
+deleteCity(city: City): void
+deletePlace(place: PlaceOfInterest): void
+deletePoint(point: PointOfInterest): void
+readTextFile(fileName: String): String[]
+writeTextFile(regionList: ArrayList<E extends Region>, fileName: String): void
+readBinaryFile(fileName: String): void
+writeBinaryFile(fileName: String): void
+generateContinents(continents: String[]): void
+generateCountries(countries: String[]): void
+generateCities(cities: String[]): void
+generatePlacesOfInterest(placesOfInterest: String[]): void
+generatePointsOfInterest(pointsOfInterest: String[]): void
+sortPoliticalRegionByChoice(choice: String, regionList: ArrayList<E extends PoliticalRegion>):void
+sortRegionListByChoice(choice: String, regionList: ArrayList<E extends Region>): void
+sortCityListByChoice(choice: String, cityList: ArrayList<City>): void
+generateMultiBarGraph(regionList: ArrayList<E extends Region>, searchOption: int, sortOption: String): void
+displayStackedBarGraph(regionList: ArrayList<E extends EnclosingRegion>, searchOption: int): void
+displayMapView(cityList: ArrayList<City>): void
SelectionView
-serialVersionUID: long
-model: RegionModel
-continentPanel: ListAndButtonPanel
-countryPanel: ListAndButtonPanel
-CityPanel: ListAndButtonPanel
-pointPanel: ListAndButtonPanel
-placePanel: ListAndButtonPanel
-menu: Menu
+SelectionView()
+actionPerformed(e: ActionEvent): void
+setModel(model: RegionModel): void
+getModel(): RegionModel
~ get methods for all Class variables
AddEditDeletePanel
-serialVersionUID: long
-jbtAdd: JButton
-jbtEdit: JButton
-jbtDelete: JButton
+AddEditDeletePanel(regionType: String)
+getJBTAdd(): JButton
+getJBTEdit(): JButton
+getJBTDelete(): JButton
Menu
-serialVersionUID: long
-jmiLoad: JMenuItem
-jmiSave: JMenuItem
-jmiImport: JMenuItem
-jmiExport: JMenuItem
-jmiCitiesWithinCountries: JMenuItem
-jmiCitiesWorldWide: JMenuItem
-jmiPOIWithinCities: JMenuItem
-jmiPOIWithinCountries: JMenuItem
-jmiPOIWithinContinents: JMenuItem
-jmiPOIWorldwide: JMenuItem
-jmiAreaBarContinent: JMenuItem
-jmiAreaBarCountry: JMenuItem
-jmiAreaBarCity: JMenuItem
-jmiAreaBarPlacesOfInterest: JMenuItem
-jmiPopBarContinent: JMenuItem
-jmiPopBarCountry: JMenuItem
-jmiPopBarCity: JMenuItem
-jmiStackedAreaCou_Con: JMenuItem
-jmiStackedAreaCit_Con: JMenuItem
-jmiStackedAreaPoi_Con: JMenuItem
-jmiStackedAreaPoi_Cou: JMenuItem
-jmiStackedAreaPoi_Cit: JMenuItem
-jmiStackedPopCou_Con: JMenuItem
-jmiStackedPopCit_Cou: JMenuItem
-jmiNeighborhoodList
-jmiRecNeighborhoodList
-jmiNeighborhoodCheck
-jmiRecNeighborhoodCheck
-jmiNeighborhoodMap
-jmiRecNeighborhoodMap
+Menu()
~ get Methods for all class variables
RegionController
-model: RegionModel
-inputView: SelectionView
-addContinentView: AddEditContinentView
-editContinentViews: ArrayList<AddEditContiner
-addCountryView: AddEditCountryView
-editCountryViews: ArrayList<AddEditCountryVi
-addCityView: AddEditCityView
-editCityViews: ArrayList<AddEditCityView>
-addPlaceView: AddEditPlaceView
-editPlaceViews: ArrayList<AddEditPlaceView>
-addPointView: AddEditPointView
-editPointViews: ArrayList<AddEditPointView>
+RegionController()
+setModel(model: RegionModel): void
+getModel(): RegionModel
+setInputView(selectionView: SelectionView): void
+getInputView(): SelectionView
+setContinentAddView(continentView: AddEditContinentView): void
+setContinentEditView(editContinent: AddEditContinentView): void
+setCountryAddView(countryView: AddEditCountryView): void
+setCountryEditView(editCountry: AddEditCountryView): void
+setCityAddView(cityView: AddEditCityView): void
+setCityEditView(editCity: AddEditCityView): void
+setPointAddView(pointView: AddEditPointView): void
+setPlaceEditView(editPlace: AddEditPlaceView): void
+setPointAddView(pointView: AddEditPointView): void
+setPointEditView(editPoint: AddEditPointView): void
~Also includes ALL ActionListener Classes ~
PoliticalRegion
-serialVersionUID: long
-population: int
-placesList: ArrayList<PlaceOfInterest>
+PoliticalRegion()
+PoliticalRegion(name: String, area: double, int: population)
+getPopulation(): int
+getPlacesList(): ArrayList<PlaceOfInterest>
+setPopoulation(population: int): void
+setPlacesList(placesList: ArrayList<PlaceOfInterest>): void
+addPlaceOfInterest(place: PlaceOfInterest): boolean
+removePlaceOfInterest(placeName: string): boolean
+toString(): String
EnclosingRegion
-serialVersionUID: long
-area: double
+EnclosingRegion()
+EnclosingRegion(regionName: String, area: double)
+getArea(): double
+setArea(area: double): void
+toString(): String
PointOfInterest
-serialVersionUID: long
-coordinates: CoordinateRegion
-regionLocation: Region
+PointOfInterest()
+PointOfInterest(name: String, area: double, pointType: String, latitude: String, longitude: String, elevation: double, regionLocationName: String)
+getType(): String
+getLatitude(): String
+getLongitude(): String
+getElevation(): double
+getRegionLocation(): Region
+getCoordinates(): CoordinateRegion
+setCoordinates(coordinates: CoordinateRegion): void
+setType(pointType: String): void
+setRegionLocation(regionLocationName: String): void
+setLatitude(latitude: String): void
+setLongitude(longitude: String): void
+setElevation(elevation: double): void
+toString(): String
StackedBarChartDisplayView
-serialVersionUID: long
-areaScalar: double
+StackedBarGraph(regionList: ArrayList<E extends Region>)
AllRegionsSingleBarPanel
+AllRegionsSingleBarPanel(regionList: ArrayList<E extends Region>)
RegionBarPanel
+RegionBarPanel(region:
RegionBar
-region: Region
+RegionBar(region: Region)
#paintComponent(g:Graphics): void
City
-serialVersionUID: long
-countryLocation: Country
-coordinates: CoordinateRegion
+City()
+City(cityName: String, area: double, population: int)
+City(cityName: String, area: double, population: int, latitude: String, longitude: String, elevation: double)
+getLatitude(): String
+getLongitude(): String
+getElevation(): double
+getCountryLocation(): Country
+getCoordinates(): CoordinateRegion
+setCoordinates(coordinates: CoordinateRegion): void
+setCountryLocation(countryLocationName: String): void
+setLatitude(latitude: String): void
+setLongitude(longitude: String): void
+setElevation(elevation: double): void
+toString(): String
Continent
-serialVersionUID: long
-countryList: ArrayList<Country>
+Continent()
+Continent(continentName: String, area: double population: int)
+getCountryList(): ArrayList<Country>
+setCountryList(countryList: ArrayList<Country>): void
+addCountry(country: Country): boolean
+removeCountry(countryName: String): boolean
+hasCountry(countryName: String): boolean
+toString(): String
Country
-serialVersionUID: long
-cityList: ArrayList<City>
-continentLocation: Continent
+Country()
+Country(countryName: String, area: double, population: int)
+getCityList(): ArrayList<City>
+getContinentLocation(): Continent
+setContinentLocation(continentLocationName: String): void
+setCityList(cityList: ArrayList<City>): void
+addCity(city: City): boolean
+removeCity(cityName: String): boolean
+hasCity(cityName: String): boolean
+toString(): String
GeoNeighborhoodCheckView
-model: RegionModel
-geoNeighbors: ArrayList<Mappable>
+GeoNeighborhoodCheckView(region1: Mappable, region2: Mappable, breadth: int, length: int model: RegionModel)
-generateGeoNeighbors(region: Mappable, breadth: int, length: int): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>
RecursiveGNListView
-geoNeighborModel: DefaultListModel<String>
-jlGeoNeighbors: JList<String>
-model: RegionModel
+GeoNeighborhoodListView(region: Mappable, breadth: int, length: int, model: RegionModel)
-populateGeoNeighborModel(region: Mappable, breadth: int, length: int): void
-generateRecursiveGeoNeighbors(region: Mappable, breadth: int, length: int, visitedRegions: ArrayList<Mappable>): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>
CoordinateRegion
-latitude: String
-longitude: String
-elevation: double
-latNeighborhood: ArrayList<CoordinateRegion>
-longNeighborhood: ArrayList<CoordinateRegion>
-latlongNeighborhood: ArrayList<CoordinateRegion>
-recLatNeighborhood: ArrayList<CoordinateRegion>
-recLongNeighborhood: ArrayList<CoordinateRegion>
-recLatLongNeighborhood: ArrayList<CoordinateRegion>
+CoordinateRegion()
+CoordinateRegion(regionName: String, latitude: String, longitude: String, elevation: double)
+getLatitude(): String
+getLongitude(): String
+getElevation(): double
+setLatitude(latitude: String): void
+setLongitude(longitude: String): void
+setElevation(elevation: double): void
+toString(): String
«Interface»
Mappable
+getName(): String
+getLatitude(): String
+getLongitude(): String
+getElevation(): double
+setLatitude(latitude: String): void
+setLongitude(longitude: String): void
+setElevation(elevation: double): void
+toString(): String
GeoNeighborhoodListView
-geoNeighborModel: DefaultListModel<String>
-jlGeoNeighbors: JList<String>
-model: RegionModel
+GeoNeighborhoodListView(region: Mappable, breadth: int, length: int, model: RegionModel)
-populateGeoNeighborModel(region: Mappable, breadth: int, length: int): void
-generateGeoNeighbors(region: Mappable, breadth: int, length: int): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>
GeoNeighborhoodMapView
-IMAGE_LOCATION: String
-model: RegionModel
+GeoNeighborhoodMapView(region: Mappable, breadth: int, length: int, model: RegionModel)
-convertLongitude(region: Mappable): int
-convertLatitude(region: Mappable): int
-generateGeoNeighbors(region: Mappable, breadth: int, length: int): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>
RecursiveGNMapView
-IMAGE_LOCATION: String
-model: RegionModel
+GeoNeighborhoodMapView(region: Mappable, breadth: int, length: int, model: RegionModel)
-convertLongitude(region: Mappable): int
-convertLatitude(region: Mappable): int
-generateRecursiveGeoNeighbors(region: Mappable, breadth: int, length: int, visitedRegions: ArrayList<Mappable>): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>
RecursiveGNCheckView
-model: RegionModel
-geoNeighbors: ArrayList<Mappable>
+RecursiveCheckView(region1: Mappable, region2: Mappable, breadth: int, length: int model: RegionModel)
-generateRecursiveGeoNeighbors(region: Mappable, breadth: int, length: int, visitedRegions: ArrayList<Mappable>): ArrayList<Mappable>
-getNumberLatValue(latitude: String): double
-getNumberLongValue(longitude: String): double
-generateAllCoordinateRegions(): ArrayList<Mappable>