

City Builder Simulation

v1.5.4

Generated by Doxygen 1.9.8

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	5
2.1 Class List	5
3 File Index	9
3.1 File List	9
4 Class Documentation	17
4.1 AggregateTraverser Class Reference	17
4.1.1 Detailed Description	18
4.1.2 Member Function Documentation	18
4.1.2.1 createCityTraverser() [1/3]	18
4.1.2.2 createCityTraverser() [2/3]	18
4.1.2.3 createCityTraverser() [3/3]	19
4.2 Airport Class Reference	19
4.2.1 Detailed Description	22
4.2.2 Constructor & Destructor Documentation	22
4.2.2.1 Airport()	22
4.2.3 Member Function Documentation	22
4.2.3.1 calculateCommute()	22
4.2.3.2 getAirportName()	22
4.3 AllocateBudgetCommand Class Reference	23
4.3.1 Detailed Description	25
4.3.2 Constructor & Destructor Documentation	25
4.3.2.1 AllocateBudgetCommand()	25
4.3.3 Member Function Documentation	26
4.3.3.1 canExecute()	26
4.3.3.2 execute()	26
4.3.3.3 executeWithValidation()	26
4.3.3.4 getAmount()	26
4.3.3.5 getDescription()	27
4.3.3.6 getName()	27
4.3.3.7 getPrevAllocation()	27
4.3.3.8 returnVal()	27
4.3.3.9 setAmount()	27
4.3.3.10 setPrevAllocation()	28
4.3.3.11 undo()	28
4.3.3.12 validateAllocation()	28
4.4 BudgetApprovalHandler Class Reference	29
4.4.1 Detailed Description	31
4.4.2 Constructor & Destructor Documentation	31

4.4.2.1 BudgetApprovalHandler()	31
4.4.3 Member Function Documentation	31
4.4.3.1 handleRequest() [1/2]	31
4.4.3.2 handleRequest() [2/2]	32
4.5 Builder Class Reference	32
4.5.1 Detailed Description	34
4.5.2 Member Function Documentation	34
4.5.2.1 build()	34
4.5.2.2 setArea()	34
4.5.2.3 setCapacity()	35
4.5.2.4 setCitizenSatisfaction()	35
4.5.2.5 setEconomicGrowth()	35
4.5.2.6 setFloors()	36
4.5.2.7 setName()	36
4.5.2.8 setResourceConsumption()	36
4.6 Building Class Reference	37
4.6.1 Detailed Description	40
4.6.2 Constructor & Destructor Documentation	41
4.6.2.1 Building() [1/4]	41
4.6.2.2 Building() [2/4]	42
4.6.2.3 Building() [3/4]	42
4.6.2.4 Building() [4/4]	42
4.6.3 Member Function Documentation	43
4.6.3.1 addJob() [1/2]	43
4.6.3.2 addJob() [2/2]	43
4.6.3.3 construct() [1/2]	43
4.6.3.4 construct() [2/2]	43
4.6.3.5 displayJobInfo() [1/2]	44
4.6.3.6 displayJobInfo() [2/2]	44
4.6.3.7 getAvailableJob() [1/2]	44
4.6.3.8 getAvailableJob() [2/2]	44
4.6.3.9 getEconomicGrowth() [1/2]	44
4.6.3.10 getEconomicGrowth() [2/2]	45
4.6.3.11 getJobs() [1/2]	45
4.6.3.12 getJobs() [2/2]	45
4.6.3.13 getName() [1/2]	45
4.6.3.14 getName() [2/2]	45
4.6.3.15 getResourceConsumption() [1/2]	46
4.6.3.16 getResourceConsumption() [2/2]	46
4.6.3.17 getSatisfaction() [1/2]	46
4.6.3.18 getSatisfaction() [2/2]	46
4.6.3.19 getType() [1/2]	46

4.6.3.20 getType() [2/2]	47
4.6.3.21 hireEmployee() [1/2]	47
4.6.3.22 hireEmployee() [2/2]	47
4.6.3.23 payTaxes() [1/2]	47
4.6.3.24 payTaxes() [2/2]	48
4.6.3.25 releaseEmployee() [1/2]	48
4.6.3.26 releaseEmployee() [2/2]	48
4.6.3.27 setName() [1/2]	49
4.6.3.28 setName() [2/2]	49
4.6.3.29 undoCollectTaxes() [1/2]	49
4.6.3.30 undoCollectTaxes() [2/2]	49
4.6.3.31 updateImpacts() [1/2]	49
4.6.3.32 updateImpacts() [2/2]	50
4.7 BuildingManager Class Reference	50
4.7.1 Detailed Description	51
4.7.2 Constructor & Destructor Documentation	51
4.7.2.1 BuildingManager()	51
4.7.3 Member Function Documentation	51
4.7.3.1 addBuilding()	51
4.7.3.2 addCitizen()	51
4.7.3.3 assignJobToCitizen()	52
4.7.3.4 findAvailableJob()	52
4.7.3.5 getBuildings()	52
4.7.3.6 listAllJobsInBuilding()	53
4.7.3.7 releaseCitizenFromJob()	54
4.8 Bus Class Reference	54
4.8.1 Detailed Description	57
4.8.2 Constructor & Destructor Documentation	58
4.8.2.1 Bus()	58
4.8.3 Member Function Documentation	58
4.8.3.1 addBus()	58
4.8.3.2 addComercialAirport()	59
4.8.3.3 addInsideRoad()	59
4.8.3.4 addPassengerTrain()	60
4.8.3.5 getBus()	60
4.8.3.6 getBusNumber()	61
4.8.3.7 getCapacity()	61
4.8.3.8 getComercialAirport()	61
4.8.3.9 getInsideRoad()	62
4.8.3.10 getPassengerTrain()	62
4.8.3.11 getRouteName()	63
4.9 Business Class Reference	63

4.9.1 Detailed Description	66
4.9.2 Constructor & Destructor Documentation	67
4.9.2.1 Business() [1/2]	67
4.9.2.2 Business() [2/2]	67
4.9.3 Member Function Documentation	67
4.9.3.1 addPolicy() [1/2]	67
4.9.3.2 addPolicy() [2/2]	67
4.9.3.3 addService() [1/2]	68
4.9.3.4 addService() [2/2]	68
4.9.3.5 calculateTax() [1/2]	68
4.9.3.6 calculateTax() [2/2]	68
4.9.3.7 payTax() [1/2]	69
4.9.3.8 payTax() [2/2]	70
4.9.3.9 payTaxes() [1/2]	70
4.9.3.10 payTaxes() [2/2]	70
4.9.3.11 removePolicy() [1/2]	71
4.9.3.12 removePolicy() [2/2]	71
4.9.3.13 removeService() [1/2]	71
4.9.3.14 removeService() [2/2]	71
4.9.3.15 setTaxCooldownPeriod() [1/2]	72
4.9.3.16 setTaxCooldownPeriod() [2/2]	72
4.9.3.17 updatePolicy() [1/2]	72
4.9.3.18 updatePolicy() [2/2]	72
4.9.3.19 updateServices() [1/2]	73
4.9.3.20 updateServices() [2/2]	73
4.9.3.21 updateTaxRate() [1/2]	73
4.9.3.22 updateTaxRate() [2/2]	73
4.10 CargoAirport Class Reference	74
4.10.1 Detailed Description	77
4.10.2 Constructor & Destructor Documentation	77
4.10.2.1 CargoAirport()	77
4.10.3 Member Function Documentation	78
4.10.3.1 addCargoAirport()	78
4.10.3.2 addInsideRoad()	78
4.10.3.3 getCargoAirport()	78
4.10.3.4 getInsideRoad()	79
4.10.3.5 getName()	79
4.11 Citizen Class Reference	79
4.11.1 Detailed Description	83
4.11.2 Constructor & Destructor Documentation	83
4.11.2.1 Citizen()	83
4.11.3 Member Function Documentation	84

4.11.3.1 addObserver()	84
4.11.3.2 addSatisfactionStrategy()	84
4.11.3.3 clone()	84
4.11.3.4 getAge()	84
4.11.3.5 getBankBalance()	85
4.11.3.6 getJob()	85
4.11.3.7 getjobobj()	85
4.11.3.8 getMarriageDuration()	85
4.11.3.9 getName()	85
4.11.3.10 getRelationshipStatus()	86
4.11.3.11 getSatisfactionLevel()	86
4.11.3.12 getTaxCooldown()	86
4.11.3.13 getTaxRate()	86
4.11.3.14 increaseBankBalance()	86
4.11.3.15 isEmployed()	87
4.11.3.16 isLeaving()	87
4.11.3.17 isOnCooldown()	87
4.11.3.18 payTaxes()	87
4.11.3.19 removeObserver()	88
4.11.3.20 searchAndApplyForJob()	88
4.11.3.21 setBankBalance()	88
4.11.3.22 setEmployed()	89
4.11.3.23 setIncome()	89
4.11.3.24 setJob()	89
4.11.3.25 setJobTitle()	89
4.11.3.26 setRelationshipStatus()	90
4.11.3.27 setSatisfactionLevel()	90
4.11.3.28 setState()	90
4.11.3.29 setTaxCooldown()	90
4.11.3.30 setTaxRate()	91
4.11.3.31 subtractBankBalance()	91
4.11.3.32 updateSatisfaction()	91
4.12 CitizenObserver Class Reference	91
4.12.1 Detailed Description	92
4.12.2 Member Function Documentation	93
4.12.2.1 update()	93
4.13 CitizenSatisfactionObserver Class Reference	93
4.13.1 Detailed Description	94
4.13.2 Member Function Documentation	94
4.13.2.1 update()	94
4.14 CitizenState Class Reference	95
4.14.1 Detailed Description	96

4.14.2 Member Function Documentation	96
4.14.2.1 handleState()	96
4.15 CityService Class Reference	96
4.15.1 Detailed Description	98
4.15.2 Constructor & Destructor Documentation	98
4.15.2.1 CityService() [1/2]	98
4.15.2.2 CityService() [2/2]	98
4.15.3 Member Function Documentation	98
4.15.3.1 allocateAdditionalBudget()	98
4.15.3.2 getBudgetAllocated()	99
4.15.3.3 getServiceName()	99
4.15.3.4 isWithinBudget()	99
4.15.3.5 provideService()	99
4.15.3.6 reduceBudget()	99
4.15.3.7 setServiceName()	100
4.15.3.8 updateBudget()	100
4.16 CityTraverser Class Reference	100
4.16.1 Detailed Description	103
4.16.2 Constructor & Destructor Documentation	103
4.16.2.1 CityTraverser() [1/3]	103
4.16.2.2 CityTraverser() [2/3]	103
4.16.2.3 CityTraverser() [3/3]	104
4.16.3 Member Function Documentation	104
4.16.3.1 getCurrentLayer()	104
4.16.3.2 operator*()	104
4.16.3.3 operator+()	105
4.16.3.4 operator++()	105
4.16.3.5 operator-()	105
4.16.3.6 operator--()	106
4.16.3.7 operator=() [1/2]	106
4.16.3.8 operator=() [2/2]	106
4.16.3.9 setState()	107
4.16.3.10 stepIn()	107
4.16.3.11 stepOut()	108
4.17 CollectTaxesCommand Class Reference	108
4.17.1 Detailed Description	111
4.17.2 Constructor & Destructor Documentation	111
4.17.2.1 CollectTaxesCommand()	111
4.17.3 Member Function Documentation	112
4.17.3.1 canExecute()	112
4.17.3.2 execute()	112
4.17.3.3 executeWithValidation()	112

4.17.3.4 getCollectedTaxes()	112
4.17.3.5 getDescription()	113
4.17.3.6 getName()	113
4.17.3.7 getTaxesCollected()	113
4.17.3.8 returnVal()	113
4.17.3.9 setCollectedTaxes()	113
4.17.3.10 undo()	114
4.17.3.11 validateCollection()	114
4.18 ComercialAirport Class Reference	114
4.18.1 Detailed Description	117
4.18.2 Constructor & Destructor Documentation	117
4.18.2.1 ComercialAirport()	117
4.18.3 Member Function Documentation	118
4.18.3.1 addComercialAirport()	118
4.18.3.2 addRoad()	118
4.18.3.3 getComercialAirport()	119
4.18.3.4 getName()	119
4.18.3.5 getRoad()	120
4.19 CommercialBuilding Class Reference	120
4.19.1 Detailed Description	125
4.19.2 Constructor & Destructor Documentation	125
4.19.2.1 CommercialBuilding()	125
4.19.3 Member Function Documentation	126
4.19.3.1 calculateEconomicImpact()	126
4.19.3.2 calculateResourceConsumption()	126
4.19.3.3 calculateSatisfactionImpact()	126
4.19.3.4 construct()	126
4.19.3.5 getType()	126
4.19.3.6 payTaxes()	126
4.19.3.7 setBusiness()	127
4.19.3.8 undoCollectTaxes()	127
4.19.3.9 updateCustomer()	127
4.19.3.10 updateImpacts()	127
4.20 CommercialBuildingBuilder Class Reference	128
4.20.1 Detailed Description	130
4.20.2 Member Function Documentation	131
4.20.2.1 build()	131
4.20.2.2 getBusinessUnits()	131
4.20.2.3 getCustomerTraffic()	131
4.20.2.4 setBusinessUnits()	131
4.20.2.5 setCustomerTraffic()	132
4.21 CreateTraverser Class Reference	132

4.21.1 Detailed Description	134
4.21.2 Member Function Documentation	134
4.21.2.1 createCityTraverser() [1/3]	134
4.21.2.2 createCityTraverser() [2/3]	134
4.21.2.3 createCityTraverser() [3/3]	135
4.22 Director Class Reference	136
4.22.1 Detailed Description	136
4.22.2 Member Function Documentation	137
4.22.2.1 constructLargeBuilding()	137
4.22.2.2 constructMediumBuilding()	137
4.22.2.3 constructSmallBuilding()	137
4.22.2.4 setBuilder()	137
4.23 EmployedState Class Reference	138
4.23.1 Detailed Description	139
4.23.2 Member Function Documentation	139
4.23.2.1 handleState()	139
4.24 EnforcePolicyCommand Class Reference	140
4.24.1 Detailed Description	142
4.24.2 Constructor & Destructor Documentation	142
4.24.2.1 EnforcePolicyCommand()	142
4.24.3 Member Function Documentation	142
4.24.3.1 canExecute()	142
4.24.3.2 execute()	143
4.24.3.3 executeWithValidation()	143
4.24.3.4 getDescription()	143
4.24.3.5 getName()	143
4.24.3.6 isPolicyEnforced()	144
4.24.3.7 returnVal()	144
4.24.3.8 setPolicyEnforced()	144
4.24.3.9 undo()	144
4.24.3.10 validateEnforcement()	144
4.25 EqualDistributionStrategy Class Reference	145
4.25.1 Detailed Description	146
4.25.2 Member Function Documentation	146
4.25.2.1 allocate()	146
4.26 FemaleCitizen Class Reference	146
4.26.1 Detailed Description	150
4.26.2 Constructor & Destructor Documentation	150
4.26.2.1 FemaleCitizen()	150
4.26.3 Member Function Documentation	151
4.26.3.1 clone()	151
4.27 FreightTrain Class Reference	151

4.27.1 Detailed Description	154
4.27.2 Constructor & Destructor Documentation	154
4.27.2.1 FreightTrain()	154
4.27.3 Member Function Documentation	155
4.27.3.1 addFreightTrain()	155
4.27.3.2 addInsideRoad()	155
4.27.3.3 getFreightTrain()	156
4.27.3.4 getInsideRoad()	156
4.27.3.5 getLength()	157
4.27.3.6 getTrainLine()	157
4.27.3.7 getWeight()	157
4.28 GovCommand Class Reference	158
4.28.1 Detailed Description	159
4.28.2 Constructor & Destructor Documentation	159
4.28.2.1 ~GovCommand()	159
4.28.3 Member Function Documentation	159
4.28.3.1 canExecute()	159
4.28.3.2 execute()	159
4.28.3.3 getDescription()	160
4.28.3.4 getName()	160
4.28.3.5 returnVal()	160
4.28.3.6 undo()	160
4.29 Government Class Reference	161
4.29.1 Detailed Description	162
4.29.2 Constructor & Destructor Documentation	162
4.29.2.1 Government()	162
4.29.3 Member Function Documentation	163
4.29.3.1 addTaxesToBudget()	163
4.29.3.2 allocateBudget()	163
4.29.3.3 collectTaxes()	163
4.29.3.4 enforcePolicy()	163
4.29.3.5 getBudget()	164
4.29.3.6 getTaxRate()	164
4.29.3.7 refundTaxes()	164
4.29.3.8 registerObserver()	164
4.29.3.9 revertBudgetAllocation()	164
4.29.3.10 setTax()	165
4.29.3.11 unregisterObserver()	165
4.29.3.12 update()	165
4.30 GovObserver Class Reference	166
4.30.1 Detailed Description	167
4.30.2 Constructor & Destructor Documentation	167

4.30.2.1 ~GovObserver()	167
4.30.3 Member Function Documentation	167
4.30.3.1 updatePolicy()	167
4.30.3.2 updateServices()	168
4.30.3.3 updateTaxRate()	168
4.31 Highway Class Reference	169
4.31.1 Detailed Description	171
4.31.2 Constructor & Destructor Documentation	171
4.31.2.1 Highway()	171
4.31.3 Member Function Documentation	172
4.31.3.1 addHighway()	172
4.31.3.2 addInsideRoad()	172
4.31.3.3 getHighway()	173
4.31.3.4 getHighwaysList()	173
4.31.3.5 getInsideRoad()	174
4.31.3.6 getInsideRoadsList()	174
4.31.3.7 getRoadName()	174
4.31.3.8 getSpeedLimit()	175
4.32 HousingSatisfactionStrategy Class Reference	175
4.32.1 Detailed Description	176
4.32.2 Member Function Documentation	177
4.32.2.1 calculateSatisfaction()	177
4.32.2.2 updateForHousingChange()	177
4.32.2.3 updateForJobChange()	177
4.32.2.4 updateForTaxChange()	178
4.33 Income Class Reference	178
4.33.1 Detailed Description	181
4.33.2 Constructor & Destructor Documentation	181
4.33.2.1 Income() [1/2]	181
4.33.2.2 Income() [2/2]	182
4.33.3 Member Function Documentation	182
4.33.3.1 addBonus()	182
4.33.3.2 applyDeductions()	182
4.33.3.3 calculateMonthlyIncome()	182
4.33.3.4 calculateTax()	183
4.33.3.5 getBaseSalary()	183
4.33.3.6 getBonus()	183
4.33.3.7 getDeductions()	183
4.33.3.8 payTaxes()	183
4.33.3.9 setTax()	184
4.34 IndustrialBuilding Class Reference	184
4.34.1 Detailed Description	189

4.34.2 Constructor & Destructor Documentation	189
4.34.2.1 IndustrialBuilding()	189
4.34.3 Member Function Documentation	189
4.34.3.1 calculateEconomicImpact()	189
4.34.3.2 calculateResourceConsumption()	190
4.34.3.3 calculateSatisfactionImpact()	190
4.34.3.4 construct()	190
4.34.3.5 getType()	190
4.34.3.6 payTaxes()	190
4.34.3.7 undoCollectTaxes()	191
4.34.3.8 updateImpacts()	191
4.34.3.9 upgradeTech()	191
4.35 IndustrialBuildingBuilder Class Reference	191
4.35.1 Detailed Description	194
4.35.2 Member Function Documentation	195
4.35.2.1 build()	195
4.35.2.2 getPollutionLevel()	195
4.35.2.3 getProductionCapacity()	195
4.35.2.4 setPollutionLevel()	195
4.35.2.5 setProductionCapacity()	196
4.36 InsideRoad Class Reference	196
4.36.1 Detailed Description	200
4.36.2 Constructor & Destructor Documentation	200
4.36.2.1 InsideRoad()	200
4.36.3 Member Function Documentation	200
4.36.3.1 addBuilding()	200
4.36.3.2 addBus()	201
4.36.3.3 addCargoAirport()	202
4.36.3.4 addComercialAirport()	202
4.36.3.5 addFreightTrain()	203
4.36.3.6 addHighway()	203
4.36.3.7 addInsideRoad()	204
4.36.3.8 addPassengerTrain()	204
4.36.3.9 addTaxi()	205
4.36.3.10 getAvgStopTime()	205
4.36.3.11 getBuilding()	206
4.36.3.12 getBus()	206
4.36.3.13 getCargoAirport()	207
4.36.3.14 getComercialAirport()	207
4.36.3.15 getFreightTrain()	208
4.36.3.16 getHighway()	208
4.36.3.17 getInsideRoad()	209

4.36.3.18 getPassengerTrain()	209
4.36.3.19 getRoadName()	210
4.36.3.20 getTaxi()	210
4.37 Jobs Class Reference	211
4.37.1 Detailed Description	212
4.37.2 Constructor & Destructor Documentation	212
4.37.2.1 Jobs()	212
4.37.3 Member Function Documentation	212
4.37.3.1 getIncome()	212
4.37.3.2 getTitle()	213
4.37.3.3 isOccupied()	213
4.38 JobSatisfactionStrategy Class Reference	213
4.38.1 Detailed Description	214
4.38.2 Member Function Documentation	215
4.38.2.1 calculateSatisfaction()	215
4.38.2.2 updateForHousingChange()	215
4.38.2.3 updateForJobChange()	215
4.38.2.4 updateForTaxChange()	216
4.39 LandmarkBuilding Class Reference	216
4.39.1 Detailed Description	221
4.39.2 Constructor & Destructor Documentation	221
4.39.2.1 LandmarkBuilding()	221
4.39.3 Member Function Documentation	222
4.39.3.1 calculateEconomicImpact()	222
4.39.3.2 calculateResourceConsumption()	222
4.39.3.3 calculateSatisfactionImpact()	222
4.39.3.4 construct()	222
4.39.3.5 getType()	222
4.39.3.6 hostEvent()	222
4.39.3.7 payTaxes()	223
4.39.3.8 undoCollectTaxes()	223
4.39.3.9 updateImpacts()	223
4.40 LandmarkBuildingBuilder Class Reference	224
4.40.1 Detailed Description	227
4.40.2 Member Function Documentation	227
4.40.2.1 build()	227
4.40.2.2 getCulturalValue()	227
4.40.2.3 getIsHistoric()	227
4.40.2.4 getVisitorCapacity()	227
4.40.2.5 setCulturalValue()	227
4.40.2.6 setIsHistoric()	228
4.40.2.7 setVisitorCapacity()	228

4.41 LeavingCityState Class Reference	228
4.41.1 Detailed Description	230
4.41.2 Member Function Documentation	230
4.41.2.1 handleState()	230
4.42 LunchRush Class Reference	230
4.42.1 Detailed Description	232
4.42.2 Member Function Documentation	232
4.42.2.1 getState()	232
4.42.2.2 getTrafficFlow()	232
4.43 MaleCitizen Class Reference	233
4.43.1 Detailed Description	236
4.43.2 Constructor & Destructor Documentation	236
4.43.2.1 MaleCitizen()	236
4.43.3 Member Function Documentation	237
4.43.3.1 clone()	237
4.44 NonPeak Class Reference	237
4.44.1 Detailed Description	238
4.44.2 Member Function Documentation	238
4.44.2.1 getState()	238
4.44.2.2 getTrafficFlow()	239
4.45 Observer Class Reference	239
4.45.1 Detailed Description	240
4.45.2 Member Function Documentation	240
4.45.2.1 update()	240
4.46 PassengerTrain Class Reference	241
4.46.1 Detailed Description	243
4.46.2 Constructor & Destructor Documentation	243
4.46.2.1 PassengerTrain()	243
4.46.3 Member Function Documentation	244
4.46.3.1 addInsideRoad()	244
4.46.3.2 addPassengerTrain()	244
4.46.3.3 getInsideRoad()	245
4.46.3.4 getPassengerTrain()	245
4.46.3.5 getTrainLine()	246
4.47 Peak Class Reference	246
4.47.1 Detailed Description	247
4.47.2 Member Function Documentation	247
4.47.2.1 getState()	247
4.47.2.2 getTrafficFlow()	248
4.48 Policy Class Reference	248
4.48.1 Detailed Description	249
4.48.2 Constructor & Destructor Documentation	249

4.48.2.1 Policy()	249
4.48.3 Member Function Documentation	249
4.48.3.1 getImpactLevel()	249
4.48.3.2 getPolicyName()	250
4.48.3.3 implement()	250
4.48.3.4 revoke()	250
4.49 PolicyCheckerHandler Class Reference	250
4.49.1 Detailed Description	252
4.49.2 Constructor & Destructor Documentation	252
4.49.2.1 PolicyCheckerHandler()	252
4.49.3 Member Function Documentation	253
4.49.3.1 handleRequest()	253
4.49.3.2 isPolicyEnforced()	253
4.49.3.3 setPolicyEnforced()	253
4.50 PopulationManager Class Reference	254
4.50.1 Detailed Description	255
4.50.2 Constructor & Destructor Documentation	255
4.50.2.1 PopulationManager()	255
4.50.3 Member Function Documentation	255
4.50.3.1 addCitizen()	255
4.50.3.2 findJobsForUnemployedCitizens()	255
4.50.3.3 getCitizens()	255
4.50.3.4 getPopulation()	256
4.51 PowerPlant Class Reference	256
4.51.1 Detailed Description	258
4.51.2 Constructor & Destructor Documentation	258
4.51.2.1 PowerPlant()	258
4.51.3 Member Function Documentation	258
4.51.3.1 registerBuilding()	258
4.51.3.2 supplyResources()	259
4.52 PriorityDistributionStrategy Class Reference	259
4.52.1 Detailed Description	260
4.52.2 Member Function Documentation	260
4.52.2.1 allocate()	260
4.53 Property Class Reference	261
4.53.1 Detailed Description	263
4.53.2 Constructor & Destructor Documentation	263
4.53.2.1 Property()	263
4.53.3 Member Function Documentation	263
4.53.3.1 calculateTax()	263
4.53.3.2 setAdditionalFees()	263
4.53.3.3 setMunicipalLevy()	264

4.53.3.4 setTax()	264
4.54 PublicTransit Class Reference	264
4.54.1 Detailed Description	267
4.54.2 Constructor & Destructor Documentation	267
4.54.2.1 PublicTransit()	267
4.54.3 Member Function Documentation	267
4.54.3.1 calculateCommute()	267
4.54.3.2 getRoute()	267
4.55 ResidentialBuilding Class Reference	268
4.55.1 Detailed Description	273
4.55.2 Constructor & Destructor Documentation	273
4.55.2.1 ResidentialBuilding()	273
4.55.3 Member Function Documentation	274
4.55.3.1 addResidents()	274
4.55.3.2 calculateEconomicImpact()	274
4.55.3.3 calculatePropertyTax()	274
4.55.3.4 calculateResourceConsumption()	275
4.55.3.5 calculateSatisfactionImpact()	275
4.55.3.6 construct()	275
4.55.3.7 getResidentialUnits()	275
4.55.3.8 getType()	275
4.55.3.9 payTaxes()	275
4.55.3.10 undoCollectTaxes()	276
4.55.3.11 updateImpacts()	276
4.55.3.12 upgradeComfort()	276
4.56 ResidentialBuildingBuilder Class Reference	277
4.56.1 Detailed Description	279
4.56.2 Member Function Documentation	280
4.56.2.1 build()	280
4.56.2.2 getComfort()	280
4.56.2.3 getResidentialUnit()	280
4.56.2.4 setComfort()	280
4.56.2.5 setResidentialUnit()	281
4.57 Resource Class Reference	281
4.57.1 Detailed Description	282
4.57.2 Constructor & Destructor Documentation	282
4.57.2.1 Resource()	282
4.57.3 Member Function Documentation	282
4.57.3.1 allocate()	282
4.57.3.2 getQuantity()	283
4.57.3.3 getType()	283
4.57.3.4 release()	283

4.58 ResourceAllocationStrategy Class Reference	283
4.58.1 Detailed Description	284
4.58.2 Member Function Documentation	284
4.58.2.1 allocate()	284
4.59 ResourceApprovalHandler Class Reference	285
4.59.1 Detailed Description	286
4.59.2 Member Function Documentation	286
4.59.2.1 handleRequest()	286
4.59.2.2 setNextHandler()	287
4.60 ResourceAvailability Class Reference	287
4.60.1 Detailed Description	288
4.60.2 Constructor & Destructor Documentation	288
4.60.2.1 ResourceAvailability()	288
4.60.3 Member Function Documentation	288
4.60.3.1 checkAvailability()	288
4.61 ResourceManager Class Reference	289
4.61.1 Detailed Description	289
4.61.2 Constructor & Destructor Documentation	290
4.61.2.1 ResourceManager()	290
4.61.3 Member Function Documentation	290
4.61.3.1 addObserver()	290
4.61.3.2 addResource()	290
4.61.3.3 allocateResources()	290
4.61.3.4 getBudget()	291
4.61.3.5 getResource()	291
4.61.3.6 releaseResources()	291
4.61.3.7 setAllocationStrategy()	292
4.62 Road Class Reference	292
4.62.1 Detailed Description	295
4.62.2 Constructor & Destructor Documentation	295
4.62.2.1 Road()	295
4.62.3 Member Function Documentation	295
4.62.3.1 calculateCommute()	295
4.62.3.2 getName()	295
4.63 Sales Class Reference	296
4.63.1 Detailed Description	298
4.63.2 Constructor & Destructor Documentation	298
4.63.2.1 Sales()	298
4.63.3 Member Function Documentation	298
4.63.3.1 calculateTax()	298
4.63.3.2 setEnvironmentalLevy()	299
4.63.3.3 setServiceFee()	299

4.63.3.4 setTax()	299
4.64 SatisfactionStrategy Class Reference	300
4.64.1 Detailed Description	301
4.64.2 Member Function Documentation	301
4.64.2.1 calculateSatisfaction()	301
4.64.2.2 updateForHousingChange()	301
4.64.2.3 updateForJobChange()	301
4.64.2.4 updateForTaxChange()	302
4.65 SatisfiedState Class Reference	302
4.65.1 Detailed Description	303
4.65.2 Member Function Documentation	303
4.65.2.1 handleState()	303
4.66 SetTaxCommand Class Reference	304
4.66.1 Detailed Description	306
4.66.2 Constructor & Destructor Documentation	306
4.66.2.1 SetTaxCommand() [1/2]	306
4.66.2.2 SetTaxCommand() [2/2]	306
4.66.3 Member Function Documentation	307
4.66.3.1 canExecute()	307
4.66.3.2 execute()	307
4.66.3.3 getDescription()	307
4.66.3.4 getName()	307
4.66.3.5 returnVal()	308
4.66.3.6 undo()	308
4.67 SewageManagement Class Reference	308
4.67.1 Detailed Description	310
4.67.2 Constructor & Destructor Documentation	310
4.67.2.1 SewageManagement()	310
4.67.3 Member Function Documentation	310
4.67.3.1 registerBuilding()	310
4.67.3.2 supplyResources()	311
4.68 Taxi Class Reference	311
4.68.1 Detailed Description	315
4.68.2 Constructor & Destructor Documentation	315
4.68.2.1 Taxi()	315
4.68.3 Member Function Documentation	315
4.68.3.1 addBuilding()	315
4.68.3.2 addCargoAirport()	316
4.68.3.3 addComercialAirport()	316
4.68.3.4 addFreightTrain()	317
4.68.3.5 addInsideRoad()	317
4.68.3.6 addPassengerTrain()	318

4.68.3.7 getBuilding()	318
4.68.3.8 getCargoAirport()	319
4.68.3.9 getComercialAirport()	319
4.68.3.10 getFreightTrain()	320
4.68.3.11 getInsideRoad()	320
4.68.3.12 getPassengerTrain()	321
4.68.3.13 getRouteName()	321
4.68.3.14 getTaxiCompany()	322
4.68.3.15 getTaxiNumber()	322
4.69 TaxSatisfactionStrategy Class Reference	322
4.69.1 Detailed Description	324
4.69.2 Member Function Documentation	325
4.69.2.1 calculateSatisfaction()	325
4.69.2.2 updateForHousingChange()	325
4.69.2.3 updateForJobChange()	326
4.69.2.4 updateForTaxChange()	326
4.70 TaxSystem Class Reference	326
4.70.1 Detailed Description	328
4.70.2 Member Function Documentation	328
4.70.2.1 addGovernment()	328
4.70.2.2 addIncomeTaxBuilding()	328
4.70.2.3 addPropertyTaxBuilding()	329
4.70.2.4 addSalesTaxBuilding()	329
4.70.2.5 addTaxRate()	329
4.70.2.6 addVATTaxPayer()	329
4.70.2.7 checkImpact()	330
4.70.2.8 collectTaxes()	330
4.70.2.9 removeIncomeTaxBuilding()	330
4.70.2.10 removePropertyTaxBuilding()	330
4.70.2.11 removeSalesTaxBuilding()	331
4.70.2.12 removeTaxRate()	331
4.70.2.13 removeVATTaxPayer()	331
4.70.2.14 setTax()	331
4.70.2.15 updateTaxRate()	332
4.71 TaxType Class Reference	332
4.71.1 Detailed Description	333
4.71.2 Constructor & Destructor Documentation	334
4.71.2.1 TaxType()	334
4.71.3 Member Function Documentation	334
4.71.3.1 calculateTax()	334
4.71.3.2 getTaxRate()	334
4.71.3.3 getTaxType()	335

4.71.3.4 setTax()	335
4.72 TrafficFlow Class Reference	335
4.72.1 Detailed Description	336
4.72.2 Member Function Documentation	336
4.72.2.1 getState()	336
4.72.2.2 getTrafficFlow()	336
4.73 Train Class Reference	337
4.73.1 Detailed Description	339
4.73.2 Constructor & Destructor Documentation	339
4.73.2.1 Train()	339
4.73.3 Member Function Documentation	339
4.73.3.1 calculateCommute()	339
4.73.3.2 getLine()	339
4.74 Transportation Class Reference	340
4.74.1 Detailed Description	341
4.74.2 Constructor & Destructor Documentation	341
4.74.2.1 Transportation()	341
4.74.2.2 ~Transportation()	341
4.74.3 Member Function Documentation	341
4.74.3.1 getTrafficFlow()	341
4.74.3.2 getType()	342
4.74.3.3 setState()	342
4.75 TransportationFactory Class Reference	342
4.75.1 Detailed Description	343
4.75.2 Member Function Documentation	344
4.75.2.1 createBus()	344
4.75.2.2 createCargoAirport()	344
4.75.2.3 createComercialAirport()	345
4.75.2.4 createFreightTrain()	345
4.75.2.5 createHighway()	346
4.75.2.6 createInsideRoad()	347
4.75.2.7 createPassengerTrain()	347
4.75.2.8 createTaxi()	348
4.76 TransportManager Class Reference	349
4.76.1 Detailed Description	350
4.76.2 Constructor & Destructor Documentation	350
4.76.2.1 TransportManager()	350
4.76.2.2 ~TransportManager()	350
4.76.3 Member Function Documentation	350
4.76.3.1 createBus()	350
4.76.3.2 createCargoAirport()	351
4.76.3.3 createComercialAirport()	351

4.76.3.4 createFreightTrain()	352
4.76.3.5 createHighway()	353
4.76.3.6 createInsideRoad()	353
4.76.3.7 createPassengerTrain()	354
4.76.3.8 createTaxi()	355
4.76.3.9 getTransportation()	355
4.77 TraverseBus Class Reference	356
4.77.1 Detailed Description	357
4.77.2 Constructor & Destructor Documentation	358
4.77.2.1 TraverseBus()	358
4.77.3 Member Function Documentation	358
4.77.3.1 getPos()	358
4.77.3.2 nextList()	359
4.77.3.3 prevList()	359
4.78 TraverseCargoAirport Class Reference	360
4.78.1 Detailed Description	361
4.78.2 Constructor & Destructor Documentation	362
4.78.2.1 TraverseCargoAirport()	362
4.78.3 Member Function Documentation	362
4.78.3.1 getPos()	362
4.78.3.2 nextList()	362
4.78.3.3 prevList()	363
4.79 TraverseComercialAirport Class Reference	363
4.79.1 Detailed Description	365
4.79.2 Constructor & Destructor Documentation	366
4.79.2.1 TraverseComercialAirport()	366
4.79.3 Member Function Documentation	366
4.79.3.1 getPos()	366
4.79.3.2 nextList()	367
4.79.3.3 prevList()	367
4.80 TraverseFreightTrain Class Reference	368
4.80.1 Detailed Description	369
4.80.2 Constructor & Destructor Documentation	370
4.80.2.1 TraverseFreightTrain()	370
4.80.3 Member Function Documentation	370
4.80.3.1 getPos()	370
4.80.3.2 nextList()	371
4.80.3.3 prevList()	371
4.81 TraverseHighway Class Reference	372
4.81.1 Detailed Description	373
4.81.2 Constructor & Destructor Documentation	374
4.81.2.1 TraverseHighway()	374

4.81.3 Member Function Documentation	374
4.81.3.1 getPos()	374
4.81.3.2 nextList()	374
4.81.3.3 prevList()	375
4.82 TraverseInsideRoad Class Reference	375
4.82.1 Detailed Description	377
4.82.2 Constructor & Destructor Documentation	378
4.82.2.1 TraverseInsideRoad()	378
4.82.3 Member Function Documentation	378
4.82.3.1 getPos()	378
4.82.3.2 nextList()	378
4.82.3.3 prevList()	379
4.83 TraversePassengerTrain Class Reference	379
4.83.1 Detailed Description	381
4.83.2 Constructor & Destructor Documentation	382
4.83.2.1 TraversePassengerTrain()	382
4.83.3 Member Function Documentation	382
4.83.3.1 getPos()	382
4.83.3.2 nextList()	383
4.83.3.3 prevList()	383
4.84 Traverser Class Reference	384
4.84.1 Detailed Description	385
4.84.2 Member Function Documentation	385
4.84.2.1 operator*()	385
4.84.2.2 operator++()	386
4.84.2.3 operator--()	386
4.85 TraverseState Class Reference	386
4.85.1 Detailed Description	387
4.85.2 Constructor & Destructor Documentation	387
4.85.2.1 TraverseState()	387
4.85.3 Member Function Documentation	388
4.85.3.1 getLayer()	388
4.85.3.2 getPos()	388
4.85.3.3 nextList()	388
4.85.3.4 prevList()	389
4.86 TraverseTaxi Class Reference	389
4.86.1 Detailed Description	390
4.86.2 Constructor & Destructor Documentation	391
4.86.2.1 TraverseTaxi()	391
4.86.3 Member Function Documentation	391
4.86.3.1 getPos()	391
4.86.3.2 nextList()	391

4.86.3.3 prevList()	392
4.87 UnemployedState Class Reference	392
4.87.1 Detailed Description	393
4.87.2 Member Function Documentation	393
4.87.2.1 handleState()	393
4.88 UnsatisfiedState Class Reference	394
4.88.1 Detailed Description	395
4.88.2 Member Function Documentation	395
4.88.2.1 handleState()	395
4.89 Utility Class Reference	396
4.89.1 Detailed Description	398
4.89.2 Constructor & Destructor Documentation	398
4.89.2.1 Utility()	398
4.89.3 Member Function Documentation	398
4.89.3.1 adjustForPopulation()	398
4.89.3.2 registerBuilding()	398
4.89.3.3 supplyResources()	399
4.90 UtilityMediator Class Reference	399
4.90.1 Detailed Description	401
4.90.2 Member Function Documentation	401
4.90.2.1 getAvailableResource()	401
4.90.2.2 produceResource()	401
4.90.2.3 releaseResources()	402
4.90.2.4 requestResources()	402
4.90.2.5 update()	402
4.91 VAT Class Reference	403
4.91.1 Detailed Description	404
4.91.2 Member Function Documentation	405
4.91.2.1 calculateTax()	405
4.91.2.2 setTax()	405
4.92 WasteManagement Class Reference	405
4.92.1 Detailed Description	408
4.92.2 Constructor & Destructor Documentation	408
4.92.2.1 WasteManagement()	408
4.92.3 Member Function Documentation	408
4.92.3.1 adjustForPopulation()	408
4.92.3.2 registerBuilding()	409
4.92.3.3 supplyResources()	409
4.93 WaterSupply Class Reference	409
4.93.1 Detailed Description	412
4.93.2 Constructor & Destructor Documentation	412
4.93.2.1 WaterSupply()	412

4.93.3 Member Function Documentation	412
4.93.3.1 adjustForPopulation()	412
4.93.3.2 registerBuilding()	413
4.93.3.3 supplyResources()	413
5 File Documentation	415
5.1 /mnt/c/users/rudie/documents/sem2/214/project/src/AggregateTraverser.h File Reference	415
5.1.1 Detailed Description	416
5.2 AggregateTraverser.h	417
5.3 /mnt/c/users/rudie/documents/sem2/214/project/src/Airport.cpp File Reference	417
5.3.1 Detailed Description	418
5.4 /mnt/c/users/rudie/documents/sem2/214/project/src/Airport.h File Reference	418
5.4.1 Detailed Description	419
5.5 Airport.h	419
5.6 /mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.cpp File Reference	420
5.6.1 Detailed Description	420
5.7 /mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.h File Reference	421
5.7.1 Detailed Description	422
5.8 AllocateBudgetCommand.h	422
5.9 /mnt/c/users/rudie/documents/sem2/214/project/src/BudgetApprovalHandler.h File Reference	423
5.9.1 Detailed Description	423
5.10 BudgetApprovalHandler.h	424
5.11 /mnt/c/users/rudie/documents/sem2/214/project/src/Builder.cpp File Reference	424
5.11.1 Detailed Description	425
5.12 /mnt/c/users/rudie/documents/sem2/214/project/src/Builder.h File Reference	426
5.12.1 Detailed Description	427
5.13 Builder.h	427
5.14 /mnt/c/users/rudie/documents/sem2/214/project/src/Building.h File Reference	428
5.14.1 Detailed Description	429
5.15 Building.h	429
5.16 /mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.cpp File Reference	430
5.16.1 Detailed Description	430
5.17 /mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.h File Reference	431
5.17.1 Detailed Description	432
5.18 BuildingManager.h	432
5.19 /mnt/c/users/rudie/documents/sem2/214/project/src/Bus.cpp File Reference	433
5.19.1 Detailed Description	433
5.20 /mnt/c/users/rudie/documents/sem2/214/project/src/Bus.h File Reference	434
5.20.1 Detailed Description	434
5.21 Bus.h	435
5.22 /mnt/c/users/rudie/documents/sem2/214/project/src/Business.cpp File Reference	436
5.22.1 Detailed Description	436

5.23 /mnt/c/users/rudie/documents/sem2/214/project/src/Business.h File Reference	437
5.23.1 Detailed Description	438
5.24 Business.h	438
5.25 /mnt/c/users/rudie/documents/sem2/214/project/src/CargoAirport.h File Reference	439
5.25.1 Detailed Description	439
5.26 CargoAirport.h	440
5.27 /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.cpp File Reference	440
5.27.1 Detailed Description	441
5.27.2 Function Documentation	441
5.27.2.1 minMax()	441
5.28 /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.h File Reference	442
5.28.1 Detailed Description	443
5.29 Citizen.h	443
5.30 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenObserver.h File Reference	445
5.30.1 Detailed Description	445
5.31 CitizenObserver.h	446
5.32 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenSatisfactionObserver.h File Reference	446
5.32.1 Detailed Description	447
5.33 CitizenSatisfactionObserver.h	447
5.34 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.cpp File Reference	447
5.34.1 Detailed Description	448
5.35 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.h File Reference	448
5.35.1 Detailed Description	449
5.36 CitizenState.h	449
5.37 /mnt/c/users/rudie/documents/sem2/214/project/src/CityService.cpp File Reference	449
5.37.1 Detailed Description	450
5.38 /mnt/c/users/rudie/documents/sem2/214/project/src/CityService.h File Reference	450
5.38.1 Detailed Description	451
5.39 CityService.h	451
5.40 /mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.cpp File Reference	451
5.40.1 Detailed Description	452
5.41 /mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.h File Reference	452
5.41.1 Detailed Description	453
5.42 CityTraverser.h	454
5.43 /mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.cpp File Reference	454
5.43.1 Detailed Description	455
5.44 /mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.h File Reference	455
5.44.1 Detailed Description	456
5.45 CollectTaxesCommand.h	456
5.46 /mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.cpp File Reference	457
5.46.1 Detailed Description	457
5.47 /mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.h File Reference	458

5.47.1 Detailed Description	458
5.48 ComercialAirport.h	459
5.49 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.cpp File Reference	459
5.49.1 Detailed Description	460
5.50 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.h File Reference	460
5.50.1 Detailed Description	461
5.51 CommercialBuilding.h	461
5.52 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.cpp File Reference	462
5.52.1 Detailed Description	462
5.53 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.h File Reference	463
5.53.1 Detailed Description	464
5.54 CommercialBuildingBuilder.h	464
5.55 /mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.cpp File Reference	465
5.55.1 Detailed Description	465
5.56 /mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.h File Reference	465
5.56.1 Detailed Description	466
5.57 createTraverser.h	466
5.58 /mnt/c/users/rudie/documents/sem2/214/project/src/Director.cpp File Reference	467
5.58.1 Detailed Description	467
5.59 Director.h	468
5.60 /mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.cpp File Reference	468
5.60.1 Detailed Description	469
5.60.2 Function Documentation	469
5.60.2.1 clamp()	469
5.61 /mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.h File Reference	469
5.61.1 Detailed Description	470
5.62 EmployedState.h	471
5.63 /mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.cpp File Reference	471
5.63.1 Detailed Description	471
5.64 /mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.h File Reference	472
5.64.1 Detailed Description	473
5.65 EnforcePolicyCommand.h	473
5.66 /mnt/c/users/rudie/documents/sem2/214/project/src/FemaleCitizen.h File Reference	474
5.66.1 Detailed Description	475
5.67 FemaleCitizen.h	475
5.68 /mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.cpp File Reference	475
5.68.1 Detailed Description	476
5.69 /mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.h File Reference	476
5.69.1 Detailed Description	477
5.70 FreightTrain.h	477
5.71 /mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.cpp File Reference	478
5.71.1 Detailed Description	478

5.72 /mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.h File Reference	478
5.72.1 Detailed Description	479
5.73 GovCommand.h	480
5.74 /mnt/c/users/rudie/documents/sem2/214/project/src/Government.cpp File Reference	480
5.74.1 Detailed Description	480
5.75 /mnt/c/users/rudie/documents/sem2/214/project/src/Government.h File Reference	481
5.75.1 Detailed Description	482
5.76 Government.h	482
5.77 /mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.cpp File Reference	483
5.77.1 Detailed Description	483
5.78 /mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.h File Reference	483
5.78.1 Detailed Description	484
5.79 GovObserver.h	484
5.80 /mnt/c/users/rudie/documents/sem2/214/project/src/Highway.cpp File Reference	484
5.80.1 Detailed Description	485
5.81 /mnt/c/users/rudie/documents/sem2/214/project/src/Highway.h File Reference	485
5.81.1 Detailed Description	486
5.82 Highway.h	487
5.83 /mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.cpp File Reference	487
5.83.1 Detailed Description	488
5.84 /mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.h File Reference	488
5.84.1 Detailed Description	489
5.85 HousingSatisfactionStrategy.h	490
5.86 /mnt/c/users/rudie/documents/sem2/214/project/src/Income.cpp File Reference	490
5.86.1 Detailed Description	491
5.87 /mnt/c/users/rudie/documents/sem2/214/project/src/Income.h File Reference	491
5.87.1 Detailed Description	492
5.88 Income.h	492
5.89 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.cpp File Reference	492
5.89.1 Detailed Description	493
5.90 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.h File Reference	493
5.90.1 Detailed Description	495
5.91 IndustrialBuilding.h	495
5.92 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.cpp File Reference	495
5.92.1 Detailed Description	496
5.93 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.h File Reference	496
5.93.1 Detailed Description	498
5.94 IndustrialBuildingBuilder.h	498
5.95 /mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.cpp File Reference	498
5.95.1 Detailed Description	499
5.96 /mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.h File Reference	499
5.96.1 Detailed Description	500

5.97 InsideRoad.h	501
5.98 /mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.cpp File Reference	502
5.98.1 Detailed Description	502
5.99 /mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.h File Reference	503
5.99.1 Detailed Description	503
5.100 Jobs.h	504
5.101 /mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.cpp File Reference	504
5.101.1 Detailed Description	505
5.102 /mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.h File Reference	505
5.102.1 Detailed Description	507
5.103 JobSatisfactionStrategy.h	507
5.104 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.cpp File Reference	507
5.104.1 Detailed Description	508
5.105 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.h File Reference	508
5.105.1 Detailed Description	510
5.106 LandmarkBuilding.h	510
5.107 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.cpp File Reference	510
5.107.1 Detailed Description	511
5.108 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.h File Reference	511
5.108.1 Detailed Description	513
5.109 LandmarkBuildingBuilder.h	513
5.110 /mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.cpp File Reference	513
5.110.1 Detailed Description	514
5.111 /mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.h File Reference	514
5.111.1 Detailed Description	515
5.112 LeavingCityState.h	515
5.113 /mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.cpp File Reference	516
5.113.1 Detailed Description	516
5.114 /mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.h File Reference	516
5.114.1 Detailed Description	517
5.115 LunchRush.h	518
5.116 /mnt/c/users/rudie/documents/sem2/214/project/src/MaleCitizen.h File Reference	518
5.116.1 Detailed Description	519
5.117 MaleCitizen.h	519
5.118 /mnt/c/users/rudie/documents/sem2/214/project/src/NonPeak.cpp File Reference	520
5.118.1 Detailed Description	520
5.119 /mnt/c/users/rudie/documents/sem2/214/project/src/NonPeak.h File Reference	520
5.119.1 Detailed Description	521
5.120 NonPeak.h	522
5.121 /mnt/c/users/rudie/documents/sem2/214/project/src/Observer.h File Reference	522
5.121.1 Detailed Description	523
5.122 Observer.h	523

5.123 /mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.cpp File Reference	523
5.123.1 Detailed Description	524
5.124 /mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.h File Reference	524
5.124.1 Detailed Description	525
5.125 PassengerTrain.h	525
5.126 /mnt/c/users/rudie/documents/sem2/214/project/src/Peak.cpp File Reference	526
5.126.1 Detailed Description	526
5.127 /mnt/c/users/rudie/documents/sem2/214/project/src/Peak.h File Reference	526
5.127.1 Detailed Description	527
5.128 Peak.h	528
5.129 /mnt/c/users/rudie/documents/sem2/214/project/src/Policy.cpp File Reference	528
5.129.1 Detailed Description	528
5.130 /mnt/c/users/rudie/documents/sem2/214/project/src/Policy.h File Reference	529
5.130.1 Detailed Description	529
5.131 Policy.h	530
5.132 /mnt/c/users/rudie/documents/sem2/214/project/src/PolicyCheckerHandler.h File Reference	530
5.132.1 Detailed Description	531
5.133 PolicyCheckerHandler.h	531
5.134 /mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.cpp File Reference	531
5.134.1 Detailed Description	532
5.135 /mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.h File Reference	532
5.135.1 Detailed Description	533
5.136 PopulationManager.h	534
5.137 /mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.cpp File Reference	534
5.137.1 Detailed Description	535
5.138 /mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.h File Reference	535
5.138.1 Detailed Description	536
5.139 PowerPlant.h	537
5.140 /mnt/c/users/rudie/documents/sem2/214/project/src/Property.cpp File Reference	537
5.140.1 Detailed Description	538
5.141 /mnt/c/users/rudie/documents/sem2/214/project/src/Property.h File Reference	538
5.141.1 Detailed Description	539
5.142 Property.h	539
5.143 /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.cpp File Reference	539
5.143.1 Detailed Description	540
5.144 /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.h File Reference	540
5.144.1 Detailed Description	541
5.145 PublicTransit.h	542
5.146 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.cpp File Reference	542
5.146.1 Detailed Description	542
5.147 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.h File Reference	543
5.147.1 Detailed Description	544

5.148 ResidentialBuilding.h	544
5.149 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.cpp File Reference	545
5.149.1 Detailed Description	545
5.150 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.h File Reference	545
5.150.1 Detailed Description	546
5.151 ResidentialBuildingBuilder.h	547
5.152 /mnt/c/users/rudie/documents/sem2/214/project/src/Resource.h File Reference	547
5.152.1 Detailed Description	548
5.153 Resource.h	549
5.154 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAllocationStrategy.h File Reference	549
5.154.1 Detailed Description	550
5.155 ResourceAllocationStrategy.h	551
5.156 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceApprovalHandler.h File Reference	551
5.156.1 Detailed Description	552
5.157 ResourceApprovalHandler.h	552
5.158 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAvailability.h File Reference	553
5.158.1 Detailed Description	553
5.159 ResourceAvailability.h	554
5.160 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceManager.h File Reference	554
5.160.1 Detailed Description	555
5.161 ResourceManager.h	555
5.162 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceType.h File Reference	556
5.162.1 Detailed Description	557
5.162.2 Enumeration Type Documentation	557
5.162.2.1 ResourceType	557
5.163 ResourceType.h	558
5.164 /mnt/c/users/rudie/documents/sem2/214/project/src/Road.cpp File Reference	558
5.164.1 Detailed Description	558
5.165 /mnt/c/users/rudie/documents/sem2/214/project/src/Road.h File Reference	559
5.165.1 Detailed Description	560
5.166 Road.h	560
5.167 /mnt/c/users/rudie/documents/sem2/214/project/src/Sales.cpp File Reference	560
5.167.1 Detailed Description	561
5.168 /mnt/c/users/rudie/documents/sem2/214/project/src/Sales.h File Reference	561
5.168.1 Detailed Description	562
5.169 Sales.h	563
5.170 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfactionStrategy.h File Reference	563
5.170.1 Detailed Description	564
5.171 SatisfactionStrategy.h	564
5.172 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.cpp File Reference	565
5.172.1 Detailed Description	565
5.173 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.h File Reference	565

5.173.1 Detailed Description	566
5.174 SatisfiedState.h	567
5.175 /mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.cpp File Reference	567
5.175.1 Detailed Description	567
5.176 /mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.h File Reference	568
5.176.1 Detailed Description	568
5.177 SetTaxCommand.h	569
5.178 /mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.cpp File Reference	569
5.178.1 Detailed Description	570
5.179 /mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.h File Reference	570
5.179.1 Detailed Description	571
5.180 SewageManagement.h	571
5.181 /mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.cpp File Reference	572
5.181.1 Detailed Description	572
5.182 /mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.h File Reference	572
5.182.1 Detailed Description	573
5.183 Taxi.h	574
5.184 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.cpp File Reference	574
5.184.1 Detailed Description	575
5.185 TaxSatisfactionStrategy.h	575
5.186 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.cpp File Reference	576
5.186.1 Detailed Description	576
5.187 TaxSystem.h	576
5.188 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.cpp File Reference	577
5.188.1 Detailed Description	578
5.189 TaxType.h	578
5.190 /mnt/c/users/rudie/documents/sem2/214/project/src/TrafficFlow.h File Reference	578
5.190.1 Detailed Description	579
5.191 TrafficFlow.h	579
5.192 /mnt/c/users/rudie/documents/sem2/214/project/src/Train.cpp File Reference	579
5.192.1 Detailed Description	580
5.193 /mnt/c/users/rudie/documents/sem2/214/project/src/Train.h File Reference	580
5.193.1 Detailed Description	581
5.194 Train.h	582
5.195 /mnt/c/users/rudie/documents/sem2/214/project/src/Transportation.cpp File Reference	582
5.195.1 Detailed Description	582
5.196 /mnt/c/users/rudie/documents/sem2/214/project/src/Transportation.h File Reference	583
5.196.1 Detailed Description	584
5.197 Transportation.h	584
5.198 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportationFactory.cpp File Reference	584
5.198.1 Detailed Description	585
5.199 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportationFactory.h File Reference	585

5.199.1 Detailed Description	586
5.200 TransportationFactory.h	587
5.201 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.cpp File Reference	587
5.201.1 Detailed Description	588
5.202 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.h File Reference	588
5.202.1 Detailed Description	589
5.203 TransportManager.h	590
5.204 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus.cpp File Reference	590
5.204.1 Detailed Description	591
5.205 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus.h File Reference	591
5.205.1 Detailed Description	592
5.206 TraverseBus.h	593
5.207 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.cpp File Reference	593
5.207.1 Detailed Description	594
5.208 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.h File Reference	594
5.208.1 Detailed Description	595
5.209 TraverseCargoAirport.h	596
5.210 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.cpp File Reference	596
5.210.1 Detailed Description	596
5.211 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.h File Reference	597
5.211.1 Detailed Description	598
5.212 TraverseComercialAirport.h	599
5.213 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.cpp File Reference	599
5.213.1 Detailed Description	599
5.214 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.h File Reference	600
5.214.1 Detailed Description	601
5.215 TraverseFreightTrain.h	602
5.216 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.cpp File Reference	602
5.216.1 Detailed Description	602
5.217 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.h File Reference	603
5.217.1 Detailed Description	604
5.218 TraverseHighway.h	605
5.219 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.cpp File Reference	605
5.219.1 Detailed Description	606
5.220 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.h File Reference	606
5.220.1 Detailed Description	607
5.221 TraverseInsideRoad.h	608
5.222 /mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.cpp File Reference	608
5.222.1 Detailed Description	608
5.223 /mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.h File Reference	609
5.223.1 Detailed Description	610
5.224 TraversePassengerTrain.h	611

5.225 /mnt/c/users/rudie/documents/sem2/214/project/src/Traverser.h File Reference	611
5.225.1 Detailed Description	612
5.226 Traverser.h	613
5.227 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.cpp File Reference	613
5.227.1 Detailed Description	613
5.228 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.h File Reference	614
5.228.1 Detailed Description	615
5.229 TraverseState.h	615
5.230 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.cpp File Reference	615
5.230.1 Detailed Description	616
5.231 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.h File Reference	616
5.231.1 Detailed Description	617
5.232 TraverseTaxi.h	618
5.233 /mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.cpp File Reference	618
5.233.1 Detailed Description	619
5.234 /mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.h File Reference	619
5.234.1 Detailed Description	620
5.235 UnemployedState.h	621
5.236 /mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.cpp File Reference	621
5.236.1 Detailed Description	621
5.237 /mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.h File Reference	622
5.237.1 Detailed Description	623
5.238 UnsatisfiedState.h	623
5.239 /mnt/c/users/rudie/documents/sem2/214/project/src/Utility.h File Reference	623
5.239.1 Detailed Description	624
5.240 Utility.h	624
5.241 UtilityManager.h	625
5.242 /mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.cpp File Reference	625
5.242.1 Detailed Description	625
5.243 /mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.h File Reference	626
5.243.1 Detailed Description	627
5.244 UtilityMediator.h	627
5.245 /mnt/c/users/rudie/documents/sem2/214/project/src/VAT.cpp File Reference	628
5.245.1 Detailed Description	628
5.246 /mnt/c/users/rudie/documents/sem2/214/project/src/VAT.h File Reference	629
5.246.1 Detailed Description	630
5.247 VAT.h	630
5.248 /mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.cpp File Reference	630
5.248.1 Detailed Description	631
5.249 /mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.h File Reference	631
5.249.1 Detailed Description	632
5.250 WasteManagement.h	633

5.251 /mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.cpp File Reference	633
5.251.1 Detailed Description	634
5.252 /mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.h File Reference	634
5.252.1 Detailed Description	635
5.253 WaterSupply.h	636
Index	637

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AggregateTraverser	17
CreateTraverser	132
Builder	32
CommercialBuildingBuilder	128
IndustrialBuildingBuilder	191
LandmarkBuildingBuilder	224
ResidentialBuildingBuilder	277
Building	37
CommercialBuilding	120
IndustrialBuilding	184
LandmarkBuilding	216
ResidentialBuilding	268
BuildingManager	50
Citizen	79
FemaleCitizen	146
MaleCitizen	233
CitizenObserver	91
CitizenSatisfactionObserver	93
CitizenState	95
EmployedState	138
LeavingCityState	228
SatisfiedState	302
UnemployedState	392
UnsatisfiedState	394
CityService	96
Director	136
GovCommand	158
AllocateBudgetCommand	23
CollectTaxesCommand	108
EnforcePolicyCommand	140
SetTaxCommand	304
Government	161
GovObserver	166

Business	63
Business	63
Jobs	211
Observer	239
UtilityMediator	399
Policy	248
PopulationManager	254
Resource	281
ResourceAllocationStrategy	283
EqualDistributionStrategy	145
PriorityDistributionStrategy	259
ResourceApprovalHandler	285
BudgetApprovalHandler	29
BudgetApprovalHandler	29
PolicyCheckerHandler	250
ResourceAvailability	287
ResourceManager	289
SatisfactionStrategy	300
HousingSatisfactionStrategy	175
JobSatisfactionStrategy	213
TaxSatisfactionStrategy	322
TaxSystem	326
TaxType	332
Income	178
Property	261
Sales	296
VAT	403
TrafficFlow	335
LunchRush	230
NonPeak	237
Peak	246
Transportation	340
Airport	19
CargoAirport	74
ComercialAirport	114
PublicTransit	264
Bus	54
Taxi	311
Road	292
Highway	169
InsideRoad	196
Train	337
FreightTrain	151
PassengerTrain	241
TransportationFactory	342
TransportManager	349
Traverser	384
CityTraverser	100
TraverseState	386
TraverseBus	356
TraverseCargoAirport	360
TraverseComercialAirport	363
TraverseFreightTrain	368
TraverseHighway	372
TraverseInsideRoad	375

TraversePassengerTrain	379
TraverseTaxi	389
Utility	396
PowerPlant	256
SewageManagement	308
WasteManagement	405
WaterSupply	409

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AggregateTraverser	Abstract base class for creating CityTraverser objects	17
Airport	Represents an airport as a type of transportation	19
AllocateBudgetCommand	Represents a command to allocate a budget to a city service	23
BudgetApprovalHandler	Handles budget approval requests	29
Builder	Base class for all builders	32
Building	Represents a building with various properties and job management capabilities	37
BuildingManager	Manages buildings and citizens	50
Bus	Represents a bus in the public transit system	54
Business	Represents a business that observes government policies and updates its state accordingly	63
CargoAirport	Represents a cargo airport	74
Citizen	Manages citizen attributes and behaviors	79
CitizenObserver	Interface for observers of Citizen objects	91
CitizenSatisfactionObserver	Observes changes in citizen satisfaction and updates their state accordingly	93
CitizenState	Abstract class representing the state of a citizen	95
CityService	Class representing a city service	96
CityTraverser	A concrete iterator for traversing transportation elements in a city	100
CollectTaxesCommand	CollectTaxesCommand class	108
ComercialAirport	Represents a commercial airport	114

CommercialBuilding	Represents a commercial building	120
CommercialBuildingBuilder	Builder class for constructing CommercialBuilding objects	128
CreateTraverser	A class to create CityTraverser instances	132
Director	For constructing buildings	136
EmployedState	For handling the employed state of a Citizen	138
EnforcePolicyCommand	For enforcing policies in the government	140
EqualDistributionStrategy	Strategy for equal distribution of resources	145
FemaleCitizen	A class representing a female citizen	146
FreightTrain	A class representing a freight train	151
GovCommand	Abstract base class for government commands	158
Government	Manages various government-related operations	161
GovObserver	Abstract base class for observers that monitor government changes	166
Highway	Represents a highway with a speed limit	169
HousingSatisfactionStrategy	Strategy for calculating and updating citizen satisfaction based on housing conditions	175
Income	Manages income-related operations	178
IndustrialBuilding	Represents an industrial building with specific attributes and behaviors	184
IndustrialBuildingBuilder	Builder class for constructing IndustrialBuilding objects	191
InsideRoad	Represents an inside road that can contain various transportation entities	196
Jobs	Manages job-related operations	211
JobSatisfactionStrategy	Strategy for calculating and updating citizen satisfaction based on job conditions	213
LandmarkBuilding	Represents a landmark building with specific attributes and behaviors	216
LandmarkBuildingBuilder	Builder class for constructing LandmarkBuilding objects	224
LeavingCityState	Handles the state of a citizen preparing to leave the city	228
LunchRush	A class representing traffic flow during lunch hours	230
MaleCitizen	A class representing a male citizen	233
NonPeak	A class to represent traffic flow during non-peak hours	237
Observer	Interface for objects that need to be notified of changes in resource types and quantities	239
PassengerTrain	A class representing a passenger train	241
Peak	A class representing peak traffic flow	246

Policy	Represents a policy with specific attributes and behaviors	248
PolicyCheckerHandler	Handles policy enforcement checks for resource approval requests	250
PopulationManager	Manages the population of citizens	254
PowerPlant	Represents a power plant that supplies power to buildings	256
PriorityDistributionStrategy	Strategy for priority-based resource allocation	259
Property	Represents a property with specific attributes and behaviors related to property tax calculation	261
PublicTransit	A class representing public transit transportation	264
ResidentialBuilding	Represents a residential building in the simulation	268
ResidentialBuildingBuilder	Builder class for constructing ResidentialBuilding objects	277
Resource	Represents a resource with a specific type and quantity	281
ResourceAllocationStrategy	Interface for resource allocation strategies	283
ResourceApprovalHandler	Base class for handling resource approval requests	285
ResourceAvailability	Checks the availability of a specific resource type and quantity	287
ResourceManager	Manages resources, handles resource allocation and release, and notifies observers	289
Road	A class representing a road as a type of transportation	292
Sales	Represents a type of tax that includes a base sales tax, an environmental levy, and a service fee	296
SatisfactionStrategy	Interface for calculating and updating citizen satisfaction	300
SatisfiedState	Represents a state where a citizen is satisfied	302
SetTaxCommand	A command to set the tax rate in the government	304
SewageManagement	A class to manage sewage services for buildings	308
Taxi	A class representing a taxi in a public transit system	311
TaxSatisfactionStrategy	A strategy for calculating and updating citizen satisfaction based on tax rates	322
TaxSystem	Manages various types of taxes, including income, property, sales, and VAT	326
TaxType	Represents a specific type of tax with a rate and type identifier	332
TrafficFlow	An abstract class that represents the traffic flow	335
Train	A class representing a train as a mode of transportation	337
Transportation	Manages traffic flow states and types of transportation	340
TransportationFactory	A factory class for creating different types of transportation objects	342
TransportManager	Manages various types of transportation objects	349

TraverseBus	A class to manage the traversal state of a bus	356
TraverseCargoAirport	A class to traverse through a cargo airport	360
TraverseComercialAirport	A class to traverse through a commercial airport	363
TraverseFreightTrain	A class to traverse through FreightTrain objects	368
TraverseHighway	A class to traverse through highways in a transportation system	372
TraverseInsideRoad	A class to traverse inside roads in a transportation system	375
TraversePassengerTrain	A class to traverse through a passenger train	379
Traverser	Interface for iterating over Transportation objects	384
TraverseState	Abstract class that provides an interface for traversing through a list of Transportation elements	386
TraverseTaxi	Manages the traversal state of a Taxi object	389
UnemployedState	A class that represents the unemployed state of a citizen	392
UnsatisfiedState	A class that represents the unsatisfied state of a citizen	394
Utility	A class that represents a utility service in the city	396
UtilityMediator	A class that manages resource distribution for utilities	399
VAT	A class that represents Value Added Tax (VAT)	403
WasteManagement	A class that represents waste management services in the city	405
WaterSupply	A class that represents water supply services in the city	409

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/mnt/c/users/rudie/documents/sem2/214/project/src/ AggregateTraverser.h	Defines the AggregateTraverser class and its interface for creating CityTraverser objects	415
/mnt/c/users/rudie/documents/sem2/214/project/src/ Airport.cpp	Implementation of the Airport class	417
/mnt/c/users/rudie/documents/sem2/214/project/src/ Airport.h	Defines the Airport class which inherits from the Transportation class	418
/mnt/c/users/rudie/documents/sem2/214/project/src/ AllocateBudgetCommand.cpp	Implementation of the AllocateBudgetCommand class	420
/mnt/c/users/rudie/documents/sem2/214/project/src/ AllocateBudgetCommand.h	Definition of the AllocateBudgetCommand class	421
/mnt/c/users/rudie/documents/sem2/214/project/src/ BudgetApprovalHandler.h	Definition of the BudgetApprovalHandler class	423
/mnt/c/users/rudie/documents/sem2/214/project/src/ Builder.cpp	Implementation of the Builder class	424
/mnt/c/users/rudie/documents/sem2/214/project/src/ Builder.h	Definition of the Builder class	426
/mnt/c/users/rudie/documents/sem2/214/project/src/ Building.h	Definition of the Building class	428
/mnt/c/users/rudie/documents/sem2/214/project/src/ BuildingManager.cpp	Implementation of the BuildingManager class	430
/mnt/c/users/rudie/documents/sem2/214/project/src/ BuildingManager.h	Definition of the BuildingManager class	431
/mnt/c/users/rudie/documents/sem2/214/project/src/ Bus.cpp	Implementation of the Bus class	433
/mnt/c/users/rudie/documents/sem2/214/project/src/ Bus.h	Header file for the Bus class	434
/mnt/c/users/rudie/documents/sem2/214/project/src/ Business.cpp	Implementation of the Business class	436
/mnt/c/users/rudie/documents/sem2/214/project/src/ Business.h	Header file for the Business class	437
/mnt/c/users/rudie/documents/sem2/214/project/src/ CargoAirport.h	Header file for the CargoAirport class	439
/mnt/c/users/rudie/documents/sem2/214/project/src/ Citizen.cpp	Implementation of the Citizen class	440
/mnt/c/users/rudie/documents/sem2/214/project/src/ Citizen.h	Header file for the Citizen class	442

/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenObserver.h	445
Header file for the CitizenObserver class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenSatisfactionObserver.h	446
Header file for the CitizenSatisfactionObserver class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.cpp	447
Implementation of the CitizenState class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.h	448
Declaration of the CitizenState class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CityService.cpp	449
Implementation of the CityService class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CityService.h	450
Declaration of the CityService class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.cpp	451
Implementation of the CityTraverser class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.h	452
Header file for the CityTraverser class, a concrete iterator for traversing transportation elements in a city	
/mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.cpp	454
Implementation of the CollectTaxesCommand class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.h	455
Declaration of the CollectTaxesCommand class	
/mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.cpp	457
Implementation of the ComercialAirport class	
/mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.h	458
Defines the ComercialAirport class which inherits from the Airport class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.cpp	459
Implementation of the CommercialBuilding class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.h	460
Declaration of the CommercialBuilding class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.cpp	462
Implementation of the CommercialBuildingBuilder class	
/mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.h	463
Declaration of the CommercialBuildingBuilder class	
/mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.cpp	465
Implementation of the CreateTraverser class for creating CityTraverser objects	
/mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.h	465
Header file for the CreateTraverser class	
/mnt/c/users/rudie/documents/sem2/214/project/src/Director.cpp	467
Implementation of the Director class for constructing buildings	
/mnt/c/users/rudie/documents/sem2/214/project/src/Director.h	468
/mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.cpp	468
Implementation of the EmployedState class for handling the employed state of a Citizen	
/mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.h	469
Declaration of the EmployedState class for handling the employed state of a Citizen	
/mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.cpp	471
Implementation of the EnforcePolicyCommand class for enforcing policies in the government	
/mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.h	472
Declaration of the EnforcePolicyCommand class for enforcing policies in the government	
/mnt/c/users/rudie/documents/sem2/214/project/src/FemaleCitizen.h	474
Definition of the FemaleCitizen class	
/mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.cpp	475
Implementation of the FreightTrain class	
/mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.h	476
Header file for the FreightTrain class	
/mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.cpp	478
Implementation of the GovCommand class	
/mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.h	478
Definition of the GovCommand class	

/mnt/c/users/rudie/documents/sem2/214/project/src/Government.cpp	Implementation of the <code>Government</code> class	480
/mnt/c/users/rudie/documents/sem2/214/project/src/Government.h	Definition of the <code>Government</code> class	481
/mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.cpp	Implementation of the <code>GovObserver</code> class	483
/mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.h	Definition of the <code>GovObserver</code> class	483
/mnt/c/users/rudie/documents/sem2/214/project/src/Highway.cpp	Implementation of the <code>Highway</code> class	484
/mnt/c/users/rudie/documents/sem2/214/project/src/Highway.h	Header file for the <code>Highway</code> class	485
/mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.cpp	Implementation of the <code>HousingSatisfactionStrategy</code> class	487
/mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.h	Definition of the <code>HousingSatisfactionStrategy</code> class	488
/mnt/c/users/rudie/documents/sem2/214/project/src/Income.cpp	Implementation of the <code>Income</code> class	490
/mnt/c/users/rudie/documents/sem2/214/project/src/Income.h	Definition of the <code>Income</code> class	491
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.cpp	Implementation of the <code>IndustrialBuilding</code> class	492
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.h	Definition of the <code>IndustrialBuilding</code> class	493
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.cpp	Implementation of the <code>IndustrialBuildingBuilder</code> class	495
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.h	Definition of the <code>IndustrialBuildingBuilder</code> class	496
/mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.cpp	Implementation of the <code>InsideRoad</code> class	498
/mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.h	Header file for the <code>InsideRoad</code> class	499
/mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.cpp	Implementation of the <code>Jobs</code> class	502
/mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.h	Definition of the <code>Jobs</code> class	503
/mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.cpp	Implementation of the <code>JobSatisfactionStrategy</code> class	504
/mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.h	Definition of the <code>JobSatisfactionStrategy</code> class	505
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.cpp	Implementation of the <code>LandmarkBuilding</code> class	507
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.h	Definition of the <code>LandmarkBuilding</code> class	508
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.cpp	Implementation of the <code>LandmarkBuildingBuilder</code> class	510
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.h	Definition of the <code>LandmarkBuildingBuilder</code> class	511
/mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.cpp	Implementation of the <code>LeavingCityState</code> class	513
/mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.h	Definition of the <code>LeavingCityState</code> class	514
/mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.cpp	Implementation of the <code>LunchRush</code> class	516
/mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.h	Header file for the <code>LunchRush</code> class	516
/mnt/c/users/rudie/documents/sem2/214/project/src/MaleCitizen.h	Definition of the <code>MaleCitizen</code> class	518

/mnt/c/users/rudie/documents/sem2/214/project/src/ NonPeak.cpp	Implementation of the NonPeak class	520
/mnt/c/users/rudie/documents/sem2/214/project/src/ NonPeak.h	Header file for the NonPeak class	520
/mnt/c/users/rudie/documents/sem2/214/project/src/ Observer.h	Definition of the Observer class	522
/mnt/c/users/rudie/documents/sem2/214/project/src/ PassengerTrain.cpp	Implementation of the PassengerTrain class	523
/mnt/c/users/rudie/documents/sem2/214/project/src/ PassengerTrain.h	Header file for the PassengerTrain class	524
/mnt/c/users/rudie/documents/sem2/214/project/src/ Peak.cpp	Implementation of the Peak class	526
/mnt/c/users/rudie/documents/sem2/214/project/src/ Peak.h	Header file for the Peak class, which inherits from TrafficFlow	526
/mnt/c/users/rudie/documents/sem2/214/project/src/ Policy.cpp	Implementation of the Policy class	528
/mnt/c/users/rudie/documents/sem2/214/project/src/ Policy.h	Definition of the Policy class	529
/mnt/c/users/rudie/documents/sem2/214/project/src/ PolicyCheckerHandler.h	Definition of the PolicyCheckerHandler class	530
/mnt/c/users/rudie/documents/sem2/214/project/src/ PopulationManager.cpp	Implementation of the PopulationManager class	531
/mnt/c/users/rudie/documents/sem2/214/project/src/ PopulationManager.h	Definition of the PopulationManager class	532
/mnt/c/users/rudie/documents/sem2/214/project/src/ PowerPlant.cpp	Implementation of the PowerPlant class	534
/mnt/c/users/rudie/documents/sem2/214/project/src/ PowerPlant.h	Definition of the PowerPlant class	535
/mnt/c/users/rudie/documents/sem2/214/project/src/ Property.cpp	Implementation of the Property class	537
/mnt/c/users/rudie/documents/sem2/214/project/src/ Property.h	Definition of the Property class	538
/mnt/c/users/rudie/documents/sem2/214/project/src/ PublicTransit.cpp	Implementation of the PublicTransit class	539
/mnt/c/users/rudie/documents/sem2/214/project/src/ PublicTransit.h	Header file for the PublicTransit class	540
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResidentialBuilding.cpp	Implementation of the ResidentialBuilding class	542
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResidentialBuilding.h	Header file for the ResidentialBuilding class	543
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResidentialBuildingBuilder.cpp	Implementation file for the ResidentialBuildingBuilder class	545
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResidentialBuildingBuilder.h	Header file for the ResidentialBuildingBuilder class	545
/mnt/c/users/rudie/documents/sem2/214/project/src/ Resource.h	Header file for the Resource class	547
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResourceAllocationStrategy.h	Header file for resource allocation strategy classes	549
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResourceApprovalHandler.h	Header file for resource approval handler classes	551
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResourceAvailability.h	Header file for the ResourceAvailability class	553
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResourceManager.h	Header file for the ResourceManager class	554
/mnt/c/users/rudie/documents/sem2/214/project/src/ ResourceType.h	Header file for the ResourceType enumeration	556
/mnt/c/users/rudie/documents/sem2/214/project/src/ Road.cpp	Implementation file for the Road class	558

/mnt/c/users/rudie/documents/sem2/214/project/src/Road.h	Defines the <code>Road</code> class which inherits from the <code>Transportation</code> class	559
/mnt/c/users/rudie/documents/sem2/214/project/src/Sales.cpp	Implementation file for the <code>Sales</code> class	560
/mnt/c/users/rudie/documents/sem2/214/project/src/Sales.h	Header file for the <code>Sales</code> class	561
/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfactionStrategy.h	Header file for the <code>SatisfactionStrategy</code> class	563
/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.cpp	Implementation file for the <code>SatisfiedState</code> class	565
/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.h	Header file for the <code>SatisfiedState</code> class	565
/mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.cpp	Implementation of the <code>SetTaxCommand</code> class	567
/mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.h	Declaration of the <code>SetTaxCommand</code> class	568
/mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.cpp	Implementation of the <code>SewageManagement</code> class	569
/mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.h	Declaration of the <code>SewageManagement</code> class	570
/mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.cpp	Implementation of the <code>Taxi</code> class	572
/mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.h	Header file for the <code>Taxi</code> class, which represents a taxi in a public transit system	572
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.cpp	Implementation of the <code>TaxSatisfactionStrategy</code> class	574
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.h		575
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.cpp	Implementation of the <code>TaxSystem</code> class	576
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.h		576
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.cpp	Implementation of the <code>TaxType</code> class	577
/mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.h		578
/mnt/c/users/rudie/documents/sem2/214/project/src/TrafficFlow.h	Defines the <code>TrafficFlow</code> interface for traffic flow measurement	578
/mnt/c/users/rudie/documents/sem2/214/project/src/Train.cpp	Implementation of the <code>Train</code> class	579
/mnt/c/users/rudie/documents/sem2/214/project/src/Train.h	Header file for the <code>Train</code> class	580
/mnt/c/users/rudie/documents/sem2/214/project/src/Transportation.cpp	Implementation of the <code>Transportation</code> class	582
/mnt/c/users/rudie/documents/sem2/214/project/src/Transportation.h	Header file for the <code>Transportation</code> class	583
/mnt/c/users/rudie/documents/sem2/214/project/src/TransportationFactory.cpp	Implementation of the <code>TransportationFactory</code> class	584
/mnt/c/users/rudie/documents/sem2/214/project/src/TransportationFactory.h	Header file for the <code>TransportationFactory</code> class	585
/mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.cpp	Implementation of the <code>TransportManager</code> class	587
/mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.h	Header file for the <code>TransportManager</code> class	588
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus.cpp	Implementation of the <code>TraverseBus</code> class	590
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus.h	Header file for the <code>TraverseBus</code> class	591
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.cpp	Implementation of the <code>TraverseCargoAirport</code> class	593

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.h	
Header file for the TraverseCargoAirport class	594
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.cpp	
Implementation of the TraverseComercialAirport class	596
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.h	
Header file for the TraverseComercialAirport class	597
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.cpp	
Implementation of the TraverseFreightTrain class	599
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.h	
Header file for the TraverseFreightTrain class	600
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.cpp	
Implementation of the TraverseHighway class	602
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.h	
Header file for the TraverseHighway class	603
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.cpp	
Implementation of the TraverseInsideRoad class	605
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.h	
Header file for the TraverseInsideRoad class	606
/mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.cpp	
Implementation of the TraversePassengerTrain class	608
/mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.h	
Header file for the TraversePassengerTrain class	609
/mnt/c/users/rudie/documents/sem2/214/project/src/Traverser.h	
Defines the Traverser interface for iterating over Transportation objects	611
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.cpp	
Implementation of the TraverseState class	613
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.h	
Defines the TraverseState class and its interface for traversing through Transportation elements	614
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.cpp	
Implementation of the TraverseTaxi class	615
/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.h	
Header file for the TraverseTaxi class	616
/mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.cpp	
Implementation of the UnemployedState class	618
/mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.h	
Declaration of the UnemployedState class	619
/mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.cpp	
Implementation of the UnsatisfiedState class	621
/mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.h	
Declaration of the UnsatisfiedState class	622
/mnt/c/users/rudie/documents/sem2/214/project/src/Utility.h	
Declaration of the Utility class	623
/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityManager.h	
/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.cpp	
Implementation of the UtilityMediator class	625
/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.h	
Declaration of the UtilityMediator class	626
/mnt/c/users/rudie/documents/sem2/214/project/src/VAT.cpp	
Implementation of the VAT class	628
/mnt/c/users/rudie/documents/sem2/214/project/src/VAT.h	
Declaration of the VAT class	629
/mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.cpp	
Implementation of the WasteManagement class	630
/mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.h	
Declaration of the WasteManagement class	631
/mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.cpp	
Implementation of the WaterSupply class	633

/mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.h	634
Declaration of the WaterSupply class	

Chapter 4

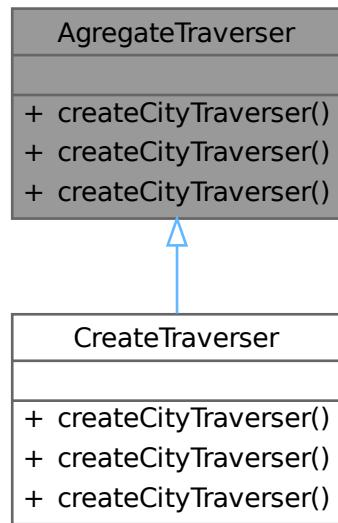
Class Documentation

4.1 AgregateTraverser Class Reference

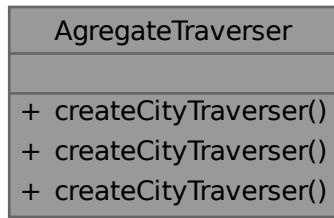
Abstract base class for creating [CityTraverser](#) objects.

```
#include <AgregateTraverser.h>
```

Inheritance diagram for AgregateTraverser:



Collaboration diagram for AggregateTraverser:



Public Member Functions

- virtual `CityTraverser * createCityTraverser ()=0`
Creates a new `CityTraverser` object.
- virtual `CityTraverser * createCityTraverser (Transportation *t)=0`
Creates a new `CityTraverser` object with a given `Transportation` object.
- virtual `CityTraverser * createCityTraverser (CityTraverser *t)=0`
Creates a new `CityTraverser` object by copying an existing `CityTraverser` object.

4.1.1 Detailed Description

Abstract base class for creating `CityTraverser` objects.

The `AggregateTraverser` class provides an interface for creating `CityTraverser` objects. It declares pure virtual functions that must be implemented by derived classes to create `CityTraverser` instances in different ways.

4.1.2 Member Function Documentation

4.1.2.1 `createCityTraverser()` [1/3]

```
virtual CityTraverser * AgregateTraverser::createCityTraverser ( ) [pure virtual]
```

Creates a new `CityTraverser` object.

Returns

A pointer to the newly created `CityTraverser` object.

Implemented in `CreateTraverser`.

4.1.2.2 `createCityTraverser()` [2/3]

```
virtual CityTraverser * AgregateTraverser::createCityTraverser (
    CityTraverser * t ) [pure virtual]
```

Creates a new `CityTraverser` object by copying an existing `CityTraverser` object.

Parameters

<i>t</i>	A pointer to an existing CityTraverser object.
----------	--

Returns

A pointer to the newly created [CityTraverser](#) object.

Implemented in [CreateTraverser](#).

4.1.2.3 `createCityTraverser()` [3/3]

```
virtual CityTraverser * AgregateTraverser::createCityTraverser (
    Transportation * t ) [pure virtual]
```

Creates a new [CityTraverser](#) object with a given [Transportation](#) object.

Parameters

<i>t</i>	A pointer to a Transportation object.
----------	---

Returns

A pointer to the newly created [CityTraverser](#) object.

Implemented in [CreateTraverser](#).

The documentation for this class was generated from the following file:

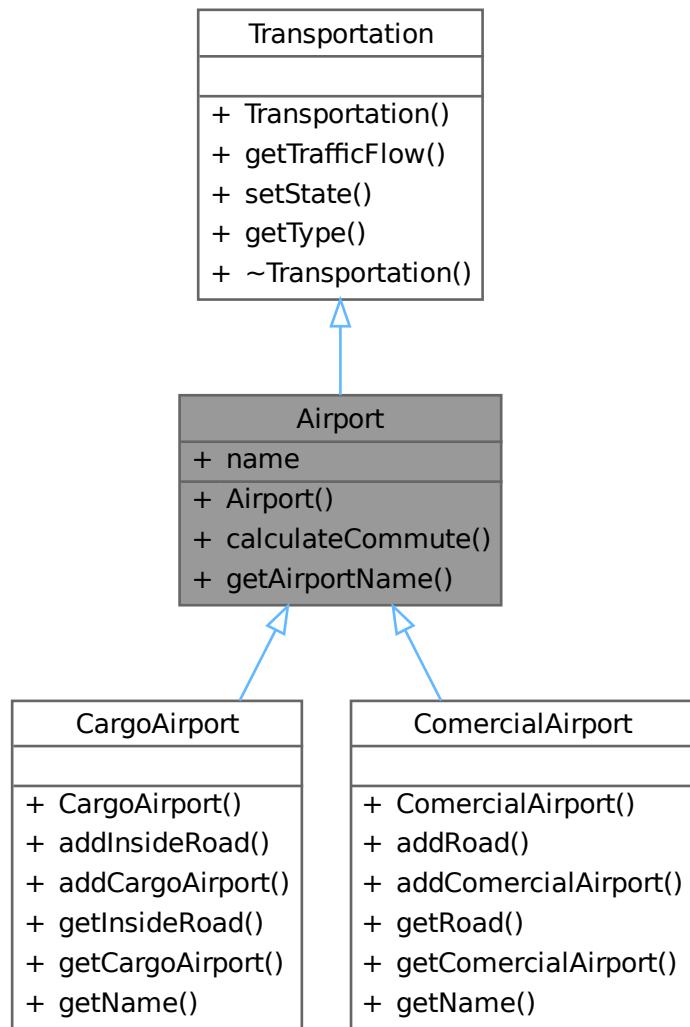
- /mnt/c/users/rudie/documents/sems/sem2/214/project/src/[AgregateTraverser.h](#)

4.2 Airport Class Reference

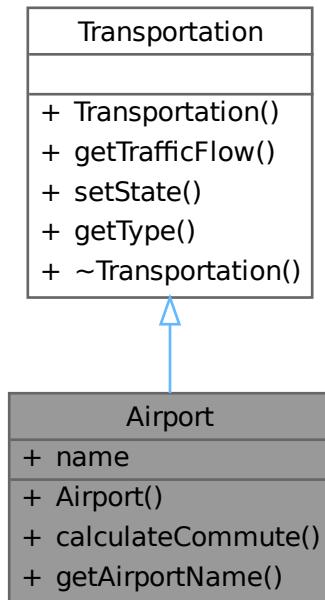
Represents an airport as a type of transportation.

```
#include <Airport.h>
```

Inheritance diagram for Airport:



Collaboration diagram for Airport:



Public Member Functions

- `Airport (char state, std::string name, char type)`
Constructs an `Airport` object.
- `float calculateCommute ()`
Calculates the commute time to the airport.
- `std::string getAirportName ()`
Gets the name of the airport.

Public Member Functions inherited from `Transportation`

- `Transportation (char state, char type)`
Constructor for the `Transportation` class.
- `float getTrafficFlow ()`
Gets the current traffic flow.
- `bool setState (char state)`
Sets the traffic flow state.
- `char getType ()`
Gets the type of transportation.
- `~Transportation ()`
Destructor for the `Transportation` class.

Public Attributes

- std::string **name**

Name of the airport.

4.2.1 Detailed Description

Represents an airport as a type of transportation.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Airport()

```
Airport::Airport (  
    char state,  
    std::string name,  
    char type )
```

Constructs an [Airport](#) object.

Construct a new [Airport](#) object.

Parameters

<i>state</i>	The state where the airport is located.
<i>name</i>	The name of the airport.
<i>type</i>	The type of transportation.

4.2.3 Member Function Documentation

4.2.3.1 calculateCommute()

```
float Airport::calculateCommute ( )
```

Calculates the commute time to the airport.

Returns

The commute time as a float.

4.2.3.2 getAirportName()

```
std::string Airport::getAirportName ( )
```

Gets the name of the airport.

Get the name of the airport.

Returns

The name of the airport as a string.

std::string The name of the airport.

The documentation for this class was generated from the following files:

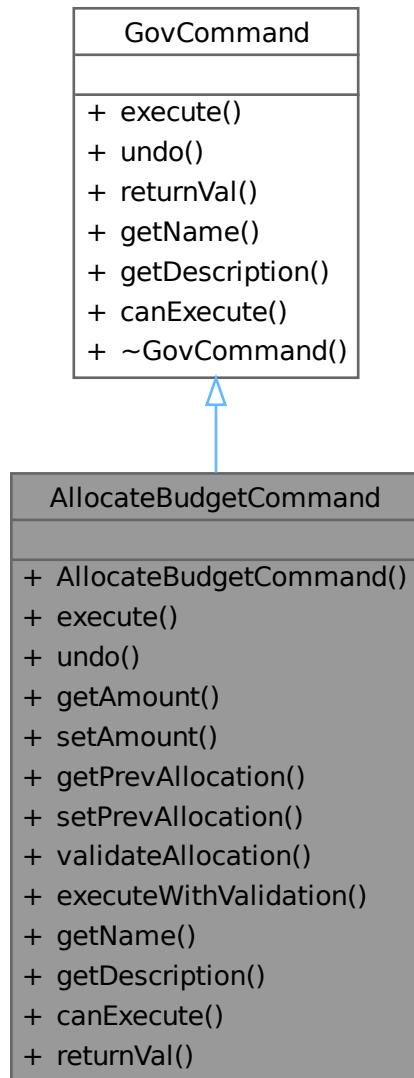
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Airport.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Airport.cpp](#)

4.3 AllocateBudgetCommand Class Reference

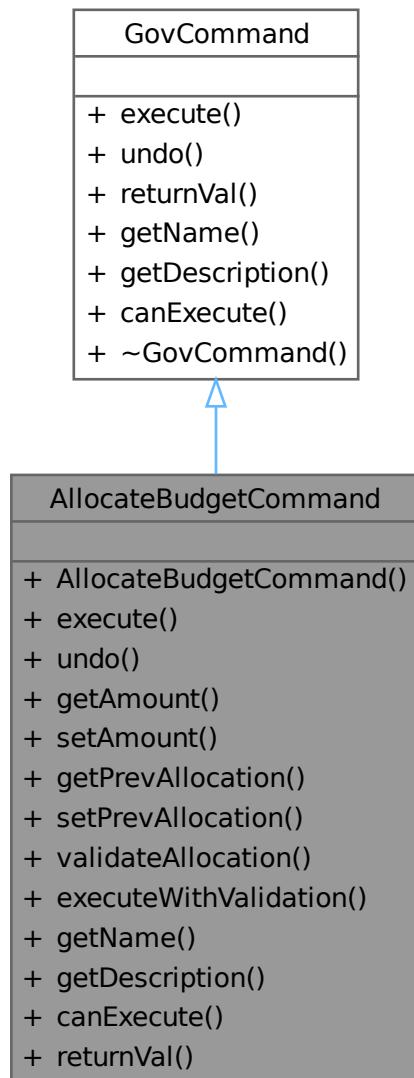
Represents a command to allocate a budget to a city service.

```
#include <AllocateBudgetCommand.h>
```

Inheritance diagram for AllocateBudgetCommand:



Collaboration diagram for AllocateBudgetCommand:



Public Member Functions

- **AllocateBudgetCommand (Government *gov, CityService &srv, double amt)**
Construct a new Allocate Budget Command object.
- void **execute ()** override
Executes the budget allocation command.
- void **undo ()** override
Undoes the budget allocation command.
- double **getAmount ()** const
Gets the amount to be allocated.
- void **setAmount (double amt)**

- `double getPrevAllocation () const`
Sets the amount to be allocated.
- `void setPrevAllocation (double prevAmt)`
Gets the previous allocation amount.
- `bool validateAllocation () const`
Sets the previous allocation amount.
- `void executeWithValidation ()`
Validates the budget allocation.
- `std::string getName () const override`
Executes the budget allocation command with validation.
- `std::string getDescription () const override`
Gets the name of the command.
- `bool canExecute () const override`
Gets the description of the command.
- `double returnVal () override`
Checks if the command can be executed.
- `double returnVal () override`
Returns the amount to be allocated.

Public Member Functions inherited from [GovCommand](#)

- `virtual ~GovCommand ()=default`
Virtual destructor.

4.3.1 Detailed Description

Represents a command to allocate a budget to a city service.

This class is used to allocate a budget to a city service. It includes methods to execute, undo, and validate the budget allocation, as well as getters and setters for the allocation amount and previous allocation amount.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 AllocateBudgetCommand()

```
AllocateBudgetCommand::AllocateBudgetCommand (
    Government * gov,
    CityService & srv,
    double amt )
```

Construct a new Allocate Budget Command object.

Parameters

<code>gov</code>	Pointer to the government object.
<code>srv</code>	Reference to the city service object.
<code>amt</code>	The amount to allocate.

4.3.3 Member Function Documentation

4.3.3.1 canExecute()

```
bool AllocateBudgetCommand::canExecute ( ) const [override], [virtual]
```

Checks if the command can be executed.

Returns

true if the command can be executed.
false if the command cannot be executed.

Implements [GovCommand](#).

4.3.3.2 execute()

```
void AllocateBudgetCommand::execute ( ) [override], [virtual]
```

Executes the budget allocation command.

Stores the previous allocation and allocates the new budget amount to the service.

Implements [GovCommand](#).

4.3.3.3 executeWithValidation()

```
void AllocateBudgetCommand::executeWithValidation ( )
```

Executes the budget allocation command with validation.

Throws an exception if the allocation is invalid.

Exceptions

<code>std::runtime_error</code>	if the allocation is invalid.
---------------------------------	-------------------------------

4.3.3.4 getAmount()

```
double AllocateBudgetCommand::getAmount ( ) const
```

Gets the amount to be allocated.

Returns

`double` The amount to be allocated.

4.3.3.5 `getDescription()`

```
std::string AllocateBudgetCommand::getDescription () const [override], [virtual]
```

Gets the description of the command.

Returns

`std::string` The description of the command.

Implements [GovCommand](#).

4.3.3.6 `getName()`

```
std::string AllocateBudgetCommand::getName () const [override], [virtual]
```

Gets the name of the command.

Returns

`std::string` The name of the command.

Implements [GovCommand](#).

4.3.3.7 `getPrevAllocation()`

```
double AllocateBudgetCommand::getPrevAllocation () const
```

Gets the previous allocation amount.

Returns

`double` The previous allocation amount.

4.3.3.8 `returnVal()`

```
double AllocateBudgetCommand::returnVal () [override], [virtual]
```

Returns the amount to be allocated.

Returns

`double` The amount to be allocated.

Implements [GovCommand](#).

4.3.3.9 `setAmount()`

```
void AllocateBudgetCommand::setAmount (
    double amt )
```

Sets the amount to be allocated.

Parameters

<i>amt</i>	The amount to be allocated.
------------	-----------------------------

Exceptions

<i>std::invalid_argument</i>	if the amount is negative.
------------------------------	----------------------------

4.3.3.10 setPrevAllocation()

```
void AllocateBudgetCommand::setPrevAllocation (
    double prevAmt )
```

Sets the previous allocation amount.

Parameters

<i>prevAmt</i>	The previous allocation amount.
----------------	---------------------------------

Exceptions

<i>std::invalid_argument</i>	if the previous allocation amount is negative.
------------------------------	--

4.3.3.11 undo()

```
void AllocateBudgetCommand::undo ( ) [override], [virtual]
```

Undoes the budget allocation command.

Reverts the budget allocation to the previous state.

Implements [GovCommand](#).

4.3.3.12 validateAllocation()

```
bool AllocateBudgetCommand::validateAllocation ( ) const
```

Validates the budget allocation.

Ensures the government has enough budget to allocate.

Returns

true if the allocation is valid.

false if the allocation is invalid.

The documentation for this class was generated from the following files:

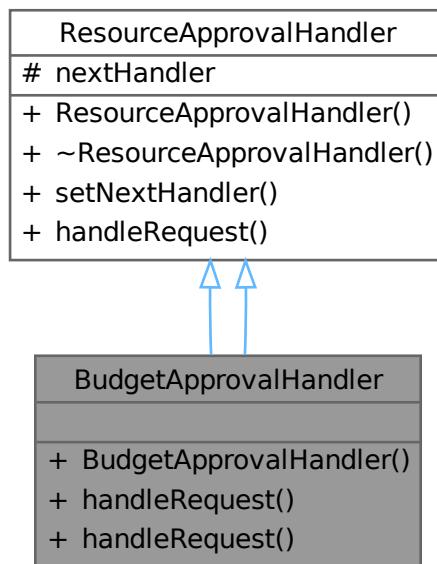
- /mnt/c/users/rudie/documents/sem2/214/project/src/[AllocateBudgetCommand.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[AllocateBudgetCommand.cpp](#)

4.4 BudgetApprovalHandler Class Reference

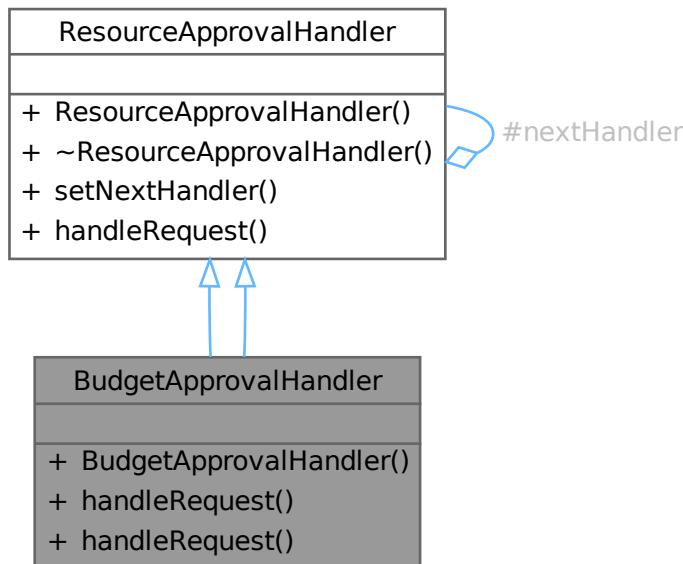
Handles budget approval requests.

```
#include <BudgetApprovalHandler.h>
```

Inheritance diagram for BudgetApprovalHandler:



Collaboration diagram for BudgetApprovalHandler:



Public Member Functions

- [BudgetApprovalHandler \(Government *gov, int cost\)](#)
Construct a new Budget Approval Handler object.
- bool [handleRequest \(ResourceType type, int quantity\) override](#)
Handles the budget approval request.
- bool [handleRequest \(ResourceType type, int quantity\) override](#)
Handles the budget approval request.

Public Member Functions inherited from [ResourceApprovalHandler](#)

- [ResourceApprovalHandler \(\)](#)
Constructs a new [ResourceApprovalHandler](#) object.
- virtual [~ResourceApprovalHandler \(\)](#)
Destroys the [ResourceApprovalHandler](#) object.
- void [setNextHandler \(ResourceApprovalHandler *handler\)](#)
Sets the next handler in the chain.

Additional Inherited Members

Protected Attributes inherited from [ResourceApprovalHandler](#)

- [ResourceApprovalHandler * nextHandler](#)
Pointer to the next handler in the chain.

4.4.1 Detailed Description

Handles budget approval requests.

This class is responsible for handling budget approval requests. It includes methods to handle requests and allocate budget to city services.

The [BudgetApprovalHandler](#) class checks if the budget allows for the requested resource.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BudgetApprovalHandler()

```
BudgetApprovalHandler::BudgetApprovalHandler (
    Government * gov,
    int cost ) [inline]
```

Construct a new Budget Approval Handler object.

Parameters

<i>gov</i>	Pointer to the government object.
<i>cost</i>	Cost of the resource.

4.4.3 Member Function Documentation

4.4.3.1 handleRequest() [1/2]

```
bool BudgetApprovalHandler::handleRequest (
    ResourceType type,
    int quantity ) [inline], [override], [virtual]
```

Handles the budget approval request.

Calculates the total cost and allocates the budget if sufficient funds are available.

Parameters

<i>type</i>	The type of resource.
<i>quantity</i>	The quantity of the resource.

Returns

true if the request is approved and budget is allocated.
false if the request is denied due to insufficient budget.

Reimplemented from [ResourceApprovalHandler](#).

4.4.3.2 handleRequest() [2/2]

```
bool BudgetApprovalHandler::handleRequest (
    ResourceType type,
    int quantity ) [inline], [override], [virtual]
```

Handles the budget approval request.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource.

Returns

True if the request is approved, false otherwise.

Reimplemented from [ResourceApprovalHandler](#).

The documentation for this class was generated from the following files:

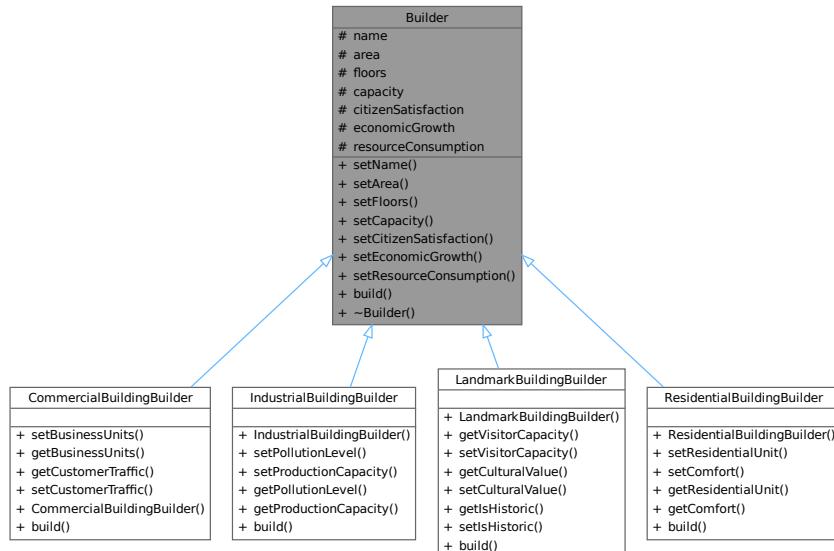
- /mnt/c/users/rudie/documents/sem2/214/project/src/[BudgetApprovalHandler.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceApprovalHandler.h](#)

4.5 Builder Class Reference

Base class for all builders.

```
#include <Builder.h>
```

Inheritance diagram for Builder:



Collaboration diagram for Builder:

Builder	
# name	
# area	
# floors	
# capacity	
# citizenSatisfaction	
# economicGrowth	
# resourceConsumption	
+ setName()	
+ setArea()	
+ setFloors()	
+ setCapacity()	
+ setCitizenSatisfaction()	
+ setEconomicGrowth()	
+ setResourceConsumption()	
+ build()	
+ ~Builder()	

Public Member Functions

- **Builder & setName (string name)**
Sets the name of the building.
- **Builder & setArea (float area)**
Sets the area of the building.
- **Builder & setFloors (int floors)**
Sets the number of floors in the building.
- **Builder & setCapacity (int capacity)**
Sets the capacity of the building.
- **Builder & setCitizenSatisfaction (float citizenSatisfaction)**
Sets the citizen satisfaction of the building.
- **Builder & setEconomicGrowth (float economicGrowth)**
Sets the economic growth of the building.
- **Builder & setResourceConsumption (float resourceConsumption)**
Sets the resource consumption of the building.
- **virtual std::unique_ptr< Building > build ()=0**
Builds the building.
- **virtual ~Builder ()=default**
Virtual destructor for the `Builder` class.

Protected Attributes

- string **name**
Name of the building.
- float **area** = 0.0f
Area of the building.
- int **floors** = 0
Number of floors in the building.
- int **capacity** = 0
Capacity of the building.
- float **citizenSatisfaction** = 0.0f
Citizen satisfaction of the building.
- float **economicGrowth** = 0.0f
Economic growth of the building.
- float **resourceConsumption** = 0.0f
Resource consumption of the building.

4.5.1 Detailed Description

Base class for all builders.

This class provides methods to set various properties of a building such as name, area, floors, capacity, citizen satisfaction, economic growth, and resource consumption.

4.5.2 Member Function Documentation

4.5.2.1 build()

```
virtual std::unique_ptr< Building > Builder::build () [pure virtual]
```

Builds the building.

Returns

`std::unique_ptr<Building>` A unique pointer to the built building.

Implemented in [CommercialBuildingBuilder](#), [IndustrialBuildingBuilder](#), [LandmarkBuildingBuilder](#), and [ResidentialBuildingBuilder](#).

4.5.2.2 setArea()

```
Builder & Builder::setArea (
    float area )
```

Sets the area of the building.

Parameters

<code>area</code>	The area of the building.
-------------------	---------------------------

Returns

`Builder&` Reference to the `Builder` object.

4.5.2.3 `setCapacity()`

```
Builder & Builder::setCapacity (
    int capacity )
```

Sets the capacity of the building.

Parameters

<code>capacity</code>	The capacity of the building.
-----------------------	-------------------------------

Returns

`Builder&` Reference to the `Builder` object.

4.5.2.4 `setCitizenSatisfaction()`

```
Builder & Builder::setCitizenSatisfaction (
    float citizenSatisfaction )
```

Sets the citizen satisfaction of the building.

Parameters

<code>citizenSatisfaction</code>	The citizen satisfaction of the building.
----------------------------------	---

Returns

`Builder&` Reference to the `Builder` object.

4.5.2.5 `setEconomicGrowth()`

```
Builder & Builder::setEconomicGrowth (
    float economicGrowth )
```

Sets the economic growth of the building.

Parameters

<code>economicGrowth</code>	The economic growth of the building.
-----------------------------	--------------------------------------

Returns

`Builder`& Reference to the `Builder` object.

4.5.2.6 setFloors()

```
Builder & Builder::setFloors (
    int floors )
```

Sets the number of floors in the building.

Parameters

<code>floors</code>	The number of floors in the building.
---------------------	---------------------------------------

Returns

`Builder`& Reference to the `Builder` object.

4.5.2.7 setName()

```
Builder & Builder::setName (
    string name )
```

Sets the name of the building.

Parameters

<code>name</code>	The name of the building.
-------------------	---------------------------

Returns

`Builder`& Reference to the `Builder` object.

4.5.2.8 setResourceConsumption()

```
Builder & Builder::setResourceConsumption (
    float resourceConsumption )
```

Sets the resource consumption of the building.

Parameters

<code>resourceConsumption</code>	The resource consumption of the building.
----------------------------------	---

Returns

`Builder&` Reference to the `Builder` object.

The documentation for this class was generated from the following files:

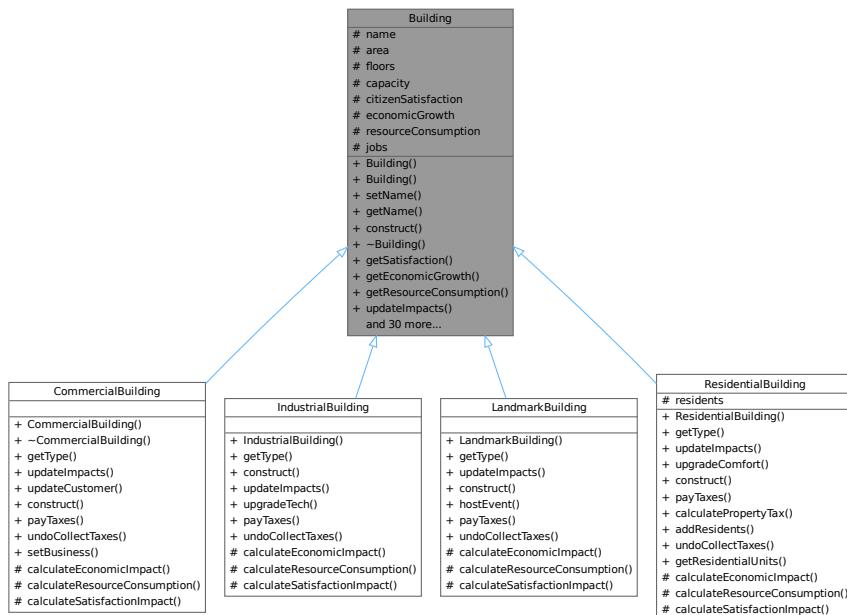
- /mnt/c/users/rudie/documents/sem2/214/project/src/`Builder.h`
- /mnt/c/users/rudie/documents/sem2/214/project/src/`Builder.cpp`

4.6 Building Class Reference

Represents a building with various properties and job management capabilities.

```
#include <Building.h>
```

Inheritance diagram for Building:



Collaboration diagram for Building:

Building
name
area
floors
capacity
citizenSatisfaction
economicGrowth
resourceConsumption
jobs
+ Building()
+ Building()
+ setName()
+ getName()
+ construct()
+ ~Building()
+ getSatisfaction()
+ getEconomicGrowth()
+ getResourceConsumption()
+ updateImpacts()
and 30 more...

Public Member Functions

- **Building** (const std::string &**name**, float **area**, int **floors**, int **capacity**, float **satisfactionImpact**, float **growthImpact**, float **consumption**)
Main constructor for setting attributes directly.
- **Building** (int **Builder**)
*Alternative constructor using an integer parameter, potentially for **Builder** pattern integration.*
- void **setName** (const std::string &**name**)
Sets the name of the building.
- std::string **getName** () const
Gets the name of the building.
- virtual void **construct** ()=0
Pure virtual function to construct the building.
- virtual ~**Building** ()=default
*Virtual destructor for the **Building** class.*
- float **getSatisfaction** () const
Gets the citizen satisfaction impact of the building.
- float **getEconomicGrowth** () const
Gets the economic growth impact of the building.

- float `getResourceConsumption () const`
Gets the resource consumption of the building.
- virtual void `updateImpacts ()=0`
Pure virtual function to update the impacts of the building.
- virtual std::string `getType () const =0`
Pure virtual function to get the type of the building.
- void `addJob (std::shared_ptr< Jobs > job)`
Adds a job to the building.
- void `listJobs () const`
Lists all jobs in the building.
- bool `hireEmployee (const std::string &jobTitle)`
Hires an employee for a job if available.
- void `releaseEmployee (const std::string &jobTitle)`
Releases an employee from a job.
- std::shared_ptr< Jobs > `getAvailableJob ()`
Provides access to an available job.
- void `displayJobInfo (const std::string &jobTitle) const`
Displays job information.
- const std::vector< std::shared_ptr< Jobs > > & `getJobs () const`
Returns the job list as a const reference.
- virtual double `payTaxes (TaxType *taxType)=0`
Pure virtual function to collect taxes from the building.
- virtual void `undoCollectTaxes ()=0`
Pure virtual function to undo collecting taxes from the building.
- `Building (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)`
Main constructor for setting attributes directly.
- `Building (int Builder)`
Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.
- void `setName (const std::string &name)`
Sets the name of the building.
- std::string `getName () const`
Gets the name of the building.
- virtual void `construct ()=0`
Pure virtual function to construct the building.
- virtual ~`Building ()=default`
Virtual destructor for the [Building](#) class.
- float `getSatisfaction () const`
Gets the citizen satisfaction impact of the building.
- float `getEconomicGrowth () const`
Gets the economic growth impact of the building.
- float `getResourceConsumption () const`
Gets the resource consumption of the building.
- virtual void `updateImpacts ()=0`
Pure virtual function to update the impacts of the building.
- virtual std::string `getType () const =0`
Pure virtual function to get the type of the building.
- void `addJob (std::shared_ptr< Jobs > job)`
Adds a job to the building.
- void `listJobs () const`
Lists all jobs in the building.

- bool `hireEmployee` (const std::string &jobTitle)
Hires an employee for a job if available.
- void `releaseEmployee` (const std::string &jobTitle)
Releases an employee from a job.
- std::shared_ptr< `Jobs` > `getAvailableJob` ()
Provides access to an available job.
- void `displayJobInfo` (const std::string &jobTitle) const
Displays job information.
- const std::vector< std::shared_ptr< `Jobs` > > & `getJobs` () const
Returns the job list as a const reference.
- virtual double `payTaxes` (`TaxType` *taxType)=0
Pure virtual function to collect taxes from the building.
- virtual void `undoCollectTaxes` ()=0
Pure virtual function to undo collecting taxes from the building.

Protected Attributes

- std::string **name**
Name of the building.
- float **area**
Area of the building.
- int **floors**
Number of floors in the building.
- int **capacity**
Capacity of the building.
- float **citizenSatisfaction**
Citizen satisfaction impact of the building.
- float **economicGrowth**
Economic growth impact of the building.
- float **resourceConsumption**
Resource consumption of the building.
- std::vector< std::shared_ptr< `Jobs` > > **jobs**
Collection of jobs as shared pointers.

4.6.1 Detailed Description

Represents a building with various properties and job management capabilities.

This class provides methods to set and get various properties of a building such as name, area, floors, capacity, citizen satisfaction, economic growth, and resource consumption. It also includes methods to manage jobs within the building.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Building() [1/4]

```
Building::Building (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float satisfactionImpact,
    float growthImpact,
    float consumption )
```

Main constructor for setting attributes directly.

Parameters

<i>name</i>	The name of the building.
<i>area</i>	The area of the building.
<i>floors</i>	The number of floors in the building.
<i>capacity</i>	The capacity of the building.
<i>satisfactionImpact</i>	The impact on citizen satisfaction.
<i>growthImpact</i>	The impact on economic growth.
<i>consumption</i>	The resource consumption of the building.

4.6.2.2 Building() [2/4]

```
Building::Building (
    int Builder )
```

Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.

Parameters

<i>Builder</i>	An integer representing the builder type.
--------------------------------	---

4.6.2.3 Building() [3/4]

```
Building::Building (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float satisfactionImpact,
    float growthImpact,
    float consumption )
```

Main constructor for setting attributes directly.

Parameters

<i>name</i>	The name of the building.
<i>area</i>	The area of the building.
<i>floors</i>	The number of floors in the building.
<i>capacity</i>	The capacity of the building.
<i>satisfactionImpact</i>	The impact on citizen satisfaction.
<i>growthImpact</i>	The impact on economic growth.
<i>consumption</i>	The resource consumption of the building.

4.6.2.4 Building() [4/4]

```
Building::Building (
```

```
int Builder )
```

Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.

Parameters

Builder	An integer representing the builder type.
-------------------------	---

4.6.3 Member Function Documentation

4.6.3.1 [addJob\(\)](#) [1/2]

```
void Building::addJob ( std::shared_ptr< Jobs > job )
```

Adds a job to the building.

Parameters

job	A shared pointer to the job to be added.
---------------------	--

4.6.3.2 [addJob\(\)](#) [2/2]

```
void Building::addJob ( std::shared_ptr< Jobs > job )
```

Adds a job to the building.

Parameters

job	A shared pointer to the job to be added.
---------------------	--

4.6.3.3 [construct\(\)](#) [1/2]

```
virtual void Building::construct ( ) [pure virtual]
```

Pure virtual function to construct the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

4.6.3.4 [construct\(\)](#) [2/2]

```
virtual void Building::construct ( ) [pure virtual]
```

Pure virtual function to construct the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

4.6.3.5 `displayJobInfo()` [1/2]

```
void Building::displayJobInfo (
    const std::string & jobTitle ) const
```

Displays job information.

Parameters

<i>jobTitle</i>	The title of the job to display information for.
-----------------	--

4.6.3.6 `displayJobInfo()` [2/2]

```
void Building::displayJobInfo (
    const std::string & jobTitle ) const
```

Displays job information.

Parameters

<i>jobTitle</i>	The title of the job to display information for.
-----------------	--

4.6.3.7 `getAvailableJob()` [1/2]

```
std::shared_ptr< Jobs > Building::getAvailableJob ( )
```

Provides access to an available job.

Returns

`std::shared_ptr<Jobs>` A shared pointer to an available job.

4.6.3.8 `getAvailableJob()` [2/2]

```
std::shared_ptr< Jobs > Building::getAvailableJob ( )
```

Provides access to an available job.

Returns

`std::shared_ptr<Jobs>` A shared pointer to an available job.

4.6.3.9 `getEconomicGrowth()` [1/2]

```
float Building::getEconomicGrowth ( ) const
```

Gets the economic growth impact of the building.

Returns

`float` The economic growth impact.

4.6.3.10 getEconomicGrowth() [2/2]

```
float Building::getEconomicGrowth ( ) const
```

Gets the economic growth impact of the building.

Returns

float The economic growth impact.

4.6.3.11 getJobs() [1/2]

```
const std::vector< std::shared_ptr< Jobs > > & Building::getJobs ( ) const
```

Returns the job list as a const reference.

Returns

const std::vector<std::shared_ptr<Jobs>>& The job list.

4.6.3.12 getJobs() [2/2]

```
const std::vector< std::shared_ptr< Jobs > > & Building::getJobs ( ) const
```

Returns the job list as a const reference.

Returns

const std::vector<std::shared_ptr<Jobs>>& The job list.

4.6.3.13 getName() [1/2]

```
std::string Building::getName ( ) const
```

Gets the name of the building.

Returns

std::string The name of the building.

4.6.3.14 getName() [2/2]

```
std::string Building::getName ( ) const
```

Gets the name of the building.

Returns

std::string The name of the building.

4.6.3.15 getResourceConsumption() [1/2]

```
float Building::getResourceConsumption ( ) const
```

Gets the resource consumption of the building.

Returns

float The resource consumption.

4.6.3.16 getResourceConsumption() [2/2]

```
float Building::getResourceConsumption ( ) const
```

Gets the resource consumption of the building.

Returns

float The resource consumption.

4.6.3.17 getSatisfaction() [1/2]

```
float Building::getSatisfaction ( ) const
```

Gets the citizen satisfaction impact of the building.

Returns

float The citizen satisfaction impact.

4.6.3.18 getSatisfaction() [2/2]

```
float Building::getSatisfaction ( ) const
```

Gets the citizen satisfaction impact of the building.

Returns

float The citizen satisfaction impact.

4.6.3.19 getType() [1/2]

```
virtual std::string Building::getType ( ) const [pure virtual]
```

Pure virtual function to get the type of the building.

Returns

std::string The type of the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

4.6.3.20 getType() [2/2]

```
virtual std::string Building::getType ( ) const [pure virtual]
```

Pure virtual function to get the type of the building.

Returns

std::string The type of the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

4.6.3.21 hireEmployee() [1/2]

```
bool Building::hireEmployee ( const std::string & jobTitle )
```

Hires an employee for a job if available.

Parameters

<i>jobTitle</i>	The title of the job to hire for.
-----------------	-----------------------------------

Returns

true if the employee was successfully hired.
false if the job is not available or already occupied.

4.6.3.22 hireEmployee() [2/2]

```
bool Building::hireEmployee ( const std::string & jobTitle )
```

Hires an employee for a job if available.

Parameters

<i>jobTitle</i>	The title of the job to hire for.
-----------------	-----------------------------------

Returns

true if the employee was successfully hired.
false if the job is not available or already occupied.

4.6.3.23 payTaxes() [1/2]

```
virtual double Building::payTaxes ( TaxType * taxType ) [pure virtual]
```

Pure virtual function to collect taxes from the building.

Parameters

<i>taxType</i>	Pointer to the tax type.
----------------	--------------------------

Returns

double The amount of taxes collected.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [ResidentialBuilding](#), and [LandmarkBuilding](#).

4.6.3.24 payTaxes() [2/2]

```
virtual double Building::payTaxes (
    TaxType * taxType ) [pure virtual]
```

Pure virtual function to collect taxes from the building.

Parameters

<i>taxType</i>	Pointer to the tax type.
----------------	--------------------------

Returns

double The amount of taxes collected.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [ResidentialBuilding](#), and [LandmarkBuilding](#).

4.6.3.25 releaseEmployee() [1/2]

```
void Building::releaseEmployee (
    const std::string & jobTitle )
```

Releases an employee from a job.

Parameters

<i>jobTitle</i>	The title of the job to release.
-----------------	----------------------------------

4.6.3.26 releaseEmployee() [2/2]

```
void Building::releaseEmployee (
    const std::string & jobTitle )
```

Releases an employee from a job.

Parameters

<i>jobTitle</i>	The title of the job to release.
-----------------	----------------------------------

4.6.3.27 setName() [1/2]

```
void Building::setName (
    const std::string & name )
```

Sets the name of the building.

Parameters

<i>name</i>	The name of the building.
-------------	---------------------------

4.6.3.28 setName() [2/2]

```
void Building::setName (
    const std::string & name )
```

Sets the name of the building.

Parameters

<i>name</i>	The name of the building.
-------------	---------------------------

4.6.3.29 undoCollectTaxes() [1/2]

```
virtual void Building::undoCollectTaxes ( ) [pure virtual]
```

Pure virtual function to undo collecting taxes from the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [ResidentialBuilding](#), and [LandmarkBuilding](#).

4.6.3.30 undoCollectTaxes() [2/2]

```
virtual void Building::undoCollectTaxes ( ) [pure virtual]
```

Pure virtual function to undo collecting taxes from the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [ResidentialBuilding](#), and [LandmarkBuilding](#).

4.6.3.31 updateImpacts() [1/2]

```
virtual void Building::updateImpacts ( ) [pure virtual]
```

Pure virtual function to update the impacts of the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

4.6.3.32 updateImpacts() [2/2]

```
virtual void Building::updateImpacts ( ) [pure virtual]
```

Pure virtual function to update the impacts of the building.

Implemented in [CommercialBuilding](#), [IndustrialBuilding](#), [LandmarkBuilding](#), and [ResidentialBuilding](#).

The documentation for this class was generated from the following files:

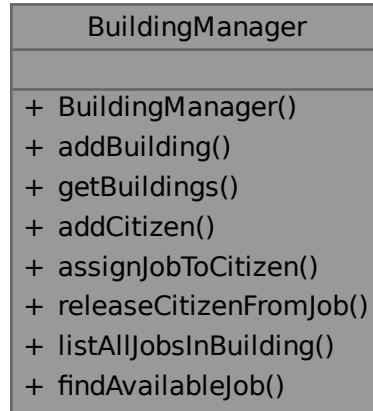
- /mnt/c/users/rudie/documents/sem2/214/project/src/Building.cpp
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Building.h](#)

4.7 BuildingManager Class Reference

Manages buildings and citizens.

```
#include <BuildingManager.h>
```

Collaboration diagram for BuildingManager:



Public Member Functions

- **BuildingManager** (const std::vector< std::shared_ptr< [Building](#) > > &buildingList)
Constructs a new [BuildingManager](#) object.
- void **addBuilding** ([Building](#) *building)
Adds a building to the list of managed buildings.
- std::vector< std::shared_ptr< [Building](#) > > **getBuildings** ()
Gets the list of managed buildings.
- void **addCitizen** ([Citizen](#) *citizen)
Adds a citizen to the list of managed citizens.

- bool `assignJobToCitizen` (const std::string &jobTitle, Citizen *citizen, Building *building)
Assigns a job to a citizen in a specific building.
- void `releaseCitizenFromJob` (Citizen *citizen)
Releases a citizen from their current job.
- void `listAllJobsInBuilding` (const Building *building) const
Lists all jobs available in a specific building.
- std::shared_ptr< Jobs > `findAvailableJob` ()
Finds an available job across all buildings.

4.7.1 Detailed Description

Manages buildings and citizens.

This class provides methods to add buildings and citizens, assign jobs to citizens, release citizens from jobs, list all jobs in a building, and find available jobs across all buildings.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 BuildingManager()

```
BuildingManager::BuildingManager (
    const std::vector< std::shared_ptr< Building > > & buildingList )
```

Constructs a new BuildingManager object.

Parameters

<code>buildingList</code>	A list of buildings to be managed.
---------------------------	------------------------------------

4.7.3 Member Function Documentation

4.7.3.1 addBuilding()

```
void BuildingManager::addBuilding (
    Building * building )
```

Adds a building to the list of managed buildings.

Parameters

<code>building</code>	Pointer to the building to be added.
-----------------------	--------------------------------------

4.7.3.2 addCitizen()

```
void BuildingManager::addCitizen (
    Citizen * citizen )
```

Adds a citizen to the list of managed citizens.

Parameters

<i>citizen</i>	Pointer to the citizen to be added.
----------------	-------------------------------------

4.7.3.3 assignJobToCitizen()

```
bool BuildingManager::assignJobToCitizen (
    const std::string & jobTitle,
    Citizen * citizen,
    Building * building )
```

Assigns a job to a citizen in a specific building.

Assigns a job to a citizen if the job is available in the specified building.

Parameters

<i>jobTitle</i>	The title of the job to be assigned.
<i>citizen</i>	Pointer to the citizen to be assigned the job.
<i>building</i>	Pointer to the building where the job is located.

Returns

true if the job was successfully assigned to the citizen.
false if the job could not be assigned.

4.7.3.4 findAvailableJob()

```
std::shared_ptr< Jobs > BuildingManager::findAvailableJob ( )
```

Finds an available job across all buildings.

Returns

std::shared_ptr<Jobs> A shared pointer to an available job, or nullptr if no jobs are available.

4.7.3.5 getBuildings()

```
std::vector< std::shared_ptr< Building > > BuildingManager::getBuildings ( )
```

Gets the list of managed buildings.

Returns

std::vector<std::shared_ptr<Building>> The list of managed buildings.

4.7.3.6 `listAllJobsInBuilding()`

```
void BuildingManager::listAllJobsInBuilding (
    const Building * building ) const
```

Lists all jobs available in a specific building.

Lists all jobs in the specified building.

Parameters

<i>building</i>	Pointer to the building whose jobs are to be listed.
-----------------	--

4.7.3.7 releaseCitizenFromJob()

```
void BuildingManager::releaseCitizenFromJob (
    Citizen * citizen )
```

Releases a citizen from their current job.

Releases a citizen from their job by updating both the citizen and the job's state.

Parameters

<i>citizen</i>	Pointer to the citizen to be released from their job.
----------------	---

The documentation for this class was generated from the following files:

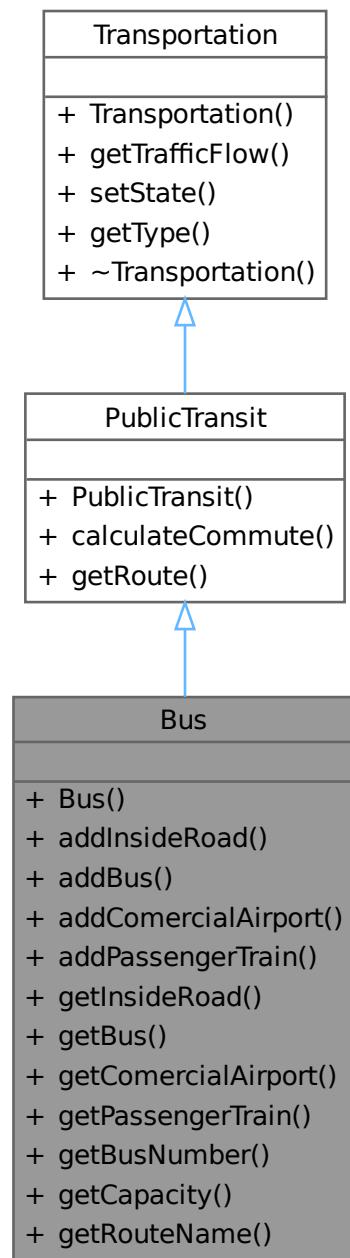
- /mnt/c/users/rudie/documents/sem2/214/project/src/[BuildingManager.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[BuildingManager.cpp](#)

4.8 Bus Class Reference

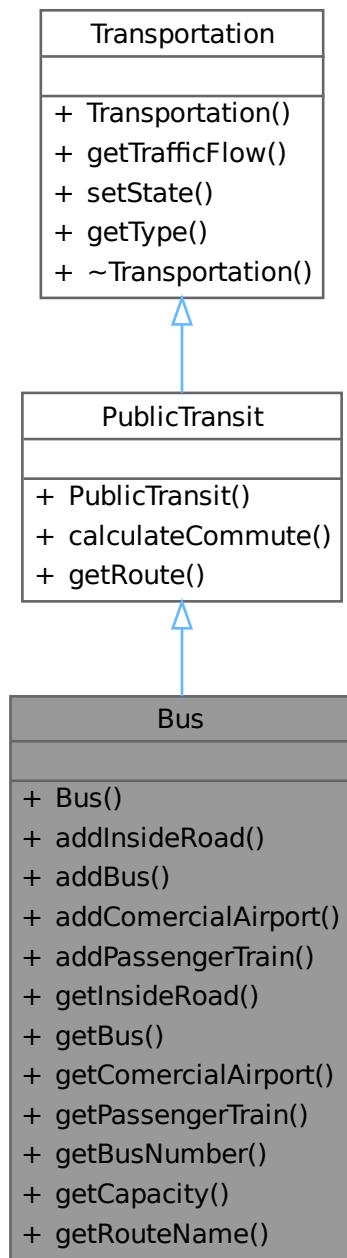
Represents a bus in the public transit system.

```
#include <Bus.h>
```

Inheritance diagram for Bus:



Collaboration diagram for Bus:



Public Member Functions

- `Bus` (char state, std::string route, int busNumber, int capacity)
Constructs a Bus object.
- bool `addInsideRoad` (`InsideRoad` *insideRoad)
Adds an InsideRoad connection to the bus.
- bool `addBus` (`Bus` *bus)

- Adds a [Bus](#) connection to the bus.
 - bool [addComercialAirport](#) ([ComercialAirport](#) *comercialAirport)
 Adds a [ComercialAirport](#) connection to the bus.
 - bool [addPassengerTrain](#) ([PassengerTrain](#) *passengerTrain)
 Adds a [PassengerTrain](#) connection to the bus.
 - [InsideRoad](#) * [getInsideRoad](#) (std::size_t x)
 Gets an [InsideRoad](#) connection by index.
 - [Bus](#) * [getBus](#) (std::size_t x)
 Gets a [Bus](#) connection by index.
 - [ComercialAirport](#) * [getComercialAirport](#) (std::size_t x)
 Gets a [ComercialAirport](#) connection by index.
 - [PassengerTrain](#) * [getPassengerTrain](#) (std::size_t x)
 Gets a [PassengerTrain](#) connection by index.
 - int [getBusNumber](#) ()
 Gets the bus number.
 - int [getCapacity](#) ()
 Gets the capacity of the bus.
 - std::string [getRouteName](#) ()
 Gets the route name of the bus.

Public Member Functions inherited from [PublicTransit](#)

- [PublicTransit](#) (char state, std::string route, char type)
Constructor for the [PublicTransit](#) class.
- float [calculateCommute](#) ()
Calculate the commute time for the public transit.
- std::string [getRoute](#) ()
Get the route of the public transit.

Public Member Functions inherited from [Transportation](#)

- [Transportation](#) (char state, char type)
Constructor for the [Transportation](#) class.
- float [getTrafficFlow](#) ()
Gets the current traffic flow.
- bool [setState](#) (char state)
Sets the traffic flow state.
- char [getType](#) ()
Gets the type of transportation.
- [~Transportation](#) ()
Destructor for the [Transportation](#) class.

4.8.1 Detailed Description

Represents a bus in the public transit system.

The [Bus](#) class inherits from [PublicTransit](#) and includes additional attributes and methods specific to buses, such as bus number, capacity, and connections to other transportation modes.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Bus()

```
Bus::Bus (
    char state,
    std::string route,
    int busNumber,
    int capacity )
```

Constructs a [Bus](#) object.

Construct a new [Bus](#) object.

Parameters

<i>state</i>	The state of the bus.
<i>route</i>	The route name of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.
<i>state</i>	The state where the bus operates.
<i>route</i>	The route of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.

4.8.3 Member Function Documentation

4.8.3.1 addBus()

```
bool Bus::addBus (
    Bus * bus )
```

Adds a [Bus](#) connection to the bus.

Adds a bus to the list of buses.

Parameters

<i>bus</i>	Pointer to the Bus object.
------------	--

Returns

True if the [Bus](#) was added successfully, false otherwise.

Parameters

<i>bus</i>	Pointer to the bus to be added.
------------	---------------------------------

Returns

true if the bus was added successfully.
false if the bus is already in the list.

4.8.3.2 addComercialAirport()

```
bool Bus::addComercialAirport (
    ComercialAirport * comercialAirport )
```

Adds a [ComercialAirport](#) connection to the bus.

Adds a commercial airport to the list of commercial airports.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport object.
-------------------------	---

Returns

True if the [ComercialAirport](#) was added successfully, false otherwise.

Parameters

<i>comercialAirport</i>	Pointer to the commercial airport to be added.
-------------------------	--

Returns

true if the commercial airport was added successfully.
false if the commercial airport is already in the list.

4.8.3.3 addInsideRoad()

```
bool Bus::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an [InsideRoad](#) connection to the bus.

Adds an inside road to the bus's route.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad object.
-------------------	---

Returns

True if the [InsideRoad](#) was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the inside road to be added.
-------------------	---

Returns

true if the inside road was added successfully.

false if the inside road is already in the list.

4.8.3.4 addPassengerTrain()

```
bool Bus::addPassengerTrain (  
    PassengerTrain * passengerTrain )
```

Adds a [PassengerTrain](#) connection to the bus.

Adds a passenger train to the list of passenger trains.

Parameters

<i>passengerTrain</i>	Pointer to the PassengerTrain object.
-----------------------	---

Returns

True if the [PassengerTrain](#) was added successfully, false otherwise.

Parameters

<i>passengerTrain</i>	Pointer to the passenger train to be added.
-----------------------	---

Returns

true if the passenger train was added successfully.

false if the passenger train is already in the list.

4.8.3.5 getBus()

```
Bus * Bus::getBus (   
    std::size_t x )
```

Gets a [Bus](#) connection by index.

Gets a bus from the list of buses.

Parameters

<i>x</i>	The index of the Bus .
----------	--

Returns

Pointer to the [Bus](#) object.

Parameters

x	The index of the bus to retrieve.
---	-----------------------------------

Returns

Bus* Pointer to the bus, or nullptr if the index is out of range.

4.8.3.6 getBusNumber()

```
int Bus::getBusNumber ( )
```

Gets the bus number.

Returns

The bus number.

```
int The bus number.
```

4.8.3.7 getCapacity()

```
int Bus::getCapacity ( )
```

Gets the capacity of the bus.

Returns

The capacity of the bus.

```
int The capacity of the bus.
```

4.8.3.8 getComercialAirport()

```
ComercialAirport * Bus::getComercialAirport (
    std::size_t x )
```

Gets a [ComercialAirport](#) connection by index.

Gets a commercial airport from the list of commercial airports.

Parameters

x	The index of the ComercialAirport .
---	---

Returns

Pointer to the [ComercialAirport](#) object.

Parameters

x	The index of the commercial airport to retrieve.
---	--

Returns

ComercialAirport* Pointer to the commercial airport, or nullptr if the index is out of range.

4.8.3.9 getInsideRoad()

```
InsideRoad * Bus::getInsideRoad (
    std::size_t x )
```

Gets an [InsideRoad](#) connection by index.

Gets an inside road from the list of inside roads.

Parameters

x	The index of the InsideRoad .
---	---

Returns

Pointer to the [InsideRoad](#) object.

Parameters

x	The index of the inside road to retrieve.
---	---

Returns

InsideRoad* Pointer to the inside road, or nullptr if the index is out of range.

4.8.3.10 getPassengerTrain()

```
PassengerTrain * Bus::getPassengerTrain (
    std::size_t x )
```

Gets a [PassengerTrain](#) connection by index.

Gets a passenger train from the list of passenger trains.

Parameters

x	The index of the PassengerTrain .
---	---

Returns

Pointer to the [PassengerTrain](#) object.

Parameters

x	The index of the passenger train to retrieve.
---	---

Returns

[PassengerTrain*](#) Pointer to the passenger train, or nullptr if the index is out of range.

4.8.3.11 getRouteName()

```
std::string Bus::getRouteName ( )
```

Gets the route name of the bus.

Returns

The route name of the bus.

`std::string` The route name of the bus.

The documentation for this class was generated from the following files:

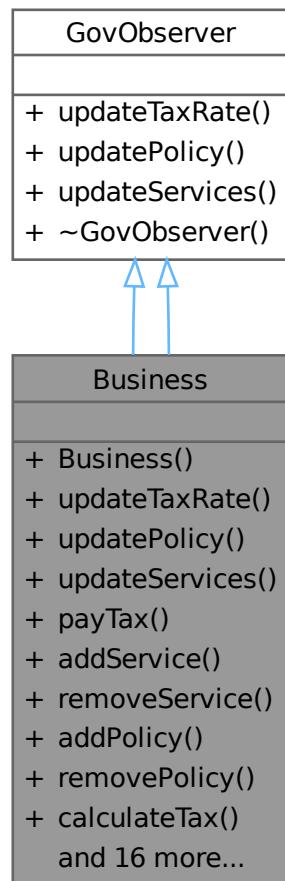
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Bus.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Bus.cpp](#)

4.9 Business Class Reference

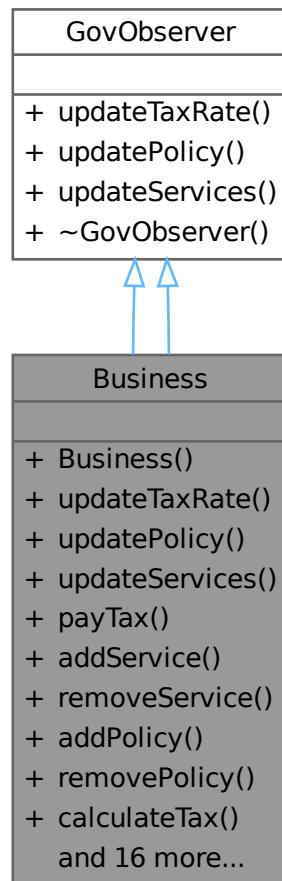
Represents a business that observes government policies and updates its state accordingly.

```
#include <Business.h>
```

Inheritance diagram for Business:



Collaboration diagram for Business:



Public Member Functions

- **Business** (double initialRevenue, double initialTaxRate)
*Constructs a new **Business** object.*
- void **updateTaxRate** (double rate) override
Updates the tax rate for the business.
- void **updatePolicy** (Policy policy) override
Updates the business policy.
- void **updateServices** (CityService service) override
Updates the services provided by the business.
- void **payTax** (double amount)
Processes the payment of tax by the business.
- void **addService** (const std::string &serviceName)
Adds a new service to the business.
- void **removeService** (const std::string &serviceName)
Removes a service from the business.
- void **addPolicy** (const Policy &policy)

- Adds a new policy to the business.
- void `removePolicy` (const `Policy` &policy)
 - Removes a policy from the business.*
- double `calculateTax` () const
 - Calculates the total tax to be paid based on the current tax rate and revenue.*
- void `printDetails` () const
 - Prints the details of the business.*
- double `payTaxes` (`TaxType` *taxType)
 - Processes the payment of taxes by the business based on the tax type.*
- void `setTaxCooldownPeriod` (int seconds)
 - Sets the cooldown period for tax payments.*
- `Business` (double initialRevenue, double initialTaxRate)
 - Constructs a new `Business` object.*
- void `updateTaxRate` (double rate) override
 - Updates the tax rate for the business.*
- void `updatePolicy` (`Policy` policy) override
 - Updates the business policy.*
- void `updateServices` (`CityService` service) override
 - Updates the services provided by the business.*
- void `payTax` (double amount)
 - Processes the payment of tax by the business.*
- void `addService` (const std::string &serviceName)
 - Adds a new service to the business.*
- void `removeService` (const std::string &serviceName)
 - Removes a service from the business.*
- void `addPolicy` (const `Policy` &policy)
 - Adds a new policy to the business.*
- void `removePolicy` (const `Policy` &policy)
 - Removes a policy from the business.*
- double `calculateTax` () const
 - Calculates the total tax to be paid based on the current tax rate and revenue.*
- void `printDetails` () const
 - Prints the details of the business.*
- double `payTaxes` (`TaxType` *taxType)
 - Processes the payment of taxes by the business based on the tax type.*
- void `setTaxCooldownPeriod` (int seconds)
 - Sets the cooldown period for tax payments.*

Public Member Functions inherited from `GovObserver`

- virtual ~`GovObserver` ()=default
 - Virtual destructor.*

4.9.1 Detailed Description

Represents a business that observes government policies and updates its state accordingly.

This class manages business operations such as updating tax rates, policies, and services, processing tax payments, and printing business details.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Business() [1/2]

```
Business::Business (
    double initialRevenue,
    double initialTaxRate )
```

Constructs a new [Business](#) object.

Parameters

<i>initialRevenue</i>	The initial revenue of the business.
<i>initialTaxRate</i>	The initial tax rate for the business.

4.9.2.2 Business() [2/2]

```
Business::Business (
    double initialRevenue,
    double initialTaxRate )
```

Constructs a new [Business](#) object.

Parameters

<i>initialRevenue</i>	The initial revenue of the business.
<i>initialTaxRate</i>	The initial tax rate for the business.

4.9.3 Member Function Documentation

4.9.3.1 addPolicy() [1/2]

```
void Business::addPolicy (
    const Policy & policy )
```

Adds a new policy to the business.

Parameters

<i>policy</i>	The policy to be added.
---------------	-------------------------

4.9.3.2 addPolicy() [2/2]

```
void Business::addPolicy (
    const Policy & policy )
```

Adds a new policy to the business.

Parameters

<i>policy</i>	The policy to be added.
---------------	-------------------------

4.9.3.3 addService() [1/2]

```
void Business::addService (
    const std::string & serviceName )
```

Adds a new service to the business.

Parameters

<i>serviceName</i>	The name of the service to be added.
--------------------	--------------------------------------

4.9.3.4 addService() [2/2]

```
void Business::addService (
    const std::string & serviceName )
```

Adds a new service to the business.

Parameters

<i>serviceName</i>	The name of the service to be added.
--------------------	--------------------------------------

4.9.3.5 calculateTax() [1/2]

```
double Business::calculateTax ( ) const
```

Calculates the total tax to be paid based on the current tax rate and revenue.

Returns

double The total tax to be paid.

4.9.3.6 calculateTax() [2/2]

```
double Business::calculateTax ( ) const
```

Calculates the total tax to be paid based on the current tax rate and revenue.

Returns

double The total tax to be paid.

4.9.3.7 payTax() [1/2]

```
void Business::payTax (
```

```
    double amount )
```

Processes the payment of tax by the business.

Parameters

<i>amount</i>	The amount of tax to be paid.
---------------	-------------------------------

4.9.3.8 payTax() [2/2]

```
void Business::payTax (
    double amount )
```

Processes the payment of tax by the business.

Parameters

<i>amount</i>	The amount of tax to be paid.
---------------	-------------------------------

4.9.3.9 payTaxes() [1/2]

```
double Business::payTaxes (
    TaxType * taxType )
```

Processes the payment of taxes by the business based on the tax type.

Parameters

<i>taxType</i>	Pointer to the tax type.
----------------	--------------------------

Returns

double The amount of taxes paid.

4.9.3.10 payTaxes() [2/2]

```
double Business::payTaxes (
    TaxType * taxType )
```

Processes the payment of taxes by the business based on the tax type.

Parameters

<i>taxType</i>	Pointer to the tax type.
----------------	--------------------------

Returns

double The amount of taxes paid.

4.9.3.11 removePolicy() [1/2]

```
void Business::removePolicy (
    const Policy & policy )
```

Removes a policy from the business.

Parameters

<i>policy</i>	The policy to be removed.
---------------	---------------------------

4.9.3.12 removePolicy() [2/2]

```
void Business::removePolicy (
    const Policy & policy )
```

Removes a policy from the business.

Parameters

<i>policy</i>	The policy to be removed.
---------------	---------------------------

4.9.3.13 removeService() [1/2]

```
void Business::removeService (
    const std::string & serviceName )
```

Removes a service from the business.

Parameters

<i>serviceName</i>	The name of the service to be removed.
--------------------	--

4.9.3.14 removeService() [2/2]

```
void Business::removeService (
    const std::string & serviceName )
```

Removes a service from the business.

Parameters

<i>serviceName</i>	The name of the service to be removed.
--------------------	--

4.9.3.15 setTaxCooldownPeriod() [1/2]

```
void Business::setTaxCooldownPeriod (
    int seconds )
```

Sets the cooldown period for tax payments.

Parameters

<i>seconds</i>	The cooldown period in seconds.
----------------	---------------------------------

4.9.3.16 setTaxCooldownPeriod() [2/2]

```
void Business::setTaxCooldownPeriod (
    int seconds )
```

Sets the cooldown period for tax payments.

Parameters

<i>seconds</i>	The cooldown period in seconds.
----------------	---------------------------------

4.9.3.17 updatePolicy() [1/2]

```
void Business::updatePolicy (
    Policy policy ) [override], [virtual]
```

Updates the business policy.

Parameters

<i>policy</i>	The new policy to be added.
---------------	-----------------------------

Implements [GovObserver](#).

4.9.3.18 updatePolicy() [2/2]

```
void Business::updatePolicy (
    Policy policy ) [override], [virtual]
```

Updates the business policy.

Parameters

<i>policy</i>	The new policy to be added.
---------------	-----------------------------

Implements [GovObserver](#).

4.9.3.19 updateServices() [1/2]

```
void Business::updateServices (
    CityService service )  [override], [virtual]
```

Updates the services provided by the business.

Parameters

<i>service</i>	The new service to be added.
----------------	------------------------------

Implements [GovObserver](#).

4.9.3.20 updateServices() [2/2]

```
void Business::updateServices (
    CityService service )  [override], [virtual]
```

Updates the services provided by the business.

Parameters

<i>service</i>	The new service to be added.
----------------	------------------------------

Implements [GovObserver](#).

4.9.3.21 updateTaxRate() [1/2]

```
void Business::updateTaxRate (
    double rate )  [override], [virtual]
```

Updates the tax rate for the business.

Parameters

<i>rate</i>	The new tax rate.
-------------	-------------------

Implements [GovObserver](#).

4.9.3.22 updateTaxRate() [2/2]

```
void Business::updateTaxRate (
    double rate )  [override], [virtual]
```

Updates the tax rate for the business.

Parameters

<i>rate</i>	The new tax rate.
-------------	-------------------

Implements [GovObserver](#).

The documentation for this class was generated from the following files:

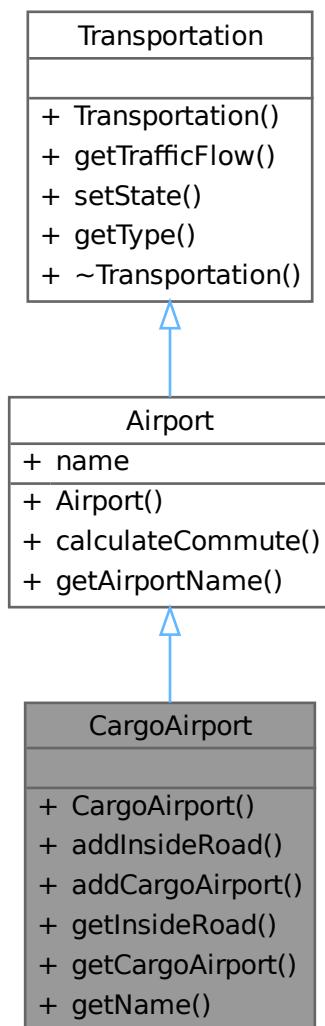
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Business.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/CargoAirport.cpp
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Business.cpp](#)

4.10 CargoAirport Class Reference

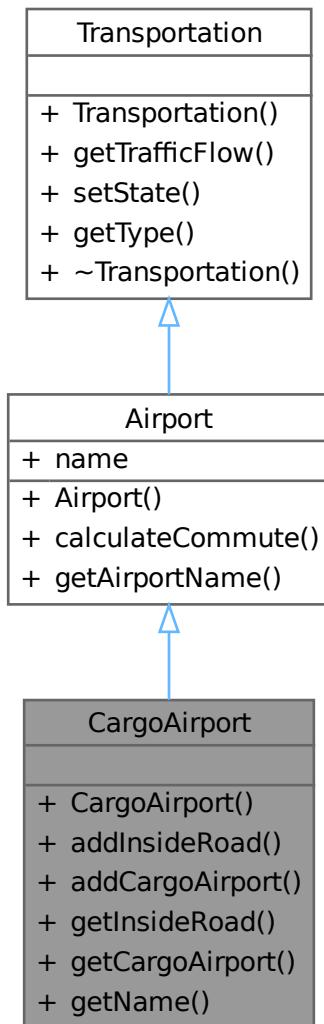
Represents a cargo airport.

```
#include <CargoAirport.h>
```

Inheritance diagram for CargoAirport:



Collaboration diagram for CargoAirport:



Public Member Functions

- `CargoAirport (char state, std::string name)`
Constructor for the `CargoAirport` class.
- `bool addInsideRoad (InsideRoad *insideRoad)`
Adds an inside road to the cargo airport.
- `bool addCargoAirport (CargoAirport *cargoAirport)`
Adds a cargo airport to the cargo airport.
- `InsideRoad * getInsideRoad (std::size_t index)`
Gets an inside road by index.
- `CargoAirport * getCargoAirport (std::size_t index)`
Gets a cargo airport by index.
- `std::string getName ()`
Gets the name of the cargo airport.

Public Member Functions inherited from [Airport](#)

- [Airport](#) (char state, std::string **name**, char type)
Constructs an [Airport](#) object.
- float [calculateCommute](#) ()
Calculates the commute time to the airport.
- std::string [getAirportName](#) ()
Gets the name of the airport.

Public Member Functions inherited from [Transportation](#)

- [Transportation](#) (char state, char type)
Constructor for the [Transportation](#) class.
- float [getTrafficFlow](#) ()
Gets the current traffic flow.
- bool [setState](#) (char state)
Sets the traffic flow state.
- char [getType](#) ()
Gets the type of transportation.
- [~Transportation](#) ()
Destructor for the [Transportation](#) class.

Additional Inherited Members

Public Attributes inherited from [Airport](#)

- std::string **name**
Name of the airport.

4.10.1 Detailed Description

Represents a cargo airport.

The [CargoAirport](#) class inherits from the [Airport](#) class and manages a collection of [InsideRoad](#) and [CargoAirport](#) objects.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 [CargoAirport\(\)](#)

```
CargoAirport::CargoAirport (
    char state,
    std::string name )
```

Constructor for the [CargoAirport](#) class.

Parameters

<i>state</i>	The state of the cargo airport.
<i>name</i>	The name of the cargo airport.

4.10.3 Member Function Documentation

4.10.3.1 addCargoAirport()

```
bool CargoAirport::addCargoAirport (
    CargoAirport * cargoAirport )
```

Adds a cargo airport to the cargo airport.

Parameters

<i>cargoAirport</i>	Pointer to the CargoAirport object to be added.
---------------------	---

Returns

True if the cargo airport was added successfully, false otherwise.

4.10.3.2 addInsideRoad()

```
bool CargoAirport::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an inside road to the cargo airport.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad object to be added.
-------------------	---

Returns

True if the inside road was added successfully, false otherwise.

4.10.3.3 getCargoAirport()

```
CargoAirport * CargoAirport::getCargoAirport (
    std::size_t index )
```

Gets a cargo airport by index.

Parameters

<i>index</i>	The index of the cargo airport to retrieve.
--------------	---

Returns

Pointer to the [CargoAirport](#) object at the specified index.

4.10.3.4 getInsideRoad()

```
InsideRoad * CargoAirport::getInsideRoad (
    std::size_t index )
```

Gets an inside road by index.

Parameters

<i>index</i>	The index of the inside road to retrieve.
--------------	---

Returns

Pointer to the [InsideRoad](#) object at the specified index.

4.10.3.5 getName()

```
std::string CargoAirport::getName ( )
```

Gets the name of the cargo airport.

Returns

The name of the cargo airport.

The documentation for this class was generated from the following file:

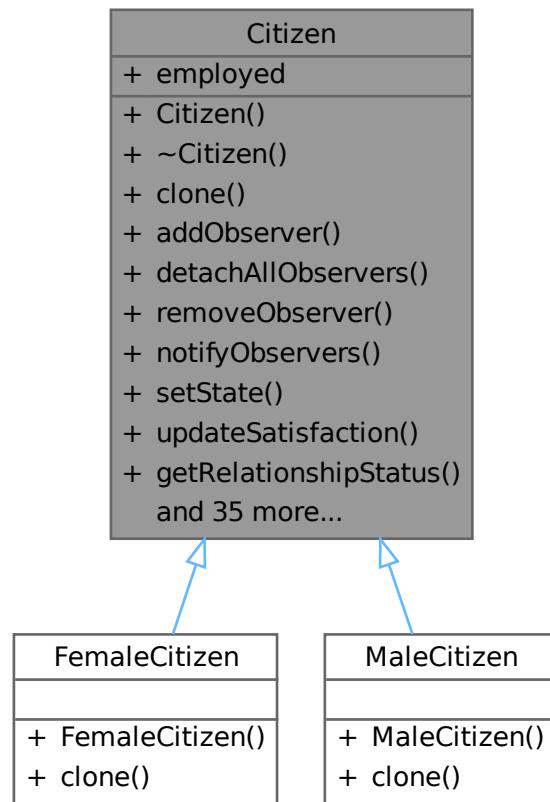
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CargoAirport.h](#)

4.11 Citizen Class Reference

Manages citizen attributes and behaviors.

```
#include <Citizen.h>
```

Inheritance diagram for Citizen:



Collaboration diagram for Citizen:

Citizen
+ employed
+ Citizen()
+ ~Citizen()
+ clone()
+ addObserver()
+ detachAllObservers()
+ removeObserver()
+ notifyObservers()
+ setState()
+ updateSatisfaction()
+ getRelationshipStatus()
and 35 more...

Public Member Functions

- **Citizen** (const std::string &name, int age)
Constructs a new [Citizen](#) object.
- virtual **~Citizen** ()
Destroys the [Citizen](#) object.
- virtual std::shared_ptr< [Citizen](#) > **clone** () const =0
Clones the citizen object.
- void **addObserver** ([CitizenObserver](#) *observer)
Adds an observer to the citizen.
- void **detachAllObservers** ()
Detaches all observers from the citizen.
- void **removeObserver** ([CitizenObserver](#) *observer)
Removes an observer from the citizen.
- void **notifyObservers** ()
Notifies all observers of changes to the citizen.
- void **setState** ([CitizenState](#) *newState)
Sets the state of the citizen.
- void **updateSatisfaction** ()
Updates the satisfaction of the citizen based on all strategies.
- std::string **getRelationshipStatus** () const
Gets the relationship status of the citizen.
- void **setRelationshipStatus** (const std::string &status)
Sets the relationship status of the citizen.
- int **getMarriageDuration** () const
Gets the duration of the citizen's marriage.

- void **resetMarriageDuration** ()
Resets the duration of the citizen's marriage to zero.
- void **incrementMarriageDuration** ()
Increments the duration of the citizen's marriage by one year.
- void **updateSatisfaction** (float adjustment)
Updates the satisfaction of the citizen by a specified adjustment.
- void **addChild** ()
Adds a child to the citizen's family.
- int **getAge** () const
Gets the age of the citizen.
- void **incrementAge** ()
Increments the age of the citizen by one year.
- void **addSatisfactionStrategy** (std::shared_ptr< SatisfactionStrategy > strategy)
Adds a satisfaction strategy to the citizen.
- void **removeSatisfactionStrategy** ()
Removes all satisfaction strategies from the citizen.
- void **setSatisfactionLevel** (double level)
Sets the satisfaction level of the citizen.
- void **depositMonthlyIncome** ()
Deposits the monthly income of the citizen into their bank balance.
- void **searchAndApplyForJob** (BuildingManager &manager, Building *building, std::string jobtitle)
Searches for and applies for a job for the citizen.
- std::string **getName** () const
Gets the name of the citizen.
- float **getSatisfactionLevel** () const
Gets the satisfaction level of the citizen.
- bool **isLeaving** () const
Checks if the citizen is leaving the city.
- void **setIncome** (std::shared_ptr< Income > income)
Sets the income of the citizen.
- void **checkAndUpdateState** ()
Checks and updates the state of the citizen based on satisfaction.
- void **setJobTitle** (const std::string &jobTitle)
Sets the job title of the citizen.
- std::string **getJob** () const
Gets the job title of the citizen.
- void **setEmployed** (bool status)
Sets the employment status of the citizen.
- bool **isEmployed** () const
Checks if the citizen is employed.
- float **getTaxRate** () const
Gets the tax rate for the citizen.
- void **setTaxRate** (float rate)
Sets the tax rate for the citizen.
- double **payTaxes** (TaxType *taxType)
Processes the payment of taxes by the citizen.
- double **getBankBalance** () const
Gets the bank balance of the citizen.
- void **setBankBalance** (double balance)
Sets the bank balance of the citizen.
- void **increaseBankBalance** (double amount)

- void **subtractBankBalance** (double amount)

Increases the bank balance of the citizen by a specified amount.
- void **setTaxCooldown** (bool status)

Decreases the bank balance of the citizen by a specified amount.
- bool **getTaxCooldown** () const

Sets the tax cooldown status for the citizen.
- bool **isOnCooldown** () const

Gets the tax cooldown status for the citizen.
- std::shared_ptr< Jobs > **getJobObj** ()

Checks if the citizen is on cooldown for tax payments.
- void **setJob** (std::shared_ptr< Jobs > job)

Gets the job object of the citizen.
- void **unsetJob** ()

Sets the job object of the citizen.
- void **unsetJob** ()

Unsets the job object of the citizen.

Public Attributes

- bool **employed** = false

Employment status of the citizen.

4.11.1 Detailed Description

Manages citizen attributes and behaviors.

This class manages citizen attributes and behaviors such as satisfaction, tax payments, job search, and relationship status.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Citizen()

```
Citizen::Citizen (
    const std::string & name,
    int age )
```

Constructs a new [Citizen](#) object.

Parameters

<i>name</i>	The name of the citizen.
<i>age</i>	The age of the citizen.

4.11.3 Member Function Documentation

4.11.3.1 addObserver()

```
void Citizen::addObserver (
    CitizenObserver * observer )
```

Adds an observer to the citizen.

Parameters

<i>observer</i>	Pointer to the observer to be added.
-----------------	--------------------------------------

4.11.3.2 addSatisfactionStrategy()

```
void Citizen::addSatisfactionStrategy (
    std::shared_ptr< SatisfactionStrategy > strategy )
```

Adds a satisfaction strategy to the citizen.

Parameters

<i>strategy</i>	Shared pointer to the satisfaction strategy.
-----------------	--

4.11.3.3 clone()

```
virtual std::shared_ptr< Citizen > Citizen::clone ( ) const [pure virtual]
```

Clones the citizen object.

Returns

`std::shared_ptr<Citizen>` A shared pointer to the cloned citizen.

Implemented in [FemaleCitizen](#), and [MaleCitizen](#).

4.11.3.4 getAge()

```
int Citizen::getAge ( ) const
```

Gets the age of the citizen.

Returns

`int` The age of the citizen.

4.11.3.5 getBankBalance()

```
double Citizen::getBankBalance ( ) const
```

Gets the bank balance of the citizen.

Returns

double The bank balance.

4.11.3.6 getJob()

```
std::string Citizen::getJob ( ) const [inline]
```

Gets the job title of the citizen.

Returns

std::string The job title.

4.11.3.7 getjobobj()

```
std::shared_ptr< Jobs > Citizen::getjobobj ( ) [inline]
```

Gets the job object of the citizen.

Returns

std::shared_ptr<Jobs> The job object.

4.11.3.8 getMarriageDuration()

```
int Citizen::getMarriageDuration ( ) const
```

Gets the duration of the citizen's marriage.

Returns

int The duration of the marriage in years.

4.11.3.9 getName()

```
std::string Citizen::getName ( ) const
```

Gets the name of the citizen.

Returns

std::string The name of the citizen.

4.11.3.10 **getRelationshipStatus()**

```
std::string Citizen::getRelationshipStatus ( ) const
```

Gets the relationship status of the citizen.

Returns

`std::string` The relationship status.

4.11.3.11 **getSatisfactionLevel()**

```
float Citizen::getSatisfactionLevel ( ) const
```

Gets the satisfaction level of the citizen.

Returns

`float` The satisfaction level.

4.11.3.12 **getTaxCooldown()**

```
bool Citizen::getTaxCooldown ( ) const
```

Gets the tax cooldown status for the citizen.

Returns

`true` if the citizen is on cooldown.

`false` if the citizen is not on cooldown.

4.11.3.13 **getTaxRate()**

```
float Citizen::getTaxRate ( ) const
```

Gets the tax rate for the citizen.

Returns

`float` The tax rate.

4.11.3.14 **increaseBankBalance()**

```
void Citizen::increaseBankBalance ( double amount )
```

Increases the bank balance of the citizen by a specified amount.

Parameters

<i>amount</i>	The amount to increase the bank balance by.
---------------	---

4.11.3.15 isEmployed()

```
bool Citizen::isEmployed () const [inline]
```

Checks if the citizen is employed.

Returns

- true if the citizen is employed.
- false if the citizen is not employed.

4.11.3.16 isLeaving()

```
bool Citizen::isLeaving () const
```

Checks if the citizen is leaving the city.

Returns

- true if the citizen is leaving.
- false if the citizen is not leaving.

4.11.3.17 isOnCooldown()

```
bool Citizen::isOnCooldown () const
```

Checks if the citizen is on cooldown for tax payments.

Returns

- true if the citizen is on cooldown.
- false if the citizen is not on cooldown.

4.11.3.18 payTaxes()

```
double Citizen::payTaxes ( TaxType * taxType )
```

Processes the payment of taxes by the citizen.

Parameters

<i>taxType</i>	Pointer to the tax type.
----------------	--------------------------

Returns

double The amount of taxes paid.

4.11.3.19 removeObserver()

```
void Citizen::removeObserver (
    CitizenObserver * observer )
```

Removes an observer from the citizen.

Parameters

<i>observer</i>	Pointer to the observer to be removed.
-----------------	--

4.11.3.20 searchAndApplyForJob()

```
void Citizen::searchAndApplyForJob (
    BuildingManager & manager,
    Building * building,
    std::string jobtitle )
```

Searches for and applies for a job for the citizen.

Parameters

<i>manager</i>	Reference to the BuildingManager .
<i>building</i>	Pointer to the building where the job is located.
<i>jobtitle</i>	The title of the job to apply for.

4.11.3.21 setBankBalance()

```
void Citizen::setBankBalance (
    double balance )
```

Sets the bank balance of the citizen.

Parameters

<i>balance</i>	The new bank balance.
----------------	-----------------------

4.11.3.22 setEmployed()

```
void Citizen::setEmployed (
    bool status ) [inline]
```

Sets the employment status of the citizen.

Parameters

<i>status</i>	The new employment status.
---------------	----------------------------

4.11.3.23 setIncome()

```
void Citizen::setIncome (
    std::shared_ptr< Income > inc )
```

Sets the income of the citizen.

Parameters

<i>income</i>	Shared pointer to the income.
<i>inc</i>	Shared pointer to the income.

4.11.3.24 setJob()

```
void Citizen::setJob (
    std::shared_ptr< Jobs > job ) [inline]
```

Sets the job object of the citizen.

Parameters

<i>job</i>	Shared pointer to the job.
------------	----------------------------

4.11.3.25 setJobTitle()

```
void Citizen::setJobTitle (
    const std::string & jobTitle ) [inline]
```

Sets the job title of the citizen.

Parameters

<i>jobTitle</i>	The new job title.
-----------------	--------------------

4.11.3.26 setRelationshipStatus()

```
void Citizen::setRelationshipStatus (
    const std::string & status )
```

Sets the relationship status of the citizen.

Parameters

<i>status</i>	The new relationship status.
---------------	------------------------------

4.11.3.27 setSatisfactionLevel()

```
void Citizen::setSatisfactionLevel (
    double level ) [inline]
```

Sets the satisfaction level of the citizen.

Parameters

<i>level</i>	The new satisfaction level.
--------------	-----------------------------

4.11.3.28 setState()

```
void Citizen::setState (
    CitizenState * newState )
```

Sets the state of the citizen.

Parameters

<i>newState</i>	Pointer to the new state.
-----------------	---------------------------

4.11.3.29 setTaxCooldown()

```
void Citizen::setTaxCooldown (
    bool status )
```

Sets the tax cooldown status for the citizen.

Parameters

<i>status</i>	The new tax cooldown status.
---------------	------------------------------

4.11.3.30 setTaxRate()

```
void Citizen::setTaxRate (
    float rate )
```

Sets the tax rate for the citizen.

Parameters

<i>rate</i>	The new tax rate.
-------------	-------------------

4.11.3.31 subtractBankBalance()

```
void Citizen::subtractBankBalance (
    double amount )
```

Decreases the bank balance of the citizen by a specified amount.

Parameters

<i>amount</i>	The amount to decrease the bank balance by.
---------------	---

4.11.3.32 updateSatisfaction()

```
void Citizen::updateSatisfaction (
    float adjustment )
```

Updates the satisfaction of the citizen by a specified adjustment.

Parameters

<i>adjustment</i>	The adjustment to the satisfaction.
-------------------	-------------------------------------

The documentation for this class was generated from the following files:

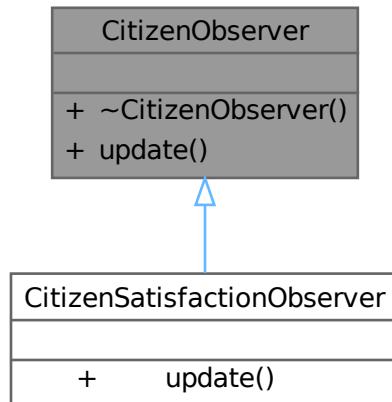
- /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.cpp

4.12 CitizenObserver Class Reference

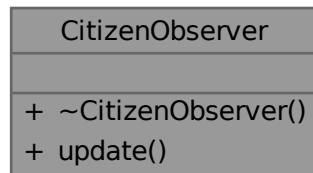
Interface for observers of [Citizen](#) objects.

```
#include <CitizenObserver.h>
```

Inheritance diagram for CitizenObserver:



Collaboration diagram for CitizenObserver:



Public Member Functions

- virtual ~**CitizenObserver** ()=default
Virtual destructor for the [CitizenObserver](#) class.
- virtual void [update](#) ([Citizen](#) *citizen)=0
Abstract update method to be implemented by concrete observers.

4.12.1 Detailed Description

Interface for observers of [Citizen](#) objects.

This class defines the interface for observers that want to be notified of changes to [Citizen](#) objects.

4.12.2 Member Function Documentation

4.12.2.1 update()

```
virtual void CitizenObserver::update (
    Citizen * citizen ) [pure virtual]
```

Abstract update method to be implemented by concrete observers.

Parameters

<i>citizen</i>	Pointer to the Citizen object that has changed.
----------------	---

Implemented in [CitizenSatisfactionObserver](#).

The documentation for this class was generated from the following file:

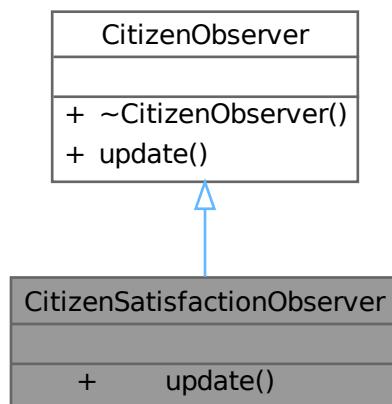
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CitizenObserver.h](#)

4.13 CitizenSatisfactionObserver Class Reference

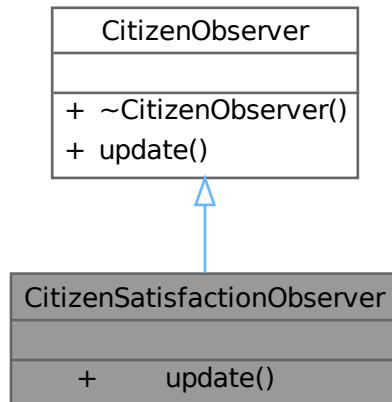
Observes changes in citizen satisfaction and updates their state accordingly.

```
#include <CitizenSatisfactionObserver.h>
```

Inheritance diagram for CitizenSatisfactionObserver:



Collaboration diagram for CitizenSatisfactionObserver:



Public Member Functions

- void [update \(Citizen *citizen\)](#) override
Updates the satisfaction and state of the observed citizen.

Public Member Functions inherited from [CitizenObserver](#)

- virtual ~[CitizenObserver \(\)](#)=default
Virtual destructor for the [CitizenObserver](#) class.

4.13.1 Detailed Description

Observes changes in citizen satisfaction and updates their state accordingly.

This class implements the [CitizenObserver](#) interface and observes changes in citizen satisfaction, updating their state based on the new satisfaction level.

4.13.2 Member Function Documentation

4.13.2.1 [update\(\)](#)

```
void CitizenSatisfactionObserver::update (
    Citizen * citizen ) [inline], [override], [virtual]
```

Updates the satisfaction and state of the observed citizen.

Parameters

<code>citizen</code>	Pointer to the Citizen object that has changed.
----------------------	---

Implements [CitizenObserver](#).

The documentation for this class was generated from the following file:

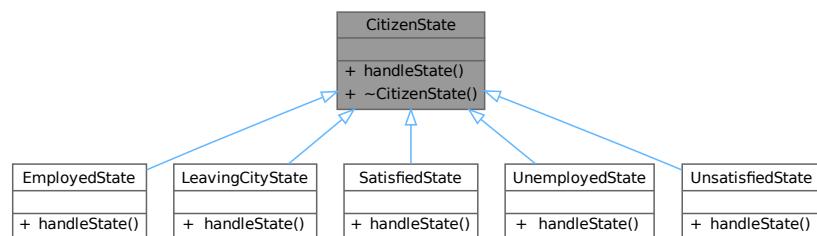
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CitizenSatisfactionObserver.h](#)

4.14 CitizenState Class Reference

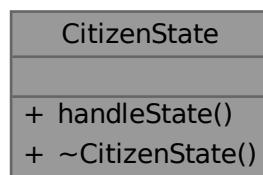
Abstract class representing the state of a citizen.

```
#include <CitizenState.h>
```

Inheritance diagram for CitizenState:



Collaboration diagram for CitizenState:



Public Member Functions

- virtual void `handleState (Citizen &citizen) const =0`
Handles the state of a citizen.
- virtual `~CitizenState ()=default`
Virtual destructor for the [CitizenState](#) class.

4.14.1 Detailed Description

Abstract class representing the state of a citizen.

4.14.2 Member Function Documentation

4.14.2.1 handleState()

```
void CitizenState::handleState (
    Citizen & citizen ) const [pure virtual]
```

Handles the state of a citizen.

This function is responsible for managing the state of the given citizen.

Parameters

<i>citizen</i>	The citizen whose state is to be handled.
----------------	---

Implemented in [EmployedState](#), [LeavingCityState](#), [SatisfiedState](#), [UnemployedState](#), and [UnsatisfiedState](#).

The documentation for this class was generated from the following files:

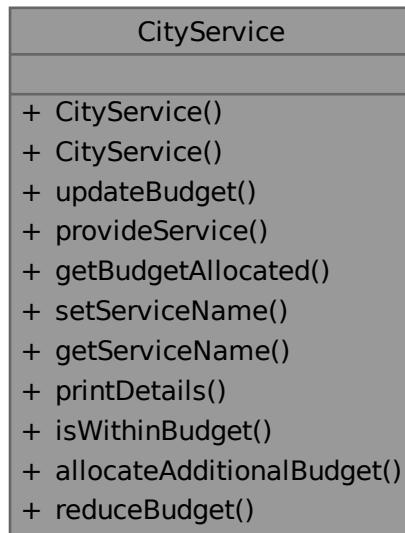
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CitizenState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CitizenState.cpp](#)

4.15 CityService Class Reference

Class representing a city service.

```
#include <CityService.h>
```

Collaboration diagram for CityService:



Public Member Functions

- **CityService ()**
Default constructor for [CityService](#).
- **CityService (const std::string &name, double initialBudget)**
Constructor for [CityService](#).
- **void updateBudget (double amount)**
Updates the budget allocated to the city service.
- **void provideService ()**
Provides the city service.
- **double getBudgetAllocated () const**
Gets the budget allocated to the city service.
- **void setServiceName (const std::string &name)**
Sets the name of the city service.
- **std::string getServiceName () const**
Gets the name of the city service.
- **void printDetails () const**
Prints the details of the city service.
- **bool isWithinBudget (double amount) const**
Checks if the service is within budget.
- **void allocateAdditionalBudget (double amount, double limit)**
Allocates additional budget to the city service.
- **void reduceBudget (double amount)**
Reduces the budget allocated to the city service.

4.15.1 Detailed Description

Class representing a city service.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 CityService() [1/2]

```
CityService::CityService ( )
```

Default constructor for [CityService](#).

Initializes the city service with default values.

4.15.2.2 CityService() [2/2]

```
CityService::CityService (
    const std::string & name,
    double initialBudget )
```

Constructor for [CityService](#).

Initializes the city service with a name and initial budget.

Parameters

<i>name</i>	The name of the city service.
<i>initialBudget</i>	The initial budget allocated to the city service.

4.15.3 Member Function Documentation

4.15.3.1 allocateAdditionalBudget()

```
void CityService::allocateAdditionalBudget (
    double amount,
    double limit )
```

Allocates additional budget to the city service.

Parameters

<i>amount</i>	The amount to add to the budget.
<i>limit</i>	The maximum limit for the budget.

4.15.3.2 `getBudgetAllocated()`

```
double CityService::getBudgetAllocated ( ) const
```

Gets the budget allocated to the city service.

Returns

The current budget allocated.

4.15.3.3 `getServiceName()`

```
std::string CityService::getServiceName ( ) const
```

Gets the name of the city service.

Returns

The name of the city service.

4.15.3.4 `isWithinBudget()`

```
bool CityService::isWithinBudget ( double amount ) const
```

Checks if the service is within budget.

Parameters

<i>amount</i>	The amount to check against the budget.
---------------	---

Returns

True if the service is within budget, false otherwise.

4.15.3.5 `provideService()`

```
void CityService::provideService ( )
```

Provides the city service.

Prints a message indicating the service being provided.

4.15.3.6 `reduceBudget()`

```
void CityService::reduceBudget ( double amount )
```

Reduces the budget allocated to the city service.

Parameters

<i>amount</i>	The amount to reduce from the budget.
---------------	---------------------------------------

4.15.3.7 setServiceName()

```
void CityService::setServiceName (
    const std::string & name )
```

Sets the name of the city service.

Parameters

<i>name</i>	The new name of the city service.
-------------	-----------------------------------

4.15.3.8 updateBudget()

```
void CityService::updateBudget (
    double amount )
```

Updates the budget allocated to the city service.

Adds the specified amount to the current budget.

Parameters

<i>amount</i>	The amount to add to the current budget.
---------------	--

Adds the specified amount to the current budget and prints the new budget.

Parameters

<i>amount</i>	The amount to add to the current budget.
---------------	--

The documentation for this class was generated from the following files:

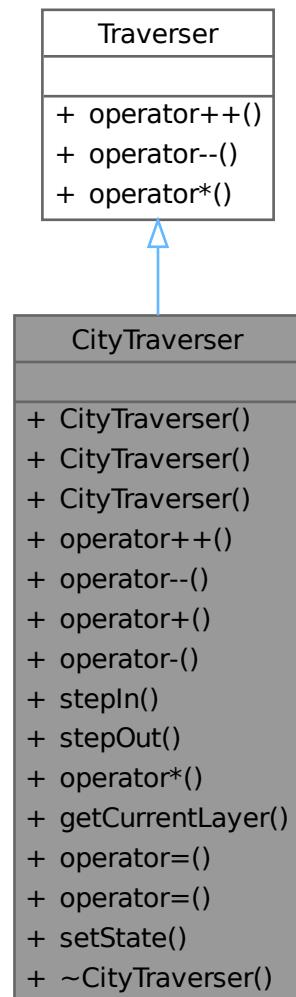
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CityService.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CityService.cpp](#)

4.16 CityTraverser Class Reference

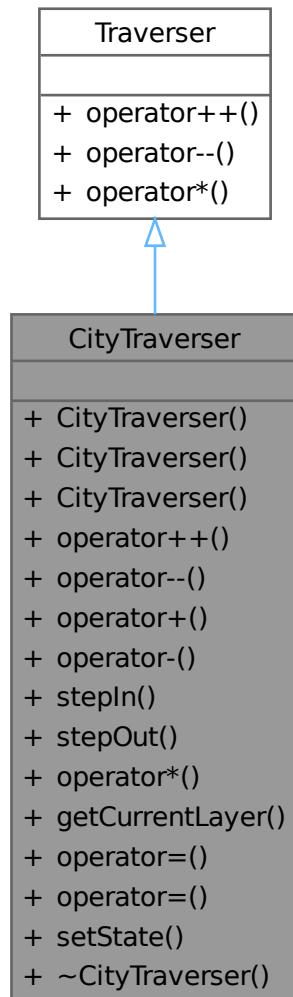
A concrete iterator for traversing transportation elements in a city.

```
#include <CityTraverser.h>
```

Inheritance diagram for CityTraverser:



Collaboration diagram for CityTraverser:



Public Member Functions

- [`CityTraverser \(\)`](#)
Default constructor for `CityTraverser`.
- [`CityTraverser \(Transportation *\)`](#)
Constructor for `CityTraverser` with a starting transportation element.
- [`CityTraverser \(CityTraverser *\)`](#)
Copy constructor for `CityTraverser`.
- [`bool operator++ \(\)`](#)
Move to the next transportation element.
- [`bool operator-- \(\)`](#)
Move to the previous transportation element.
- [`bool operator+ \(\)`](#)
Move forward in the current layer.

- bool [operator- \(\)](#)
Move backward in the current layer.
- bool [stepIn \(\)](#)
Step into the current layer of transportation elements.
- bool [stepOut \(\)](#)
Step out to the current layer of transportation elements.
- [Transportation * operator* \(\)](#)
Dereference operator to access the current transportation element.
- [Transportation * getCurrentLayer \(\)](#)
Get the current layer of transportation elements.
- bool [operator= \(Transportation *t\)](#)
Assignment operator to set the current transportation layer.
- bool [operator= \(CityTraverser *t\)](#)
Assignment operator to copy from another [CityTraverser](#) object.
- bool [setState \(\)](#)
Set the current state of traversal.
- [~CityTraverser \(\)](#)
Destructor for [CityTraverser](#).

4.16.1 Detailed Description

A concrete iterator for traversing transportation elements in a city.

The [CityTraverser](#) class provides functionality to iterate over various transportation elements within a city. It maintains the current state of traversal and allows moving forward and backward through the elements, as well as stepping into and out of layers.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 [CityTraverser\(\)](#) [1/3]

```
CityTraverser::CityTraverser ( )
```

Default constructor for [CityTraverser](#).

Initializes the [CityTraverser](#) with default values.

4.16.2.2 [CityTraverser\(\)](#) [2/3]

```
CityTraverser::CityTraverser (   
    Transportation * t )
```

Constructor for [CityTraverser](#) with a starting transportation element.

Constructor for [CityTraverser](#) with a [Transportation](#) object.

Parameters

<i>t</i>	Pointer to the starting transportation element.
----------	---

Initializes the [CityTraverser](#) with a given [Transportation](#) object.

Parameters

<code>t</code>	Pointer to a Transportation object.
----------------	---

4.16.2.3 [CityTraverser\(\)](#) [3/3]

```
CityTraverser::CityTraverser (
    CityTraverser * t )
```

Copy constructor for [CityTraverser](#).

Parameters

<code>t</code>	Pointer to another CityTraverser object to copy from.
----------------	---

Initializes the [CityTraverser](#) with another [CityTraverser](#) object.

Parameters

<code>t</code>	Pointer to a CityTraverser object.
----------------	--

4.16.3 Member Function Documentation

4.16.3.1 [getCurrentLayer\(\)](#)

```
Transportation * CityTraverser::getCurrentLayer ( )
```

Get the current layer of transportation elements.

Get the current layer.

Returns

Pointer to the current layer of transportation elements.

Pointer to the current [Transportation](#) layer.

4.16.3.2 [operator*\(\)](#)

```
Transportation * CityTraverser::operator* ( ) [virtual]
```

Dereference operator to access the current transportation element.

Dereference operator for [CityTraverser](#).

Returns

Pointer to the current transportation element.

Pointer to the current [Transportation](#) element.

Implements [Traverser](#).

4.16.3.3 operator+()

```
bool CityTraverser::operator+ ( )
```

Move forward in the current layer.

Move to the next list in the current layer.

Returns

True if successful, false otherwise.

4.16.3.4 operator++()

```
bool CityTraverser::operator++ ( ) [virtual]
```

Move to the next transportation element.

Increment operator for [CityTraverser](#).

Returns

True if successful, false otherwise.

Moves to the next element in the current layer.

Returns

True if successful, false otherwise.

Implements [Traverser](#).

4.16.3.5 operator-()

```
bool CityTraverser::operator- ( )
```

Move backward in the current layer.

Move to the previous list in the current layer.

Returns

True if successful, false otherwise.

4.16.3.6 operator--()

```
bool CityTraverser::operator-- ( ) [virtual]
```

Move to the previous transportation element.

Decrement operator for [CityTraverser](#).

Returns

True if successful, false otherwise.

Moves to the previous element in the current layer.

Returns

True if successful, false otherwise.

Implements [Traverser](#).

4.16.3.7 operator=() [1/2]

```
bool CityTraverser::operator= (
    CityTraverser * t )
```

Assignment operator to copy from another [CityTraverser](#) object.

Assignment operator for [CityTraverser](#) with another [CityTraverser](#) object.

Parameters

<i>t</i>	Pointer to the CityTraverser object to copy from.
----------	---

Returns

True if successful, false otherwise.

Parameters

<i>t</i>	Pointer to a CityTraverser object.
----------	--

Returns

True if successful, false otherwise.

4.16.3.8 operator=() [2/2]

```
bool CityTraverser::operator= (
    Transportation * t )
```

Assignment operator to set the current transportation layer.

Assignment operator for [CityTraverser](#) with a [Transportation](#) object.

Parameters

<i>t</i>	Pointer to the transportation element to set.
----------	---

Returns

True if successful, false otherwise.

Parameters

<i>t</i>	Pointer to a Transportation object.
----------	---

Returns

True if successful, false otherwise.

4.16.3.9 `setState()`

```
bool CityTraverser::setState ( )
```

Set the current state of traversal.

Set the state based on the current data type.

Returns

True if successful, false otherwise.

4.16.3.10 `stepIn()`

```
bool CityTraverser::stepIn ( )
```

Step into the current layer of transportation elements.

Step into the current element.

Returns

True if successful, false otherwise.

4.16.3.11 stepOut()

```
bool CityTraverser::stepOut ( )
```

Step out to the current layer of transportation elements.

Step out to the previous layer.

Returns

True if successful, false otherwise.

The documentation for this class was generated from the following files:

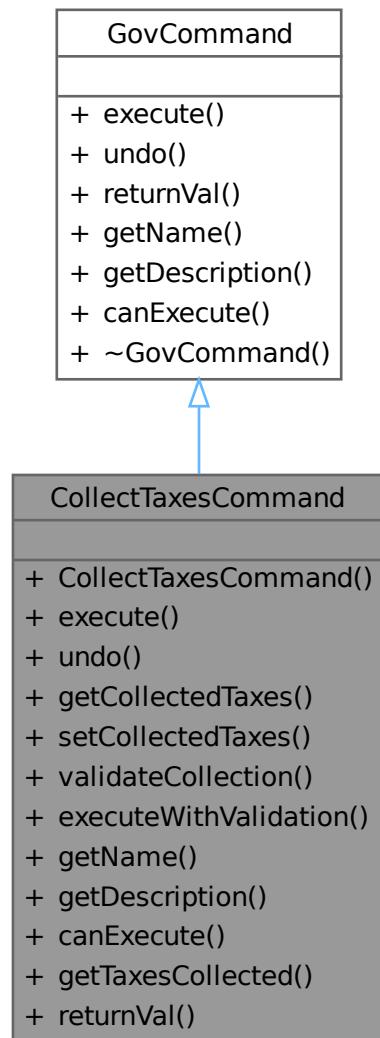
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CityTraverser.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CityTraverser.cpp](#)

4.17 CollectTaxesCommand Class Reference

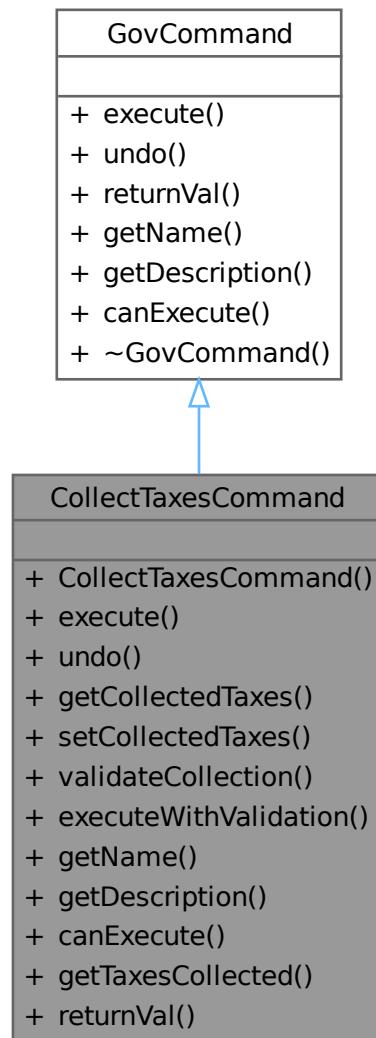
[CollectTaxesCommand](#) class.

```
#include <CollectTaxesCommand.h>
```

Inheritance diagram for CollectTaxesCommand:



Collaboration diagram for CollectTaxesCommand:



Public Member Functions

- `CollectTaxesCommand (Government *gov, std::shared_ptr< Building > building, TaxType *taxType)`
Constructor for CollectTaxesCommand.
- void `execute ()` override
Executes the tax collection command.
- void `undo ()` override
Undoes the tax collection command.
- double `getCollectedTaxes () const`
Gets the amount of taxes collected.
- void `setCollectedTaxes (double amount)`
Sets the amount of taxes collected.

- bool `validateCollection () const`
Validates the tax collection.
- void `executeWithValidation ()`
Executes the tax collection command with validation.
- std::string `getName () const override`
Gets the name of the command.
- std::string `getDescription () const override`
Gets the description of the command.
- bool `canExecute () const override`
Checks if the command can be executed.
- double `getTaxesCollected ()`
Gets the amount of taxes collected.
- double `returnVal ()`
Returns the amount of taxes collected.

Public Member Functions inherited from GovCommand

- virtual ~`GovCommand ()=default`
Virtual destructor.

4.17.1 Detailed Description

`CollectTaxesCommand` class.

This class represents a command to collect taxes from the government.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 CollectTaxesCommand()

```
CollectTaxesCommand::CollectTaxesCommand (
    Government * gov,
    std::shared_ptr< Building > building,
    TaxType * taxType )
```

Constructor for `CollectTaxesCommand`.

Initializes the command with the government object.

Parameters

<code>gov</code>	Pointer to the <code>Government</code> object.
<code>building</code>	Shared pointer to the <code>Building</code> object.
<code>taxType</code>	Pointer to the <code>TaxType</code> object.

4.17.3 Member Function Documentation

4.17.3.1 canExecute()

```
bool CollectTaxesCommand::canExecute ( ) const [override], [virtual]
```

Checks if the command can be executed.

Returns

True if the command can be executed, false otherwise.

Implements [GovCommand](#).

4.17.3.2 execute()

```
void CollectTaxesCommand::execute ( ) [override], [virtual]
```

Executes the tax collection command.

Collects taxes from the government and stores the collected amount.

Implements [GovCommand](#).

4.17.3.3 executeWithValidation()

```
void CollectTaxesCommand::executeWithValidation ( )
```

Executes the tax collection command with validation.

Throws an exception if the collection is invalid.

Exceptions

<code>std::runtime_error</code>	if the tax rate is invalid.
---------------------------------	-----------------------------

4.17.3.4 getCollectedTaxes()

```
double CollectTaxesCommand::getCollectedTaxes ( ) const
```

Gets the amount of taxes collected.

Returns

The amount of taxes collected.

4.17.3.5 getDescription()

```
std::string CollectTaxesCommand::getDescription () const [override], [virtual]
```

Gets the description of the command.

Returns

The description of the command.

Implements [GovCommand](#).

4.17.3.6 getName()

```
std::string CollectTaxesCommand::getName () const [override], [virtual]
```

Gets the name of the command.

Returns

The name of the command.

Implements [GovCommand](#).

4.17.3.7 getTaxesCollected()

```
double CollectTaxesCommand::getTaxesCollected ()
```

Gets the amount of taxes collected.

Returns

The amount of taxes collected.

4.17.3.8 returnVal()

```
double CollectTaxesCommand::returnVal () [virtual]
```

Returns the amount of taxes collected.

Returns

The amount of taxes collected.

Implements [GovCommand](#).

4.17.3.9 setCollectedTaxes()

```
void CollectTaxesCommand::setCollectedTaxes (
    double amount )
```

Sets the amount of taxes collected.

Parameters

<i>amount</i>	The amount of taxes collected.
---------------	--------------------------------

Exceptions

<i>std::invalid_argument</i>	if the amount is negative.
------------------------------	----------------------------

4.17.3.10 undo()

```
void CollectTaxesCommand::undo () [override], [virtual]
```

Undoes the tax collection command.

Refunds the collected taxes to the government.

Implements [GovCommand](#).

4.17.3.11 validateCollection()

```
bool CollectTaxesCommand::validateCollection () const
```

Validates the tax collection.

Ensures the government has a valid tax rate.

Returns

True if the tax rate is valid, false otherwise.

The documentation for this class was generated from the following files:

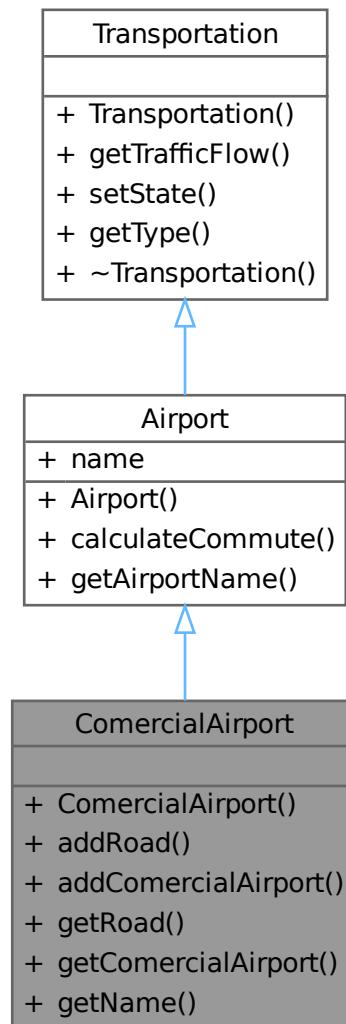
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CollectTaxesCommand.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CollectTaxesCommand.cpp](#)

4.18 ComercialAirport Class Reference

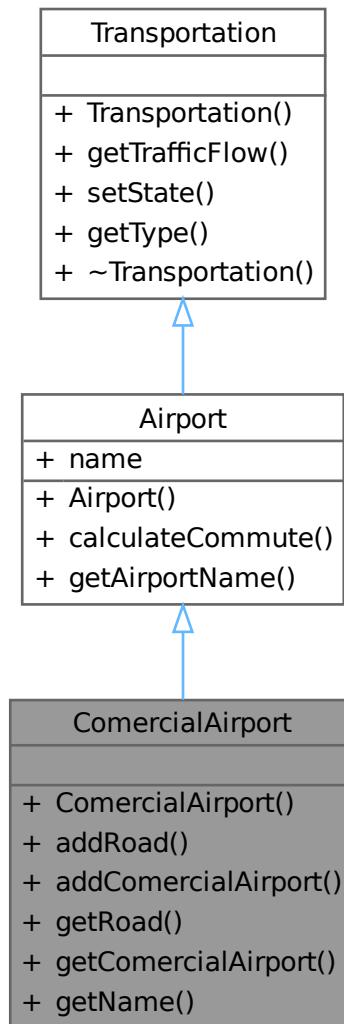
Represents a commercial airport.

```
#include <ComercialAirport.h>
```

Inheritance diagram for ComercialAirport:



Collaboration diagram for ComercialAirport:



Public Member Functions

- `ComercialAirport (char state, std::string name)`
Constructor for `ComercialAirport`.
- `bool addRoad (InsideRoad *road)`
Adds a road to the airport.
- `bool addComercialAirport (ComercialAirport *comercialAirport)`
Adds a commercial airport to the list.
- `InsideRoad * getRoad (std::size_t index)`
Gets a road by index.
- `ComercialAirport * getComercialAirport (std::size_t index)`
Gets a commercial airport by index.
- `std::string getName ()`
Gets the name of the airport.

Public Member Functions inherited from [Airport](#)

- [Airport](#) (char state, std::string **name**, char type)
Constructs an [Airport](#) object.
- float [calculateCommute](#) ()
Calculates the commute time to the airport.
- std::string [getAirportName](#) ()
Gets the name of the airport.

Public Member Functions inherited from [Transportation](#)

- [Transportation](#) (char state, char type)
Constructor for the [Transportation](#) class.
- float [getTrafficFlow](#) ()
Gets the current traffic flow.
- bool [setState](#) (char state)
Sets the traffic flow state.
- char [getType](#) ()
Gets the type of transportation.
- [~Transportation](#) ()
Destructor for the [Transportation](#) class.

Additional Inherited Members

Public Attributes inherited from [Airport](#)

- std::string **name**
Name of the airport.

4.18.1 Detailed Description

Represents a commercial airport.

The [ComercialAirport](#) class inherits from the [Airport](#) class and provides additional functionality specific to commercial airports, such as connecting roads and other commercial airports.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 ComercialAirport()

```
ComercialAirport::ComercialAirport (
    char state,
    std::string name )
```

Constructor for [ComercialAirport](#).

Parameters

<i>state</i>	The state of the airport.
<i>name</i>	The name of the airport.

Initializes the commercial airport with a state and name.

Parameters

<i>state</i>	The state of the airport.
<i>name</i>	The name of the airport.

4.18.3 Member Function Documentation

4.18.3.1 addComercialAirport()

```
bool ComercialAirport::addComercialAirport (
    ComercialAirport * comercialAirport )
```

Adds a commercial airport to the list.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport object to be added.
-------------------------	---

Returns

true if the commercial airport was added successfully, false otherwise.

Adds a commercial airport to the list of commercial airports if it is not already present.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport object.
-------------------------	---

Returns

True if the commercial airport was added, false otherwise.

4.18.3.2 addRoad()

```
bool ComercialAirport::addRoad (
    InsideRoad * road )
```

Adds a road to the airport.

Adds a road to the commercial airport.

Parameters

<i>road</i>	Pointer to the InsideRoad object to be added.
-------------	---

Returns

true if the road was added successfully, false otherwise.

Adds a road to the list of roads if it is not already present.

Parameters

<i>road</i>	Pointer to the InsideRoad object.
-------------	---

Returns

True if the road was added, false otherwise.

4.18.3.3 [getComercialAirport\(\)](#)

```
ComercialAirport * ComercialAirport::getComercialAirport ( std::size_t index )
```

Gets a commercial airport by index.

Parameters

<i>index</i>	The index of the commercial airport to retrieve.
--------------	--

Returns

Pointer to the [ComercialAirport](#) object at the specified index.

Retrieves a commercial airport from the list of commercial airports by its index.

Parameters

<i>index</i>	The index of the commercial airport.
--------------	--------------------------------------

Returns

Pointer to the [ComercialAirport](#) object if found, nullptr otherwise.

4.18.3.4 [getName\(\)](#)

```
std::string ComercialAirport::getName ( )
```

Gets the name of the airport.

Gets the name of the commercial airport.

Returns

The name of the airport.
The name of the commercial airport.

4.18.3.5 `getRoad()`

```
InsideRoad * ComercialAirport::getRoad (
    std::size_t index )
```

Gets a road by index.

Parameters

<i>index</i>	The index of the road to retrieve.
--------------	------------------------------------

Returns

Pointer to the `InsideRoad` object at the specified index.

Retrieves a road from the list of roads by its index.

Parameters

<i>index</i>	The index of the road.
--------------	------------------------

Returns

Pointer to the `InsideRoad` object if found, `nullptr` otherwise.

The documentation for this class was generated from the following files:

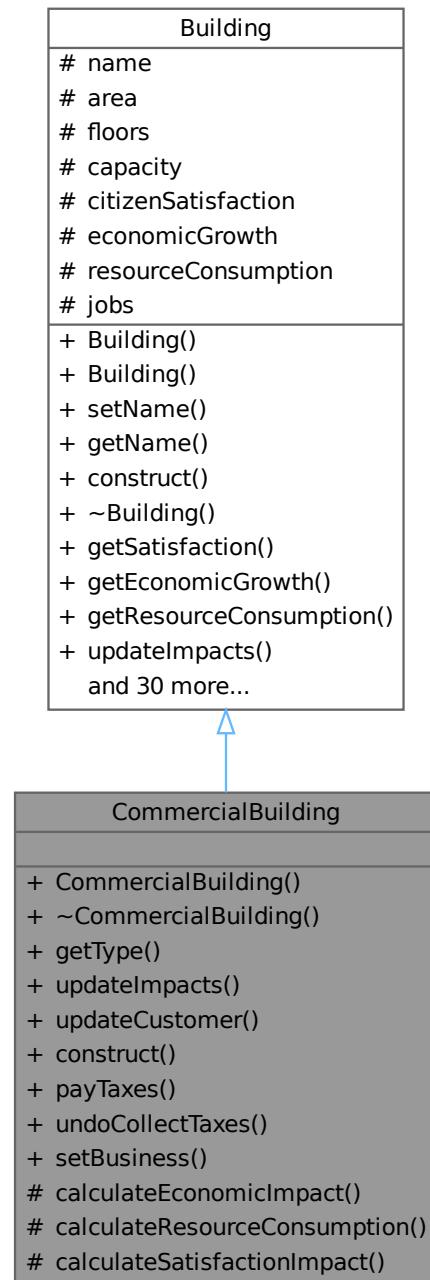
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ComercialAirport.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ComercialAirport.cpp](#)

4.19 CommercialBuilding Class Reference

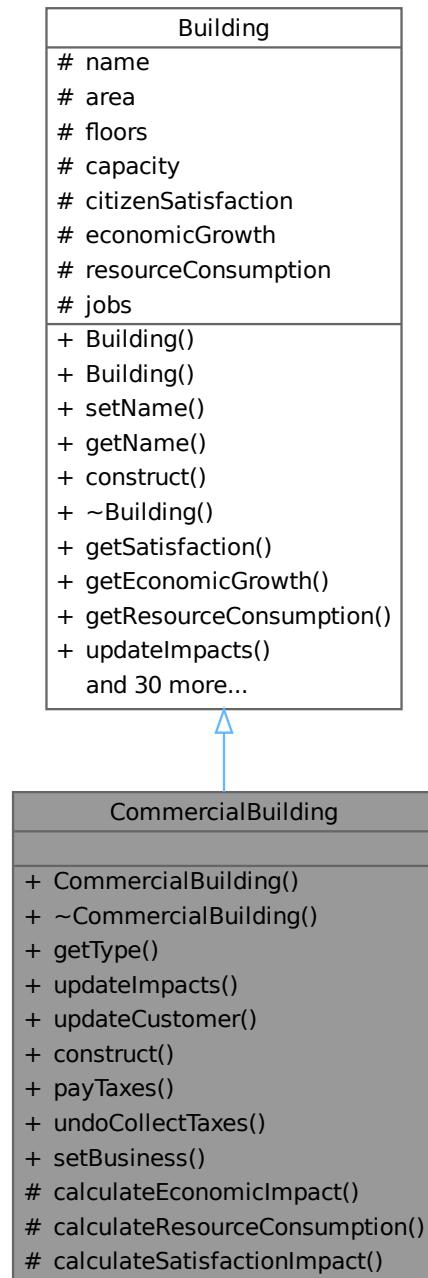
Represents a commercial building.

```
#include <CommercialBuilding.h>
```

Inheritance diagram for CommercialBuilding:



Collaboration diagram for CommercialBuilding:



Public Member Functions

- `CommercialBuilding` (const std::string &`name`, float `area`, int `floors`, int `capacity`, float `citizenSatisfaction`, float `economicGrowth`, float `resourceConsumption`, int `businessUnits`, float `customerTraffic`)
Constructor for CommercialBuilding.
- `~CommercialBuilding ()`
Destructor for CommercialBuilding.

- std::string **getType** () const override
Gets the type of the building.
- void **updateImpacts** () override
Updates the impacts of the building.
- void **updateCustomer** (int traffic)
Updates the customer traffic.
- void **construct** () override
Constructs the commercial building.
- double **payTaxes** (TaxType *taxType)
Pays taxes for the building.
- void **undoCollectTaxes** ()
Undoes the tax collection from the building.
- void **setBusiness** (Business *business)
Sets the business in the building.

Public Member Functions inherited from Building

- **Building** (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)
Main constructor for setting attributes directly.
- **Building** (int Builder)
*Alternative constructor using an integer parameter, potentially for **Builder** pattern integration.*
- void **setName** (const std::string &name)
Sets the name of the building.
- std::string **getName** () const
Gets the name of the building.
- virtual ~**Building** ()=default
*Virtual destructor for the **Building** class.*
- float **getSatisfaction** () const
Gets the citizen satisfaction impact of the building.
- float **getEconomicGrowth** () const
Gets the economic growth impact of the building.
- float **getResourceConsumption** () const
Gets the resource consumption of the building.
- void **addJob** (std::shared_ptr<Jobs> job)
Adds a job to the building.
- void **listJobs** () const
Lists all jobs in the building.
- bool **hireEmployee** (const std::string &jobTitle)
Hires an employee for a job if available.
- void **releaseEmployee** (const std::string &jobTitle)
Releases an employee from a job.
- std::shared_ptr<Jobs> **getAvailableJob** ()
Provides access to an available job.
- void **displayJobInfo** (const std::string &jobTitle) const
Displays job information.
- const std::vector<std::shared_ptr<Jobs>> & **getJobs** () const
Returns the job list as a const reference.
- **Building** (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)

- Main constructor for setting attributes directly.
- Building (int Builder)
 - Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.
- void [setName](#) (const std::string &name)
 - Sets the name of the building.
- std::string [getName](#) () const
 - Gets the name of the building.
- virtual ~Building ()=default
 - Virtual destructor for the [Building](#) class.
- float [getSatisfaction](#) () const
 - Gets the citizen satisfaction impact of the building.
- float [getEconomicGrowth](#) () const
 - Gets the economic growth impact of the building.
- float [getResourceConsumption](#) () const
 - Gets the resource consumption of the building.
- void [addJob](#) (std::shared_ptr<Jobs> job)
 - Adds a job to the building.
- void [listJobs](#) () const
 - Lists all jobs in the building.
- bool [hireEmployee](#) (const std::string &jobTitle)
 - Hires an employee for a job if available.
- void [releaseEmployee](#) (const std::string &jobTitle)
 - Releases an employee from a job.
- std::shared_ptr<Jobs> [getAvailableJob](#) ()
 - Provides access to an available job.
- void [displayJobInfo](#) (const std::string &jobTitle) const
 - Displays job information.
- const std::vector<std::shared_ptr<Jobs>> & [getJobs](#) () const
 - Returns the job list as a const reference.

Protected Member Functions

- void [calculateEconomicImpact](#) ()
 - Calculates the economic impact of the building.
- void [calculateResourceConsumption](#) ()
 - Calculates the resource consumption of the building.
- void [calculateSatisfactionImpact](#) ()
 - Calculates the satisfaction impact of the building.

Additional Inherited Members

Protected Attributes inherited from Building

- std::string **name**
 - Name of the building.
- float **area**
 - Area of the building.
- int **floors**
 - Number of floors in the building.

- int **capacity**
Capacity of the building.
- float **citizenSatisfaction**
Citizen satisfaction impact of the building.
- float **economicGrowth**
Economic growth impact of the building.
- float **resourceConsumption**
Resource consumption of the building.
- std::vector< std::shared_ptr< [Jobs](#) > > **jobs**
Collection of jobs as shared pointers.

4.19.1 Detailed Description

Represents a commercial building.

The [CommercialBuilding](#) class inherits from the [Building](#) class and provides additional functionality specific to commercial buildings, such as managing business units and customer traffic.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 [CommercialBuilding\(\)](#)

```
CommercialBuilding::CommercialBuilding (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float citizenSatisfaction,
    float economicGrowth,
    float resourceConsumption,
    int businessUnits,
    float customerTraffic )
```

Constructor for [CommercialBuilding](#).

Initializes the commercial building with the given parameters.

Parameters

<i>name</i>	The name of the building.
<i>area</i>	The area of the building.
<i>floors</i>	The number of floors in the building.
<i>capacity</i>	The capacity of the building.
<i>citizenSatisfaction</i>	The citizen satisfaction level.
<i>economicGrowth</i>	The economic growth impact.
<i>resourceConsumption</i>	The resource consumption level.
<i>businessUnits</i>	The number of business units in the building.
<i>customerTraffic</i>	The customer traffic level.

4.19.3 Member Function Documentation

4.19.3.1 calculateEconomicImpact()

```
void CommercialBuilding::calculateEconomicImpact ( ) [protected], [virtual]
```

Calculates the economic impact of the building.

Implements [Building](#).

4.19.3.2 calculateResourceConsumption()

```
void CommercialBuilding::calculateResourceConsumption ( ) [protected], [virtual]
```

Calculates the resource consumption of the building.

Implements [Building](#).

4.19.3.3 calculateSatisfactionImpact()

```
void CommercialBuilding::calculateSatisfactionImpact ( ) [protected], [virtual]
```

Calculates the satisfaction impact of the building.

Implements [Building](#).

4.19.3.4 construct()

```
void CommercialBuilding::construct ( ) [override], [virtual]
```

Constructs the commercial building.

Implements [Building](#).

4.19.3.5 getType()

```
std::string CommercialBuilding::getType ( ) const [override], [virtual]
```

Gets the type of the building.

Returns

The type of the building.

Implements [Building](#).

4.19.3.6 payTaxes()

```
double CommercialBuilding::payTaxes ( TaxType * taxType ) [virtual]
```

Pays taxes for the building.

Parameters

<i>taxType</i>	Pointer to the TaxType object.
----------------	--

Returns

The total amount of taxes collected.

Implements [Building](#).

4.19.3.7 setBusiness()

```
void CommercialBuilding::setBusiness (  
    Business * business )
```

Sets the business in the building.

Parameters

<i>business</i>	Pointer to the Business object.
-----------------	---

4.19.3.8 undoCollectTaxes()

```
void CommercialBuilding::undoCollectTaxes ( ) [virtual]
```

Undoes the tax collection from the building.

Implements [Building](#).

4.19.3.9 updateCustomer()

```
void CommercialBuilding::updateCustomer (   
    int traffic )
```

Updates the customer traffic.

Parameters

<i>traffic</i>	The amount of traffic to add.
----------------	-------------------------------

4.19.3.10 updateImpacts()

```
void CommercialBuilding::updateImpacts ( ) [override], [virtual]
```

Updates the impacts of the building.

Calculates the economic impact, resource consumption, and satisfaction impact.

Implements [Building](#).

The documentation for this class was generated from the following files:

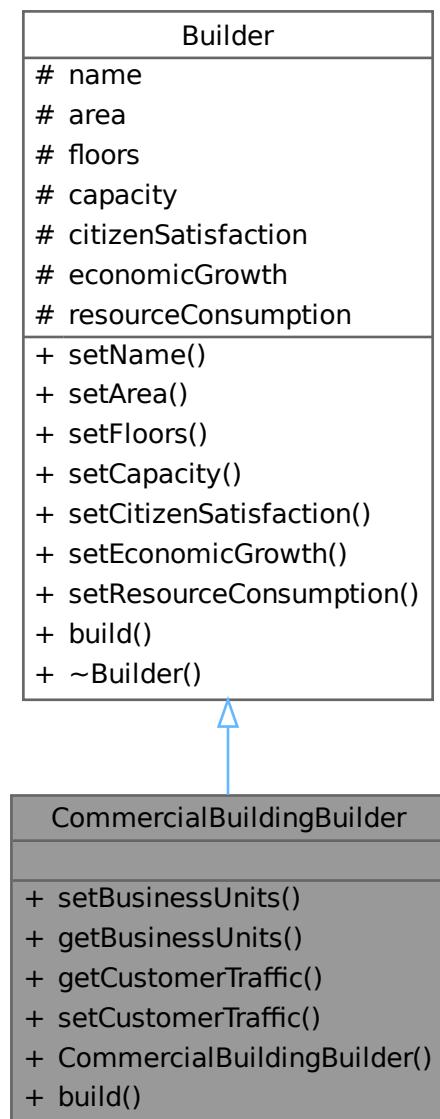
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CommercialBuilding.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CommercialBuilding.cpp](#)

4.20 CommercialBuildingBuilder Class Reference

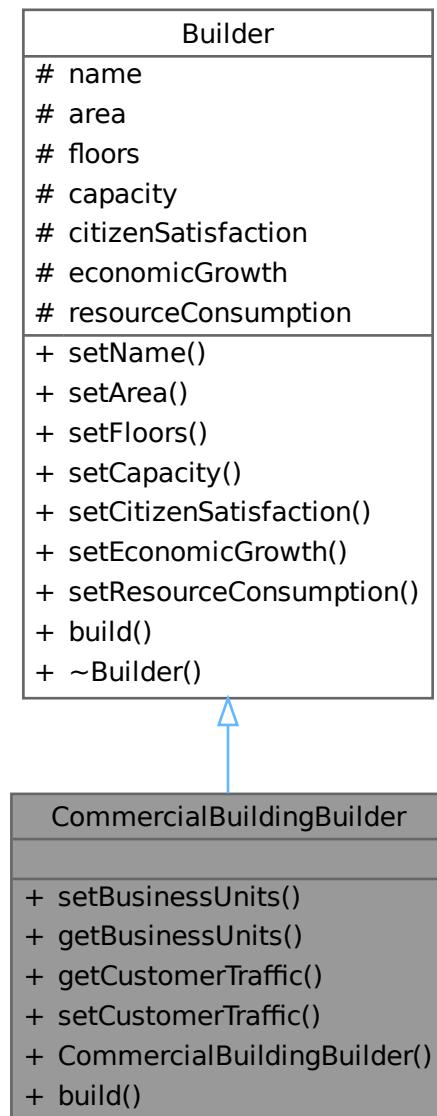
[Builder](#) class for constructing [CommercialBuilding](#) objects.

```
#include <CommercialBuildingBuilder.h>
```

Inheritance diagram for CommercialBuildingBuilder:



Collaboration diagram for CommercialBuildingBuilder:



Public Member Functions

- [CommercialBuildingBuilder](#) & [setBusinessUnits](#) (int units)
Sets the number of business units in the building.
- int [getBusinessUnits](#) ()
Gets the number of business units in the building.
- float [getCustomerTraffic](#) ()
Gets the customer traffic in the building.
- [CommercialBuildingBuilder](#) & [setCustomerTraffic](#) (float customerTraffic)
Sets the customer traffic in the building.

- **CommercialBuildingBuilder ()**
Default constructor for [CommercialBuildingBuilder](#).
- std::unique_ptr< Building > build () override
Builds the commercial building and sets all the attributes of the building.

Public Member Functions inherited from [Builder](#)

- [Builder & setName \(string name\)](#)
Sets the name of the building.
- [Builder & setArea \(float area\)](#)
Sets the area of the building.
- [Builder & setFloors \(int floors\)](#)
Sets the number of floors in the building.
- [Builder & setCapacity \(int capacity\)](#)
Sets the capacity of the building.
- [Builder & setCitizenSatisfaction \(float citizenSatisfaction\)](#)
Sets the citizen satisfaction of the building.
- [Builder & setEconomicGrowth \(float economicGrowth\)](#)
Sets the economic growth of the building.
- [Builder & setResourceConsumption \(float resourceConsumption\)](#)
Sets the resource consumption of the building.
- virtual ~[Builder \(\)=default](#)
Virtual destructor for the [Builder](#) class.

Additional Inherited Members

Protected Attributes inherited from [Builder](#)

- string **name**
Name of the building.
- float **area** = 0.0f
Area of the building.
- int **floors** = 0
Number of floors in the building.
- int **capacity** = 0
Capacity of the building.
- float **citizenSatisfaction** = 0.0f
Citizen satisfaction of the building.
- float **economicGrowth** = 0.0f
Economic growth of the building.
- float **resourceConsumption** = 0.0f
Resource consumption of the building.

4.20.1 Detailed Description

[Builder](#) class for constructing [CommercialBuilding](#) objects.

The [CommercialBuildingBuilder](#) class provides methods to set various attributes of a commercial building and build the final [CommercialBuilding](#) object.

4.20.2 Member Function Documentation

4.20.2.1 build()

```
std::unique_ptr< Building > CommercialBuildingBuilder::build ( ) [override], [virtual]
```

Builds the commercial building and sets all the attributes of the building.

Returns

A unique pointer to the built [CommercialBuilding](#) object.

Implements [Builder](#).

4.20.2.2 getBusinessUnits()

```
int CommercialBuildingBuilder::getBusinessUnits ( )
```

Gets the number of business units in the building.

Returns

The number of business units.

4.20.2.3 getCustomerTraffic()

```
float CommercialBuildingBuilder::getCustomerTraffic ( )
```

Gets the customer traffic in the building.

Returns

The customer traffic.

4.20.2.4 setBusinessUnits()

```
CommercialBuildingBuilder & CommercialBuildingBuilder::setBusinessUnits ( int units )
```

Sets the number of business units in the building.

Parameters

<i>units</i>	The number of business units.
--------------	-------------------------------

Returns

Reference to the current builder object.

4.20.2.5 setCustomerTraffic()

```
CommercialBuildingBuilder & CommercialBuildingBuilder::setCustomerTraffic (  
    float traffic )
```

Sets the customer traffic in the building.

Parameters

<i>customerTraffic</i>	The customer traffic.
------------------------	-----------------------

Returns

Reference to the current builder object.

Parameters

<i>traffic</i>	The customer traffic.
----------------	-----------------------

Returns

Reference to the current builder object.

The documentation for this class was generated from the following files:

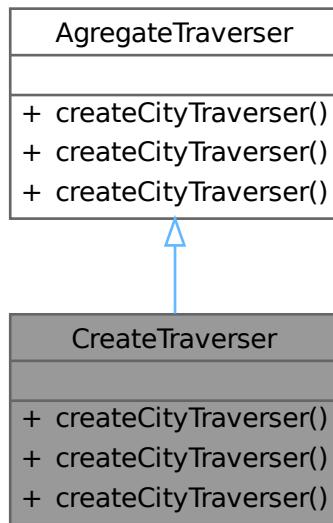
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CommercialBuildingBuilder.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[CommercialBuildingBuilder.cpp](#)

4.21 CreateTraverser Class Reference

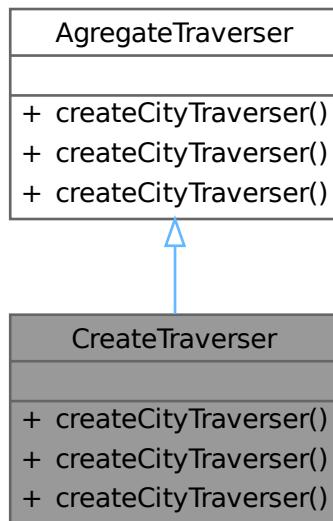
A class to create [CityTraverser](#) instances.

```
#include <createTraverser.h>
```

Inheritance diagram for CreateTraverser:



Collaboration diagram for CreateTraverser:



Public Member Functions

- `CityTraverser * createCityTraverser ()`

Creates a new [CityTraverser](#) instance.

- `CityTraverser * createCityTraverser (Transportation *t)`

Creates a new [CityTraverser](#) instance with a specified [Transportation](#) object.

- `CityTraverser * createCityTraverser (CityTraverser *t)`

Creates a new [CityTraverser](#) instance by copying an existing [CityTraverser](#) object.

4.21.1 Detailed Description

A class to create [CityTraverser](#) instances.

The [CreateTraverser](#) class inherits from [AgregateTraverser](#) and provides methods to create [CityTraverser](#) objects.

4.21.2 Member Function Documentation

4.21.2.1 `createCityTraverser()` [1/3]

```
CityTraverser * CreateTraverser::createCityTraverser ( ) [virtual]
```

Creates a new [CityTraverser](#) instance.

Creates a new [CityTraverser](#) object.

Returns

A pointer to the newly created [CityTraverser](#) instance.

This method creates a new instance of the [CityTraverser](#) class without any parameters.

Returns

`CityTraverser*` Pointer to the newly created [CityTraverser](#) object.

Implements [AgregateTraverser](#).

4.21.2.2 `createCityTraverser()` [2/3]

```
CityTraverser * CreateTraverser::createCityTraverser (
    CityTraverser * t ) [virtual]
```

Creates a new [CityTraverser](#) instance by copying an existing [CityTraverser](#) object.

Creates a new [CityTraverser](#) object by copying another [CityTraverser](#) object.

Parameters

<code>t</code>	A pointer to an existing CityTraverser object.
----------------	--

Returns

A pointer to the newly created [CityTraverser](#) instance.

This method creates a new instance of the [CityTraverser](#) class by copying an existing [CityTraverser](#) object. If the parameter is null, it creates a [CityTraverser](#) without any parameters.

Parameters

<i>t</i>	Pointer to a CityTraverser object.
----------	--

Returns

[CityTraverser](#)* Pointer to the newly created [CityTraverser](#) object.

Implements [AgreateTraverser](#).

4.21.2.3 createCityTraverser() [3/3]

```
CityTraverser * CreateTraverser::createCityTraverser (
    Transportation * t ) [virtual]
```

Creates a new [CityTraverser](#) instance with a specified [Transportation](#) object.

Creates a new [CityTraverser](#) object with a [Transportation](#) parameter.

Parameters

<i>t</i>	A pointer to a Transportation object.
----------	---

Returns

A pointer to the newly created [CityTraverser](#) instance.

This method creates a new instance of the [CityTraverser](#) class with a [Transportation](#) parameter. If the parameter is null, it creates a [CityTraverser](#) without any parameters.

Parameters

<i>t</i>	Pointer to a Transportation object.
----------	---

Returns

[CityTraverser](#)* Pointer to the newly created [CityTraverser](#) object.

Implements [AgreateTraverser](#).

The documentation for this class was generated from the following files:

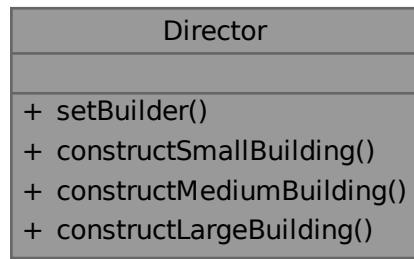
- /mnt/c/users/rudie/documents/sem2/214/project/src/[createTraverser.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[createTraverser.cpp](#)

4.22 Director Class Reference

The [Director](#) class for constructing buildings.

```
#include <Director.h>
```

Collaboration diagram for Director:



Public Member Functions

- void [setBuilder \(Builder &builder\)](#)
Sets the builder object.
- std::unique_ptr<[Building](#)> [constructSmallBuilding \(\)](#)
Constructs a small building.
- std::unique_ptr<[Building](#)> [constructMediumBuilding \(\)](#)
Constructs a medium building.
- std::unique_ptr<[Building](#)> [constructLargeBuilding \(\)](#)
Constructs a large building.

4.22.1 Detailed Description

The [Director](#) class for constructing buildings.

This class provides methods to construct buildings of different sizes using a [Builder](#) object.

Version

1.0

Date

2024-11-04

4.22.2 Member Function Documentation

4.22.2.1 constructLargeBuilding()

```
std::unique_ptr< Building > Director::constructLargeBuilding ( )
```

Constructs a large building.

This method constructs a large building with predefined parameters.

Returns

```
std::unique_ptr<Building> Pointer to the newly constructed large building.
```

4.22.2.2 constructMediumBuilding()

```
std::unique_ptr< Building > Director::constructMediumBuilding ( )
```

Constructs a medium building.

This method constructs a medium building with predefined parameters.

Returns

```
std::unique_ptr<Building> Pointer to the newly constructed medium building.
```

4.22.2.3 constructSmallBuilding()

```
std::unique_ptr< Building > Director::constructSmallBuilding ( )
```

Constructs a small building.

This method constructs a small building with predefined parameters.

Returns

```
std::unique_ptr<Building> Pointer to the newly constructed small building.
```

4.22.2.4 setBuilder()

```
void Director::setBuilder (  
    Builder & builder )
```

Sets the builder object.

This method sets the builder object that will be used to construct buildings.

Parameters

<code>builder</code>	Reference to a Builder object.
----------------------	--

The documentation for this class was generated from the following files:

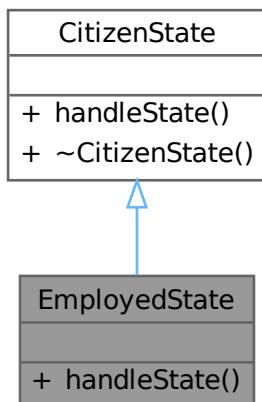
- /mnt/c/users/rudie/documents/sem2/214/project/src/Director.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Director.cpp](#)

4.23 EmployedState Class Reference

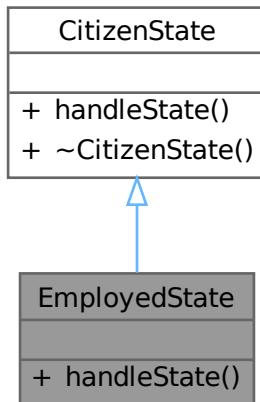
The [EmployedState](#) class for handling the employed state of a [Citizen](#).

```
#include <EmployedState.h>
```

Inheritance diagram for EmployedState:



Collaboration diagram for EmployedState:



Public Member Functions

- void `handleState (Citizen &citizen) const override`
Handles the state of a `Citizen` when they are employed.

Public Member Functions inherited from `CitizenState`

- virtual `~CitizenState ()=default`
Virtual destructor for the `CitizenState` class.

4.23.1 Detailed Description

The `EmployedState` class for handling the employed state of a `Citizen`.

This class provides methods to handle the state of a `Citizen` when they are employed.

4.23.2 Member Function Documentation

4.23.2.1 `handleState()`

```
void EmployedState::handleState (
    Citizen & citizen ) const [override], [virtual]
```

Handles the state of a `Citizen` when they are employed.

This method updates the satisfaction level of the `Citizen` when they are in the employed state.

Parameters

<i>citizen</i>	Reference to the Citizen object.
----------------	--

Implements [CitizenState](#).

The documentation for this class was generated from the following files:

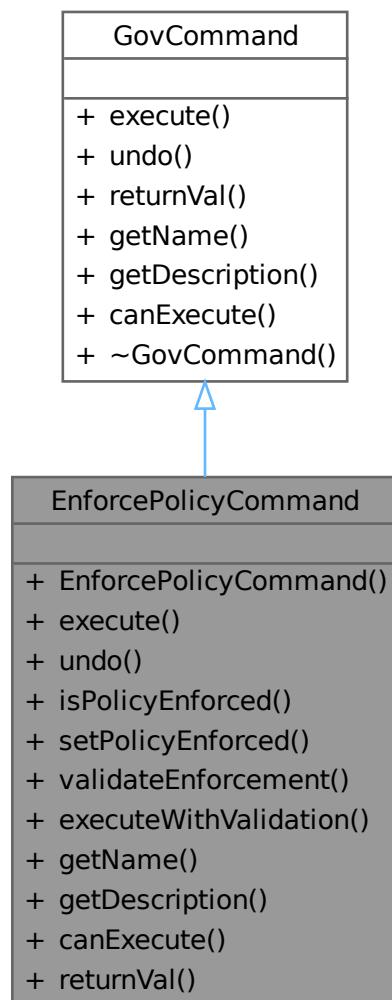
- /mnt/c/users/rudie/documents/sem2/214/project/src/[EmployedState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[EmployedState.cpp](#)

4.24 EnforcePolicyCommand Class Reference

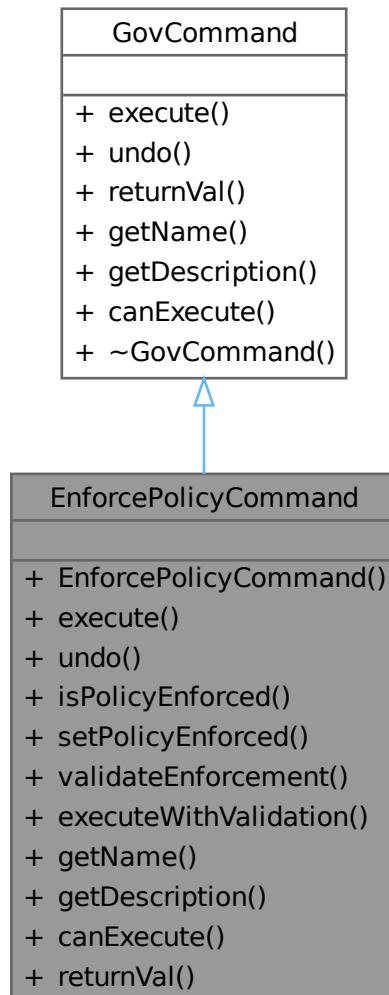
The [EnforcePolicyCommand](#) class for enforcing policies in the government.

```
#include <EnforcePolicyCommand.h>
```

Inheritance diagram for EnforcePolicyCommand:



Collaboration diagram for EnforcePolicyCommand:



Public Member Functions

- **EnforcePolicyCommand (Government *gov, Policy pol)**
Constructor for `EnforcePolicyCommand`.
- void **execute ()** override
Executes the policy enforcement command.
- void **undo ()** override
Undoes the policy enforcement command.
- bool **isPolicyEnforced ()** const
Checks if the policy is enforced.
- void **setPolicyEnforced (bool enforced)**
Sets the enforcement status of the policy.
- bool **validateEnforcement ()** const
Validates the policy enforcement.

- void `executeWithValidation ()`
Executes the policy enforcement command with validation.
- std::string `getName () const override`
Gets the name of the command.
- std::string `getDescription () const override`
Gets the description of the command.
- bool `canExecute () const override`
Checks if the command can be executed.
- double `returnVal () override`
Returns an appropriate value based on the policy enforcement.

Public Member Functions inherited from [GovCommand](#)

- virtual ~[GovCommand](#) ()=default
Virtual destructor.

4.24.1 Detailed Description

The [EnforcePolicyCommand](#) class for enforcing policies in the government.

This class represents a command to enforce a policy in the government.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 EnforcePolicyCommand()

```
EnforcePolicyCommand::EnforcePolicyCommand (
    Government * gov,
    Policy pol )
```

Constructor for [EnforcePolicyCommand](#).

Initializes the command with the government object and the policy to enforce.

Parameters

<code>gov</code>	Pointer to the Government object.
<code>pol</code>	The Policy to be enforced.

4.24.3 Member Function Documentation

4.24.3.1 canExecute()

```
bool EnforcePolicyCommand::canExecute ( ) const [override], [virtual]
```

Checks if the command can be executed.

Returns

true if the command can be executed, false otherwise.

Implements [GovCommand](#).

4.24.3.2 execute()

```
void EnforcePolicyCommand::execute ( ) [override], [virtual]
```

Executes the policy enforcement command.

Enforces the specified policy in the government.

Implements [GovCommand](#).

4.24.3.3 executeWithValidation()

```
void EnforcePolicyCommand::executeWithValidation ( )
```

Executes the policy enforcement command with validation.

Throws an exception if the enforcement is invalid.

4.24.3.4 getDescription()

```
std::string EnforcePolicyCommand::getDescription ( ) const [override], [virtual]
```

Gets the description of the command.

Returns

The description of the command.

Implements [GovCommand](#).

4.24.3.5 getName()

```
std::string EnforcePolicyCommand::getName ( ) const [override], [virtual]
```

Gets the name of the command.

Returns

The name of the command.

Implements [GovCommand](#).

4.24.3.6 `isPolicyEnforced()`

```
bool EnforcePolicyCommand::isPolicyEnforced ( ) const
```

Checks if the policy is enforced.

Returns

true if the policy is enforced, false otherwise.

4.24.3.7 `returnVal()`

```
double EnforcePolicyCommand::returnVal ( ) [override], [virtual]
```

Returns an appropriate value based on the policy enforcement.

Returns

A double value based on the policy enforcement.

Implements [GovCommand](#).

4.24.3.8 `setPolicyEnforced()`

```
void EnforcePolicyCommand::setPolicyEnforced ( bool enforced )
```

Sets the enforcement status of the policy.

Parameters

<code>enforced</code>	The enforcement status to be set.
-----------------------	-----------------------------------

4.24.3.9 `undo()`

```
void EnforcePolicyCommand::undo ( ) [override], [virtual]
```

Undoes the policy enforcement command.

Reverts the policy enforcement if applicable.

Implements [GovCommand](#).

4.24.3.10 `validateEnforcement()`

```
bool EnforcePolicyCommand::validateEnforcement ( ) const
```

Validates the policy enforcement.

Ensures the policy is valid for enforcement.

Returns

true if the policy is valid for enforcement, false otherwise.

The documentation for this class was generated from the following files:

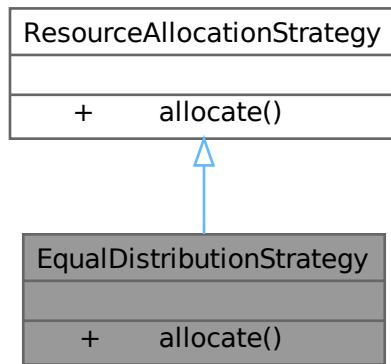
- /mnt/c/users/rudie/documents/sem2/214/project/src/[EnforcePolicyCommand.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[EnforcePolicyCommand.cpp](#)

4.25 EqualDistributionStrategy Class Reference

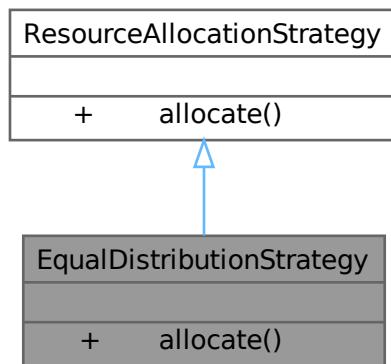
Strategy for equal distribution of resources.

```
#include <ResourceAllocationStrategy.h>
```

Inheritance diagram for EqualDistributionStrategy:



Collaboration diagram for EqualDistributionStrategy:



Public Member Functions

- bool [allocate \(ResourceType type, int quantity\) override](#)
Allocates resources equally.

4.25.1 Detailed Description

Strategy for equal distribution of resources.

The [EqualDistributionStrategy](#) class implements the [ResourceAllocationStrategy](#) interface and provides logic for equal distribution of resources.

4.25.2 Member Function Documentation

4.25.2.1 [allocate\(\)](#)

```
bool EqualDistributionStrategy::allocate (
    ResourceType type,
    int quantity ) [inline], [override], [virtual]
```

Allocates resources equally.

Parameters

<code>type</code>	The type of the resource.
<code>quantity</code>	The quantity of the resource to allocate.

Returns

True if the allocation was successful, false otherwise.

Implements [ResourceAllocationStrategy](#).

The documentation for this class was generated from the following file:

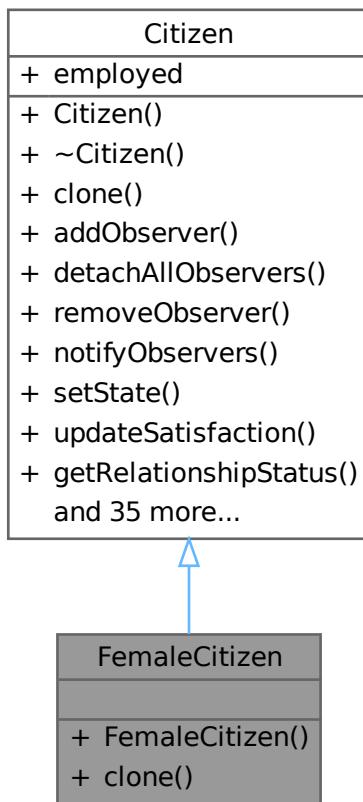
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceAllocationStrategy.h](#)

4.26 FemaleCitizen Class Reference

A class representing a female citizen.

```
#include <FemaleCitizen.h>
```

Inheritance diagram for FemaleCitizen:



Collaboration diagram for FemaleCitizen:



Public Member Functions

- `FemaleCitizen` (const std::string &name, int age)
Constructor for `FemaleCitizen`.
- `std::shared_ptr< Citizen > clone () const override`
Clone method to create a copy of the `FemaleCitizen` object.

Public Member Functions inherited from `Citizen`

- `Citizen` (const std::string &name, int age)
Constructs a new `Citizen` object.
- `virtual ~Citizen ()`
Destroys the `Citizen` object.
- `void addObserver (CitizenObserver *observer)`
Adds an observer to the citizen.
- `void detachAllObservers ()`
Detaches all observers from the citizen.
- `void removeObserver (CitizenObserver *observer)`

- **void notifyObservers ()**
Removes an observer from the citizen.
- **void setState (CitizenState *newState)**
Notifies all observers of changes to the citizen.
- **void updateSatisfaction ()**
Sets the state of the citizen.
- **void updateSatisfaction (float adjustment)**
Updates the satisfaction of the citizen based on all strategies.
- **std::string getRelationshipStatus () const**
Gets the relationship status of the citizen.
- **void setRelationshipStatus (const std::string &status)**
Sets the relationship status of the citizen.
- **int getMarriageDuration () const**
Gets the duration of the citizen's marriage.
- **void resetMarriageDuration ()**
Resets the duration of the citizen's marriage to zero.
- **void incrementMarriageDuration ()**
Increments the duration of the citizen's marriage by one year.
- **void updateSatisfaction (float adjustment)**
Updates the satisfaction of the citizen by a specified adjustment.
- **void addChild ()**
Adds a child to the citizen's family.
- **int getAge () const**
Gets the age of the citizen.
- **void incrementAge ()**
Increments the age of the citizen by one year.
- **void addSatisfactionStrategy (std::shared_ptr< SatisfactionStrategy > strategy)**
Adds a satisfaction strategy to the citizen.
- **void removeSatisfactionStrategy ()**
Removes all satisfaction strategies from the citizen.
- **void setSatisfactionLevel (double level)**
Sets the satisfaction level of the citizen.
- **void depositMonthlyIncome ()**
Deposits the monthly income of the citizen into their bank balance.
- **void searchAndApplyForJob (BuildingManager &manager, Building *building, std::string jobtitle)**
Searches for and applies for a job for the citizen.
- **std::string getName () const**
Gets the name of the citizen.
- **float getSatisfactionLevel () const**
Gets the satisfaction level of the citizen.
- **bool isLeaving () const**
Checks if the citizen is leaving the city.
- **void setIncome (std::shared_ptr< Income > income)**
Sets the income of the citizen.
- **void checkAndUpdateState ()**
Checks and updates the state of the citizen based on satisfaction.
- **void setJobTitle (const std::string &jobTitle)**
Sets the job title of the citizen.
- **std::string getJob () const**
Gets the job title of the citizen.
- **void setEmployed (bool status)**
Sets the employment status of the citizen.

- bool `isEmployed () const`
Checks if the citizen is employed.
- float `getTaxRate () const`
Gets the tax rate for the citizen.
- void `setTaxRate (float rate)`
Sets the tax rate for the citizen.
- double `payTaxes (TaxType *taxType)`
Processes the payment of taxes by the citizen.
- double `getBankBalance () const`
Gets the bank balance of the citizen.
- void `setBankBalance (double balance)`
Sets the bank balance of the citizen.
- void `increaseBankBalance (double amount)`
Increases the bank balance of the citizen by a specified amount.
- void `subtractBankBalance (double amount)`
Decreases the bank balance of the citizen by a specified amount.
- void `setTaxCooldown (bool status)`
Sets the tax cooldown status for the citizen.
- bool `getTaxCooldown () const`
Gets the tax cooldown status for the citizen.
- bool `isOnCooldown () const`
Checks if the citizen is on cooldown for tax payments.
- std::shared_ptr<Jobs> `getjobobj ()`
Gets the job object of the citizen.
- void `setJob (std::shared_ptr<Jobs> job)`
Sets the job object of the citizen.
- void `unsetJob ()`
Unsets the job object of the citizen.

Additional Inherited Members

Public Attributes inherited from [Citizen](#)

- bool `employed = false`
Employment status of the citizen.

4.26.1 Detailed Description

A class representing a female citizen.

The [FemaleCitizen](#) class inherits from the [Citizen](#) class and provides a specific implementation for female citizens. It includes a clone method to create a copy of the object.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 [FemaleCitizen\(\)](#)

```
FemaleCitizen::FemaleCitizen (
    const std::string & name,
    int age ) [inline]
```

Constructor for [FemaleCitizen](#).

Parameters

<i>name</i>	The name of the female citizen.
<i>age</i>	The age of the female citizen.

4.26.3 Member Function Documentation

4.26.3.1 clone()

```
std::shared_ptr< Citizen > FemaleCitizen::clone ( ) const [inline], [override], [virtual]
```

Clone method to create a copy of the [FemaleCitizen](#) object.

Returns

A shared pointer to the cloned [FemaleCitizen](#) object.

Implements [Citizen](#).

The documentation for this class was generated from the following file:

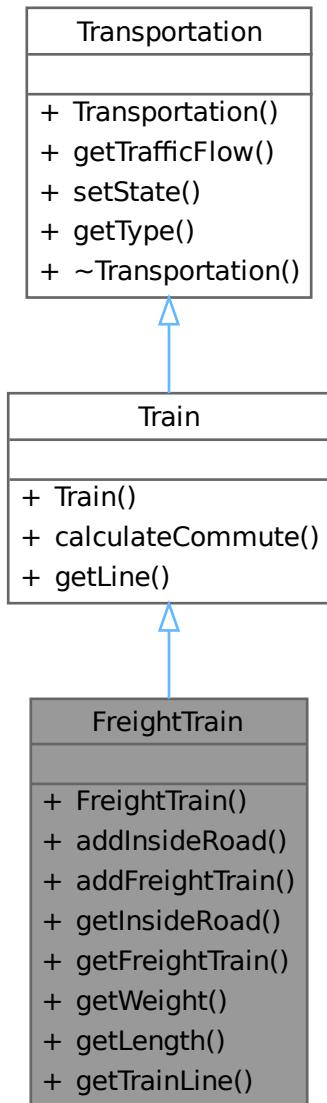
- /mnt/c/users/rudie/documents/sem2/214/project/src/[FemaleCitizen.h](#)

4.27 FreightTrain Class Reference

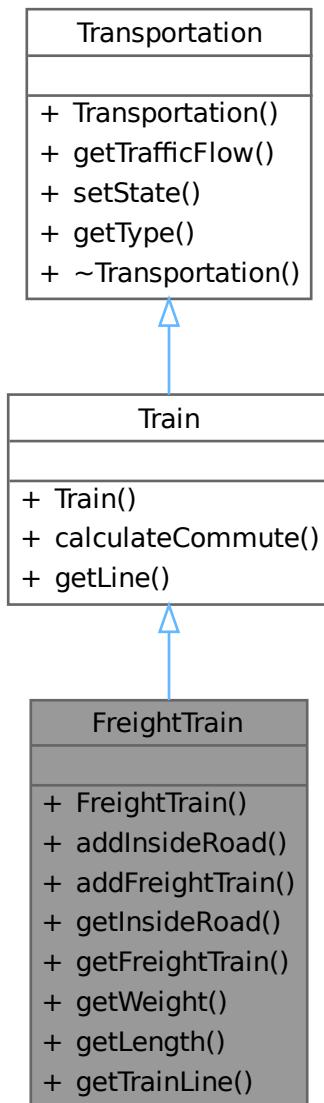
A class representing a freight train.

```
#include <FreightTrain.h>
```

Inheritance diagram for FreightTrain:



Collaboration diagram for FreightTrain:



Public Member Functions

- `FreightTrain (char state, std::string line, float weight, float length)`
Constructor for the `FreightTrain` class.
- `bool addInsideRoad (InsideRoad *insideRoad)`
Adds an `InsideRoad` object to the freight train.
- `bool addFreightTrain (FreightTrain *freightTrain)`
Adds a `FreightTrain` object to the freight train.
- `InsideRoad *getInsideRoad (size_t index)`
Retrieves an `InsideRoad` object by index.
- `FreightTrain *getFreightTrain (size_t index)`

- Retrieves a [FreightTrain](#) object by index.
- float [getWeight \(\)](#)
Gets the weight of the freight train.
- float [getLength \(\)](#)
Gets the length of the freight train.
- std::string [getTrainLine \(\)](#)
Gets the line on which the train operates.

Public Member Functions inherited from [Train](#)

- [Train](#) (char state, std::string line, char type)
Construct a new [Train](#) object.
- float [calculateCommute \(\)](#)
Calculate the commute time for the train.
- std::string [getLine \(\)](#)
Get the line on which the train operates.

Public Member Functions inherited from [Transportation](#)

- [Transportation](#) (char state, char type)
Constructor for the [Transportation](#) class.
- float [getTrafficFlow \(\)](#)
Gets the current traffic flow.
- bool [setState \(char state\)](#)
Sets the traffic flow state.
- char [getType \(\)](#)
Gets the type of transportation.
- [~Transportation \(\)](#)
Destructor for the [Transportation](#) class.

4.27.1 Detailed Description

A class representing a freight train.

The [FreightTrain](#) class inherits from the [Train](#) class and includes additional attributes specific to freight trains, such as weight and length. It also manages collections of [InsideRoad](#) and [FreightTrain](#) objects.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 [FreightTrain\(\)](#)

```
FreightTrain::FreightTrain (
    char state,
    std::string line,
    float weight,
    float length )
```

Constructor for the [FreightTrain](#) class.

Constructor for [FreightTrain](#).

Parameters

<i>state</i>	The state of the train.
<i>line</i>	The line on which the train operates.
<i>weight</i>	The weight of the freight train.
<i>length</i>	The length of the freight train.
<i>state</i>	The state of the freight train.
<i>line</i>	The line on which the freight train operates.
<i>weight</i>	The weight of the freight train.
<i>length</i>	The length of the freight train.

4.27.3 Member Function Documentation**4.27.3.1 addFreightTrain()**

```
bool FreightTrain::addFreightTrain (
    FreightTrain * freightTrain )
```

Adds a [FreightTrain](#) object to the freight train.

Adds another [FreightTrain](#) to the freight train.

Parameters

<i>freightTrain</i>	Pointer to the FreightTrain object to be added.
---------------------	---

Returns

true if the [FreightTrain](#) was added successfully, false otherwise.

Parameters

<i>freightTrain</i>	Pointer to the FreightTrain to be added.
---------------------	--

Returns

True if the [FreightTrain](#) was added successfully, false if it already exists.

4.27.3.2 addInsideRoad()

```
bool FreightTrain::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an [InsideRoad](#) object to the freight train.

Adds an [InsideRoad](#) to the freight train.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad object to be added.
-------------------	---

Returns

true if the [InsideRoad](#) was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad to be added.
-------------------	--

Returns

True if the [InsideRoad](#) was added successfully, false if it already exists.

4.27.3.3 [getFreightTrain\(\)](#)

```
FreightTrain * FreightTrain::getFreightTrain ( size_t index )
```

Retrieves a [FreightTrain](#) object by index.

Gets a [FreightTrain](#) by index.

Parameters

<i>index</i>	The index of the FreightTrain object to retrieve.
--------------	---

Returns

Pointer to the [FreightTrain](#) object at the specified index.

Parameters

<i>index</i>	The index of the FreightTrain .
--------------	---

Returns

Pointer to the [FreightTrain](#) if the index is valid, nullptr otherwise.

4.27.3.4 [getInsideRoad\(\)](#)

```
InsideRoad * FreightTrain::getInsideRoad ( size_t index )
```

Retrieves an [InsideRoad](#) object by index.

Gets an [InsideRoad](#) by index.

Parameters

<i>index</i>	The index of the InsideRoad object to retrieve.
--------------	---

Returns

Pointer to the [InsideRoad](#) object at the specified index.

Parameters

<i>index</i>	The index of the InsideRoad .
--------------	---

Returns

Pointer to the [InsideRoad](#) if the index is valid, nullptr otherwise.

4.27.3.5 `getLength()`

```
float FreightTrain::getLength ( )
```

Gets the length of the freight train.

Returns

The length of the freight train.

4.27.3.6 `getTrainLine()`

```
std::string FreightTrain::getTrainLine ( )
```

Gets the line on which the train operates.

Gets the line on which the freight train operates.

Returns

The line on which the train operates.

The line of the freight train.

4.27.3.7 `getWeight()`

```
float FreightTrain::getWeight ( )
```

Gets the weight of the freight train.

Returns

The weight of the freight train.

The documentation for this class was generated from the following files:

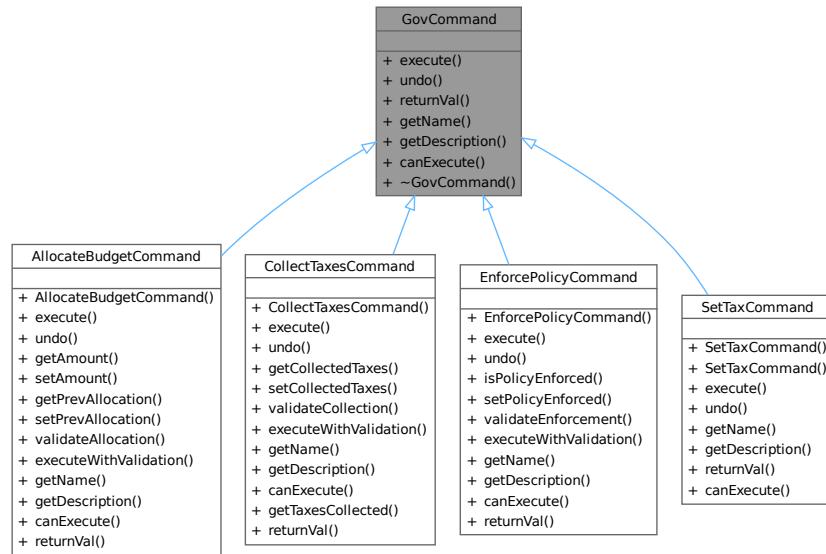
- /mnt/c/users/rudie/documents/sem2/214/project/src/[FreightTrain.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[FreightTrain.cpp](#)

4.28 GovCommand Class Reference

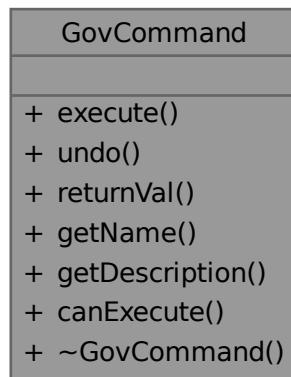
Abstract base class for government commands.

```
#include <GovCommand.h>
```

Inheritance diagram for GovCommand:



Collaboration diagram for GovCommand:



Public Member Functions

- virtual void `execute ()=0`
Executes the command.
- virtual void `undo ()=0`
Undoes the command.
- virtual double `returnVal ()=0`
Returns a value associated with the command.
- virtual std::string `getName () const =0`
Gets the name of the command.
- virtual std::string `getDescription () const =0`
Gets the description of the command.
- virtual bool `canExecute () const =0`
Checks if the command can be executed.
- virtual `~GovCommand ()=default`
Virtual destructor.

4.28.1 Detailed Description

Abstract base class for government commands.

This class serves as a base class for various government-related commands. Derived classes should override the methods to provide specific command execution logic.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 `~GovCommand()`

```
virtual GovCommand::~GovCommand ( ) [virtual], [default]
```

Virtual destructor.

Ensures derived class destructors are called correctly.

4.28.3 Member Function Documentation

4.28.3.1 `canExecute()`

```
bool GovCommand::canExecute ( ) const [pure virtual]
```

Checks if the command can be executed.

This method should be overridden by derived classes to provide execution validation.

Returns

True if the command can be executed, false otherwise.

Implemented in [AllocateBudgetCommand](#), [CollectTaxesCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

4.28.3.2 `execute()`

```
void GovCommand::execute ( ) [pure virtual]
```

Executes the command.

This method should be overridden by derived classes to implement specific command execution logic.

This method should be overridden by derived classes to implement specific command execution logic.

Exceptions

<i>const</i>	char* Exception indicating the method is not yet implemented.
--------------	---

Implemented in [AllocateBudgetCommand](#), [CollectTaxesCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

4.28.3.3 `getDescription()`

```
std::string GovCommand::getDescription ( ) const [pure virtual]
```

Gets the description of the command.

This method should be overridden by derived classes to provide the command description.

Returns

The description of the command.

Implemented in [AllocateBudgetCommand](#), [CollectTaxesCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

4.28.3.4 `getName()`

```
std::string GovCommand::getName ( ) const [pure virtual]
```

Gets the name of the command.

This method should be overridden by derived classes to provide the command name.

Returns

The name of the command.

Implemented in [AllocateBudgetCommand](#), [CollectTaxesCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

4.28.3.5 `returnVal()`

```
double GovCommand::returnVal ( ) [pure virtual]
```

Returns a value associated with the command.

This method can be overridden by derived classes to return a specific value.

Returns

A double value associated with the command.

Implemented in [CollectTaxesCommand](#), [AllocateBudgetCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

4.28.3.6 `undo()`

```
void GovCommand::undo ( ) [pure virtual]
```

Undoes the command.

This method should be overridden by derived classes to implement specific command undo logic.

This method should be overridden by derived classes to implement specific command undo logic.

Exceptions

<code>const</code>	char* Exception indicating the method is not yet implemented.
--------------------	---

Implemented in [AllocateBudgetCommand](#), [CollectTaxesCommand](#), [EnforcePolicyCommand](#), and [SetTaxCommand](#).

The documentation for this class was generated from the following files:

- /mnt/c/users/rudie/documents/sem2/214/project/src/[GovCommand.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[GovCommand.cpp](#)

4.29 Government Class Reference

Manages various government-related operations.

```
#include <Government.h>
```

Collaboration diagram for Government:

Government
+ Government() + setTax() + getTaxRate() + notifyCitizen() + notifyBusinesses() + notifyServices() + allocateBudget() + revertBudgetAllocation() + enforcePolicy() + update() and 7 more...

Public Member Functions

- [Government](#) (std::string name)
Constructor for Government.
- void [setTax](#) (double rate)
Sets the tax rate for the government.
- double [getTaxRate](#) () const

- Gets the current tax rate.
- void **notifyCitizen** ()
 - Notifies citizens of changes.*
- void **notifyBusinesses** ()
 - Notifies businesses of changes.*
- void **notifyServices** ()
 - Notifies services of changes.*
- void **allocateBudget** ([CityService](#) &service, double amount)
 - Allocates budget to a city service.*
- void **revertBudgetAllocation** ([CityService](#) &service, double amount)
 - Reverts the budget allocation to a city service.*
- void **enforcePolicy** ([Policy](#) policy)
 - Enforces a policy in the government.*
- void **update** (int newPopulation)
 - Updates the population of the government.*
- double **collectTaxes** ()
 - Collects taxes based on the tax rate.*
- void **refundTaxes** (double amount)
 - Refunds collected taxes.*
- void **addTaxesToBudget** (double amount)
 - Adds taxes to the budget.*
- void **registerObserver** ([GovObserver](#) *observer)
 - Registers an observer to the government.*
- void **unregisterObserver** ([GovObserver](#) *observer)
 - Unregisters an observer from the government.*
- void **notifyObservers** ()
 - Notifies all registered observers of changes.*
- double **getBudget** () const
 - Gets the current budget.*

4.29.1 Detailed Description

Manages various government-related operations.

The [Government](#) class manages various government-related operations such as setting tax rates, allocating budgets, and notifying observers.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 [Government\(\)](#)

```
Government::Government (
    std::string name )
```

Constructor for [Government](#).

Parameters

<i>name</i>	The name of the government.
-------------	-----------------------------

4.29.3 Member Function Documentation

4.29.3.1 addTaxesToBudget()

```
void Government::addTaxesToBudget (  
    double amount )
```

Adds taxes to the budget.

Parameters

<i>amount</i>	The amount of taxes to add.
---------------	-----------------------------

4.29.3.2 allocateBudget()

```
void Government::allocateBudget (   
     CityService & service,  
     double amount )
```

Allocates budget to a city service.

Parameters

<i>service</i>	The city service to allocate the budget to.
<i>amount</i>	The amount to allocate.

4.29.3.3 collectTaxes()

```
double Government::collectTaxes ( )
```

Collects taxes based on the tax rate.

Returns

The amount of taxes collected.

4.29.3.4 enforcePolicy()

```
void Government::enforcePolicy (   
     Policy policy )
```

Enforces a policy in the government.

Parameters

<i>policy</i>	The policy to enforce.
---------------	------------------------

4.29.3.5 `getBudget()`

```
double Government::getBudget ( ) const
```

Gets the current budget.

Returns

The current budget.

4.29.3.6 `getTaxRate()`

```
double Government::getTaxRate ( ) const
```

Gets the current tax rate.

Returns

The current tax rate.

4.29.3.7 `refundTaxes()`

```
void Government::refundTaxes (  
    double amount )
```

Refunds collected taxes.

Parameters

<i>amount</i>	The amount of taxes to refund.
---------------	--------------------------------

4.29.3.8 `registerObserver()`

```
void Government::registerObserver (   
    GovObserver * observer )
```

Registers an observer to the government.

Parameters

<i>observer</i>	The observer to register.
-----------------	---------------------------

4.29.3.9 `revertBudgetAllocation()`

```
void Government::revertBudgetAllocation (   
    CityService & service,  
    double amount )
```

Reverts the budget allocation to a city service.

Parameters

<i>service</i>	The city service to revert the budget allocation from.
<i>amount</i>	The amount to revert.

4.29.3.10 setTax()

```
void Government::setTax (
    double rate )
```

Sets the tax rate for the government.

Parameters

<i>rate</i>	The new tax rate.
-------------	-------------------

4.29.3.11 unregisterObserver()

```
void Government::unregisterObserver (
    GovObserver * observer )
```

Unregisters an observer from the government.

Parameters

<i>observer</i>	The observer to unregister.
-----------------	-----------------------------

4.29.3.12 update()

```
void Government::update (
    int newPopulation )
```

Updates the population of the government.

Parameters

<i>newPopulation</i>	The new population value.
----------------------	---------------------------

The documentation for this class was generated from the following files:

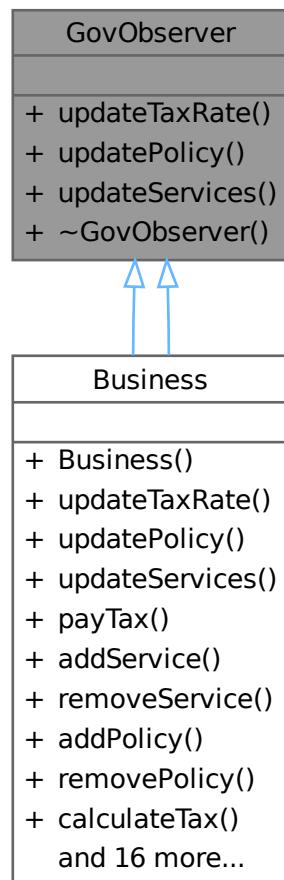
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Government.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Government.cpp](#)

4.30 GovObserver Class Reference

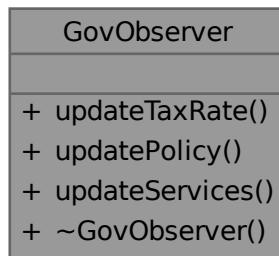
Abstract base class for observers that monitor government changes.

```
#include <GovObserver.h>
```

Inheritance diagram for GovObserver:



Collaboration diagram for GovObserver:



Public Member Functions

- virtual void `updateTaxRate` (double rate)=0
Updates the tax rate for the observer.
- virtual void `updatePolicy` (`Policy` policy)=0
Updates the policy for the observer.
- virtual void `updateServices` (`CityService` services)=0
Updates the services for the observer.
- virtual `~GovObserver` ()=default
Virtual destructor.

4.30.1 Detailed Description

Abstract base class for observers that monitor government changes.

This class provides methods to update the tax rate, policy, and services for the observer.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 `~GovObserver()`

```
virtual GovObserver::~GovObserver ( ) [virtual], [default]
```

Virtual destructor.

Ensures derived class destructors are called correctly.

4.30.3 Member Function Documentation

4.30.3.1 `updatePolicy()`

```
void GovObserver::updatePolicy (
    Policy policy ) [pure virtual]
```

Updates the policy for the observer.

Parameters

<i>policy</i>	The new policy.
<i>policy</i>	The new policy.

Prints a message indicating the policy update.

Implemented in [Business](#), and [Business](#).

4.30.3.2 updateServices()

```
void GovObserver::updateServices (
    CityService services ) [pure virtual]
```

Updates the services for the observer.

Parameters

<i>services</i>	The new services.
<i>services</i>	The new services.

Prints a message indicating the services update.

Implemented in [Business](#), and [Business](#).

4.30.3.3 updateTaxRate()

```
void GovObserver::updateTaxRate (
    double rate ) [pure virtual]
```

Updates the tax rate for the observer.

Parameters

<i>rate</i>	The new tax rate.
<i>rate</i>	The new tax rate.

Prints the updated tax rate to the console.

Implemented in [Business](#), and [Business](#).

The documentation for this class was generated from the following files:

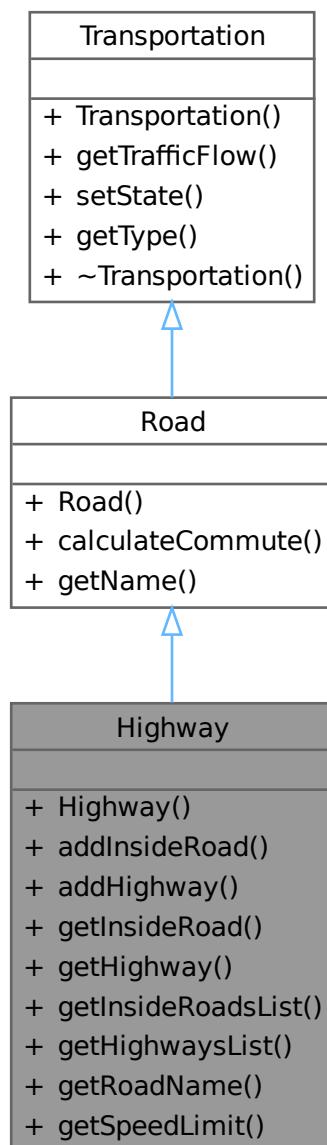
- /mnt/c/users/rudie/documents/sem2/214/project/src/[GovObserver.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[GovObserver.cpp](#)

4.31 Highway Class Reference

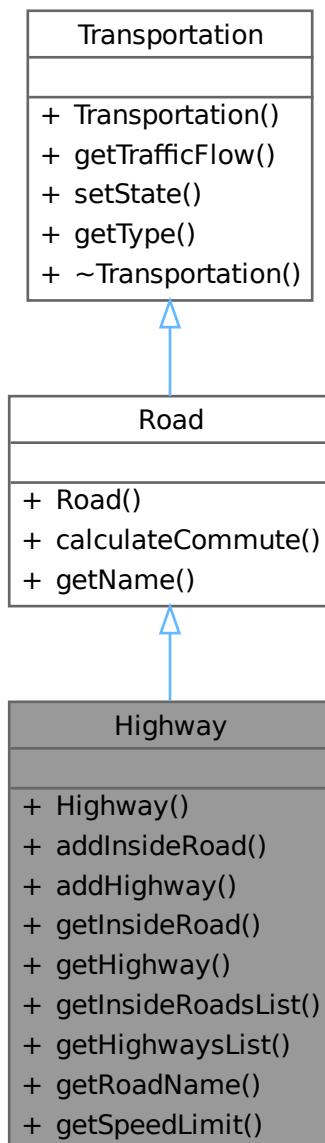
Represents a highway with a speed limit.

```
#include <Highway.h>
```

Inheritance diagram for Highway:



Collaboration diagram for Highway:



Public Member Functions

- `Highway (char state, std::string roadName, float speedLimit)`
Constructs a `Highway` object.
- `bool addInsideRoad (InsideRoad *insideRoad)`
Adds an inside road to the highway.
- `bool addHighway (Highway *highway)`
Adds another highway to this highway.
- `InsideRoad * getInsideRoad (std::size_t x)`
Gets an inside road by index.

- `Highway * getHighway (std::size_t x)`
Gets a highway by index.
- `std::vector< InsideRoad * > getInsideRoadsList ()`
Gets the list of inside roads.
- `std::vector< Highway * > getHighwaysList ()`
Gets the list of highways.
- `std::string getRoadName ()`
Gets the name of the highway.
- `float getSpeedLimit ()`
Gets the speed limit of the highway.

Public Member Functions inherited from [Road](#)

- `Road (char state, std::string roadName, char type)`
Constructs a new [Road](#) object.
- `float calculateCommute ()`
Calculates the commute time on the road.
- `std::string getName ()`
Gets the name of the road.

Public Member Functions inherited from [Transportation](#)

- `Transportation (char state, char type)`
Constructor for the [Transportation](#) class.
- `float getTrafficFlow ()`
Gets the current traffic flow.
- `bool setState (char state)`
Sets the traffic flow state.
- `char getType ()`
Gets the type of transportation.
- `~Transportation ()`
Destructor for the [Transportation](#) class.

4.31.1 Detailed Description

Represents a highway with a speed limit.

The [Highway](#) class inherits from the [Road](#) class and includes additional attributes and methods specific to highways, such as a speed limit and lists of inside roads and other highways.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 [Highway\(\)](#)

```
Highway::Highway (
    char state,
    std::string roadName,
    float speedLimit )
```

Constructs a [Highway](#) object.

Constructor for [Highway](#).

Parameters

<i>state</i>	The state of the highway.
<i>roadName</i>	The name of the highway.
<i>speedLimit</i>	The speed limit of the highway.

4.31.3 Member Function Documentation**4.31.3.1 addHighway()**

```
bool Highway::addHighway (
    Highway * highway )
```

Adds another highway to this highway.

Adds another [Highway](#) to the highway.

Parameters

<i>highway</i>	Pointer to the highway to be added.
----------------	-------------------------------------

Returns

true if the highway was added successfully, false otherwise.

Parameters

<i>highway</i>	Pointer to the Highway to be added.
----------------	---

Returns

True if the [Highway](#) was added successfully, false if it already exists.

4.31.3.2 addInsideRoad()

```
bool Highway::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an inside road to the highway.

Adds an [InsideRoad](#) to the highway.

Parameters

<i>insideRoad</i>	Pointer to the inside road to be added.
-------------------	---

Returns

true if the inside road was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad to be added.
-------------------	--

Returns

True if the [InsideRoad](#) was added successfully, false if it already exists.

4.31.3.3 getHighway()

```
Highway * Highway::getHighway (
```

```
    std::size_t x )
```

Gets a highway by index.

Gets a [Highway](#) by index.

Parameters

<i>x</i>	The index of the highway.
----------	---------------------------

Returns

Pointer to the highway at the specified index.

Parameters

<i>x</i>	The index of the Highway .
----------	--

Returns

Pointer to the [Highway](#) if the index is valid, nullptr otherwise.

4.31.3.4 getHighwaysList()

```
std::vector< Highway * > Highway::getHighwaysList ( )
```

Gets the list of highways.

Gets the list of Highways.

Returns

Vector of pointers to the highways.

A vector of pointers to Highways.

4.31.3.5 `getInsideRoad()`

```
InsideRoad * Highway::getInsideRoad ( std::size_t x )
```

Gets an inside road by index.

Gets an [InsideRoad](#) by index.

Parameters

x	The index of the inside road.
---	-------------------------------

Returns

Pointer to the inside road at the specified index.

Parameters

x	The index of the InsideRoad .
---	---

Returns

Pointer to the [InsideRoad](#) if the index is valid, nullptr otherwise.

4.31.3.6 `getInsideRoadsList()`

```
std::vector< InsideRoad * > Highway::getInsideRoadsList ( )
```

Gets the list of inside roads.

Gets the list of InsideRoads.

Returns

Vector of pointers to the inside roads.

A vector of pointers to InsideRoads.

4.31.3.7 `getRoadName()`

```
std::string Highway::getRoadName ( )
```

Gets the name of the highway.

Returns

The name of the highway.

4.31.3.8 getSpeedLimit()

```
float Highway::getSpeedLimit ( )
```

Gets the speed limit of the highway.

Returns

The speed limit of the highway.

The documentation for this class was generated from the following files:

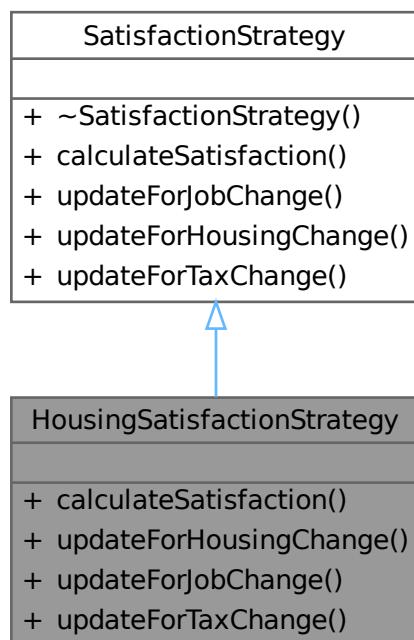
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Highway.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Highway.cpp](#)

4.32 HousingSatisfactionStrategy Class Reference

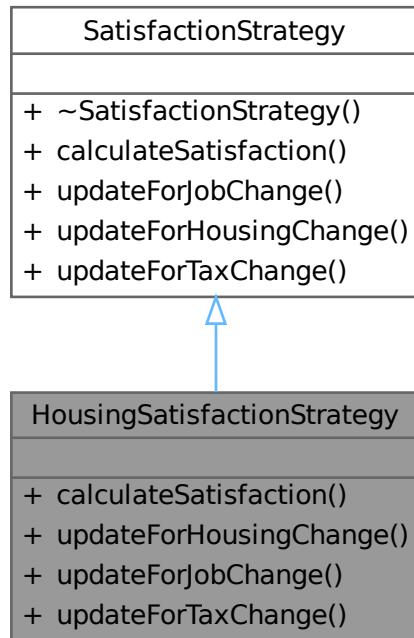
Strategy for calculating and updating citizen satisfaction based on housing conditions.

```
#include <HousingSatisfactionStrategy.h>
```

Inheritance diagram for HousingSatisfactionStrategy:



Collaboration diagram for `HousingSatisfactionStrategy`:



Public Member Functions

- float `calculateSatisfaction` (const `Citizen` &citizen) override
Calculates the satisfaction level of a citizen based on housing conditions.
- void `updateForHousingChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in housing conditions.
- void `updateForJobChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in job conditions.
- void `updateForTaxChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in tax conditions.

Public Member Functions inherited from `SatisfactionStrategy`

- virtual ~`SatisfactionStrategy` ()=default
Virtual destructor for `SatisfactionStrategy`.

4.32.1 Detailed Description

Strategy for calculating and updating citizen satisfaction based on housing conditions.

The `HousingSatisfactionStrategy` class provides methods to calculate and update citizen satisfaction based on housing conditions.

4.32.2 Member Function Documentation

4.32.2.1 calculateSatisfaction()

```
float HousingSatisfactionStrategy::calculateSatisfaction (
    const Citizen & citizen ) [override], [virtual]
```

Calculates the satisfaction level of a citizen based on housing conditions.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be calculated.
----------------	---

Returns

The calculated satisfaction level.

Implements [SatisfactionStrategy](#).

4.32.2.2 updateForHousingChange()

```
void HousingSatisfactionStrategy::updateForHousingChange (
    Citizen & citizen ) [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in housing conditions.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.32.2.3 updateForJobChange()

```
void HousingSatisfactionStrategy::updateForJobChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in job conditions.

This method is not implemented for housing satisfaction strategy.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.32.2.4 updateForTaxChange()

```
void HousingSatisfactionStrategy::updateForTaxChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in tax conditions.

This method is not implemented for housing satisfaction strategy.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

The documentation for this class was generated from the following files:

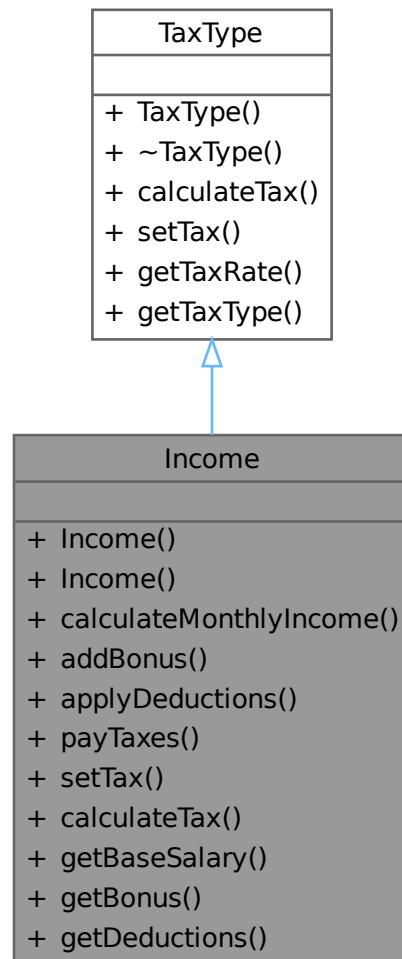
- /mnt/c/users/rudie/documents/sem2/214/project/src/[HousingSatisfactionStrategy.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[HousingSatisfactionStrategy.cpp](#)

4.33 Income Class Reference

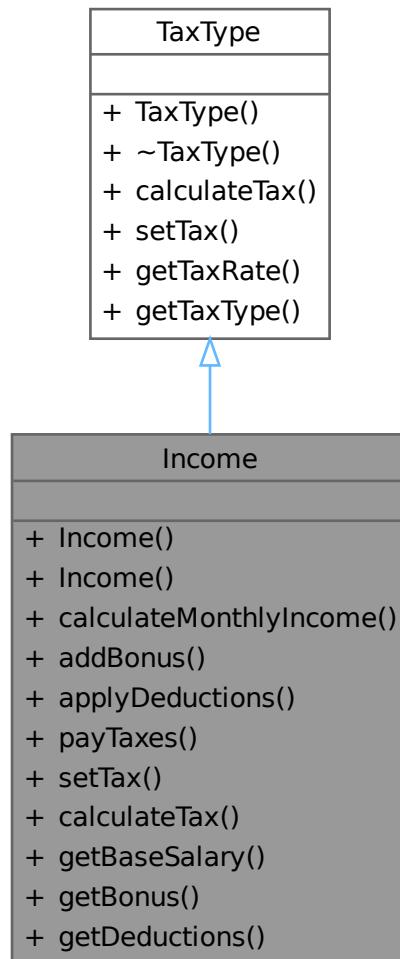
Manages income-related operations.

```
#include <Income.h>
```

Inheritance diagram for Income:



Collaboration diagram for Income:



Public Member Functions

- **Income** (double salary, double rate, const char category)
Constructor to initialize with base salary, tax rate, and category.
- **Income** (double salary)
Constructor to initialize with base salary.
- double **calculateMonthlyIncome** () const
Calculates the monthly income considering base salary, bonuses, and deductions.
- void **addBonus** (double amount)
Adds a bonus to the income for the current month.
- void **applyDeductions** (double amount)
Applies deductions for the current month.
- double **payTaxes** (`TaxType` &taxType)
Calculates and pays taxes based on the tax type.
- void **setTax** (double rate) override

- `double calculateTax (double val) override`
Calculates the tax based on a given value.
- `double getBaseSalary () const`
Gets the base salary.
- `double getBonus () const`
Gets the bonus.
- `double getDeductions () const`
Gets the deductions.

Public Member Functions inherited from `TaxType`

- `TaxType (double rate, char type)`
Constructs a new `TaxType` object.
- `virtual ~TaxType ()`
Virtual Destructor.
- `virtual double getTaxRate ()`
Gets the current tax rate.
- `char getTaxType ()`
Gets the tax type identifier.

4.33.1 Detailed Description

Manages income-related operations.

The `Income` class provides methods to calculate monthly income, add bonuses, apply deductions, and calculate taxes.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 `Income()` [1/2]

```
Income::Income (
    double salary,
    double rate,
    const char category ) [inline]
```

Constructor to initialize with base salary, tax rate, and category.

Parameters

<code>salary</code>	The base salary.
<code>rate</code>	The tax rate.
<code>category</code>	The category type.

4.33.2.2 Income() [2/2]

```
Income::Income (
    double salary ) [inline]
```

Constructor to initialize with base salary.

Parameters

<i>salary</i>	The base salary.
---------------	------------------

4.33.3 Member Function Documentation

4.33.3.1 addBonus()

```
void Income::addBonus (
    double amount )
```

Adds a bonus to the income for the current month.

Parameters

<i>amount</i>	The amount of the bonus to add.
---------------	---------------------------------

4.33.3.2 applyDeductions()

```
void Income::applyDeductions (
    double amount )
```

Applies deductions for the current month.

Parameters

<i>amount</i>	The amount of deductions to apply.
---------------	------------------------------------

4.33.3.3 calculateMonthlyIncome()

```
double Income::calculateMonthlyIncome ( ) const
```

Calculates the monthly income considering base salary, bonuses, and deductions.

Returns

The calculated monthly income.

4.33.3.4 calculateTax()

```
double Income::calculateTax (
    double val ) [override], [virtual]
```

Calculates the tax based on a given value.

Parameters

<i>val</i>	The value to calculate the tax on.
------------	------------------------------------

Returns

The calculated tax.

Reimplemented from [TaxType](#).

4.33.3.5 getBaseSalary()

```
double Income::getBaseSalary ( ) const [inline]
```

Gets the base salary.

Returns

The base salary.

4.33.3.6 getBonus()

```
double Income::getBonus ( ) const [inline]
```

Gets the bonus.

Returns

The bonus.

4.33.3.7 getDeductions()

```
double Income::getDeductions ( ) const [inline]
```

Gets the deductions.

Returns

The deductions.

4.33.3.8 payTaxes()

```
double Income::payTaxes (
    TaxType & taxType )
```

Calculates and pays taxes based on the tax type.

Parameters

<i>taxType</i>	The type of tax to calculate.
----------------	-------------------------------

Returns

The amount of taxes paid.

4.33.3.9 setTax()

```
void Income::setTax ( double taxRate ) [override], [virtual]
```

Sets the tax rate for the income.

Parameters

<i>rate</i>	The tax rate to set.
<i>taxRate</i>	The tax rate to set.

Reimplemented from [TaxType](#).

The documentation for this class was generated from the following files:

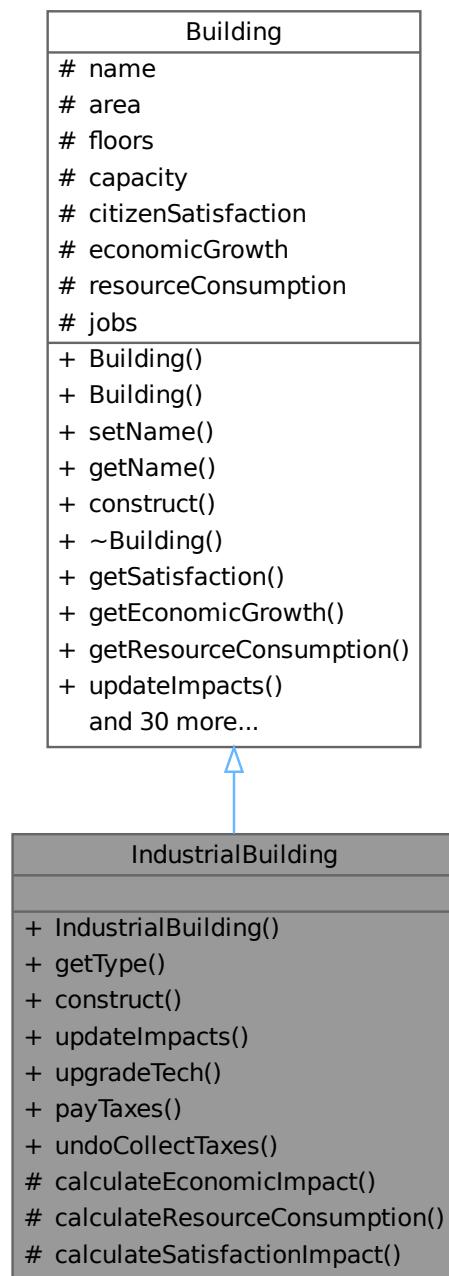
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Income.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Income.cpp](#)

4.34 IndustrialBuilding Class Reference

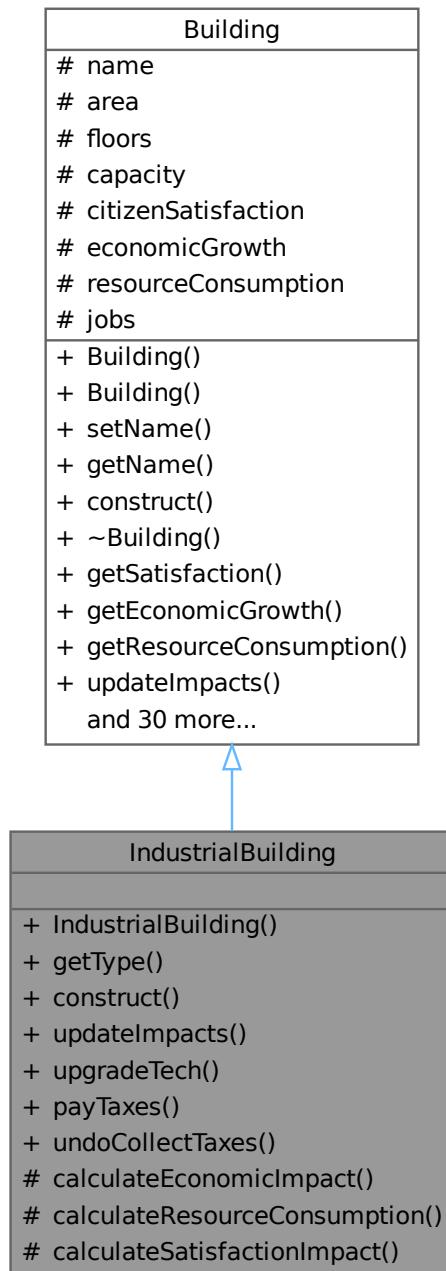
Represents an industrial building with specific attributes and behaviors.

```
#include <IndustrialBuilding.h>
```

Inheritance diagram for IndustrialBuilding:



Collaboration diagram for IndustrialBuilding:



Public Member Functions

- `IndustrialBuilding (const std::string &name, float area, int floors, int capacity, float citizenSatisfaction, float economicGrowth, float resourceConsumption, float pollutionLevel, float productionCapacity)`
Constructor for IndustrialBuilding.
- string `getType ()` const override
Gets the type of the building.

- void **construct** () override
Constructs the building.
- void **updateImpacts** () override
Updates the impacts of the building by calculating economic impact, resource consumption, and satisfaction impact.
- void **upgradeTech** (float techLevel)
Upgrades the technology level of the building.
- double **payTaxes** (TaxType *taxType)
Calculates and pays taxes for the building.
- void **undoCollectTaxes** ()
Undoes the tax collection from the building.

Public Member Functions inherited from [Building](#)

- **Building** (const std::string &**name**, float **area**, int **floors**, int **capacity**, float **satisfactionImpact**, float **growthImpact**, float **consumption**)
Main constructor for setting attributes directly.
- **Building** (int **Builder**)
Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.
- void **setName** (const std::string &**name**)
Sets the name of the building.
- std::string **getName** () const
Gets the name of the building.
- virtual ~**Building** ()=default
Virtual destructor for the [Building](#) class.
- float **getSatisfaction** () const
Gets the citizen satisfaction impact of the building.
- float **getEconomicGrowth** () const
Gets the economic growth impact of the building.
- float **getResourceConsumption** () const
Gets the resource consumption of the building.
- void **addJob** (std::shared_ptr< [Jobs](#) > job)
Adds a job to the building.
- void **listJobs** () const
Lists all jobs in the building.
- bool **hireEmployee** (const std::string &**jobTitle**)
Hires an employee for a job if available.
- void **releaseEmployee** (const std::string &**jobTitle**)
Releases an employee from a job.
- std::shared_ptr< [Jobs](#) > **getAvailableJob** ()
Provides access to an available job.
- void **displayJobInfo** (const std::string &**jobTitle**) const
Displays job information.
- const std::vector< std::shared_ptr< [Jobs](#) > > & **getJobs** () const
Returns the job list as a const reference.
- **Building** (const std::string &**name**, float **area**, int **floors**, int **capacity**, float **satisfactionImpact**, float **growthImpact**, float **consumption**)
Main constructor for setting attributes directly.
- **Building** (int **Builder**)
Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.
- void **setName** (const std::string &**name**)

- std::string **getName** () const

Sets the name of the building.
- virtual ~**Building** ()=default

Virtual destructor for the [Building](#) class.
- float **getSatisfaction** () const

Gets the citizen satisfaction impact of the building.
- float **getEconomicGrowth** () const

Gets the economic growth impact of the building.
- float **getResourceConsumption** () const

Gets the resource consumption of the building.
- void **addJob** (std::shared_ptr< [Jobs](#) > job)

Adds a job to the building.
- void **listJobs** () const

Lists all jobs in the building.
- bool **hireEmployee** (const std::string &jobTitle)

Hires an employee for a job if available.
- void **releaseEmployee** (const std::string &jobTitle)

Releases an employee from a job.
- std::shared_ptr< [Jobs](#) > **getAvailableJob** ()

Provides access to an available job.
- void **displayJobInfo** (const std::string &jobTitle) const

Displays job information.
- const std::vector< std::shared_ptr< [Jobs](#) > > & **getJobs** () const

Returns the job list as a const reference.

Protected Member Functions

- void **calculateEconomicImpact** ()

Calculates the economic impact of the building.
- void **calculateResourceConsumption** ()

Calculates the resource consumption of the building.
- void **calculateSatisfactionImpact** ()

Calculates the satisfaction impact of the building.

Additional Inherited Members

Protected Attributes inherited from [Building](#)

- std::string **name**

Name of the building.
- float **area**

Area of the building.
- int **floors**

Number of floors in the building.
- int **capacity**

Capacity of the building.
- float **citizenSatisfaction**

Citizen satisfaction impact of the building.

- float **economicGrowth**
Economic growth impact of the building.
- float **resourceConsumption**
Resource consumption of the building.
- std::vector< std::shared_ptr< [Jobs](#) >> **jobs**
Collection of jobs as shared pointers.

4.34.1 Detailed Description

Represents an industrial building with specific attributes and behaviors.

The [IndustrialBuilding](#) class inherits from the [Building](#) class and provides specific implementations for an industrial building, including methods to calculate impacts, upgrade technology, and manage taxes.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 [IndustrialBuilding\(\)](#)

```
IndustrialBuilding::IndustrialBuilding (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float citizenSatisfaction,
    float economicGrowth,
    float resourceConsumption,
    float pollutionLevel,
    float productionCapacity )
```

Constructor for [IndustrialBuilding](#).

Parameters

<i>name</i>	The name of the industrial building.
<i>area</i>	The area of the industrial building.
<i>floors</i>	The number of floors in the industrial building.
<i>capacity</i>	The capacity of the industrial building.
<i>citizenSatisfaction</i>	The citizen satisfaction level.
<i>economicGrowth</i>	The economic growth contributed by the building.
<i>resourceConsumption</i>	The resource consumption of the building.
<i>pollutionLevel</i>	The pollution level of the building.
<i>productionCapacity</i>	The production capacity of the building.

4.34.3 Member Function Documentation

4.34.3.1 [calculateEconomicImpact\(\)](#)

```
void IndustrialBuilding::calculateEconomicImpact ( ) [protected], [virtual]
```

Calculates the economic impact of the building.

Implements [Building](#).

4.34.3.2 calculateResourceConsumption()

```
void IndustrialBuilding::calculateResourceConsumption () [protected], [virtual]
```

Calculates the resource consumption of the building.

Implements [Building](#).

4.34.3.3 calculateSatisfactionImpact()

```
void IndustrialBuilding::calculateSatisfactionImpact () [protected], [virtual]
```

Calculates the satisfaction impact of the building.

Implements [Building](#).

4.34.3.4 construct()

```
void IndustrialBuilding::construct () [override], [virtual]
```

Constructs the building.

Implements [Building](#).

4.34.3.5 getType()

```
std::string IndustrialBuilding::getType () const [override], [virtual]
```

Gets the type of the building.

Returns

The type of the building.

Implements [Building](#).

4.34.3.6 payTaxes()

```
double IndustrialBuilding::payTaxes (
    TaxType * taxType ) [virtual]
```

Calculates and pays taxes for the building.

Pays taxes for the building.

Parameters

<i>taxType</i>	The type of tax to calculate.
----------------	-------------------------------

Returns

The amount of taxes paid.

Implements [Building](#).

4.34.3.7 undoCollectTaxes()

```
void IndustrialBuilding::undoCollectTaxes ( ) [virtual]
```

Undoes the tax collection from the building.

Implements [Building](#).

4.34.3.8 updateImpacts()

```
void IndustrialBuilding::updateImpacts ( ) [override], [virtual]
```

Updates the impacts of the building by calculating economic impact, resource consumption, and satisfaction impact.

Implements [Building](#).

4.34.3.9 upgradeTech()

```
void IndustrialBuilding::upgradeTech ( float techLevel )
```

Upgrades the technology level of the building.

Parameters

<i>techLevel</i>	The new technology level to upgrade to.
------------------	---

The documentation for this class was generated from the following files:

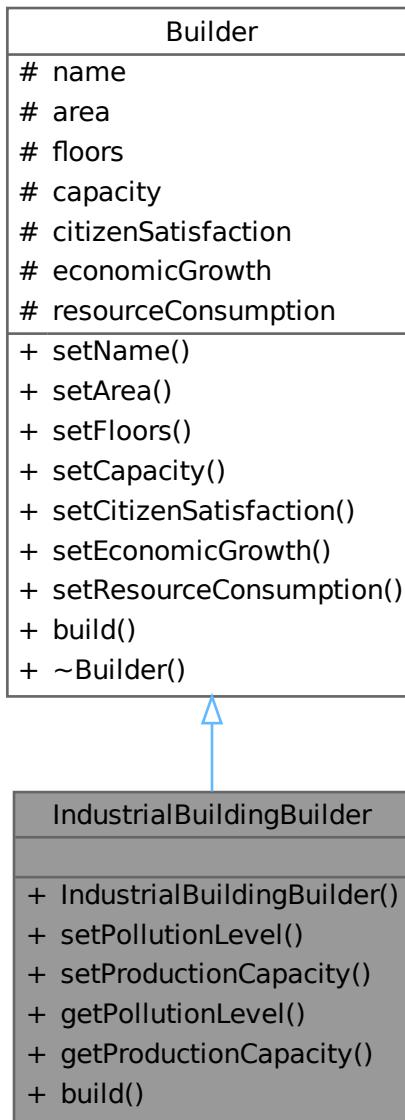
- /mnt/c/users/rudie/documents/sem2/214/project/src/[IndustrialBuilding.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[IndustrialBuilding.cpp](#)

4.35 IndustrialBuildingBuilder Class Reference

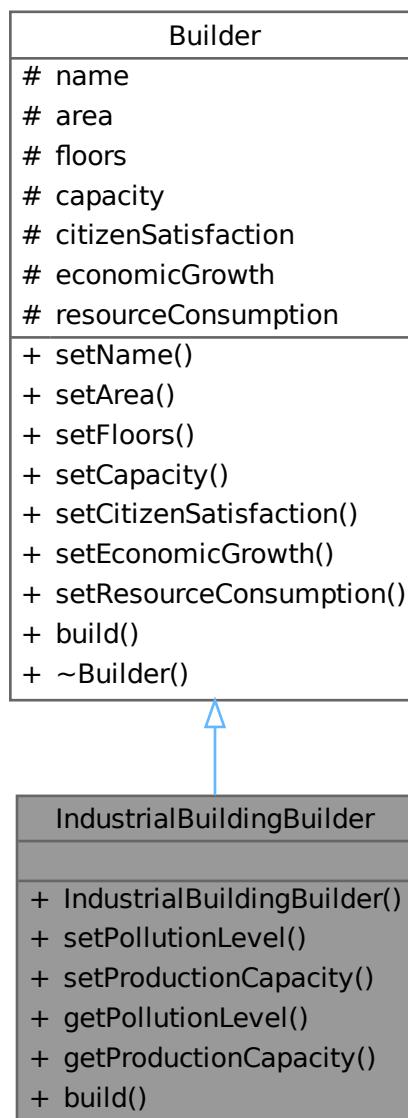
[Builder](#) class for constructing [IndustrialBuilding](#) objects.

```
#include <IndustrialBuildingBuilder.h>
```

Inheritance diagram for IndustrialBuildingBuilder:



Collaboration diagram for IndustrialBuildingBuilder:



Public Member Functions

- **IndustrialBuildingBuilder ()**
Constructor for IndustrialBuildingBuilder.
- **IndustrialBuildingBuilder & setPollutionLevel (float level)**
Sets the pollution level of the building.
- **IndustrialBuildingBuilder & setProductionCapacity (float capacity)**
Sets the production capacity of the building.
- **float getPollutionLevel ()**
Gets the pollution level of the building.

- float `getProductionCapacity ()`
Gets the production capacity of the building.
- std::unique_ptr< Building > `build ()` override
Builds the industrial building and sets all the attributes of the building.

Public Member Functions inherited from [Builder](#)

- `Builder & setName (string name)`
Sets the name of the building.
- `Builder & setArea (float area)`
Sets the area of the building.
- `Builder & setFloors (int floors)`
Sets the number of floors in the building.
- `Builder & setCapacity (int capacity)`
Sets the capacity of the building.
- `Builder & setCitizenSatisfaction (float citizenSatisfaction)`
Sets the citizen satisfaction of the building.
- `Builder & setEconomicGrowth (float economicGrowth)`
Sets the economic growth of the building.
- `Builder & setResourceConsumption (float resourceConsumption)`
Sets the resource consumption of the building.
- virtual ~`Builder ()`=default
Virtual destructor for the `Builder` class.

Additional Inherited Members

Protected Attributes inherited from [Builder](#)

- string `name`
Name of the building.
- float `area` = 0.0f
Area of the building.
- int `floors` = 0
Number of floors in the building.
- int `capacity` = 0
Capacity of the building.
- float `citizenSatisfaction` = 0.0f
Citizen satisfaction of the building.
- float `economicGrowth` = 0.0f
Economic growth of the building.
- float `resourceConsumption` = 0.0f
Resource consumption of the building.

4.35.1 Detailed Description

[Builder](#) class for constructing [IndustrialBuilding](#) objects.

The [IndustrialBuildingBuilder](#) class provides methods to set attributes and build an [IndustrialBuilding](#) object.

4.35.2 Member Function Documentation

4.35.2.1 build()

```
std::unique_ptr< Building > IndustrialBuildingBuilder::build ( ) [override], [virtual]
```

Builds the industrial building and sets all the attributes of the building.

Returns

A unique pointer to the constructed [IndustrialBuilding](#) object.

Implements [Builder](#).

4.35.2.2 getPollutionLevel()

```
float IndustrialBuildingBuilder::getPollutionLevel ( )
```

Gets the pollution level of the building.

Returns

The pollution level of the building.

4.35.2.3 getProductionCapacity()

```
float IndustrialBuildingBuilder::getProductionCapacity ( )
```

Gets the production capacity of the building.

Returns

The production capacity of the building.

4.35.2.4 setPollutionLevel()

```
IndustrialBuildingBuilder & IndustrialBuildingBuilder::setPollutionLevel ( float level )
```

Sets the pollution level of the building.

Parameters

<i>level</i>	The pollution level to set.
--------------	-----------------------------

Returns

Reference to the current [IndustrialBuildingBuilder](#) object.

4.35.2.5 **setProductionCapacity()**

```
IndustrialBuildingBuilder & IndustrialBuildingBuilder::setProductionCapacity (
    float capacity )
```

Sets the production capacity of the building.

Parameters

<i>capacity</i>	The production capacity to set.
-----------------	---------------------------------

Returns

Reference to the current [IndustrialBuildingBuilder](#) object.

The documentation for this class was generated from the following files:

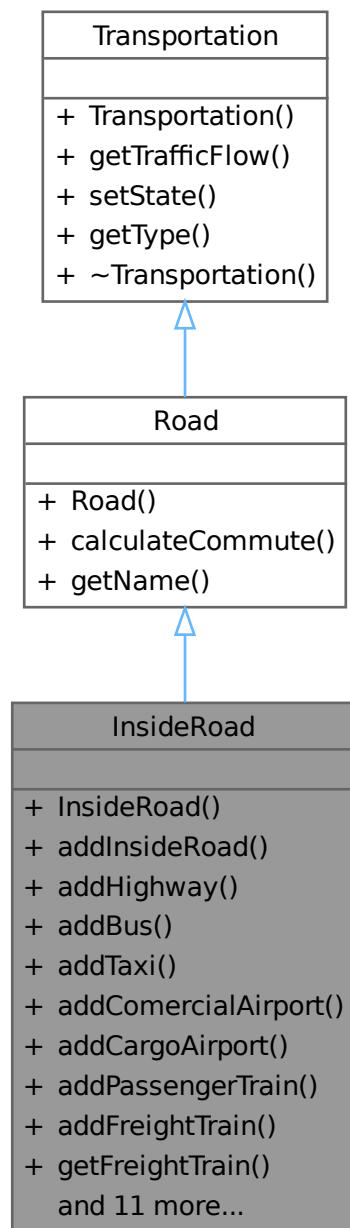
- /mnt/c/users/rudie/documents/sem2/214/project/src/[IndustrialBuildingBuilder.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[IndustrialBuildingBuilder.cpp](#)

4.36 InsideRoad Class Reference

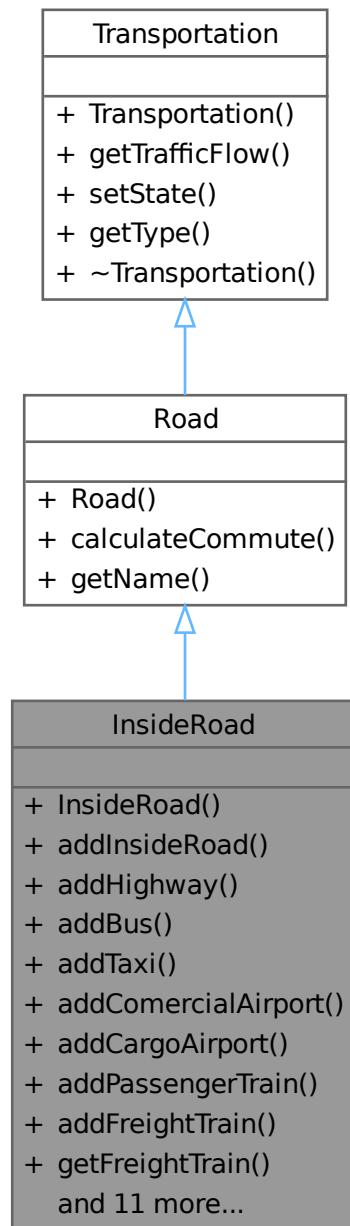
Represents an inside road that can contain various transportation entities.

```
#include <InsideRoad.h>
```

Inheritance diagram for InsideRoad:



Collaboration diagram for InsideRoad:



Public Member Functions

- `InsideRoad` (char state, std::string roadName, float avgStopTime)
Constructor for the `InsideRoad` class.
- bool `addInsideRoad` (`InsideRoad` *insideRoad)
Adds an inside road to this road.
- bool `addHighway` (`Highway` *highway)

- `bool addBus (Bus *bus)`
Adds a highway to this road.
- `bool addTaxi (Taxi *taxi)`
Adds a bus to this road.
- `bool addComercialAirport (ComercialAirport *comercialAirport)`
Adds a taxi to this road.
- `bool addCargoAirport (CargoAirport *cargoAirport)`
Adds a commercial airport to this road.
- `bool addPassengerTrain (PassengerTrain *passengerTrain)`
Adds a cargo airport to this road.
- `bool addFreightTrain (FreightTrain *freightTrain)`
Adds a passenger train to this road.
- `FreightTrain * getFreightTrain (std::size_t x)`
Adds a freight train to this road.
- `bool addBuilding (Building *building)`
Gets a freight train from the list of freight trains.
- `PassengerTrain * getPassengerTrain (std::size_t x)`
Adds a building to the road.
- `Highway * getHighway (std::size_t x)`
Gets a passenger train from the list of passenger trains.
- `InsideRoad * getInsideRoad (std::size_t x)`
Gets a highway from the list of highways.
- `Bus * getBus (std::size_t x)`
Gets an inside road from the list of inside roads.
- `Taxi * getTaxi (std::size_t x)`
Gets a bus from the list of buses.
- `ComercialAirport * getComercialAirport (std::size_t x)`
Gets a taxi from the list of taxis.
- `CargoAirport * getCargoAirport (std::size_t x)`
Gets a commercial airport from the list of commercial airports.
- `Building * getBuilding (std::size_t x)`
Gets a cargo airport from the list of cargo airports.
- `float getAvgStopTime ()`
Gets a building from the list of buildings.
- `std::string getRoadName ()`
Gets the average stop time on this road.
- `Gets the name of this road.`

Public Member Functions inherited from Road

- `Road (char state, std::string roadName, char type)`
Constructs a new Road object.
- `float calculateCommute ()`
Calculates the commute time on the road.
- `std::string getName ()`
Gets the name of the road.

Public Member Functions inherited from [Transportation](#)

- [Transportation](#) (char state, char type)
Constructor for the [Transportation](#) class.
- float [getTrafficFlow](#) ()
Gets the current traffic flow.
- bool [setState](#) (char state)
Sets the traffic flow state.
- char [getType](#) ()
Gets the type of transportation.
- [~Transportation](#) ()
Destructor for the [Transportation](#) class.

4.36.1 Detailed Description

Represents an inside road that can contain various transportation entities.

The [InsideRoad](#) class inherits from the [Road](#) class and adds functionality to manage different types of transportation entities such as buses, taxis, airports, and trains.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 [InsideRoad\(\)](#)

```
InsideRoad::InsideRoad (
    char state,
    std::string roadName,
    float avgStopTime )
```

Constructor for the [InsideRoad](#) class.

Constructor for [InsideRoad](#).

Parameters

<i>state</i>	The state of the road.
<i>roadName</i>	The name of the road.
<i>avgStopTime</i>	The average stop time on this road.
<i>state</i>	The state of the inside road.
<i>roadName</i>	The name of the inside road.
<i>avgStopTime</i>	The average stop time on the inside road.

4.36.3 Member Function Documentation

4.36.3.1 [addBuilding\(\)](#)

```
bool InsideRoad::addBuilding (
    Building * building )
```

Adds a building to the road.

Adds a [Building](#) to the inside road.

This function adds a building to the road's list of buildings.

Parameters

<i>building</i>	Pointer to the building to be added.
-----------------	--------------------------------------

Returns

true if the building was added successfully, false otherwise.

Parameters

<i>building</i>	Pointer to the Building to be added.
-----------------	--

Returns

True if the [Building](#) was added successfully, false if it already exists.

4.36.3.2 addBus()

```
bool InsideRoad::addBus (
    Bus * bus )
```

Adds a bus to this road.

Adds a [Bus](#) to the inside road.

Parameters

<i>bus</i>	Pointer to the bus to be added.
------------	---------------------------------

Returns

True if the bus was added successfully, false otherwise.

Parameters

<i>bus</i>	Pointer to the Bus to be added.
------------	---

Returns

True if the [Bus](#) was added successfully, false if it already exists.

4.36.3.3 addCargoAirport()

```
bool InsideRoad::addCargoAirport (
    CargoAirport * cargoAirport )
```

Adds a cargo airport to this road.

Adds a [CargoAirport](#) to the inside road.

Parameters

<i>cargoAirport</i>	Pointer to the cargo airport to be added.
---------------------	---

Returns

True if the cargo airport was added successfully, false otherwise.

Parameters

<i>cargoAirport</i>	Pointer to the CargoAirport to be added.
---------------------	--

Returns

True if the [CargoAirport](#) was added successfully, false if it already exists.

4.36.3.4 addComercialAirport()

```
bool InsideRoad::addComercialAirport (
    ComercialAirport * comercialAirport )
```

Adds a commercial airport to this road.

Adds a [ComercialAirport](#) to the inside road.

Parameters

<i>comercialAirport</i>	Pointer to the commercial airport to be added.
-------------------------	--

Returns

True if the commercial airport was added successfully, false otherwise.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport to be added.
-------------------------	--

Returns

True if the [ComercialAirport](#) was added successfully, false if it already exists.

4.36.3.5 addFreightTrain()

```
bool InsideRoad::addFreightTrain (  
    FreightTrain * freightTrain )
```

Adds a freight train to this road.

Adds a [FreightTrain](#) to the inside road.

Parameters

<i>freightTrain</i>	Pointer to the freight train to be added.
---------------------	---

Returns

True if the freight train was added successfully, false otherwise.

Parameters

<i>freightTrain</i>	Pointer to the FreightTrain to be added.
---------------------	--

Returns

True if the [FreightTrain](#) was added successfully, false if it already exists.

4.36.3.6 addHighway()

```
bool InsideRoad::addHighway (  
    Highway * highway )
```

Adds a highway to this road.

Adds a [Highway](#) to the inside road.

Parameters

<i>highway</i>	Pointer to the highway to be added.
----------------	-------------------------------------

Returns

True if the highway was added successfully, false otherwise.

Parameters

<i>highway</i>	Pointer to the Highway to be added.
----------------	---

Returns

True if the [Highway](#) was added successfully, false if it already exists.

4.36.3.7 addInsideRoad()

```
bool InsideRoad::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an inside road to this road.

Adds an [InsideRoad](#) to the inside road.

Parameters

<i>insideRoad</i>	Pointer to the inside road to be added.
-------------------	---

Returns

True if the inside road was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad to be added.
-------------------	--

Returns

True if the [InsideRoad](#) was added successfully, false if it already exists.

4.36.3.8 addPassengerTrain()

```
bool InsideRoad::addPassengerTrain (
    PassengerTrain * passengerTrain )
```

Adds a passenger train to this road.

Adds a [PassengerTrain](#) to the inside road.

Parameters

<i>passengerTrain</i>	Pointer to the passenger train to be added.
-----------------------	---

Returns

True if the passenger train was added successfully, false otherwise.

Parameters

<i>passengerTrain</i>	Pointer to the PassengerTrain to be added.
-----------------------	--

Returns

True if the [PassengerTrain](#) was added successfully, false if it already exists.

4.36.3.9 addTaxi()

```
bool InsideRoad::addTaxi (   
    Taxi * taxi )
```

Adds a taxi to this road.

Adds a [Taxi](#) to the inside road.

Parameters

<i>taxi</i>	Pointer to the taxi to be added.
-------------	----------------------------------

Returns

True if the taxi was added successfully, false otherwise.

Parameters

<i>taxi</i>	Pointer to the Taxi to be added.
-------------	--

Returns

True if the [Taxi](#) was added successfully, false if it already exists.

4.36.3.10 getAvgStopTime()

```
float InsideRoad::getAvgStopTime ( )
```

Gets the average stop time on this road.

Gets the average stop time on the inside road.

Returns

The average stop time.

4.36.3.11 getBuilding()

```
Building * InsideRoad::getBuilding ( std::size_t x )
```

Gets a building from the list of buildings.

Gets a [Building](#) by index.

Parameters

x	Index of the building to be retrieved.
---	--

Returns

Pointer to the building at the specified index.

Parameters

x	The index of the Building .
---	---

Returns

Pointer to the [Building](#) if the index is valid, nullptr otherwise.

4.36.3.12 getBus()

```
Bus * InsideRoad::getBus ( std::size_t x )
```

Gets a bus from the list of buses.

Gets a [Bus](#) by index.

Parameters

x	Index of the bus to be retrieved.
---	-----------------------------------

Returns

Pointer to the bus at the specified index.

Parameters

x	The index of the Bus .
---	--

Returns

Pointer to the [Bus](#) if the index is valid, nullptr otherwise.

4.36.3.13 getCargoAirport()

```
CargoAirport * InsideRoad::getCargoAirport (   
    std::size_t x )
```

Gets a cargo airport from the list of cargo airports.

Gets a [CargoAirport](#) by index.

Parameters

x	Index of the cargo airport to be retrieved.
---	---

Returns

Pointer to the cargo airport at the specified index.

Parameters

x	The index of the CargoAirport .
---	---

Returns

Pointer to the [CargoAirport](#) if the index is valid, nullptr otherwise.

4.36.3.14 getComercialAirport()

```
ComercialAirport * InsideRoad::getComercialAirport (   
    std::size_t x )
```

Gets a commercial airport from the list of commercial airports.

Gets a [ComercialAirport](#) by index.

Parameters

x	Index of the commercial airport to be retrieved.
---	--

Returns

Pointer to the commercial airport at the specified index.

Parameters

x	The index of the ComercialAirport .
---	---

Returns

Pointer to the [ComercialAirport](#) if the index is valid, nullptr otherwise.

4.36.3.15 getFreightTrain()

```
FreightTrain * InsideRoad::getFreightTrain (
    std::size_t x )
```

Gets a freight train from the list of freight trains.

Gets a [FreightTrain](#) by index.

Parameters

x	Index of the freight train to be retrieved.
---	---

Returns

Pointer to the freight train at the specified index.

Parameters

x	The index of the FreightTrain .
---	---

Returns

Pointer to the [FreightTrain](#) if the index is valid, nullptr otherwise.

4.36.3.16 getHighway()

```
Highway * InsideRoad::getHighway (
    std::size_t x )
```

Gets a highway from the list of highways.

Gets a [Highway](#) by index.

Parameters

x	Index of the highway to be retrieved.
---	---------------------------------------

Returns

Pointer to the highway at the specified index.

Parameters

x	The index of the Highway .
---	--

Returns

Pointer to the [Highway](#) if the index is valid, nullptr otherwise.

4.36.3.17 getInsideRoad()

```
InsideRoad * InsideRoad::getInsideRoad ( std::size_t x )
```

Gets an inside road from the list of inside roads.

Gets an [InsideRoad](#) by index.

Parameters

x	Index of the inside road to be retrieved.
---	---

Returns

Pointer to the inside road at the specified index.

Parameters

x	The index of the InsideRoad .
---	---

Returns

Pointer to the [InsideRoad](#) if the index is valid, nullptr otherwise.

4.36.3.18 getPassengerTrain()

```
PassengerTrain * InsideRoad::getPassengerTrain ( std::size_t x )
```

Gets a passenger train from the list of passenger trains.

Gets a [PassengerTrain](#) by index.

Parameters

x	Index of the passenger train to be retrieved.
---	---

Returns

Pointer to the passenger train at the specified index.

Parameters

x	The index of the PassengerTrain .
---	---

Returns

Pointer to the [PassengerTrain](#) if the index is valid, nullptr otherwise.

4.36.3.19 getRoadName()

```
std::string InsideRoad::getRoadName ( )
```

Gets the name of this road.

Gets the name of the inside road.

Returns

The name of the road.

The name of the inside road.

4.36.3.20 getTaxi()

```
Taxi * InsideRoad::getTaxi (
    std::size_t x )
```

Gets a taxi from the list of taxis.

Gets a [Taxi](#) by index.

Parameters

x	Index of the taxi to be retrieved.
---	------------------------------------

Returns

Pointer to the taxi at the specified index.

Parameters

x	The index of the Taxi .
---	---

Returns

Pointer to the [Taxi](#) if the index is valid, nullptr otherwise.

The documentation for this class was generated from the following files:

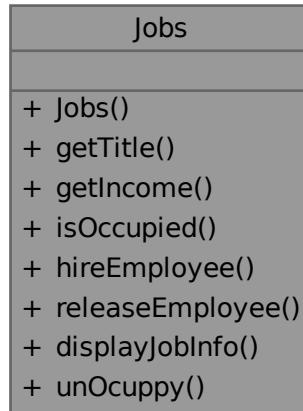
- /mnt/c/users/rudie/documents/sem2/214/project/src/[InsideRoad.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[InsideRoad.cpp](#)

4.37 Jobs Class Reference

Manages job-related operations.

```
#include <Jobs.h>
```

Collaboration diagram for Jobs:



Public Member Functions

- [Jobs](#) (const std::string &jobTitle, double salary)
Constructor that initializes a job with a title and salary.
- std::string [getTitle](#) () const
Gets the job title.
- std::shared_ptr<[Income](#)> [getIncome](#) () const
Gets the income associated with the job.
- bool [isOccupied](#) () const

- Checks if the job is occupied.
- void **hireEmployee ()**
Marks the job as occupied when an employee is hired.
- void **releaseEmployee ()**
Releases the job, marking it as available.
- void **displayJobInfo () const**
Displays job information for debugging or informational purposes.
- void **unOccupy ()**
Marks the job as unoccupied.

4.37.1 Detailed Description

Manages job-related operations.

The [Jobs](#) class manages job-related operations such as hiring and releasing employees, and displaying job information.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 Jobs()

```
Jobs::Jobs (
    const std::string & jobTitle,
    double salary )
```

Constructor that initializes a job with a title and salary.

Constructor initializes job title and income based on provided salary.

Parameters

<i>jobTitle</i>	The title of the job.
<i>salary</i>	The salary for the job.

4.37.3 Member Function Documentation

4.37.3.1 getIncome()

```
std::shared_ptr< Income > Jobs::getIncome ( ) const [inline]
```

Gets the income associated with the job.

Returns

A shared pointer to the [Income](#) object.

4.37.3.2 getTitle()

```
std::string Jobs::getTitle () const [inline]
```

Gets the job title.

Returns

The job title.

4.37.3.3 isOccupied()

```
bool Jobs::isOccupied () const [inline]
```

Checks if the job is occupied.

Returns

True if the job is occupied, false otherwise.

The documentation for this class was generated from the following files:

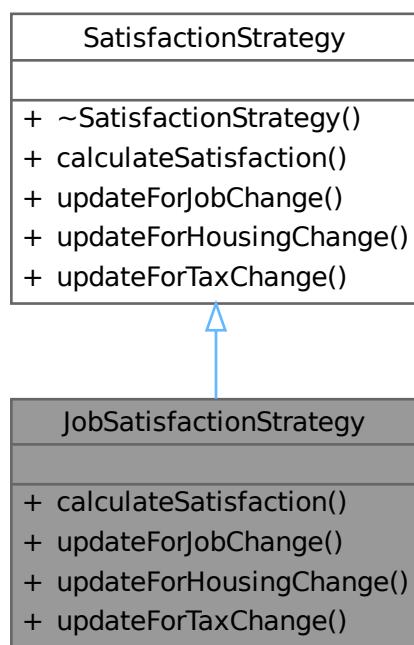
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Jobs.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Jobs.cpp](#)

4.38 JobSatisfactionStrategy Class Reference

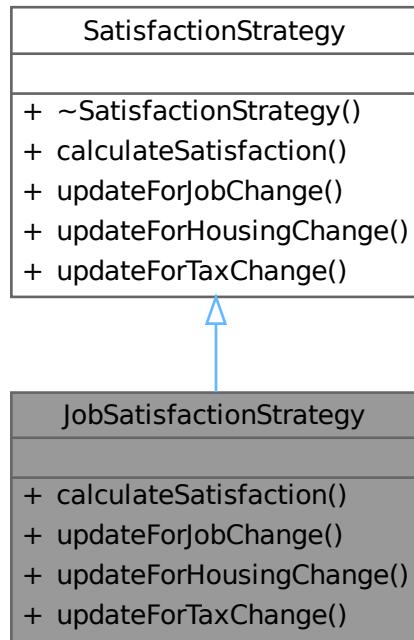
Strategy for calculating and updating citizen satisfaction based on job conditions.

```
#include <JobSatisfactionStrategy.h>
```

Inheritance diagram for JobSatisfactionStrategy:



Collaboration diagram for JobSatisfactionStrategy:



Public Member Functions

- float `calculateSatisfaction` (const `Citizen` &citizen) override
Calculates the satisfaction level of a citizen based on job conditions.
- void `updateForJobChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in job conditions.
- void `updateForHousingChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in housing conditions.
- void `updateForTaxChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen due to a change in tax conditions.

Public Member Functions inherited from `SatisfactionStrategy`

- virtual `~SatisfactionStrategy` ()=default
Virtual destructor for `SatisfactionStrategy`.

4.38.1 Detailed Description

Strategy for calculating and updating citizen satisfaction based on job conditions.

The `JobSatisfactionStrategy` class provides methods to calculate and update citizen satisfaction based on job conditions.

4.38.2 Member Function Documentation

4.38.2.1 calculateSatisfaction()

```
float JobSatisfactionStrategy::calculateSatisfaction (
    const Citizen & citizen ) [override], [virtual]
```

Calculates the satisfaction level of a citizen based on job conditions.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be calculated.
----------------	---

Returns

The calculated satisfaction level.

Implements [SatisfactionStrategy](#).

4.38.2.2 updateForHousingChange()

```
void JobSatisfactionStrategy::updateForHousingChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in housing conditions.

This method is not implemented for job satisfaction strategy.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.38.2.3 updateForJobChange()

```
void JobSatisfactionStrategy::updateForJobChange (
    Citizen & citizen ) [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in job conditions.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.38.2.4 updateForTaxChange()

```
void JobSatisfactionStrategy::updateForTaxChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen due to a change in tax conditions.

This method is not implemented for job satisfaction strategy.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

The documentation for this class was generated from the following files:

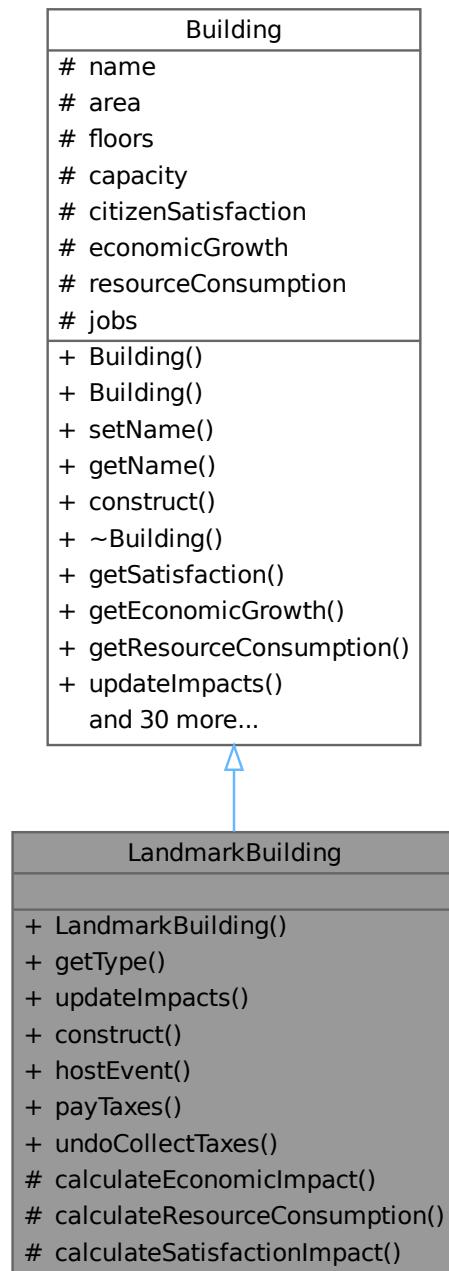
- /mnt/c/users/rudie/documents/sem2/214/project/src/[JobSatisfactionStrategy.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[JobSatisfactionStrategy.cpp](#)

4.39 LandmarkBuilding Class Reference

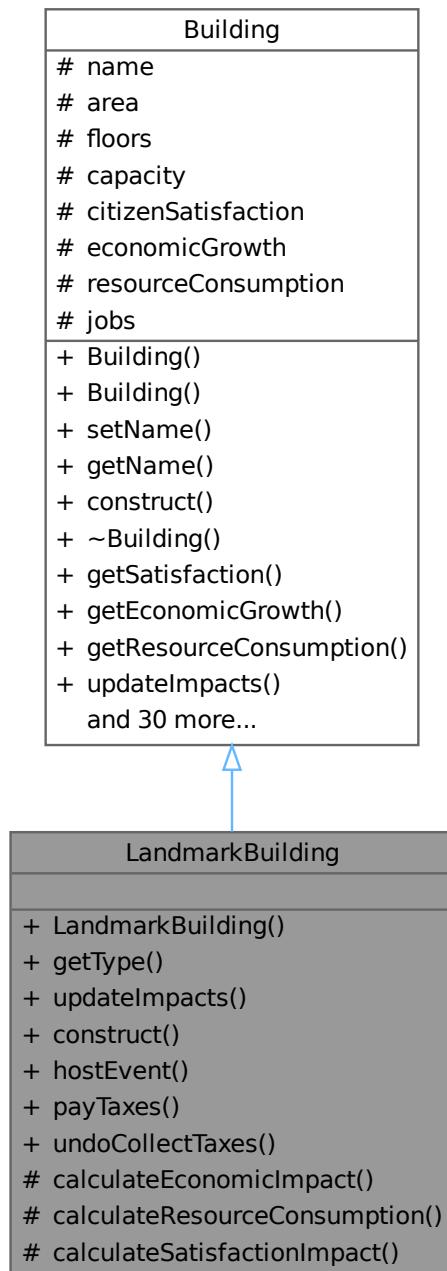
Represents a landmark building with specific attributes and behaviors.

```
#include <LandmarkBuilding.h>
```

Inheritance diagram for LandmarkBuilding:



Collaboration diagram for LandmarkBuilding:



Public Member Functions

- `LandmarkBuilding` (const std::string &`name`, float `area`, int `floors`, int `capacity`, float `citizenSatisfactionImpact`, float `economicGrowthImpact`, float `resourceConsumption`, int `visitorCapacity`, float `culturalValue`, bool `isHistoric`)
Constructor for LandmarkBuilding.
- string `getType` () const override

- Gets the type of the building.
- void **updateImpacts** () override

Updates the impacts of the building by calculating economic impact, resource consumption, and satisfaction impact.
- void **construct** () override

Constructs the building.
- void **hostEvent** (int visitors)

Hosts an event in the building.
- double **payTaxes** (TaxType *taxType) override

Pays taxes for the building.
- void **undoCollectTaxes** () override

Undoes the tax collection from the building.

Public Member Functions inherited from Building

- Building (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)

Main constructor for setting attributes directly.
- Building (int Builder)

Alternative constructor using an integer parameter, potentially for *Builder* pattern integration.
- void **setName** (const std::string &name)

Sets the name of the building.
- std::string **getName** () const

Gets the name of the building.
- virtual ~Building ()=default

Virtual destructor for the *Building* class.
- float **getSatisfaction** () const

Gets the citizen satisfaction impact of the building.
- float **getEconomicGrowth** () const

Gets the economic growth impact of the building.
- float **getResourceConsumption** () const

Gets the resource consumption of the building.
- void **addJob** (std::shared_ptr<Jobs> job)

Adds a job to the building.
- void **listJobs** () const

Lists all jobs in the building.
- bool **hireEmployee** (const std::string &jobTitle)

Hires an employee for a job if available.
- void **releaseEmployee** (const std::string &jobTitle)

Releases an employee from a job.
- std::shared_ptr<Jobs> **getAvailableJob** ()

Provides access to an available job.
- void **displayJobInfo** (const std::string &jobTitle) const

Displays job information.
- const std::vector<std::shared_ptr<Jobs>> & **getJobs** () const

Returns the job list as a const reference.
- Building (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)

Main constructor for setting attributes directly.
- Building (int Builder)

Alternative constructor using an integer parameter, potentially for *Builder* pattern integration.

- void `setName` (const std::string &`name`)
Sets the name of the building.
- std::string `getName` () const
Gets the name of the building.
- virtual ~**Building** ()=default
Virtual destructor for the `Building` class.
- float `getSatisfaction` () const
Gets the citizen satisfaction impact of the building.
- float `getEconomicGrowth` () const
Gets the economic growth impact of the building.
- float `getResourceConsumption` () const
Gets the resource consumption of the building.
- void `addJob` (std::shared_ptr< `Jobs` > `job`)
Adds a job to the building.
- void `listJobs` () const
Lists all jobs in the building.
- bool `hireEmployee` (const std::string &`jobTitle`)
Hires an employee for a job if available.
- void `releaseEmployee` (const std::string &`jobTitle`)
Releases an employee from a job.
- std::shared_ptr< `Jobs` > `getAvailableJob` ()
Provides access to an available job.
- void `displayJobInfo` (const std::string &`jobTitle`) const
Displays job information.
- const std::vector< std::shared_ptr< `Jobs` > > & `getJobs` () const
Returns the job list as a const reference.

Protected Member Functions

- void `calculateEconomicImpact` ()
Calculates the economic impact of the building.
- void `calculateResourceConsumption` ()
Calculates the resource consumption of the building.
- void `calculateSatisfactionImpact` ()
Calculates the satisfaction impact of the building.

Additional Inherited Members

Protected Attributes inherited from Building

- std::string **name**
Name of the building.
- float **area**
Area of the building.
- int **floors**
Number of floors in the building.
- int **capacity**
Capacity of the building.
- float **citizenSatisfaction**

- float **economicGrowth**
Economic growth impact of the building.
- float **resourceConsumption**
Resource consumption of the building.
- std::vector< std::shared_ptr< [Jobs](#) >> **jobs**
Collection of jobs as shared pointers.

4.39.1 Detailed Description

Represents a landmark building with specific attributes and behaviors.

The [LandmarkBuilding](#) class inherits from the [Building](#) class and provides specific implementations for a landmark building, including methods to calculate impacts, host events, and manage taxes.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 [LandmarkBuilding\(\)](#)

```
LandmarkBuilding::LandmarkBuilding (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float citizenSatisfaction,
    float economicGrowth,
    float resourceConsumption,
    int visitorsPerDay,
    float entranceFee,
    bool hasGuidedTours )
```

Constructor for [LandmarkBuilding](#).

Parameters

<i>name</i>	The name of the landmark building.
<i>area</i>	The area of the landmark building.
<i>floors</i>	The number of floors in the landmark building.
<i>capacity</i>	The capacity of the landmark building.
<i>citizenSatisfactionImpact</i>	The citizen satisfaction level.
<i>economicGrowthImpact</i>	The economic growth contributed by the building.
<i>resourceConsumption</i>	The resource consumption of the building.
<i>visitorCapacity</i>	The visitor capacity of the landmark building.
<i>culturalValue</i>	The cultural value of the landmark building.
<i>isHistoric</i>	Indicates if the landmark building is historic.
<i>name</i>	The name of the landmark building.
<i>area</i>	The area of the landmark building.
<i>floors</i>	The number of floors in the landmark building.
<i>capacity</i>	The capacity of the landmark building.
<i>citizenSatisfaction</i>	The citizen satisfaction level.

Parameters

<i>economicGrowth</i>	The economic growth contributed by the building.
<i>resourceConsumption</i>	The resource consumption of the building.
<i>visitorsPerDay</i>	The number of visitors per day.
<i>entranceFee</i>	The entrance fee for the landmark building.
<i>hasGuidedTours</i>	Indicates if the landmark building has guided tours.

4.39.3 Member Function Documentation**4.39.3.1 calculateEconomicImpact()**

```
void LandmarkBuilding::calculateEconomicImpact ( ) [protected], [virtual]
```

Calculates the economic impact of the building.

Implements [Building](#).

4.39.3.2 calculateResourceConsumption()

```
void LandmarkBuilding::calculateResourceConsumption ( ) [protected], [virtual]
```

Calculates the resource consumption of the building.

Implements [Building](#).

4.39.3.3 calculateSatisfactionImpact()

```
void LandmarkBuilding::calculateSatisfactionImpact ( ) [protected], [virtual]
```

Calculates the satisfaction impact of the building.

Implements [Building](#).

4.39.3.4 construct()

```
void LandmarkBuilding::construct ( ) [override], [virtual]
```

Constructs the building.

Implements [Building](#).

4.39.3.5 getType()

```
string LandmarkBuilding::getType ( ) const [override], [virtual]
```

Gets the type of the building.

Returns

The type of the building.

Implements [Building](#).

4.39.3.6 hostEvent()

```
void LandmarkBuilding::hostEvent (
    int visitors )
```

Hosts an event in the building.

Parameters

<code>visitors</code>	The number of visitors attending the event.
-----------------------	---

4.39.3.7 payTaxes()

```
double LandmarkBuilding::payTaxes (
    TaxType * taxType ) [override], [virtual]
```

Pays taxes for the building.

Parameters

<code>taxType</code>	The type of tax to calculate.
----------------------	-------------------------------

Returns

The amount of taxes paid.

Implements [Building](#).

4.39.3.8 undoCollectTaxes()

```
void LandmarkBuilding::undoCollectTaxes ( ) [override], [virtual]
```

Undoes the tax collection from the building.

Implements [Building](#).

4.39.3.9 updateImpacts()

```
void LandmarkBuilding::updateImpacts ( ) [override], [virtual]
```

Updates the impacts of the building by calculating economic impact, resource consumption, and satisfaction impact.

Implements [Building](#).

The documentation for this class was generated from the following files:

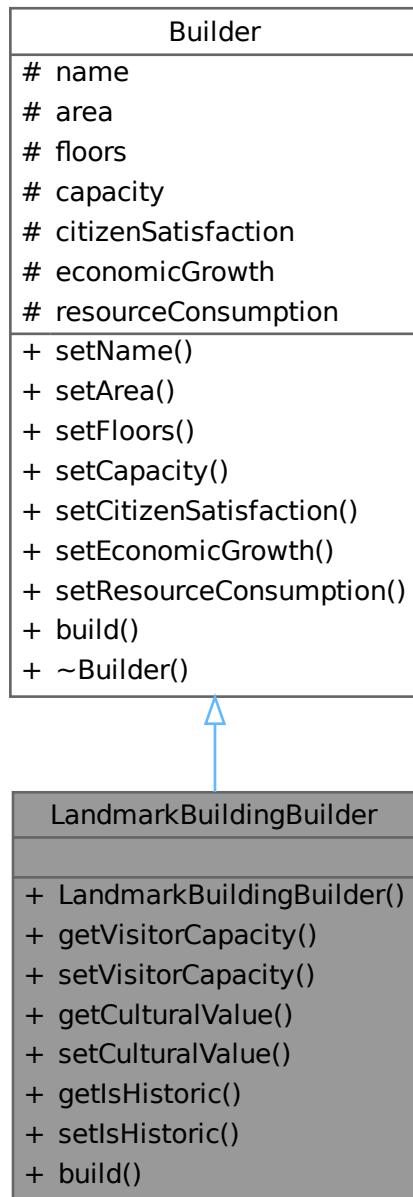
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LandmarkBuilding.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LandmarkBuilding.cpp](#)

4.40 LandmarkBuildingBuilder Class Reference

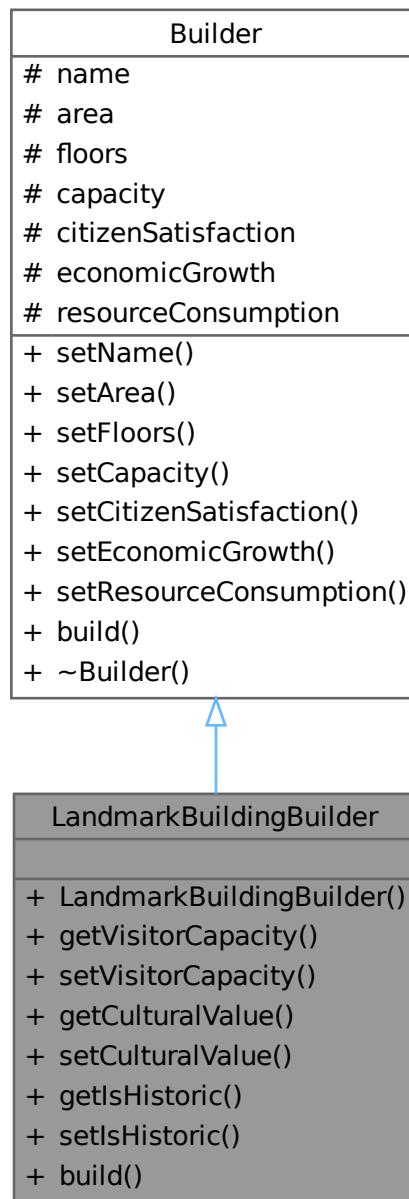
[Builder](#) class for constructing [LandmarkBuilding](#) objects.

```
#include <LandmarkBuildingBuilder.h>
```

Inheritance diagram for LandmarkBuildingBuilder:



Collaboration diagram for LandmarkBuildingBuilder:



Public Member Functions

- **LandmarkBuildingBuilder ()**
Constructor for `LandmarkBuildingBuilder`.
- int **getVisitorCapacity ()**
Gets the visitor capacity of the building.
- **LandmarkBuildingBuilder & setVisitorCapacity (int visitorCapacity)**
Sets the visitor capacity of the building.

- float `getCulturalValue ()`
Gets the cultural value of the building.
- `LandmarkBuildingBuilder & setCulturalValue (float culturalValue)`
Sets the cultural value of the building.
- bool `getIsHistoric ()`
Gets whether the building is historic.
- `LandmarkBuildingBuilder & setIsHistoric (bool isHistoric)`
Sets whether the building is historic.
- std::unique_ptr< `Building` > `build ()` override
Builds the landmark building and sets all the attributes of the building.

Public Member Functions inherited from `Builder`

- `Builder & setName (string name)`
Sets the name of the building.
- `Builder & setArea (float area)`
Sets the area of the building.
- `Builder & setFloors (int floors)`
Sets the number of floors in the building.
- `Builder & setCapacity (int capacity)`
Sets the capacity of the building.
- `Builder & setCitizenSatisfaction (float citizenSatisfaction)`
Sets the citizen satisfaction of the building.
- `Builder & setEconomicGrowth (float economicGrowth)`
Sets the economic growth of the building.
- `Builder & setResourceConsumption (float resourceConsumption)`
Sets the resource consumption of the building.
- virtual ~`Builder` ()=default
Virtual destructor for the `Builder` class.

Additional Inherited Members

Protected Attributes inherited from `Builder`

- string **name**
Name of the building.
- float **area** = 0.0f
Area of the building.
- int **floors** = 0
Number of floors in the building.
- int **capacity** = 0
Capacity of the building.
- float **citizenSatisfaction** = 0.0f
Citizen satisfaction of the building.
- float **economicGrowth** = 0.0f
Economic growth of the building.
- float **resourceConsumption** = 0.0f
Resource consumption of the building.

4.40.1 Detailed Description

[Builder](#) class for constructing [LandmarkBuilding](#) objects.

The [LandmarkBuildingBuilder](#) class provides methods to set attributes and build a [LandmarkBuilding](#) object.

4.40.2 Member Function Documentation

4.40.2.1 build()

```
std::unique_ptr< Building > LandmarkBuildingBuilder::build ( ) [override], [virtual]
```

Builds the landmark building and sets all the attributes of the building.

Returns

A unique pointer to the constructed [LandmarkBuilding](#) object.

Implements [Builder](#).

4.40.2.2 getCultureValue()

```
float LandmarkBuildingBuilder::getCultureValue ( )
```

Gets the cultural value of the building.

Returns

The cultural value of the building.

4.40.2.3 getIsHistoric()

```
bool LandmarkBuildingBuilder::getIsHistoric ( )
```

Gets whether the building is historic.

Returns

True if the building is historic, false otherwise.

4.40.2.4 getVisitorCapacity()

```
int LandmarkBuildingBuilder::getVisitorCapacity ( )
```

Gets the visitor capacity of the building.

Returns

The visitor capacity of the building.

4.40.2.5 setCultureValue()

```
LandmarkBuildingBuilder & LandmarkBuildingBuilder::setCultureValue ( float culturalValue )
```

Sets the cultural value of the building.

Parameters

<i>culturalValue</i>	The cultural value to set.
----------------------	----------------------------

Returns

Reference to the current [LandmarkBuildingBuilder](#) object.

4.40.2.6 setIsHistoric()

```
LandmarkBuildingBuilder & LandmarkBuildingBuilder::setIsHistoric (   
    bool isHistoric )
```

Sets whether the building is historic.

Parameters

<i>isHistoric</i>	True if the building is historic, false otherwise.
-------------------	--

Returns

Reference to the current [LandmarkBuildingBuilder](#) object.

4.40.2.7 setVisitorCapacity()

```
LandmarkBuildingBuilder & LandmarkBuildingBuilder::setVisitorCapacity (   
    int visitorCapacity )
```

Sets the visitor capacity of the building.

Parameters

<i>visitorCapacity</i>	The visitor capacity to set.
------------------------	------------------------------

Returns

Reference to the current [LandmarkBuildingBuilder](#) object.

The documentation for this class was generated from the following files:

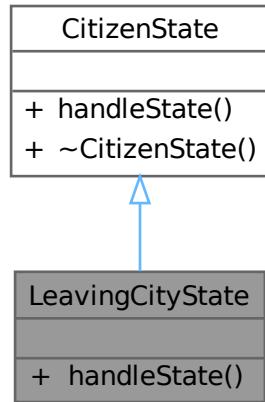
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LandmarkBuildingBuilder.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LandmarkBuildingBuilder.cpp](#)

4.41 LeavingCityState Class Reference

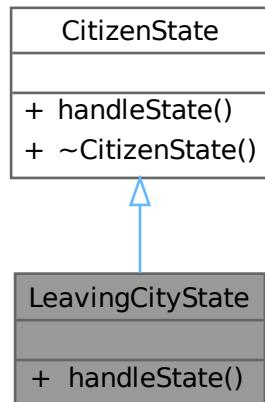
Handles the state of a citizen preparing to leave the city.

```
#include <LeavingCityState.h>
```

Inheritance diagram for LeavingCityState:



Collaboration diagram for LeavingCityState:



Public Member Functions

- void `handleState (Citizen &citizen) const override`
Handles the state of a citizen preparing to leave the city.

Public Member Functions inherited from `CitizenState`

- virtual `~CitizenState ()=default`
Virtual destructor for the `CitizenState` class.

4.41.1 Detailed Description

Handles the state of a citizen preparing to leave the city.

The [LeavingCityState](#) class provides the implementation for handling the state of a citizen who is preparing to leave the city.

4.41.2 Member Function Documentation

4.41.2.1 handleState()

```
void LeavingCityState::handleState (
    Citizen & citizen ) const [override], [virtual]
```

Handles the state of a citizen preparing to leave the city.

Parameters

<i>citizen</i>	The citizen whose state is being handled.
<i>citizen</i>	The citizen whose state is being handled.

This method drastically reduces the satisfaction level of the citizen as they prepare to leave.

Implements [CitizenState](#).

The documentation for this class was generated from the following files:

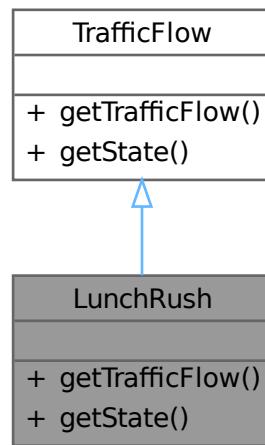
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LeavingCityState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LeavingCityState.cpp](#)

4.42 LunchRush Class Reference

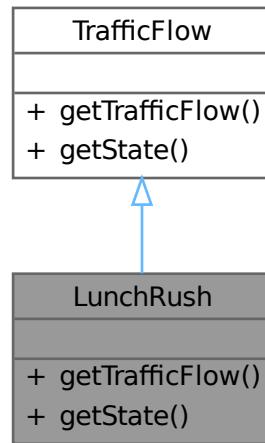
A class representing traffic flow during lunch hours.

```
#include <LunchRush.h>
```

Inheritance diagram for LunchRush:



Collaboration diagram for LunchRush:



Public Member Functions

- float [getTrafficFlow \(\)](#)
Get the traffic flow value.
- char [getState \(\)](#)
Get the state of the traffic flow.

4.42.1 Detailed Description

A class representing traffic flow during lunch hours.

The [LunchRush](#) class inherits from the [TrafficFlow](#) class and provides specific implementations for traffic flow and state during lunch hours.

4.42.2 Member Function Documentation

4.42.2.1 getState()

```
char LunchRush::getState ( ) [virtual]
```

Get the state of the traffic flow.

Gets the state during the lunch rush period.

Returns

The state of the traffic flow as a char.

The state.

Implements [TrafficFlow](#).

4.42.2.2 getTrafficFlow()

```
float LunchRush::getTrafficFlow ( ) [virtual]
```

Get the traffic flow value.

Gets the traffic flow during the lunch rush period.

Returns

The traffic flow value as a float.

The traffic flow.

Implements [TrafficFlow](#).

The documentation for this class was generated from the following files:

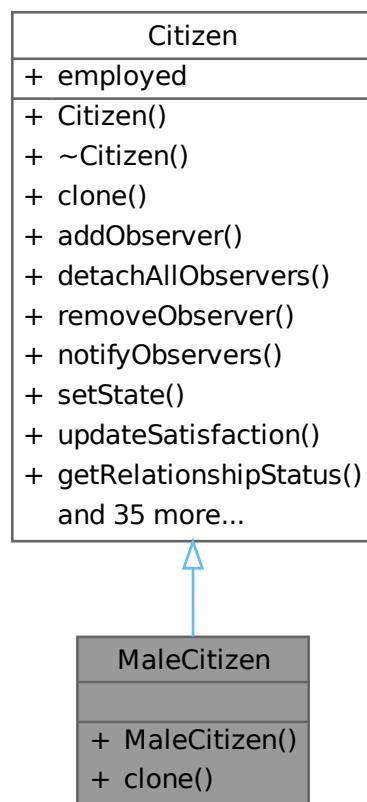
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LunchRush.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[LunchRush.cpp](#)

4.43 MaleCitizen Class Reference

A class representing a male citizen.

```
#include <MaleCitizen.h>
```

Inheritance diagram for MaleCitizen:



Collaboration diagram for MaleCitizen:



Public Member Functions

- `MaleCitizen` (const std::string &name, int age)
Constructor for MaleCitizen.
- `std::shared_ptr< Citizen > clone () const override`
Clone method to create a copy of the MaleCitizen object.

Public Member Functions inherited from `Citizen`

- `Citizen` (const std::string &name, int age)
Constructs a new Citizen object.
- `virtual ~Citizen ()`
Destroys the Citizen object.
- `void addObserver (CitizenObserver *observer)`
Adds an observer to the citizen.
- `void detachAllObservers ()`
Detaches all observers from the citizen.
- `void removeObserver (CitizenObserver *observer)`

- **void notifyObservers ()**
Removes an observer from the citizen.
- **void setState (CitizenState *newState)**
Notifies all observers of changes to the citizen.
- **void updateSatisfaction ()**
Sets the state of the citizen.
- **std::string getRelationshipStatus () const**
Gets the relationship status of the citizen.
- **void setRelationshipStatus (const std::string &status)**
Sets the relationship status of the citizen.
- **int getMarriageDuration () const**
Gets the duration of the citizen's marriage.
- **void resetMarriageDuration ()**
Resets the duration of the citizen's marriage to zero.
- **void incrementMarriageDuration ()**
Increments the duration of the citizen's marriage by one year.
- **void updateSatisfaction (float adjustment)**
Updates the satisfaction of the citizen by a specified adjustment.
- **void addChild ()**
Adds a child to the citizen's family.
- **int getAge () const**
Gets the age of the citizen.
- **void incrementAge ()**
Increments the age of the citizen by one year.
- **void addSatisfactionStrategy (std::shared_ptr< SatisfactionStrategy > strategy)**
Adds a satisfaction strategy to the citizen.
- **void removeSatisfactionStrategy ()**
Removes all satisfaction strategies from the citizen.
- **void setSatisfactionLevel (double level)**
Sets the satisfaction level of the citizen.
- **void depositMonthlyIncome ()**
Deposits the monthly income of the citizen into their bank balance.
- **void searchAndApplyForJob (BuildingManager &manager, Building *building, std::string jobtitle)**
Searches for and applies for a job for the citizen.
- **std::string getName () const**
Gets the name of the citizen.
- **float getSatisfactionLevel () const**
Gets the satisfaction level of the citizen.
- **bool isLeaving () const**
Checks if the citizen is leaving the city.
- **void setIncome (std::shared_ptr< Income > income)**
Sets the income of the citizen.
- **void checkAndUpdateState ()**
Checks and updates the state of the citizen based on satisfaction.
- **void setJobTitle (const std::string &jobTitle)**
Sets the job title of the citizen.
- **std::string getJob () const**
Gets the job title of the citizen.
- **void setEmployed (bool status)**
Sets the employment status of the citizen.

- bool `isEmployed () const`
Checks if the citizen is employed.
- float `getTaxRate () const`
Gets the tax rate for the citizen.
- void `setTaxRate (float rate)`
Sets the tax rate for the citizen.
- double `payTaxes (TaxType *taxType)`
Processes the payment of taxes by the citizen.
- double `getBankBalance () const`
Gets the bank balance of the citizen.
- void `setBankBalance (double balance)`
Sets the bank balance of the citizen.
- void `increaseBankBalance (double amount)`
Increases the bank balance of the citizen by a specified amount.
- void `subtractBankBalance (double amount)`
Decreases the bank balance of the citizen by a specified amount.
- void `setTaxCooldown (bool status)`
Sets the tax cooldown status for the citizen.
- bool `getTaxCooldown () const`
Gets the tax cooldown status for the citizen.
- bool `isOnCooldown () const`
Checks if the citizen is on cooldown for tax payments.
- std::shared_ptr<Jobs> `getjobobj ()`
Gets the job object of the citizen.
- void `setJob (std::shared_ptr<Jobs> job)`
Sets the job object of the citizen.
- void `unsetJob ()`
Unsets the job object of the citizen.

Additional Inherited Members

Public Attributes inherited from [Citizen](#)

- bool `employed = false`
Employment status of the citizen.

4.43.1 Detailed Description

A class representing a male citizen.

The [MaleCitizen](#) class inherits from the [Citizen](#) class and provides a specific implementation for male citizens. It includes a clone method to create a copy of the object.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 [MaleCitizen\(\)](#)

```
MaleCitizen::MaleCitizen (
    const std::string & name,
    int age ) [inline]
```

Constructor for [MaleCitizen](#).

Parameters

<i>name</i>	The name of the male citizen.
<i>age</i>	The age of the male citizen.

4.43.3 Member Function Documentation

4.43.3.1 clone()

```
std::shared_ptr< Citizen > MaleCitizen::clone ( ) const [inline], [override], [virtual]
```

Clone method to create a copy of the [MaleCitizen](#) object.

Returns

A shared pointer to the cloned [MaleCitizen](#) object.

Implements [Citizen](#).

The documentation for this class was generated from the following file:

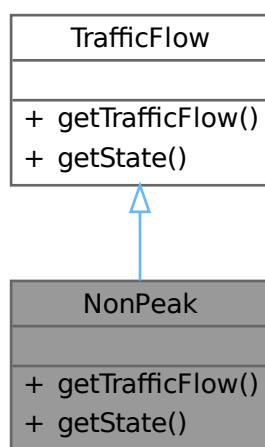
- /mnt/c/users/rudie/documents/sem2/214/project/src/[MaleCitizen.h](#)

4.44 NonPeak Class Reference

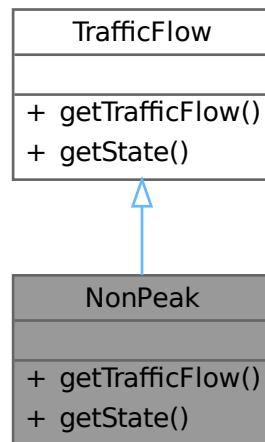
A class to represent traffic flow during non-peak hours.

```
#include <NonPeak.h>
```

Inheritance diagram for NonPeak:



Collaboration diagram for NonPeak:



Public Member Functions

- float `getTrafficFlow ()`
Get the traffic flow rate.
- char `getState ()`
Get the state of traffic flow.

4.44.1 Detailed Description

A class to represent traffic flow during non-peak hours.

The `NonPeak` class inherits from the `TrafficFlow` class and provides specific implementations for traffic flow and state during non-peak hours.

4.44.2 Member Function Documentation

4.44.2.1 getState()

```
char NonPeak::getState ( ) [virtual]
```

Get the state of traffic flow.

Gets the state during non-peak hours.

Returns

char The state representing non-peak hours.
The state.

Implements `TrafficFlow`.

4.44.2.2 getTrafficFlow()

```
float NonPeak::getTrafficFlow ( ) [virtual]
```

Get the traffic flow rate.

Gets the traffic flow during non-peak hours.

Returns

float The traffic flow rate during non-peak hours.

The traffic flow.

Implements [TrafficFlow](#).

The documentation for this class was generated from the following files:

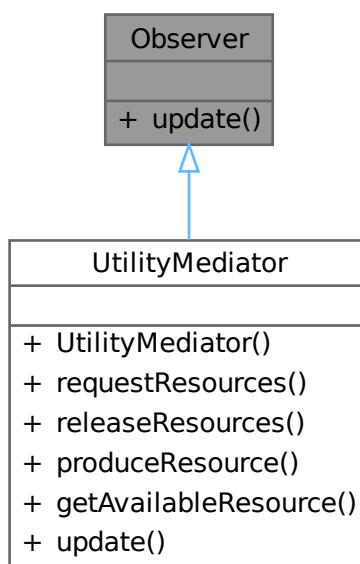
- /mnt/c/users/rudie/documents/sem2/214/project/src/[NonPeak.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[NonPeak.cpp](#)

4.45 Observer Class Reference

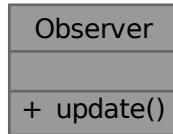
Interface for objects that need to be notified of changes in resource types and quantities.

```
#include <Observer.h>
```

Inheritance diagram for Observer:



Collaboration diagram for Observer:



Public Member Functions

- virtual void `update (ResourceType type, int quantity)=0`
Updates the observer with the new resource type and quantity.

4.45.1 Detailed Description

Interface for objects that need to be notified of changes in resource types and quantities.

The `Observer` class provides an interface for objects that need to be notified of changes in resource types and quantities.

4.45.2 Member Function Documentation

4.45.2.1 update()

```

virtual void Observer::update (
    ResourceType type,
    int quantity ) [pure virtual]
  
```

Updates the observer with the new resource type and quantity.

Parameters

<code>type</code>	The type of the resource.
<code>quantity</code>	The quantity of the resource.

Implemented in [UtilityMediator](#).

The documentation for this class was generated from the following file:

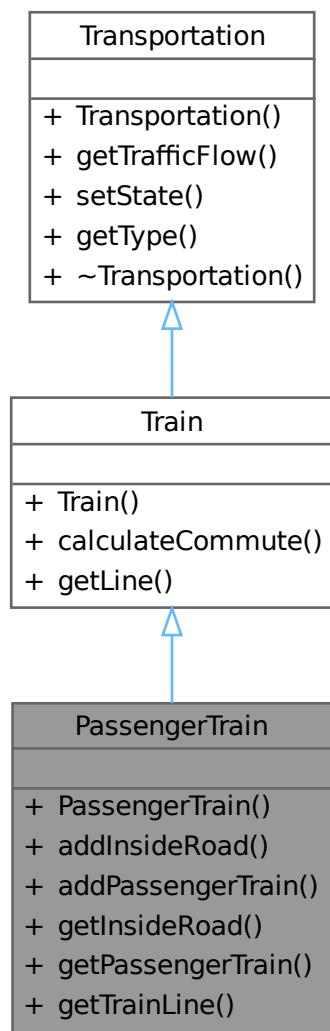
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Observer.h](#)

4.46 PassengerTrain Class Reference

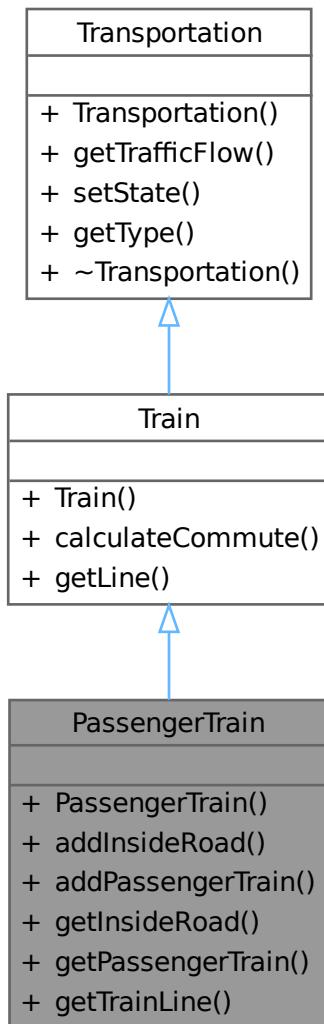
A class representing a passenger train.

```
#include <PassengerTrain.h>
```

Inheritance diagram for PassengerTrain:



Collaboration diagram for PassengerTrain:



Public Member Functions

- `PassengerTrain (char state, std::string line)`
Constructor for the `PassengerTrain` class.
- `bool addInsideRoad (InsideRoad *insideRoad)`
Adds an `InsideRoad` object to the train.
- `bool addPassengerTrain (PassengerTrain *passengerTrain)`
Adds a `PassengerTrain` object to the train.
- `InsideRoad * getInsideRoad (std::size_t x)`
Gets an `InsideRoad` object from the train.
- `PassengerTrain * getPassengerTrain (std::size_t x)`
Gets a `PassengerTrain` object from the train.
- `std::string getTrainLine ()`
Gets the line on which the train operates.

Public Member Functions inherited from Train

- `Train` (char state, std::string line, char type)

Construct a new `Train` object.

- float `calculateCommute ()`

Calculate the commute time for the train.

- std::string `getLine ()`

Get the line on which the train operates.

Public Member Functions inherited from Transportation

- `Transportation` (char state, char type)

Constructor for the `Transportation` class.

- float `getTrafficFlow ()`

Gets the current traffic flow.

- bool `setState (char state)`

Sets the traffic flow state.

- char `getType ()`

Gets the type of transportation.

- `~Transportation ()`

Destructor for the `Transportation` class.

4.46.1 Detailed Description

A class representing a passenger train.

The `PassengerTrain` class inherits from the `Train` class and manages a collection of `InsideRoad` and `PassengerTrain` objects.

4.46.2 Constructor & Destructor Documentation

4.46.2.1 PassengerTrain()

```
PassengerTrain::PassengerTrain (
    char state,
    std::string line )
```

Constructor for the `PassengerTrain` class.

Constructor for `PassengerTrain`.

Parameters

<code>state</code>	The state of the train.
<code>line</code>	The line on which the train operates.
<code>state</code>	The state of the passenger train.
<code>line</code>	The line on which the passenger train operates.

4.46.3 Member Function Documentation

4.46.3.1 addInsideRoad()

```
bool PassengerTrain::addInsideRoad (
    InsideRoad * insideRoad )
```

Adds an [InsideRoad](#) object to the train.

Adds an [InsideRoad](#) to the passenger train.

Parameters

<i>insideRoad</i>	A pointer to the InsideRoad object to be added.
-------------------	---

Returns

true if the [InsideRoad](#) was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad to be added.
-------------------	--

Returns

True if the [InsideRoad](#) was added successfully, false if it already exists.

4.46.3.2 addPassengerTrain()

```
bool PassengerTrain::addPassengerTrain (
    PassengerTrain * passengerTrain )
```

Adds a [PassengerTrain](#) object to the train.

Adds another [PassengerTrain](#) to the passenger train.

Parameters

<i>passengerTrain</i>	A pointer to the PassengerTrain object to be added.
-----------------------	---

Returns

true if the [PassengerTrain](#) was added successfully, false otherwise.

Parameters

<i>passengerTrain</i>	Pointer to the PassengerTrain to be added.
-----------------------	--

Returns

True if the [PassengerTrain](#) was added successfully, false if it already exists.

4.46.3.3 getInsideRoad()

```
InsideRoad * PassengerTrain::getInsideRoad (   
    std::size_t x )
```

Gets an [InsideRoad](#) object from the train.

Gets an [InsideRoad](#) by index.

Parameters

x	The index of the InsideRoad object to be retrieved.
---	---

Returns

A pointer to the [InsideRoad](#) object.

Parameters

x	The index of the InsideRoad .
---	---

Returns

Pointer to the [InsideRoad](#) if the index is valid, nullptr otherwise.

4.46.3.4 getPassengerTrain()

```
PassengerTrain * PassengerTrain::getPassengerTrain (   
    std::size_t x )
```

Gets a [PassengerTrain](#) object from the train.

Gets a [PassengerTrain](#) by index.

Parameters

x	The index of the PassengerTrain object to be retrieved.
---	---

Returns

A pointer to the [PassengerTrain](#) object.

Parameters

x	The index of the PassengerTrain .
---	---

Returns

Pointer to the [PassengerTrain](#) if the index is valid, nullptr otherwise.

4.46.3.5 getTrainLine()

```
std::string PassengerTrain::getTrainLine ( )
```

Gets the line on which the train operates.

Gets the line on which the passenger train operates.

Returns

A string representing the train line.

The line of the passenger train.

The documentation for this class was generated from the following files:

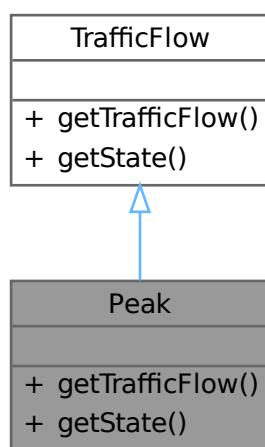
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PassengerTrain.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PassengerTrain.cpp](#)

4.47 Peak Class Reference

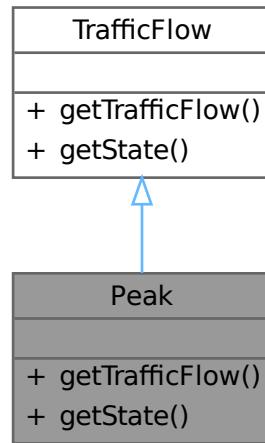
A class representing peak traffic flow.

```
#include <Peak.h>
```

Inheritance diagram for Peak:



Collaboration diagram for Peak:



Public Member Functions

- float [getTrafficFlow \(\)](#)
Get the traffic flow value.
- char [getState \(\)](#)
Get the state of the traffic flow.

4.47.1 Detailed Description

A class representing peak traffic flow.

The `Peak` class inherits from the `TrafficFlow` class and represents a peak traffic flow state with a specific traffic flow value.

4.47.2 Member Function Documentation

4.47.2.1 getState()

```
char Peak::getState ( ) [virtual]
```

Get the state of the traffic flow.

Gets the state during peak hours.

Returns

The state of the traffic flow as a char.

The state.

Implements [TrafficFlow](#).

4.47.2.2 getTrafficFlow()

```
float Peak::getTrafficFlow ( ) [virtual]
```

Get the traffic flow value.

Gets the traffic flow during peak hours.

Returns

The traffic flow value as a float.

The traffic flow.

Implements [TrafficFlow](#).

The documentation for this class was generated from the following files:

- /mnt/c/users/rudie/documents/sem2/214/project/src/[Peak.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Peak.cpp](#)

4.48 Policy Class Reference

Represents a policy with specific attributes and behaviors.

```
#include <Policy.h>
```

Collaboration diagram for Policy:

Policy
+ Policy() + Policy() + implement() + revoke() + getPolicyName() + getImpactLevel()

Public Member Functions

- **Policy ()**
Default constructor for Policy.
- **Policy (const std::string &name, const std::string &impact)**
Constructor for Policy with parameters.
- **void implement ()**
Implements the policy.
- **void revoke ()**
Revokes the policy.
- **std::string getPolicyName () const**
Gets the name of the policy.
- **std::string getImpactLevel () const**
Gets the impact level of the policy.

4.48.1 Detailed Description

Represents a policy with specific attributes and behaviors.

The [Policy](#) class provides methods to implement and revoke a policy, as well as to get the policy's name and impact level.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 Policy()

```
Policy::Policy (
    const std::string & name,
    const std::string & impact )
```

Constructor for [Policy](#) with parameters.

Parameters

<i>name</i>	The name of the policy.
<i>impact</i>	The impact level of the policy.

4.48.3 Member Function Documentation

4.48.3.1 getImpactLevel()

```
std::string Policy::getImpactLevel ( ) const
```

Gets the impact level of the policy.

Returns

The impact level of the policy.

4.48.3.2 `getPolicyName()`

```
std::string Policy::getPolicyName ( ) const
```

Gets the name of the policy.

Returns

The name of the policy.

4.48.3.3 `implement()`

```
void Policy::implement ( )
```

Implements the policy.

Prints a message indicating the policy implementation with its impact level.

4.48.3.4 `revoke()`

```
void Policy::revoke ( )
```

Revokes the policy.

Prints a message indicating the policy revocation with its impact level.

The documentation for this class was generated from the following files:

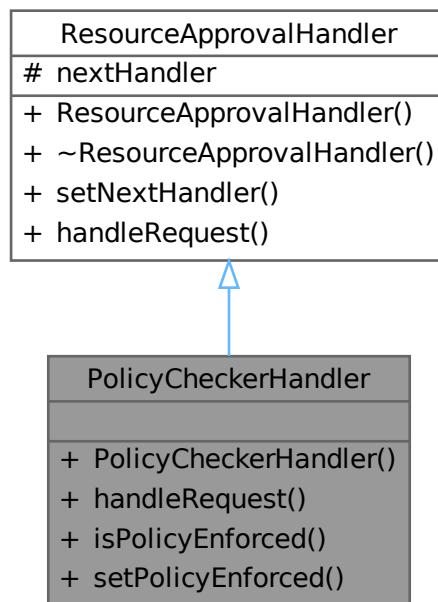
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Policy.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Policy.cpp](#)

4.49 PolicyCheckerHandler Class Reference

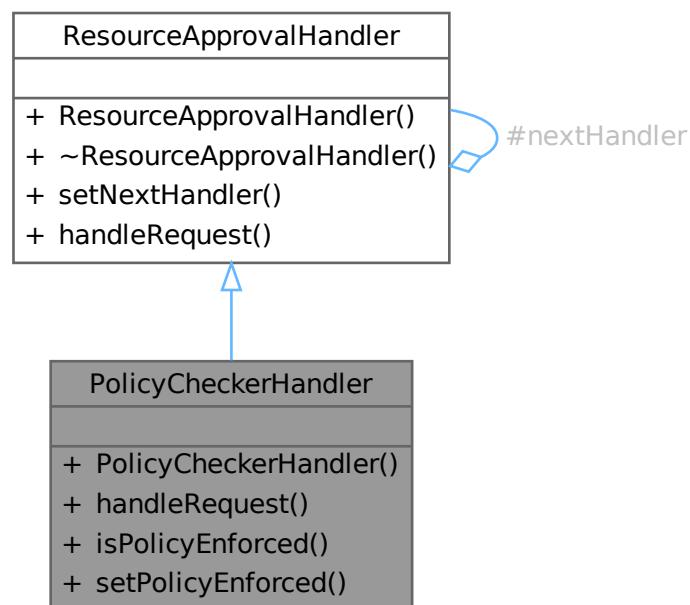
Handles policy enforcement checks for resource approval requests.

```
#include <PolicyCheckerHandler.h>
```

Inheritance diagram for PolicyCheckerHandler:



Collaboration diagram for PolicyCheckerHandler:



Public Member Functions

- **PolicyCheckerHandler (Government *gov)**
Constructor for PolicyCheckerHandler.
- **bool handleRequest (ResourceType type, int quantity) override**
Handles the resource approval request.
- **bool isPolicyEnforced () const**
Determines if a policy was enforced.
- **void setPolicyEnforced (bool enforced)**
Sets the policy enforcement state.

Public Member Functions inherited from ResourceApprovalHandler

- **ResourceApprovalHandler ()**
Constructs a new ResourceApprovalHandler object.
- **virtual ~ResourceApprovalHandler ()**
Destroys the ResourceApprovalHandler object.
- **void setNextHandler (ResourceApprovalHandler *handler)**
Sets the next handler in the chain.

Additional Inherited Members

Protected Attributes inherited from ResourceApprovalHandler

- **ResourceApprovalHandler * nextHandler**
Pointer to the next handler in the chain.

4.49.1 Detailed Description

Handles policy enforcement checks for resource approval requests.

The [PolicyCheckerHandler](#) class inherits from [ResourceApprovalHandler](#) and provides methods to enforce policies and handle resource approval requests based on policy enforcement.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 PolicyCheckerHandler()

```
PolicyCheckerHandler::PolicyCheckerHandler (
    Government * gov ) [inline]
```

Constructor for [PolicyCheckerHandler](#).

Parameters

gov	Pointer to the Government object.
-----	---

4.49.3 Member Function Documentation

4.49.3.1 handleRequest()

```
bool PolicyCheckerHandler::handleRequest (
    ResourceType type,
    int quantity ) [inline], [override], [virtual]
```

Handles the resource approval request.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource.

Returns

True if the request is approved, false otherwise.

Reimplemented from [ResourceApprovalHandler](#).

4.49.3.2 isPolicyEnforced()

```
bool PolicyCheckerHandler::isPolicyEnforced () const [inline]
```

Determines if a policy was enforced.

Returns

True if the policy was enforced, false otherwise.

4.49.3.3 setPolicyEnforced()

```
void PolicyCheckerHandler::setPolicyEnforced (
    bool enforced ) [inline]
```

Sets the policy enforcement state.

Parameters

<i>enforced</i>	True if the policy was enforced, false otherwise.
-----------------	---

The documentation for this class was generated from the following file:

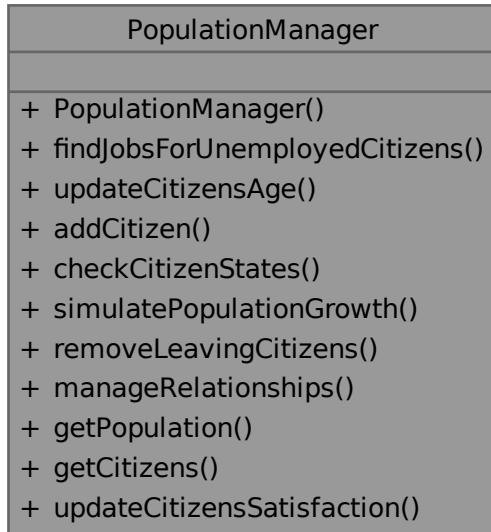
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PolicyCheckerHandler.h](#)

4.50 PopulationManager Class Reference

Manages the population of citizens.

```
#include <PopulationManager.h>
```

Collaboration diagram for PopulationManager:



Public Member Functions

- **PopulationManager ()**
Constructor for PopulationManager.
- **void findJobsForUnemployedCitizens (BuildingManager &buildingManager)**
Finds jobs for unemployed citizens using the BuildingManager.
- **void updateCitizensAge ()**
Updates the age of all citizens each cycle.
- **void addCitizen (std::shared_ptr< Citizen > citizen)**
Adds a citizen to the population.
- **void checkCitizenStates ()**
Checks and updates citizens' states based on satisfaction levels.
- **void simulatePopulationGrowth ()**
Simulates population growth randomly by adding or removing citizens.
- **void removeLeavingCitizens ()**
Removes citizens who are leaving the city.
- **void manageRelationships ()**
Manages relationships between citizens.
- **int getPopulation ()**
Gets the current population count.
- **const std::vector< std::shared_ptr< Citizen > > & getCitizens () const**
Gets the list of all citizens.
- **void updateCitizensSatisfaction ()**
Updates satisfaction for all eligible citizens.

4.50.1 Detailed Description

Manages the population of citizens.

The [PopulationManager](#) class manages the population of citizens, including adding, removing, and updating citizens. It also handles job assignments, relationship management, and satisfaction updates.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 PopulationManager()

```
PopulationManager::PopulationManager ( )
```

Constructor for [PopulationManager](#).

Initializes the [PopulationManager](#) with job and housing satisfaction strategies.

4.50.3 Member Function Documentation

4.50.3.1 addCitizen()

```
void PopulationManager::addCitizen (
    std::shared_ptr< Citizen > citizen )
```

Adds a citizen to the population.

Parameters

<i>citizen</i>	The citizen to be added.
----------------	--------------------------

4.50.3.2 findJobsForUnemployedCitizens()

```
void PopulationManager::findJobsForUnemployedCitizens (
    BuildingManager & buildingManager )
```

Finds jobs for unemployed citizens using the [BuildingManager](#).

Parameters

<i>buildingManager</i>	The BuildingManager to use for finding jobs.
------------------------	--

4.50.3.3 getCitizens()

```
const std::vector< std::shared_ptr< Citizen > > & PopulationManager::getCitizens ( ) const
```

Gets the list of all citizens.

Returns

A vector of shared pointers to all citizens.

4.50.3.4 getPopulation()

```
int PopulationManager::getPopulation ( )
```

Gets the current population count.

Returns

The current population count.

The documentation for this class was generated from the following files:

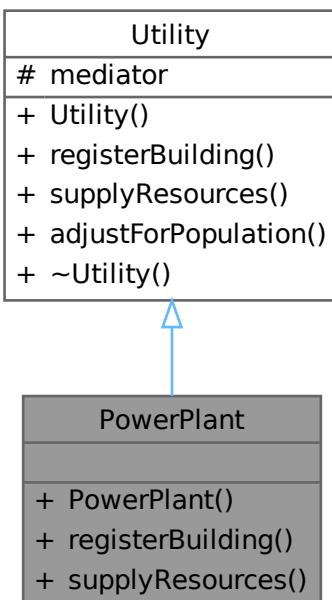
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PopulationManager.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PopulationManager.cpp](#)

4.51 PowerPlant Class Reference

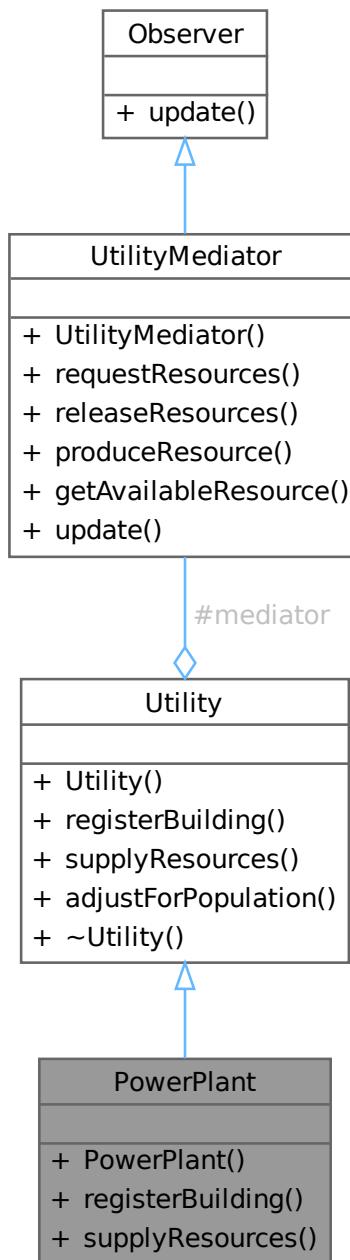
Represents a power plant that supplies power to buildings.

```
#include <PowerPlant.h>
```

Inheritance diagram for PowerPlant:



Collaboration diagram for PowerPlant:



Public Member Functions

- `PowerPlant (UtilityMediator *mediator)`
Constructor for PowerPlant.
- `void registerBuilding (Building *building) override`
Registers a building to receive power.
- `void supplyResources (Building *building) override`
Supplies power to a building.

Public Member Functions inherited from Utility

- **Utility (UtilityMediator *mediator)**
Constructor for the Utility class.
- virtual void **adjustForPopulation** (int newPopulation)=0
Adjusts the utility service for a new population size.
- virtual ~**Utility** ()=default
Virtual destructor for the Utility class.

Additional Inherited Members

Protected Attributes inherited from Utility

- **UtilityMediator * mediator**
Reference to the mediator for managing resources.

4.51.1 Detailed Description

Represents a power plant that supplies power to buildings.

The **PowerPlant** class inherits from the **Utility** class and provides methods to register buildings, supply resources, and generate electricity.

4.51.2 Constructor & Destructor Documentation

4.51.2.1 PowerPlant()

```
PowerPlant::PowerPlant (
    UtilityMediator * mediator )
```

Constructor for **PowerPlant**.

Default constructor for **PowerPlant**.

Parameters

mediator	Pointer to the UtilityMediator .
-----------------	---

4.51.3 Member Function Documentation

4.51.3.1 registerBuilding()

```
void PowerPlant::registerBuilding (
    Building * building ) [override], [virtual]
```

Registers a building to receive power.

Parameters

<i>building</i>	Pointer to the building to be registered.
-----------------	---

Implements [Utility](#).

4.51.3.2 supplyResources()

```
void PowerPlant::supplyResources (
    Building * building ) [override], [virtual]
```

Supplies power to a building.

Parameters

<i>building</i>	Pointer to the building to receive power.
-----------------	---

Implements [Utility](#).

The documentation for this class was generated from the following files:

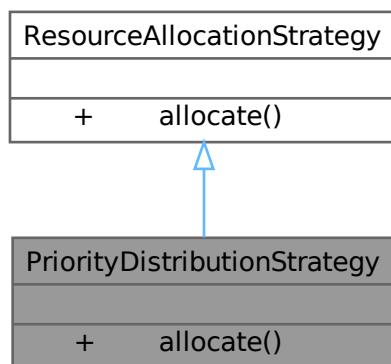
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PowerPlant.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[PowerPlant.cpp](#)

4.52 PriorityDistributionStrategy Class Reference

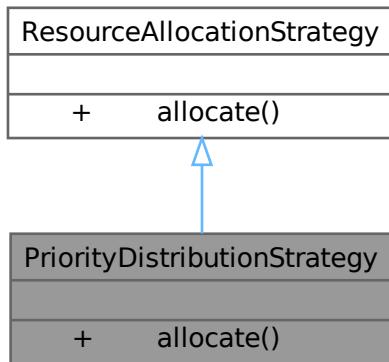
Strategy for priority-based resource allocation.

```
#include <ResourceAllocationStrategy.h>
```

Inheritance diagram for PriorityDistributionStrategy:



Collaboration diagram for PriorityDistributionStrategy:



Public Member Functions

- bool `allocate (ResourceType type, int quantity) override`
Allocates resources based on priority.

4.52.1 Detailed Description

Strategy for priority-based resource allocation.

The [PriorityDistributionStrategy](#) class implements the [ResourceAllocationStrategy](#) interface and provides logic for priority-based allocation of resources.

4.52.2 Member Function Documentation

4.52.2.1 `allocate()`

```
bool PriorityDistributionStrategy::allocate (
    ResourceType type,
    int quantity ) [inline], [override], [virtual]
```

Allocates resources based on priority.

Parameters

<code>type</code>	The type of the resource.
<code>quantity</code>	The quantity of the resource to allocate.

Returns

True if the allocation was successful, false otherwise.

Implements [ResourceAllocationStrategy](#).

The documentation for this class was generated from the following file:

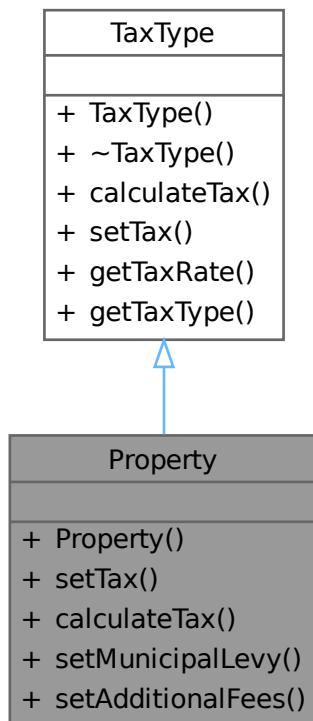
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceAllocationStrategy.h](#)

4.53 Property Class Reference

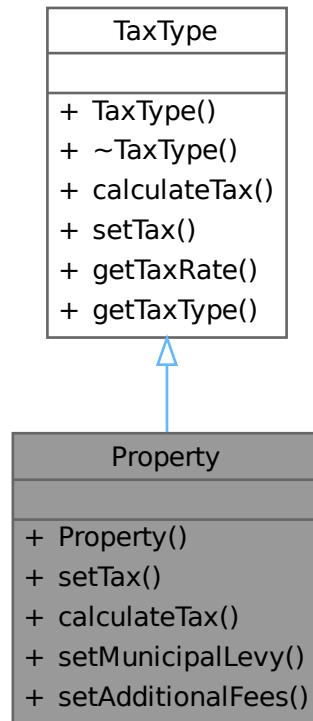
Represents a property with specific attributes and behaviors related to property tax calculation.

```
#include <Property.h>
```

Inheritance diagram for Property:



Collaboration diagram for Property:



Public Member Functions

- **Property** (double rate, double levy, double fees)
Constructor to initialize base property tax rate, municipal levy, and additional fees.
- void **setTax** (double rate) override
Sets the base property tax rate.
- double **calculateTax** (double propertyValue) override
Calculates the total property tax for a given property value.
- void **setMunicipalLevy** (double levy)
Sets the municipal levy rate.
- void **setAdditionalFees** (double fees)
Sets additional fixed fees.

Public Member Functions inherited from **TaxType**

- **TaxType** (double rate, char type)
*Constructs a new **TaxType** object.*
- virtual ~**TaxType** ()
Virtual Destructor.
- virtual double **getTaxRate** ()
Gets the current tax rate.
- char **getTaxType** ()
Gets the tax type identifier.

4.53.1 Detailed Description

Represents a property with specific attributes and behaviors related to property tax calculation.

The [Property](#) class inherits from the [TaxType](#) class and provides methods to set tax rates, calculate property tax, and set additional fees.

4.53.2 Constructor & Destructor Documentation

4.53.2.1 [Property\(\)](#)

```
Property::Property (
    double rate,
    double levy,
    double fees )
```

Constructor to initialize base property tax rate, municipal levy, and additional fees.

Parameters

<i>rate</i>	The base property tax rate.
<i>levy</i>	The municipal levy rate.
<i>fees</i>	Additional fixed fees.

4.53.3 Member Function Documentation

4.53.3.1 [calculateTax\(\)](#)

```
double Property::calculateTax (
    double propertyValue ) [override], [virtual]
```

Calculates the total property tax for a given property value.

Parameters

<i>propertyValue</i>	The value of the property.
----------------------	----------------------------

Returns

The total property tax.

Reimplemented from [TaxType](#).

4.53.3.2 [setAdditionalFees\(\)](#)

```
void Property::setAdditionalFees (
    double fees )
```

Sets additional fixed fees.

Parameters

<i>fees</i>	The additional fixed fees.
-------------	----------------------------

4.53.3.3 setMunicipalLevy()

```
void Property::setMunicipalLevy (
    double levy )
```

Sets the municipal levy rate.

Parameters

<i>levy</i>	The municipal levy rate.
-------------	--------------------------

4.53.3.4 setTax()

```
void Property::setTax (
    double rate ) [override], [virtual]
```

Sets the base property tax rate.

Parameters

<i>rate</i>	The base property tax rate.
-------------	-----------------------------

Reimplemented from [TaxType](#).

The documentation for this class was generated from the following files:

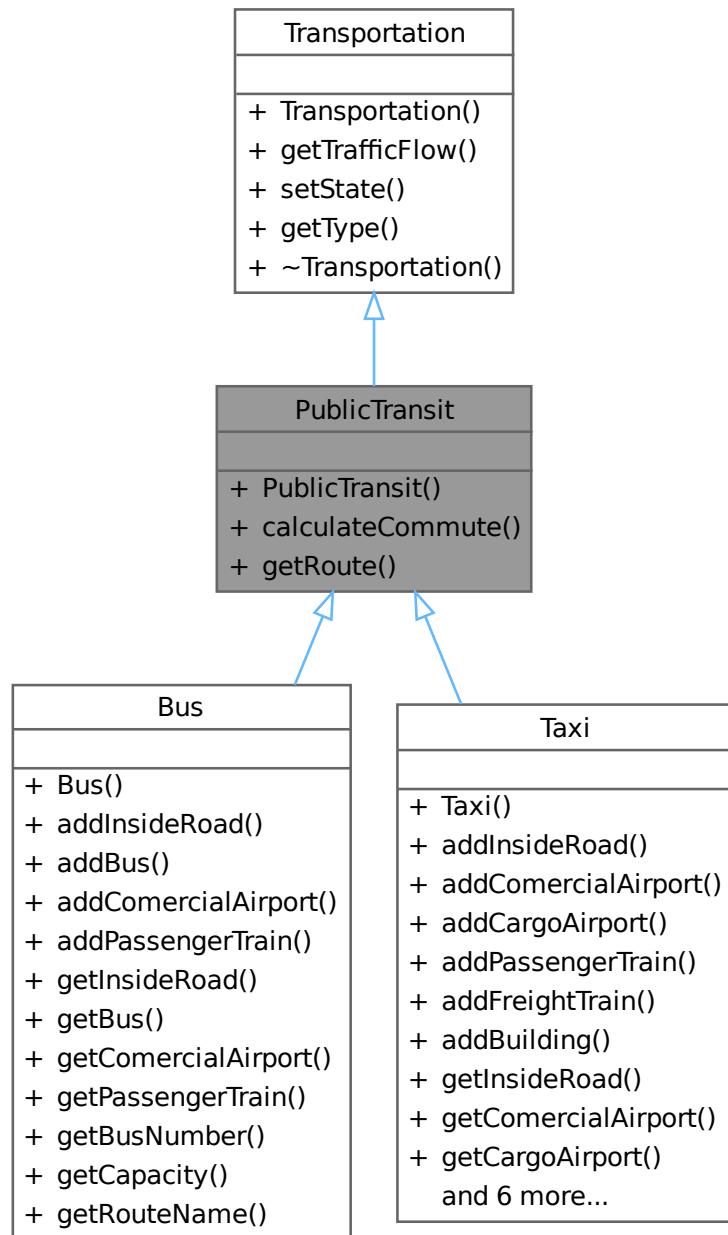
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Property.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Property.cpp](#)

4.54 PublicTransit Class Reference

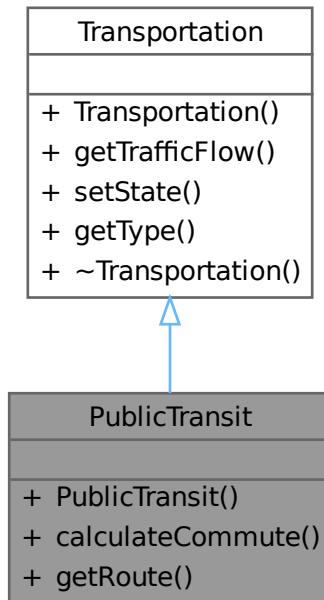
A class representing public transit transportation.

```
#include <PublicTransit.h>
```

Inheritance diagram for PublicTransit:



Collaboration diagram for PublicTransit:



Public Member Functions

- **PublicTransit** (char state, std::string route, char type)
Constructor for the [PublicTransit](#) class.
- float **calculateCommute** ()
Calculate the commute time for the public transit.
- std::string **getRoute** ()
Get the route of the public transit.

Public Member Functions inherited from [Transportation](#)

- **Transportation** (char state, char type)
Constructor for the [Transportation](#) class.
- float **getTrafficFlow** ()
Gets the current traffic flow.
- bool **setState** (char state)
Sets the traffic flow state.
- char **getType** ()
Gets the type of transportation.
- **~Transportation** ()
Destructor for the [Transportation](#) class.

4.54.1 Detailed Description

A class representing public transit transportation.

The [PublicTransit](#) class inherits from the [Transportation](#) class and adds specific attributes and methods related to public transit systems.

4.54.2 Constructor & Destructor Documentation

4.54.2.1 PublicTransit()

```
PublicTransit::PublicTransit (
    char state,
    std::string route,
    char type )
```

Constructor for the [PublicTransit](#) class.

Constructor for [PublicTransit](#).

Parameters

<i>state</i>	The state of the transportation.
<i>route</i>	The route of the public transit.
<i>type</i>	The type of the transportation.
<i>state</i>	The state of the public transit.
<i>route</i>	The route of the public transit.
<i>type</i>	The type of the public transit.

4.54.3 Member Function Documentation

4.54.3.1 calculateCommute()

```
float PublicTransit::calculateCommute ( )
```

Calculate the commute time for the public transit.

Returns

The calculated commute time.

4.54.3.2 getRoute()

```
std::string PublicTransit::getRoute ( )
```

Get the route of the public transit.

Gets the route of the public transit.

Returns

The route as a string.
The route of the public transit.

The documentation for this class was generated from the following files:

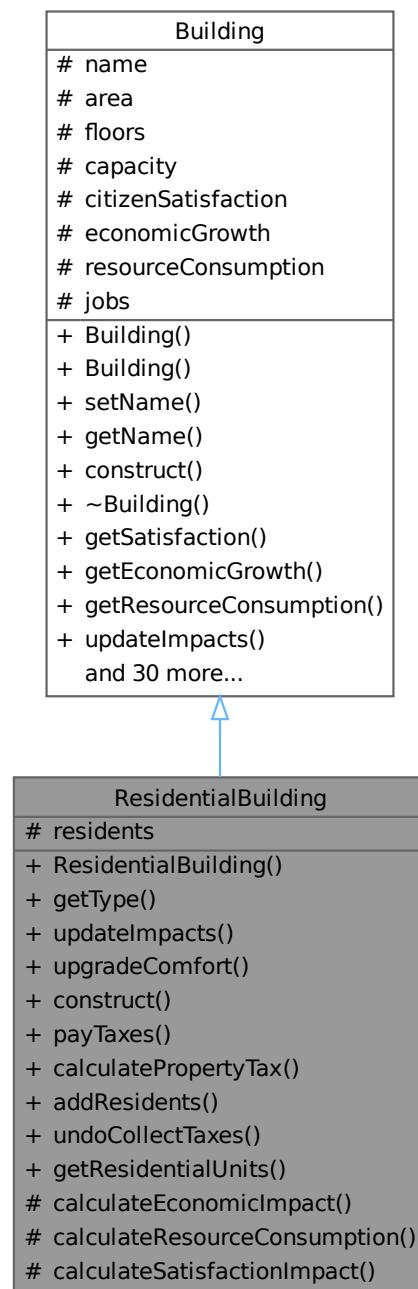
- /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.cpp

4.55 ResidentialBuilding Class Reference

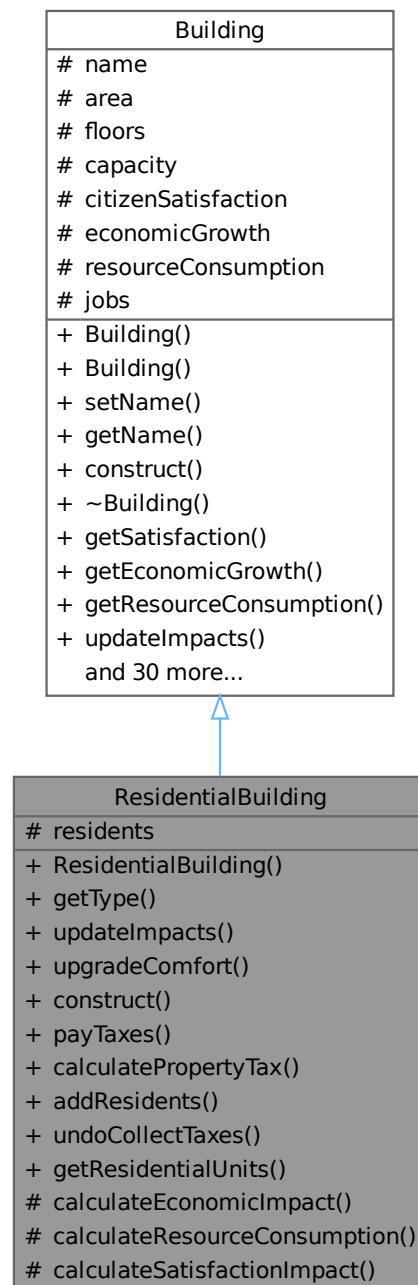
Represents a residential building in the simulation.

```
#include <ResidentialBuilding.h>
```

Inheritance diagram for ResidentialBuilding:



Collaboration diagram for ResidentialBuilding:



Public Member Functions

- **ResidentialBuilding** (const std::string &**name**, float **area**, int **floors**, int **capacity**, float **citizenSatisfaction**, float **economicGrowth**, float **resourceConsumption**, int **residentialUnits**, float **comfortLevel**)
*Constructs a new **ResidentialBuilding** object.*
- std::string **getType** () const override
Gets the type of the building.

- void `updateImpacts () override`
Updates the impacts of the building.
- void `upgradeComfort (float comfort)`
Upgrades the comfort level of the building.
- void `construct () override`
Constructs the building.
- double `payTaxes (TaxType *taxType)`
Pays taxes based on the given tax type.
- double `calculatePropertyTax ()`
Calculates the property tax for the building.
- void `addResidents (Citizen *citizen)`
Adds a resident to the building.
- void `undoCollectTaxes ()`
Undoes the collection of taxes.
- int `getResidentialUnits () const`
Gets the number of residential units in the building.

Public Member Functions inherited from [Building](#)

- `Building (const std::string &name, float area, int floors, int capacity, float satisfactionImpact, float growthImpact, float consumption)`
Main constructor for setting attributes directly.
- `Building (int Builder)`
Alternative constructor using an integer parameter, potentially for [Builder](#) pattern integration.
- void `setName (const std::string &name)`
Sets the name of the building.
- std::string `getName () const`
Gets the name of the building.
- virtual ~[Building](#) ()=default
Virtual destructor for the [Building](#) class.
- float `getSatisfaction () const`
Gets the citizen satisfaction impact of the building.
- float `getEconomicGrowth () const`
Gets the economic growth impact of the building.
- float `getResourceConsumption () const`
Gets the resource consumption of the building.
- void `addJob (std::shared_ptr< Jobs > job)`
Adds a job to the building.
- void `listJobs () const`
Lists all jobs in the building.
- bool `hireEmployee (const std::string &jobTitle)`
Hires an employee for a job if available.
- void `releaseEmployee (const std::string &jobTitle)`
Releases an employee from a job.
- std::shared_ptr< Jobs > `getAvailableJob ()`
Provides access to an available job.
- void `displayJobInfo (const std::string &jobTitle) const`
Displays job information.
- const std::vector< std::shared_ptr< Jobs > > & `getJobs () const`
Returns the job list as a const reference.

- `Building` (const std::string &`name`, float `area`, int `floors`, int `capacity`, float `satisfactionImpact`, float `growthImpact`, float `consumption`)
Main constructor for setting attributes directly.
- `Building` (int `Builder`)
Alternative constructor using an integer parameter, potentially for `Builder` pattern integration.
- void `setName` (const std::string &`name`)
Sets the name of the building.
- std::string `getName` () const
Gets the name of the building.
- virtual ~`Building` ()=default
Virtual destructor for the `Building` class.
- float `getSatisfaction` () const
Gets the citizen satisfaction impact of the building.
- float `getEconomicGrowth` () const
Gets the economic growth impact of the building.
- float `getResourceConsumption` () const
Gets the resource consumption of the building.
- void `addJob` (std::shared_ptr< `Jobs` > `job`)
Adds a job to the building.
- void `listJobs` () const
Lists all jobs in the building.
- bool `hireEmployee` (const std::string &`jobTitle`)
Hires an employee for a job if available.
- void `releaseEmployee` (const std::string &`jobTitle`)
Releases an employee from a job.
- std::shared_ptr< `Jobs` > `getAvailableJob` ()
Provides access to an available job.
- void `displayJobInfo` (const std::string &`jobTitle`) const
Displays job information.
- const std::vector< std::shared_ptr< `Jobs` > > & `getJobs` () const
Returns the job list as a const reference.

Protected Member Functions

- void `calculateEconomicImpact` () override
Calculates the economic impact of the building.
- void `calculateResourceConsumption` () override
Calculates the resource consumption of the building.
- void `calculateSatisfactionImpact` () override
Calculates the satisfaction impact of the building.

Protected Attributes

- std::vector< `Citizen` * > `residents`
The list of residents in the building.

Protected Attributes inherited from Building

- std::string **name**
Name of the building.
- float **area**
Area of the building.
- int **floors**
Number of floors in the building.
- int **capacity**
Capacity of the building.
- float **citizenSatisfaction**
Citizen satisfaction impact of the building.
- float **economicGrowth**
Economic growth impact of the building.
- float **resourceConsumption**
Resource consumption of the building.
- std::vector< std::shared_ptr< **Jobs** > > **jobs**
Collection of jobs as shared pointers.

4.55.1 Detailed Description

Represents a residential building in the simulation.

The [ResidentialBuilding](#) class inherits from the [Building](#) class and includes additional attributes and methods specific to residential buildings, such as residential units, comfort level, and methods to manage residents and calculate taxes.

4.55.2 Constructor & Destructor Documentation

4.55.2.1 [ResidentialBuilding\(\)](#)

```
ResidentialBuilding::ResidentialBuilding (
    const std::string & name,
    float area,
    int floors,
    int capacity,
    float citizenSatisfaction,
    float economicGrowth,
    float resourceConsumption,
    int residentialUnits,
    float comfortLevel )
```

Constructs a new [ResidentialBuilding](#) object.

Constructor for [ResidentialBuilding](#).

Parameters

<i>name</i>	The name of the building.
<i>area</i>	The area of the building.
<i>floors</i>	The number of floors in the building.

Parameters

<i>capacity</i>	The capacity of the building.
<i>citizenSatisfaction</i>	The satisfaction level of the citizens.
<i>economicGrowth</i>	The economic growth associated with the building.
<i>resourceConsumption</i>	The resource consumption of the building.
<i>residentialUnits</i>	The number of residential units in the building.
<i>comfortLevel</i>	The comfort level of the building.
<i>name</i>	The name of the residential building.
<i>area</i>	The area of the residential building.
<i>floors</i>	The number of floors in the residential building.
<i>capacity</i>	The capacity of the residential building.
<i>citizenSatisfaction</i>	The citizen satisfaction level.
<i>economicGrowth</i>	The economic growth contributed by the building.
<i>resourceConsumption</i>	The resource consumption of the building.
<i>residentialUnits</i>	The number of residential units in the building.
<i>comfortLevel</i>	The comfort level of the building.

4.55.3 Member Function Documentation**4.55.3.1 addResidents()**

```
void ResidentialBuilding::addResidents (
    Citizen * citizen )
```

Adds a resident to the building.

Adds residents to the building.

Parameters

<i>citizen</i>	The citizen to be added.
<i>citizen</i>	Pointer to the citizen to be added.

4.55.3.2 calculateEconomicImpact()

```
void ResidentialBuilding::calculateEconomicImpact ( ) [override], [protected], [virtual]
```

Calculates the economic impact of the building.

Implements [Building](#).

4.55.3.3 calculatePropertyTax()

```
double ResidentialBuilding::calculatePropertyTax ( )
```

Calculates the property tax for the building.

Returns

- The property tax amount.
- The calculated property tax.

4.55.3.4 calculateResourceConsumption()

```
void ResidentialBuilding::calculateResourceConsumption ( ) [override], [protected], [virtual]
```

Calculates the resource consumption of the building.

Implements [Building](#).

4.55.3.5 calculateSatisfactionImpact()

```
void ResidentialBuilding::calculateSatisfactionImpact ( ) [override], [protected], [virtual]
```

Calculates the satisfaction impact of the building.

Implements [Building](#).

4.55.3.6 construct()

```
void ResidentialBuilding::construct ( ) [override], [virtual]
```

Constructs the building.

Implements [Building](#).

4.55.3.7 getResidentialUnits()

```
int ResidentialBuilding::getResidentialUnits ( ) const
```

Gets the number of residential units in the building.

Returns

- The number of residential units.

4.55.3.8 getType()

```
std::string ResidentialBuilding::getType ( ) const [override], [virtual]
```

Gets the type of the building.

Returns

- The type of the building as a string.
- The type of the building.

Implements [Building](#).

4.55.3.9 payTaxes()

```
double ResidentialBuilding::payTaxes ( TaxType * taxType ) [virtual]
```

Pays taxes based on the given tax type.

Pays taxes for the building.

Parameters

<i>taxType</i>	The type of tax to be paid.
----------------	-----------------------------

Returns

The amount of taxes paid.

Parameters

<i>taxType</i>	The type of tax to calculate.
----------------	-------------------------------

Returns

The amount of taxes paid.

Implements [Building](#).

4.55.3.10 undoCollectTaxes()

```
void ResidentialBuilding::undoCollectTaxes ( ) [virtual]
```

Undoes the collection of taxes.

Undoes the tax collection from the building.

Implements [Building](#).

4.55.3.11 updateImpacts()

```
void ResidentialBuilding::updateImpacts ( ) [override], [virtual]
```

Updates the impacts of the building.

Updates the impacts of the building by calculating economic impact, resource consumption, and satisfaction impact.

Implements [Building](#).

4.55.3.12 upgradeComfort()

```
void ResidentialBuilding::upgradeComfort (
    float comfort )
```

Upgrades the comfort level of the building.

Parameters

<i>comfort</i>	The new comfort level.
<i>comfort</i>	The new comfort level to upgrade to.

The documentation for this class was generated from the following files:

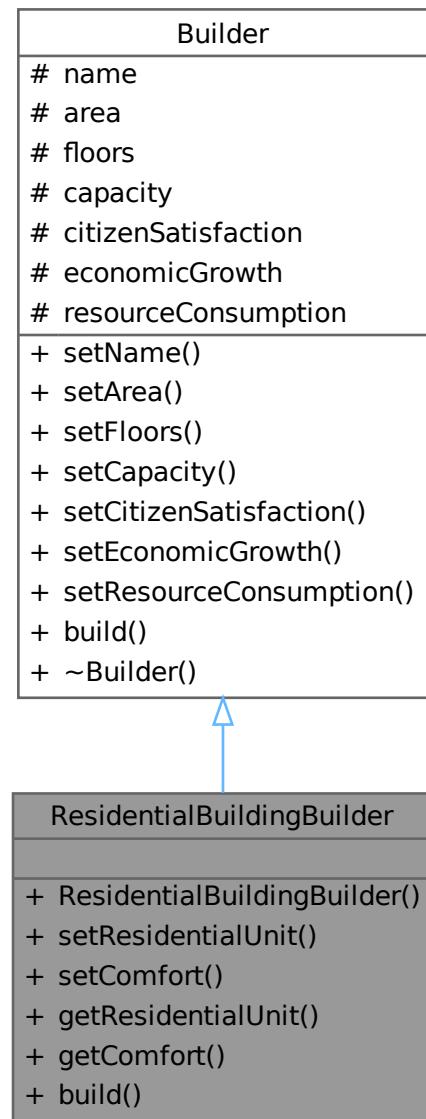
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResidentialBuilding.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResidentialBuilding.cpp](#)

4.56 ResidentialBuildingBuilder Class Reference

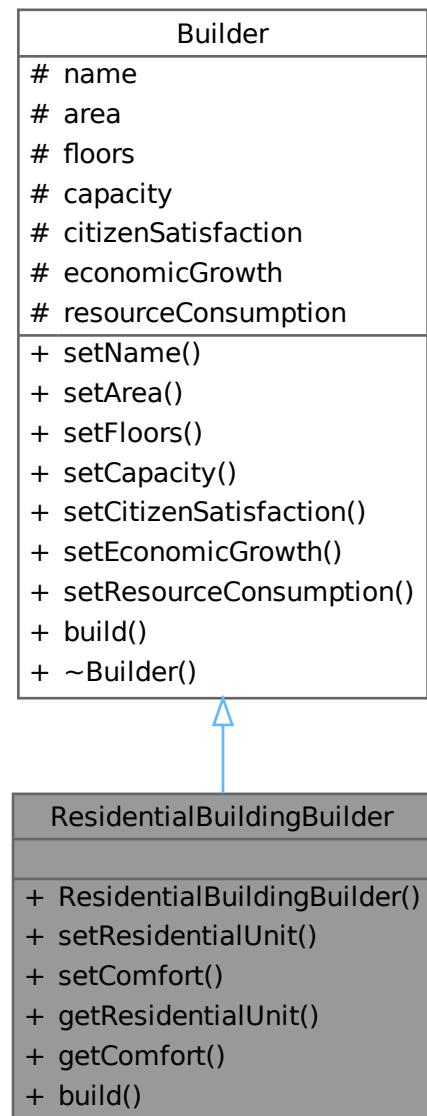
[Builder](#) class for constructing [ResidentialBuilding](#) objects.

```
#include <ResidentialBuildingBuilder.h>
```

Inheritance diagram for ResidentialBuildingBuilder:



Collaboration diagram for ResidentialBuildingBuilder:



Public Member Functions

- **ResidentialBuildingBuilder ()**
Constructs a new `ResidentialBuildingBuilder` object.
- **ResidentialBuildingBuilder & setResidentialUnit (int unit)**
Sets the number of residential units in the building.
- **ResidentialBuildingBuilder & setComfort (float comfort)**
Sets the comfort level of the building.
- **int getResidentialUnit ()**
Gets the number of residential units in the building.

- float `getComfort ()`
Gets the comfort level of the building.
- std::unique_ptr< Building > `build ()` override
Builds the residential building and sets all the attributes of the building.

Public Member Functions inherited from `Builder`

- `Builder & setName (string name)`
Sets the name of the building.
- `Builder & setArea (float area)`
Sets the area of the building.
- `Builder & setFloors (int floors)`
Sets the number of floors in the building.
- `Builder & setCapacity (int capacity)`
Sets the capacity of the building.
- `Builder & setCitizenSatisfaction (float citizenSatisfaction)`
Sets the citizen satisfaction of the building.
- `Builder & setEconomicGrowth (float economicGrowth)`
Sets the economic growth of the building.
- `Builder & setResourceConsumption (float resourceConsumption)`
Sets the resource consumption of the building.
- virtual ~`Builder` ()=default
Virtual destructor for the `Builder` class.

Additional Inherited Members

Protected Attributes inherited from `Builder`

- string `name`
Name of the building.
- float `area` = 0.0f
Area of the building.
- int `floors` = 0
Number of floors in the building.
- int `capacity` = 0
Capacity of the building.
- float `citizenSatisfaction` = 0.0f
Citizen satisfaction of the building.
- float `economicGrowth` = 0.0f
Economic growth of the building.
- float `resourceConsumption` = 0.0f
Resource consumption of the building.

4.56.1 Detailed Description

`Builder` class for constructing `ResidentialBuilding` objects.

The `ResidentialBuildingBuilder` class provides methods to set specific attributes for a `ResidentialBuilding` object and then construct the object.

4.56.2 Member Function Documentation

4.56.2.1 build()

```
std::unique_ptr< Building > ResidentialBuildingBuilder::build ( ) [override], [virtual]
```

Builds the residential building and sets all the attributes of the building.

Returns

A unique pointer to the constructed [ResidentialBuilding](#) object.

Implements [Builder](#).

4.56.2.2 getComfort()

```
float ResidentialBuildingBuilder::getComfort ( )
```

Gets the comfort level of the building.

Returns

The comfort level.

4.56.2.3 getResidentialUnit()

```
int ResidentialBuildingBuilder::getResidentialUnit ( )
```

Gets the number of residential units in the building.

Returns

The number of residential units.

4.56.2.4 setComfort()

```
ResidentialBuildingBuilder & ResidentialBuildingBuilder::setComfort ( float comfort )
```

Sets the comfort level of the building.

Parameters

<i>comfort</i>	The comfort level.
----------------	--------------------

Returns

A reference to the [ResidentialBuildingBuilder](#) object.

4.56.2.5 setResidentialUnit()

```
ResidentialBuildingBuilder & ResidentialBuildingBuilder::setResidentialUnit ( int unit )
```

Sets the number of residential units in the building.

Parameters

<i>unit</i>	The number of residential units.
-------------	----------------------------------

Returns

A reference to the [ResidentialBuildingBuilder](#) object.

The documentation for this class was generated from the following files:

- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResidentialBuildingBuilder.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResidentialBuildingBuilder.cpp](#)

4.57 Resource Class Reference

Represents a resource with a specific type and quantity.

```
#include <Resource.h>
```

Collaboration diagram for Resource:

Resource
+ Resource()
+ allocate()
+ release()
+ getType()
+ getQuantity()

Public Member Functions

- **Resource** (`ResourceType` `resourceType`, `int initialQuantity`)

Constructs a new `Resource` object.
- **bool allocate** (`int amount`)

Allocates a specified amount of the resource.
- **void release** (`int amount`)

Releases a specified amount of the resource.
- **ResourceType getType** () `const`

Gets the type of the resource.
- **int getQuantity** () `const`

Gets the quantity of the resource.

4.57.1 Detailed Description

Represents a resource with a specific type and quantity.

The `Resource` class provides methods to allocate and release the resource.

4.57.2 Constructor & Destructor Documentation

4.57.2.1 Resource()

```
Resource::Resource (
    ResourceType resourceType,
    int initialQuantity ) [inline]
```

Constructs a new `Resource` object.

Parameters

<code>resourceType</code>	The type of the resource.
<code>initialQuantity</code>	The initial quantity of the resource.

4.57.3 Member Function Documentation

4.57.3.1 allocate()

```
bool Resource::allocate (
    int amount ) [inline]
```

Allocates a specified amount of the resource.

Parameters

<code>amount</code>	The amount to allocate.
---------------------	-------------------------

Returns

True if the allocation was successful, false otherwise.

4.57.3.2 getQuantity()

```
int Resource::getQuantity ( ) const [inline]
```

Gets the quantity of the resource.

Returns

The quantity of the resource.

4.57.3.3 getType()

```
ResourceType Resource::getType ( ) const [inline]
```

Gets the type of the resource.

Returns

The type of the resource.

4.57.3.4 release()

```
void Resource::release ( int amount ) [inline]
```

Releases a specified amount of the resource.

Parameters

<i>amount</i>	The amount to release.
---------------	------------------------

The documentation for this class was generated from the following file:

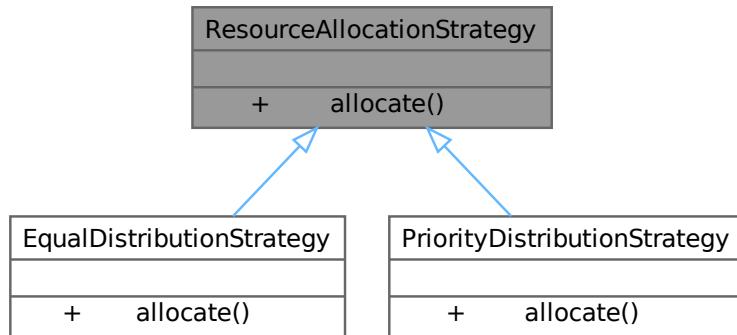
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Resource.h](#)

4.58 ResourceAllocationStrategy Class Reference

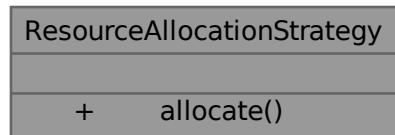
Interface for resource allocation strategies.

```
#include <ResourceAllocationStrategy.h>
```

Inheritance diagram for ResourceAllocationStrategy:



Collaboration diagram for ResourceAllocationStrategy:



Public Member Functions

- virtual bool `allocate (ResourceType type, int quantity)=0`
Allocates a specified quantity of a resource type.

4.58.1 Detailed Description

Interface for resource allocation strategies.

The `ResourceAllocationStrategy` class provides a pure virtual method for allocating resources.

4.58.2 Member Function Documentation

4.58.2.1 `allocate()`

```

virtual bool ResourceAllocationStrategy::allocate (
    ResourceType type,
    int quantity ) [pure virtual]
  
```

Allocates a specified quantity of a resource type.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource to allocate.

Returns

True if the allocation was successful, false otherwise.

Implemented in [PriorityDistributionStrategy](#), and [EqualDistributionStrategy](#).

The documentation for this class was generated from the following file:

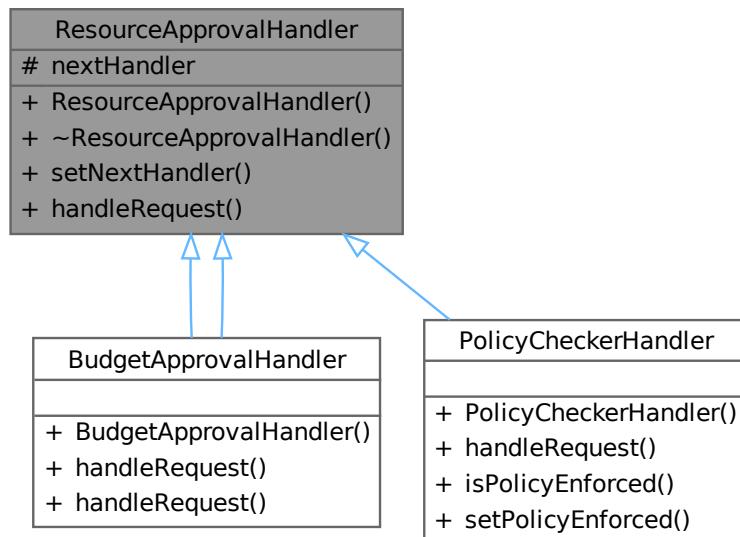
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceAllocationStrategy.h](#)

4.59 ResourceApprovalHandler Class Reference

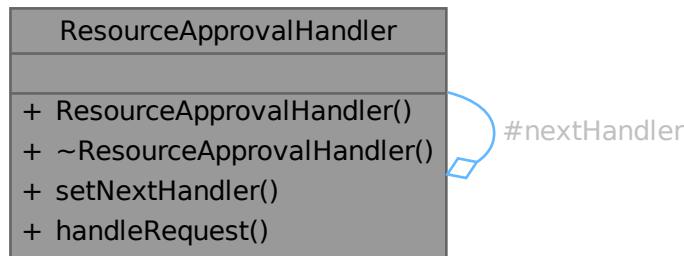
Base class for handling resource approval requests.

```
#include <ResourceApprovalHandler.h>
```

Inheritance diagram for ResourceApprovalHandler:



Collaboration diagram for ResourceApprovalHandler:



Public Member Functions

- **ResourceApprovalHandler ()**
Constructs a new [ResourceApprovalHandler](#) object.
- **virtual ~ResourceApprovalHandler ()**
Destroys the [ResourceApprovalHandler](#) object.
- **void setNextHandler ([ResourceApprovalHandler](#) *handler)**
Sets the next handler in the chain.
- **virtual bool handleRequest ([ResourceType](#) type, int quantity)**
Handles the resource request.

Protected Attributes

- **ResourceApprovalHandler * nextHandler**
Pointer to the next handler in the chain.

4.59.1 Detailed Description

Base class for handling resource approval requests.

The [ResourceApprovalHandler](#) class provides a chain of responsibility pattern for approving resource requests. Derived classes implement specific approval logic.

4.59.2 Member Function Documentation

4.59.2.1 handleRequest()

```

virtual bool ResourceApprovalHandler::handleRequest (
    ResourceType type,
    int quantity ) [inline], [virtual]

```

Handles the resource request.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource.

Returns

True if the request is approved, false otherwise.

Reimplemented in [BudgetApprovalHandler](#), [PolicyCheckerHandler](#), and [BudgetApprovalHandler](#).

4.59.2.2 setNextHandler()

```
void ResourceApprovalHandler::setNextHandler (
    ResourceApprovalHandler * handler ) [inline]
```

Sets the next handler in the chain.

Parameters

<i>handler</i>	Pointer to the next handler.
----------------	------------------------------

The documentation for this class was generated from the following file:

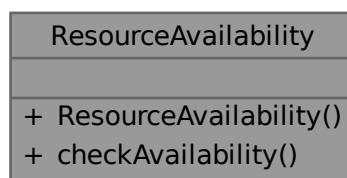
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceApprovalHandler.h](#)

4.60 ResourceAvailability Class Reference

Checks the availability of a specific resource type and quantity.

```
#include <ResourceAvailability.h>
```

Collaboration diagram for ResourceAvailability:



Public Member Functions

- [ResourceAvailability \(ResourceManager *manager\)](#)
Constructs a new ResourceAvailability object.
- [bool checkAvailability \(ResourceType type, int quantity\) const](#)
Checks if a given quantity of a specific resource type is available.

4.60.1 Detailed Description

Checks the availability of a specific resource type and quantity.

The [ResourceAvailability](#) class provides a method to check if a given quantity of a specific resource type is available.

4.60.2 Constructor & Destructor Documentation

4.60.2.1 ResourceAvailability()

```
ResourceAvailability::ResourceAvailability (
    ResourceManager * manager ) [inline]
```

Constructs a new [ResourceAvailability](#) object.

Parameters

<code>manager</code>	Pointer to the resource manager.
----------------------	----------------------------------

4.60.3 Member Function Documentation

4.60.3.1 checkAvailability()

```
bool ResourceAvailability::checkAvailability (
    ResourceType type,
    int quantity ) const [inline]
```

Checks if a given quantity of a specific resource type is available.

Parameters

<code>type</code>	The type of the resource.
<code>quantity</code>	The quantity of the resource.

Returns

True if the resource is available, false otherwise.

The documentation for this class was generated from the following file:

- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceAvailability.h](#)

4.61 ResourceManager Class Reference

Manages resources, handles resource allocation and release, and notifies observers.

```
#include <ResourceManager.h>
```

Collaboration diagram for ResourceManager:

ResourceManager
+ ResourceManager() + allocateResources() + releaseResources() + setAllocationStrategy() + getBudget() + addObserver() + getResource() + addResource()

Public Member Functions

- **ResourceManager (int initialBudget)**
Constructs a new ResourceManager object.
- **bool allocateResources (ResourceType type, int quantity)**
Allocates resources of a specific type and quantity.
- **void releaseResources (ResourceType type, int quantity)**
Releases resources of a specific type and quantity.
- **void setAllocationStrategy (ResourceAllocationStrategy *strategy)**
Sets the resource allocation strategy.
- **int getBudget () const**
Gets the current budget.
- **void addObserver (Observer *observer)**
Adds an observer to the resource manager.
- **Resource * getResource (ResourceType type) const**
Gets a resource of a specific type.
- **void addResource (ResourceType type, Resource *resource)**
Adds a resource to the resource manager.

4.61.1 Detailed Description

Manages resources, handles resource allocation and release, and notifies observers.

The [ResourceManager](#) class provides methods to allocate and release resources, set allocation strategies, manage budget, and notify observers about resource changes.

4.61.2 Constructor & Destructor Documentation

4.61.2.1 ResourceManager()

```
ResourceManager::ResourceManager (
    int initialBudget ) [inline]
```

Constructs a new [ResourceManager](#) object.

Parameters

<i>initialBudget</i>	The initial budget for the resource manager.
----------------------	--

4.61.3 Member Function Documentation

4.61.3.1 addObserver()

```
void ResourceManager::addObserver (
    Observer * observer ) [inline]
```

Adds an observer to the resource manager.

Parameters

<i>observer</i>	Pointer to the observer.
-----------------	--------------------------

4.61.3.2 addResource()

```
void ResourceManager::addResource (
    ResourceType type,
    Resource * resource ) [inline]
```

Adds a resource to the resource manager.

Parameters

<i>type</i>	The type of the resource.
<i>resource</i>	Pointer to the Resource object.

4.61.3.3 allocateResources()

```
bool ResourceManager::allocateResources (
    ResourceType type,
    int quantity ) [inline]
```

Allocates resources of a specific type and quantity.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource to allocate.

Returns

True if the allocation was successful, false otherwise.

4.61.3.4 getBudget()

```
int ResourceManager::getBudget ( ) const [inline]
```

Gets the current budget.

Returns

The current budget.

4.61.3.5 getResource()

```
Resource * ResourceManager::getResource (
    ResourceType type ) const [inline]
```

Gets a resource of a specific type.

Parameters

<i>type</i>	The type of the resource.
-------------	---------------------------

Returns

Pointer to the [Resource](#) object, or nullptr if not found.

4.61.3.6 releaseResources()

```
void ResourceManager::releaseResources (
    ResourceType type,
    int quantity ) [inline]
```

Releases resources of a specific type and quantity.

Parameters

<i>type</i>	The type of the resource.
<i>quantity</i>	The quantity of the resource to release.

4.61.3.7 **setAllocationStrategy()**

```
void ResourceManager::setAllocationStrategy (  
    ResourceAllocationStrategy * strategy ) [inline]
```

Sets the resource allocation strategy.

Parameters

<i>strategy</i>	Pointer to the resource allocation strategy.
-----------------	--

The documentation for this class was generated from the following file:

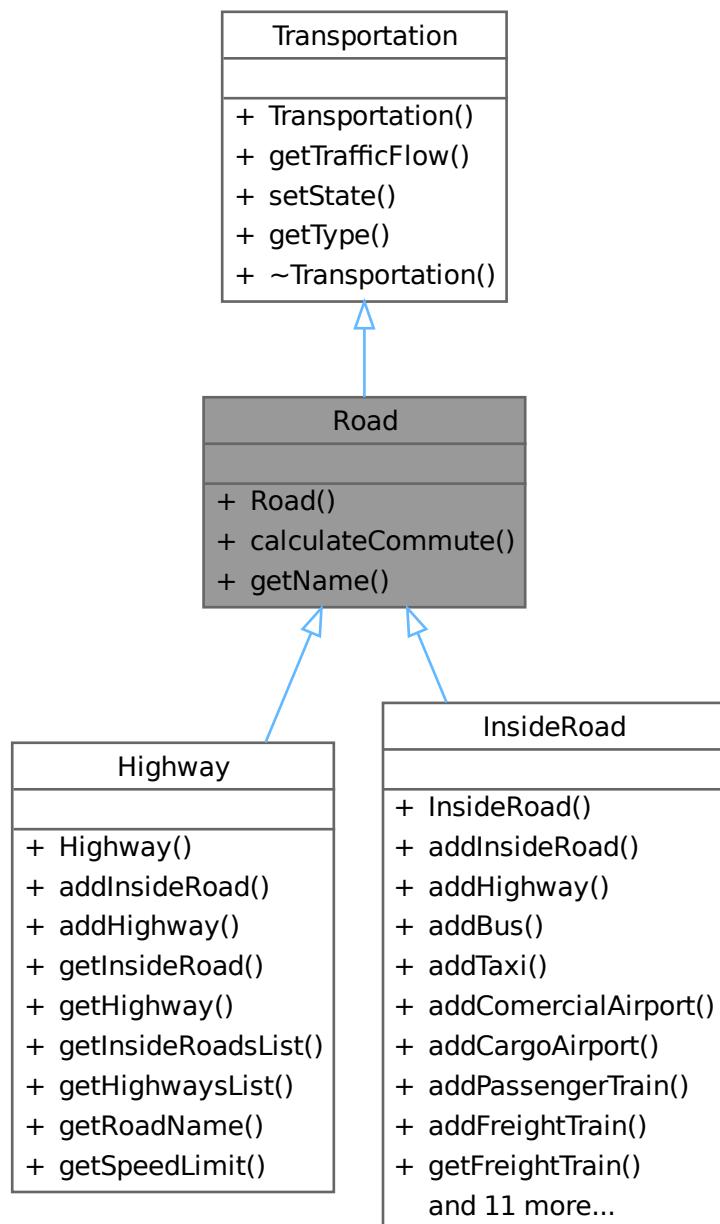
- /mnt/c/users/rudie/documents/sem2/214/project/src/[ResourceManager.h](#)

4.62 Road Class Reference

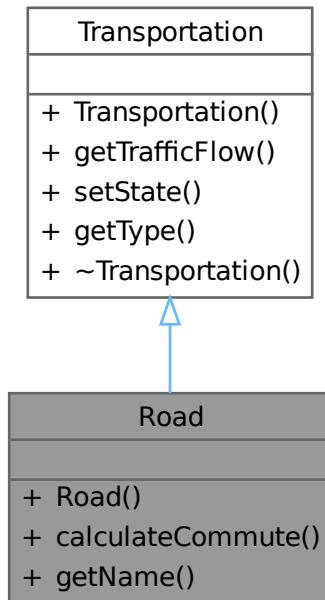
A class representing a road as a type of transportation.

```
#include <Road.h>
```

Inheritance diagram for Road:



Collaboration diagram for Road:



Public Member Functions

- `Road` (char state, std::string roadName, char type)
Constructs a new `Road` object.
- float `calculateCommute` ()
Calculates the commute time on the road.
- std::string `getName` ()
Gets the name of the road.

Public Member Functions inherited from `Transportation`

- `Transportation` (char state, char type)
Constructor for the `Transportation` class.
- float `getTrafficFlow` ()
Gets the current traffic flow.
- bool `setState` (char state)
Sets the traffic flow state.
- char `getType` ()
Gets the type of transportation.
- `~Transportation` ()
Destructor for the `Transportation` class.

4.62.1 Detailed Description

A class representing a road as a type of transportation.

The [Road](#) class inherits from the [Transportation](#) class and adds specific attributes and methods related to roads.

4.62.2 Constructor & Destructor Documentation

4.62.2.1 Road()

```
Road::Road (  
    char state,  
    std::string roadName,  
    char type )
```

Constructs a new [Road](#) object.

Parameters

<i>state</i>	The state where the road is located.
<i>roadName</i>	The name of the road.
<i>type</i>	The type of the road.
<i>state</i>	The state of the road.
<i>roadName</i>	The name of the road.
<i>type</i>	The type of the road.

4.62.3 Member Function Documentation

4.62.3.1 calculateCommute()

```
float Road::calculateCommute ( )
```

Calculates the commute time on the road.

Returns

The calculated commute time.

4.62.3.2 getName()

```
std::string Road::getName ( )
```

Gets the name of the road.

Returns

The name of the road.

The documentation for this class was generated from the following files:

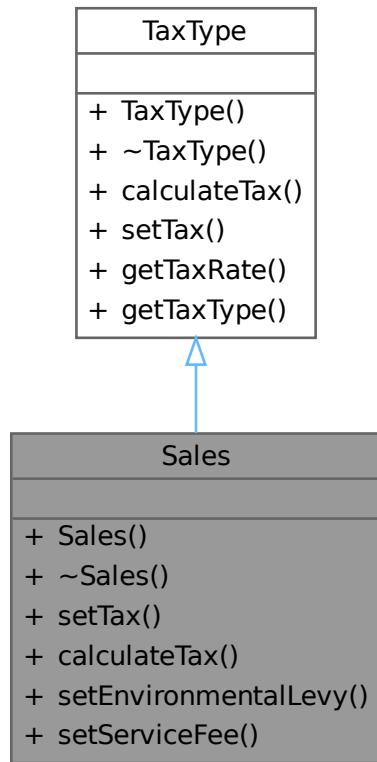
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Road.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Road.cpp](#)

4.63 Sales Class Reference

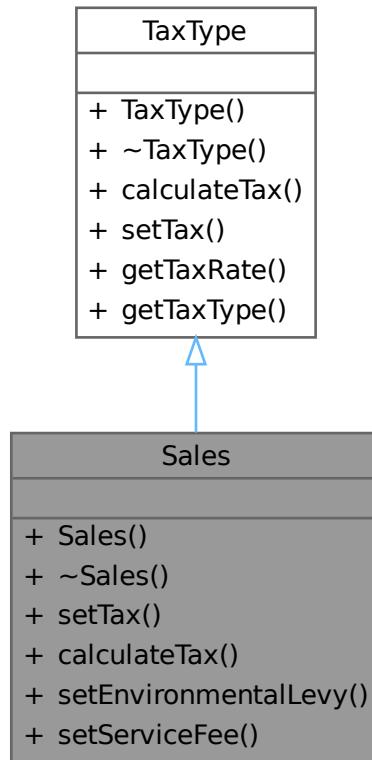
Represents a type of tax that includes a base sales tax, an environmental levy, and a service fee.

```
#include <Sales.h>
```

Inheritance diagram for Sales:



Collaboration diagram for Sales:



Public Member Functions

- **Sales** (double rate, double levy, double fee)
*Constructs a new **Sales** object.*
- virtual **~Sales** () override
*Destroys the **Sales** object.*
- void **setTax** (double rate) override
Sets the base sales tax rate.
- double **calculateTax** (double saleAmount) override
Calculates the total sales tax based on the sale amount.
- void **setEnvironmentalLevy** (double levy)
Sets the environmental levy rate.
- void **setServiceFee** (double fee)
Sets the fixed service fee.

Public Member Functions inherited from **TaxType**

- **TaxType** (double rate, char type)
*Constructs a new **TaxType** object.*

- virtual ~**TaxType** ()
Virtual Destructor.
- virtual double **getTaxRate** ()
Gets the current tax rate.
- char **getTaxType** ()
Gets the tax type identifier.

4.63.1 Detailed Description

Represents a type of tax that includes a base sales tax, an environmental levy, and a service fee.

The [Sales](#) class inherits from the [TaxType](#) class and provides methods to set and calculate the total sales tax based on a sale amount.

4.63.2 Constructor & Destructor Documentation

4.63.2.1 Sales()

```
Sales::Sales (
    double rate,
    double levy,
    double fee )
```

Constructs a new [Sales](#) object.

Parameters

<i>rate</i>	The base sales tax rate.
<i>levy</i>	The environmental levy rate.
<i>fee</i>	The fixed service fee.

4.63.3 Member Function Documentation

4.63.3.1 calculateTax()

```
double Sales::calculateTax (
    double saleAmount ) [override], [virtual]
```

Calculates the total sales tax based on the sale amount.

Calculates the total sales tax for a given sale amount.

Parameters

<i>saleAmount</i>	The amount of the sale.
-------------------	-------------------------

Returns

The total sales tax.

Reimplemented from [TaxType](#).

4.63.3.2 setEnvironmentalLevy()

```
void Sales::setEnvironmentalLevy (
    double levy )
```

Sets the environmental levy rate.

Parameters

<i>levy</i>	The new environmental levy rate.
-------------	----------------------------------

4.63.3.3 setServiceFee()

```
void Sales::setServiceFee (
    double fee )
```

Sets the fixed service fee.

Parameters

<i>fee</i>	The new service fee.
------------	----------------------

4.63.3.4 setTax()

```
void Sales::setTax (
    double rate ) [override], [virtual]
```

Sets the base sales tax rate.

Parameters

<i>rate</i>	The new sales tax rate.
-------------	-------------------------

Reimplemented from [TaxType](#).

The documentation for this class was generated from the following files:

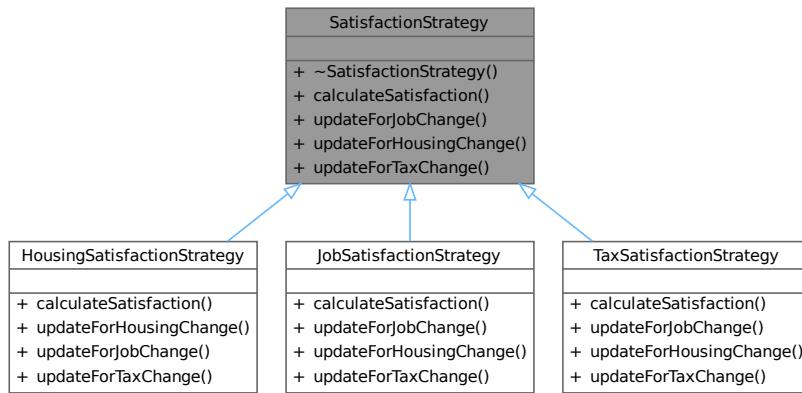
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Sales.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Sales.cpp](#)

4.64 SatisfactionStrategy Class Reference

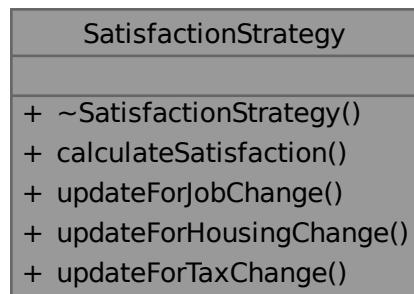
Interface for calculating and updating citizen satisfaction.

```
#include <SatisfactionStrategy.h>
```

Inheritance diagram for SatisfactionStrategy:



Collaboration diagram for SatisfactionStrategy:



Public Member Functions

- virtual ~**SatisfactionStrategy** ()=default
Virtual destructor for SatisfactionStrategy.
- virtual float **calculateSatisfaction** (const **Citizen** &citizen)=0
Calculates the satisfaction of a citizen.
- virtual void **updateForJobChange** (**Citizen** &citizen)=0
Updates the satisfaction of a citizen for a job change.
- virtual void **updateForHousingChange** (**Citizen** &citizen)=0
Updates the satisfaction of a citizen for a housing change.
- virtual void **updateForTaxChange** (**Citizen** &citizen)=0
Updates the satisfaction of a citizen for a tax change.

4.64.1 Detailed Description

Interface for calculating and updating citizen satisfaction.

The [SatisfactionStrategy](#) class provides methods to calculate citizen satisfaction and update it based on changes in job, housing, and taxes.

4.64.2 Member Function Documentation

4.64.2.1 calculateSatisfaction()

```
virtual float SatisfactionStrategy::calculateSatisfaction (
    const Citizen & citizen ) [pure virtual]
```

Calculates the satisfaction of a citizen.

Parameters

<i>citizen</i>	The citizen whose satisfaction is to be calculated.
----------------	---

Returns

The calculated satisfaction level.

Implemented in [HousingSatisfactionStrategy](#), [JobSatisfactionStrategy](#), and [TaxSatisfactionStrategy](#).

4.64.2.2 updateForHousingChange()

```
virtual void SatisfactionStrategy::updateForHousingChange (
    Citizen & citizen ) [pure virtual]
```

Updates the satisfaction of a citizen for a housing change.

Parameters

<i>citizen</i>	The citizen whose satisfaction is to be updated.
----------------	--

Implemented in [HousingSatisfactionStrategy](#), [JobSatisfactionStrategy](#), and [TaxSatisfactionStrategy](#).

4.64.2.3 updateForJobChange()

```
virtual void SatisfactionStrategy::updateForJobChange (
    Citizen & citizen ) [pure virtual]
```

Updates the satisfaction of a citizen for a job change.

Parameters

<i>citizen</i>	The citizen whose satisfaction is to be updated.
----------------	--

Implemented in [HousingSatisfactionStrategy](#), [JobSatisfactionStrategy](#), and [TaxSatisfactionStrategy](#).

4.64.2.4 updateForTaxChange()

```
virtual void SatisfactionStrategy::updateForTaxChange (
    Citizen & citizen ) [pure virtual]
```

Updates the satisfaction of a citizen for a tax change.

Parameters

<i>citizen</i>	The citizen whose satisfaction is to be updated.
----------------	--

Implemented in [HousingSatisfactionStrategy](#), [JobSatisfactionStrategy](#), and [TaxSatisfactionStrategy](#).

The documentation for this class was generated from the following file:

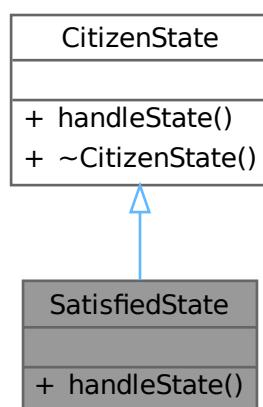
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SatisfactionStrategy.h](#)

4.65 SatisfiedState Class Reference

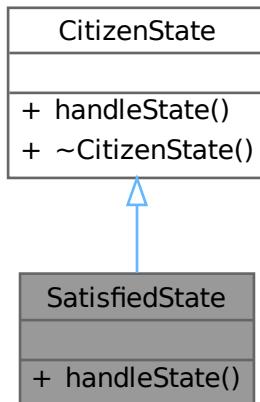
Represents a state where a citizen is satisfied.

```
#include <SatisfiedState.h>
```

Inheritance diagram for SatisfiedState:



Collaboration diagram for SatisfiedState:



Public Member Functions

- void `HandleState (Citizen &citizen) const override`
Handles the satisfied state for a citizen.

Public Member Functions inherited from [CitizenState](#)

- virtual `~CitizenState ()=default`
Virtual destructor for the [CitizenState](#) class.

4.65.1 Detailed Description

Represents a state where a citizen is satisfied.

The [SatisfiedState](#) class provides methods to handle the state and update the citizen's satisfaction level.

4.65.2 Member Function Documentation

4.65.2.1 `handleState()`

```
void SatisfiedState::handleState (
    Citizen & citizen ) const [override], [virtual]
```

Handles the satisfied state for a citizen.

Parameters

<i>citizen</i>	The citizen whose state is being handled.
----------------	---

Implements [CitizenState](#).

The documentation for this class was generated from the following files:

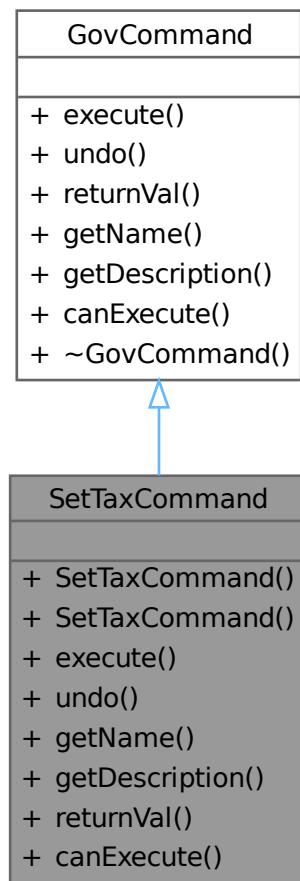
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SatisfiedState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SatisfiedState.cpp](#)

4.66 SetTaxCommand Class Reference

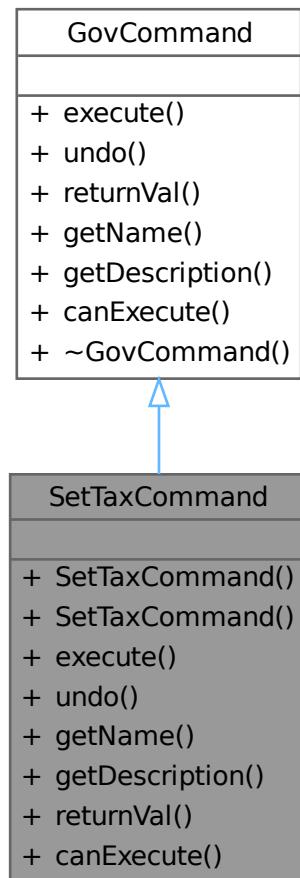
A command to set the tax rate in the government.

```
#include <SetTaxCommand.h>
```

Inheritance diagram for SetTaxCommand:



Collaboration diagram for SetTaxCommand:



Public Member Functions

- **`SetTaxCommand (Government *gov, double rate)`**
Constructor for `SetTaxCommand`.
- **`SetTaxCommand (Government *gov, TaxSystem *taxSys, double rate, char category)`**
Constructor for `SetTaxCommand` with additional parameters.
- `void execute () override`
Executes the set tax command.
- `void undo () override`
Undoes the set tax command.
- `std::string getName () const override`
Gets the name of the command.
- `std::string getDescription () const override`
Gets the description of the command.
- `double returnVal () override`
Returns the tax rate.
- `bool canExecute () const override`
Checks if the command can be executed.

Public Member Functions inherited from [GovCommand](#)

- virtual [~GovCommand \(\)=default](#)

Virtual destructor.

4.66.1 Detailed Description

A command to set the tax rate in the government.

This class represents a command to set the tax rate in the government.

4.66.2 Constructor & Destructor Documentation

4.66.2.1 [SetTaxCommand\(\)](#) [1/2]

```
SetTaxCommand::SetTaxCommand (
    Government * gov,
    double rate )
```

Constructor for [SetTaxCommand](#).

Initializes the command with the government object and the new tax rate.

Parameters

<i>gov</i>	Pointer to the Government object.
<i>rate</i>	The new tax rate to be set.

4.66.2.2 [SetTaxCommand\(\)](#) [2/2]

```
SetTaxCommand::SetTaxCommand (
    Government * gov,
    TaxSystem * taxSys,
    double rate,
    char category )
```

Constructor for [SetTaxCommand](#) with additional parameters.

Initializes the command with the government object, tax system, new tax rate, and category.

Parameters

<i>gov</i>	Pointer to the Government object.
<i>taxSys</i>	Pointer to the TaxSystem object.
<i>rate</i>	The new tax rate to be set.
<i>category</i>	The category of the tax.

4.66.3 Member Function Documentation

4.66.3.1 canExecute()

```
bool SetTaxCommand::canExecute ( ) const [override], [virtual]
```

Checks if the command can be executed.

Returns

True if the command can be executed, false otherwise.

Implements [GovCommand](#).

4.66.3.2 execute()

```
void SetTaxCommand::execute ( ) [override], [virtual]
```

Executes the set tax command.

Stores the previous tax rate and sets the new tax rate.

Implements [GovCommand](#).

4.66.3.3 getDescription()

```
std::string SetTaxCommand::getDescription ( ) const [override], [virtual]
```

Gets the description of the command.

Returns

The description of the command.

Implements [GovCommand](#).

4.66.3.4 getName()

```
std::string SetTaxCommand::getName ( ) const [override], [virtual]
```

Gets the name of the command.

Returns

The name of the command.

Implements [GovCommand](#).

4.66.3.5 `returnVal()`

```
double SetTaxCommand::returnVal ( ) [override], [virtual]
```

Returns the tax rate.

Returns

The tax rate.

Implements [GovCommand](#).

4.66.3.6 `undo()`

```
void SetTaxCommand::undo ( ) [override], [virtual]
```

Undoes the set tax command.

Reverts the tax rate to the previous rate.

Implements [GovCommand](#).

The documentation for this class was generated from the following files:

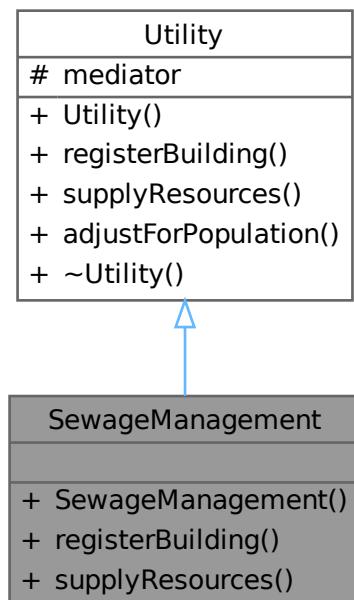
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SetTaxCommand.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SetTaxCommand.cpp](#)

4.67 SewageManagement Class Reference

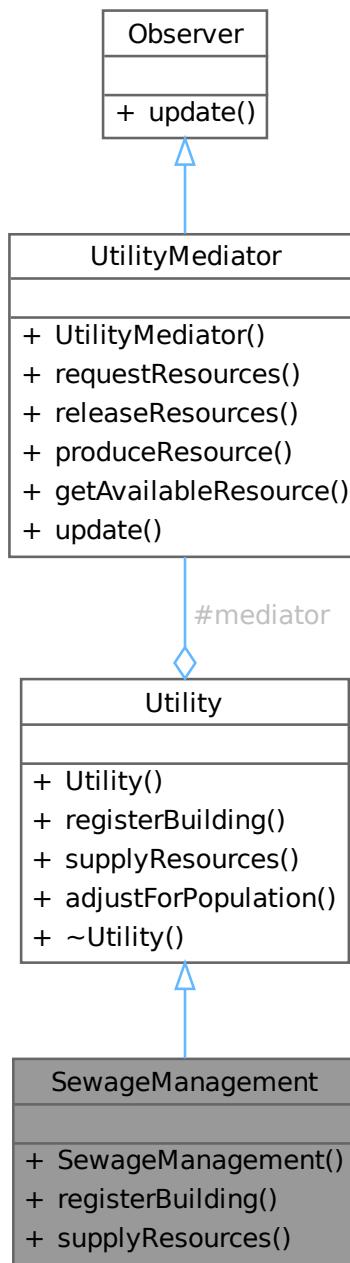
A class to manage sewage services for buildings.

```
#include <SewageManagement.h>
```

Inheritance diagram for SewageManagement:



Collaboration diagram for SewageManagement:



Public Member Functions

- `SewageManagement (UtilityMediator *mediator)`
Constructor for SewageManagement.
- `void registerBuilding (Building *building) override`
Registers a building with the sewage management service.
- `void supplyResources (Building *building) override`
Supplies sewage management services to a building.

Public Member Functions inherited from Utility

- **Utility** (`UtilityMediator *mediator`)
Constructor for the Utility class.
- **virtual void adjustForPopulation** (`int newPopulation)=0`
Adjusts the utility service for a new population size.
- **virtual ~Utility** ()=`default`
Virtual destructor for the Utility class.

Additional Inherited Members

Protected Attributes inherited from Utility

- **UtilityMediator * mediator**
Reference to the mediator for managing resources.

4.67.1 Detailed Description

A class to manage sewage services for buildings.

This class represents the sewage management service, which provides sewage services to buildings.

4.67.2 Constructor & Destructor Documentation

4.67.2.1 SewageManagement()

```
SewageManagement::SewageManagement (
    UtilityMediator * mediator )
```

Constructor for **SewageManagement**.

Initializes the sewage management service with the given utility mediator.

Parameters

<code>mediator</code>	Pointer to the <code>UtilityMediator</code> object.
-----------------------	---

4.67.3 Member Function Documentation

4.67.3.1 registerBuilding()

```
void SewageManagement::registerBuilding (
    Building * building ) [override], [virtual]
```

Registers a building with the sewage management service.

Parameters

<i>building</i>	Pointer to the Building object to be registered.
-----------------	--

Implements [Utility](#).

4.67.3.2 supplyResources()

```
void SewageManagement::supplyResources (
    Building * building ) [override], [virtual]
```

Supplies sewage management services to a building.

Requests resources from the mediator and provides sewage management services if resources are available.

Parameters

<i>building</i>	Pointer to the Building object to receive services.
-----------------	---

Implements [Utility](#).

The documentation for this class was generated from the following files:

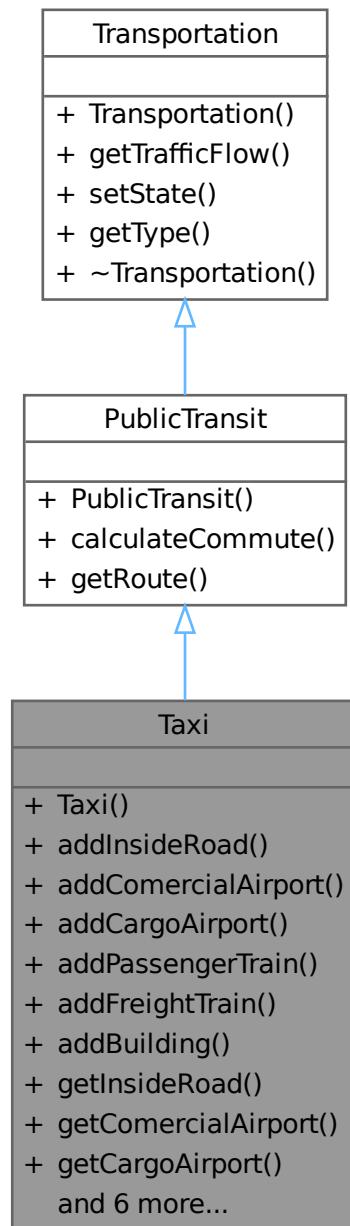
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SewageManagement.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[SewageManagement.cpp](#)

4.68 Taxi Class Reference

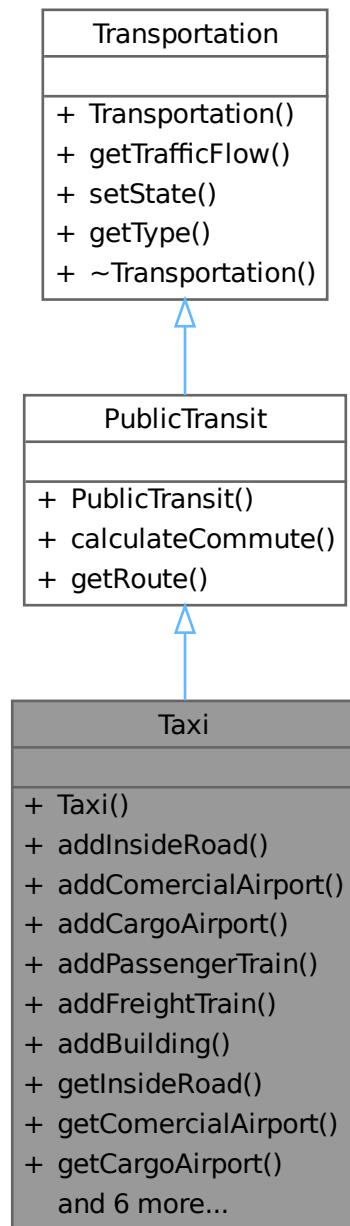
A class representing a taxi in a public transit system.

```
#include <Taxi.h>
```

Inheritance diagram for Taxi:



Collaboration diagram for Taxi:



Public Member Functions

- `Taxi` (char state, std::string route, std::string taxiCompany, int taxiNumber)
Constructor for the `Taxi` class.
- bool `addInsideRoad` (`InsideRoad` *insideRoad)
Adds an inside road to the taxi's accessible roads.
- bool `addComercialAirport` (`ComercialAirport` *comercialAirport)

- Adds a commercial airport to the taxi's accessible airports.
- bool `addCargoAirport (CargoAirport *cargoAirport)`
 Adds a cargo airport to the taxi's accessible airports.
- bool `addPassengerTrain (PassengerTrain *passengerTrain)`
 Adds a passenger train to the taxi's accessible trains.
- bool `addFreightTrain (FreightTrain *freightTrain)`
 Adds a freight train to the taxi's accessible trains.
- bool `addBuilding (Building *building)`
 Adds a building to the taxi's accessible buildings.
- `InsideRoad * getInsideRoad (std::size_t x)`
 Gets an inside road from the taxi's accessible roads.
- `ComercialAirport * getComercialAirport (std::size_t x)`
 Gets a commercial airport from the taxi's accessible airports.
- `CargoAirport * getCargoAirport (std::size_t x)`
 Gets a cargo airport from the taxi's accessible airports.
- `PassengerTrain * getPassengerTrain (std::size_t x)`
 Gets a passenger train from the taxi's accessible trains.
- `FreightTrain * getFreightTrain (std::size_t x)`
 Gets a freight train from the taxi's accessible trains.
- `Building * getBuilding (std::size_t x)`
 Gets a building from the taxi's accessible buildings.
- std::string `getRouteName ()`
 Gets the route name of the taxi.
- std::string `getTaxiCompany ()`
 Gets the company that owns the taxi.
- int `getTaxiNumber ()`
 Gets the unique number identifying the taxi.

Public Member Functions inherited from [PublicTransit](#)

- `PublicTransit (char state, std::string route, char type)`
 Constructor for the [PublicTransit](#) class.
- float `calculateCommute ()`
 Calculate the commute time for the public transit.
- std::string `getRoute ()`
 Get the route of the public transit.

Public Member Functions inherited from [Transportation](#)

- `Transportation (char state, char type)`
 Constructor for the [Transportation](#) class.
- float `getTrafficFlow ()`
 Gets the current traffic flow.
- bool `setState (char state)`
 Sets the traffic flow state.
- char `getType ()`
 Gets the type of transportation.
- `~Transportation ()`
 Destructor for the [Transportation](#) class.

4.68.1 Detailed Description

A class representing a taxi in a public transit system.

4.68.2 Constructor & Destructor Documentation

4.68.2.1 Taxi()

```
Taxi::Taxi (
    char state,
    std::string route,
    std::string taxiCompany,
    int taxiNumber )
```

Constructor for the [Taxi](#) class.

Constructor for [Taxi](#).

Parameters

<i>state</i>	The state of the taxi.
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The company that owns the taxi.
<i>taxisNumber</i>	The unique number identifying the taxi.

Initializes the taxi with the given state, route, taxi company, and taxi number.

Parameters

<i>state</i>	The state of the taxi.
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The company that owns the taxi.
<i>taxisNumber</i>	The number of the taxi.

4.68.3 Member Function Documentation

4.68.3.1 addBuilding()

```
bool Taxi::addBuilding (
    Building * building )
```

Adds a building to the taxi's accessible buildings.

Adds a building to the taxi's route.

Parameters

<i>building</i>	Pointer to the Building object.
-----------------	---

Returns

True if the building was added successfully, false otherwise.

Parameters

<i>building</i>	Pointer to the Building object.
-----------------	---

Returns

True if the building was added, false if it was already present.

4.68.3.2 addCargoAirport()

```
bool Taxi::addCargoAirport (
    CargoAirport * cargoAirport )
```

Adds a cargo airport to the taxi's accessible airports.

Adds a cargo airport to the taxi's route.

Parameters

<i>cargoAirport</i>	Pointer to the CargoAirport object.
---------------------	---

Returns

True if the airport was added successfully, false otherwise.

Parameters

<i>cargoAirport</i>	Pointer to the CargoAirport object.
---------------------	---

Returns

True if the cargo airport was added, false if it was already present.

4.68.3.3 addComercialAirport()

```
bool Taxi::addComercialAirport (
    ComercialAirport * comercialAirport )
```

Adds a commercial airport to the taxi's accessible airports.

Adds a commercial airport to the taxi's route.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport object.
-------------------------	---

Returns

True if the airport was added successfully, false otherwise.

Parameters

<i>comercialAirport</i>	Pointer to the ComercialAirport object.
-------------------------	---

Returns

True if the commercial airport was added, false if it was already present.

4.68.3.4 addFreightTrain()

```
bool Taxi::addFreightTrain (  
    FreightTrain * freightTrain )
```

Adds a freight train to the taxi's accessible trains.

Adds a freight train to the taxi's route.

Parameters

<i>freightTrain</i>	Pointer to the FreightTrain object.
---------------------	---

Returns

True if the train was added successfully, false otherwise.

Parameters

<i>freightTrain</i>	Pointer to the FreightTrain object.
---------------------	---

Returns

True if the freight train was added, false if it was already present.

4.68.3.5 addInsideRoad()

```
bool Taxi::addInsideRoad (  
    InsideRoad * insideRoad )
```

Adds an inside road to the taxi's accessible roads.

Adds an inside road to the taxi's route.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad object.
-------------------	---

Returns

True if the road was added successfully, false otherwise.

Parameters

<i>insideRoad</i>	Pointer to the InsideRoad object.
-------------------	---

Returns

True if the inside road was added, false if it was already present.

4.68.3.6 addPassengerTrain()

```
bool Taxi::addPassengerTrain (  
    PassengerTrain * passengerTrain )
```

Adds a passenger train to the taxi's accessible trains.

Adds a passenger train to the taxi's route.

Parameters

<i>passengerTrain</i>	Pointer to the PassengerTrain object.
-----------------------	---

Returns

True if the train was added successfully, false otherwise.

Parameters

<i>passengerTrain</i>	Pointer to the PassengerTrain object.
-----------------------	---

Returns

True if the passenger train was added, false if it was already present.

4.68.3.7 getBuilding()

```
Building * Taxi::getBuilding ( std::size_t x )
```

Gets a building from the taxi's accessible buildings.

Gets a building by index.

Parameters

x	The index of the building.
---	----------------------------

Returns

Pointer to the [Building](#) object.

Parameters

x	The index of the building.
---	----------------------------

Returns

Pointer to the [Building](#) object, or nullptr if the index is out of range.

4.68.3.8 `getCargoAirport()`

```
CargoAirport * Taxi::getCargoAirport ( std::size_t x )
```

Gets a cargo airport from the taxi's accessible airports.

Gets a cargo airport by index.

Parameters

x	The index of the cargo airport.
---	---------------------------------

Returns

Pointer to the [CargoAirport](#) object.

Parameters

x	The index of the cargo airport.
---	---------------------------------

Returns

Pointer to the [CargoAirport](#) object, or nullptr if the index is out of range.

4.68.3.9 `getComercialAirport()`

```
ComercialAirport * Taxi::getComercialAirport ( std::size_t x )
```

Gets a commercial airport from the taxi's accessible airports.

Gets a commercial airport by index.

Parameters

x	The index of the commercial airport.
---	--------------------------------------

Returns

Pointer to the [ComercialAirport](#) object.

Parameters

x	The index of the commercial airport.
---	--------------------------------------

Returns

Pointer to the [ComercialAirport](#) object, or nullptr if the index is out of range.

4.68.3.10 getFreightTrain()

```
FreightTrain * Taxi::getFreightTrain (
    std::size_t x )
```

Gets a freight train from the taxi's accessible trains.

Gets a freight train by index.

Parameters

x	The index of the freight train.
---	---------------------------------

Returns

Pointer to the [FreightTrain](#) object.

Parameters

x	The index of the freight train.
---	---------------------------------

Returns

Pointer to the [FreightTrain](#) object, or nullptr if the index is out of range.

4.68.3.11 getInsideRoad()

```
InsideRoad * Taxi::getInsideRoad (
    std::size_t x )
```

Gets an inside road from the taxi's accessible roads.

Gets an inside road by index.

Parameters

x	The index of the inside road.
---	-------------------------------

Returns

Pointer to the [InsideRoad](#) object.

Parameters

x	The index of the inside road.
---	-------------------------------

Returns

Pointer to the [InsideRoad](#) object, or nullptr if the index is out of range.

4.68.3.12 getPassengerTrain()

```
PassengerTrain * Taxi::getPassengerTrain ( std::size_t x )
```

Gets a passenger train from the taxi's accessible trains.

Gets a passenger train by index.

Parameters

x	The index of the passenger train.
---	-----------------------------------

Returns

Pointer to the [PassengerTrain](#) object.

Parameters

x	The index of the passenger train.
---	-----------------------------------

Returns

Pointer to the [PassengerTrain](#) object, or nullptr if the index is out of range.

4.68.3.13 getRouteName()

```
std::string Taxi::getRouteName ( )
```

Gets the route name of the taxi.

Returns

The route name.

4.68.3.14 getTaxiCompany()

```
std::string Taxi::getTaxiCompany ()
```

Gets the company that owns the taxi.

Gets the taxi company name.

Returns

The taxi company name.

4.68.3.15 getTaxiNumber()

```
int Taxi::getTaxiNumber ()
```

Gets the unique number identifying the taxi.

Gets the taxi number.

Returns

The taxi number.

The documentation for this class was generated from the following files:

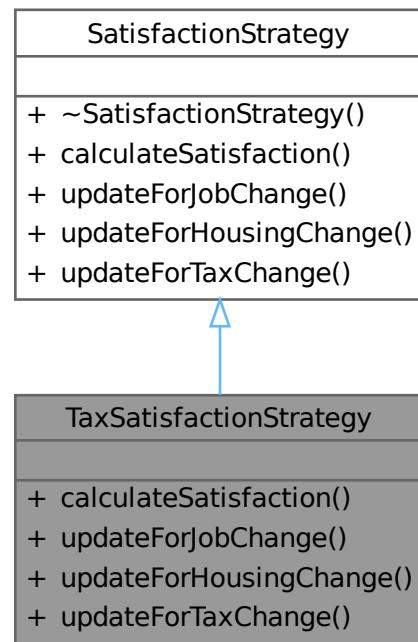
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Taxi.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Taxi.cpp](#)

4.69 TaxSatisfactionStrategy Class Reference

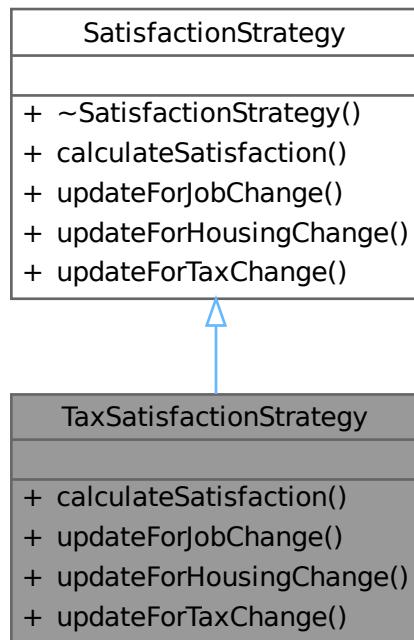
A strategy for calculating and updating citizen satisfaction based on tax rates.

```
#include <TaxSatisfactionStrategy.h>
```

Inheritance diagram for TaxSatisfactionStrategy:



Collaboration diagram for TaxSatisfactionStrategy:



Public Member Functions

- float `calculateSatisfaction` (const `Citizen` &citizen) override
Calculates the satisfaction level of a citizen based on the tax rate.
- void `updateForJobChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen based on a job change.
- void `updateForHousingChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen based on a housing change.
- void `updateForTaxChange` (`Citizen` &citizen) override
Updates the satisfaction level of a citizen based on a tax change.

Public Member Functions inherited from `SatisfactionStrategy`

- virtual `~SatisfactionStrategy` ()=default
Virtual destructor for `SatisfactionStrategy`.

4.69.1 Detailed Description

A strategy for calculating and updating citizen satisfaction based on tax rates.

This class provides methods to calculate and update the satisfaction level of a citizen based on the current tax rate. It inherits from the `SatisfactionStrategy` interface.

Version

1.0

Date

2024-11-04

4.69.2 Member Function Documentation

4.69.2.1 calculateSatisfaction()

```
float TaxSatisfactionStrategy::calculateSatisfaction (
    const Citizen & citizen ) [override], [virtual]
```

Calculates the satisfaction level of a citizen based on the tax rate.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be calculated.
----------------	---

Returns

The calculated satisfaction level.

This function calculates the satisfaction level of a citizen by reducing their current satisfaction level based on the tax rate. Higher tax rates result in lower satisfaction levels.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be calculated.
----------------	---

Returns

The calculated satisfaction level.

Implements [SatisfactionStrategy](#).

4.69.2.2 updateForHousingChange()

```
void TaxSatisfactionStrategy::updateForHousingChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen based on a housing change.

This method is currently not implemented.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.69.2.3 updateForJobChange()

```
void TaxSatisfactionStrategy::updateForJobChange (
    Citizen & citizen ) [inline], [override], [virtual]
```

Updates the satisfaction level of a citizen based on a job change.

This method is currently not implemented.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

4.69.2.4 updateForTaxChange()

```
void TaxSatisfactionStrategy::updateForTaxChange (
    Citizen & citizen ) [override], [virtual]
```

Updates the satisfaction level of a citizen based on a tax change.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

This function updates the satisfaction level of a citizen by applying a small impact based on the change in tax rate.

Parameters

<i>citizen</i>	The citizen whose satisfaction level is to be updated.
----------------	--

Implements [SatisfactionStrategy](#).

The documentation for this class was generated from the following files:

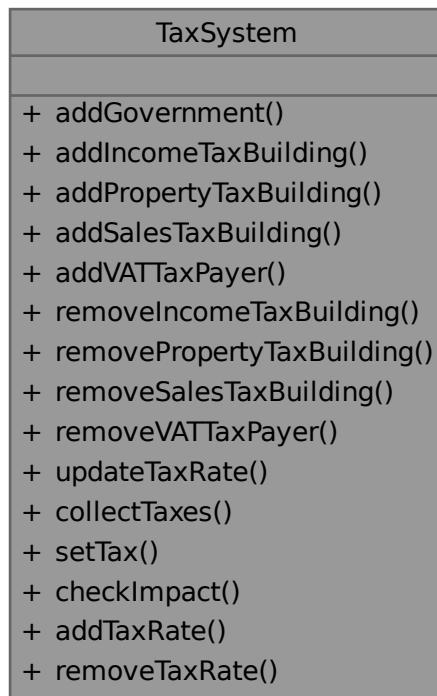
- /mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TaxSatisfactionStrategy.cpp](#)

4.70 TaxSystem Class Reference

Manages various types of taxes, including income, property, sales, and [VAT](#).

```
#include <TaxSystem.h>
```

Collaboration diagram for TaxSystem:



Public Member Functions

- void **addGovernment** (**Government** *gov)
Adds a government to the tax system.
- void **addIncomeTaxBuilding** (**Building** *building)
Adds a building to the income tax list.
- void **addPropertyTaxBuilding** (**Building** *building)
Adds a building to the property tax list.
- void **addSalesTaxBuilding** (**Building** *building)
Adds a building to the sales tax list.
- void **addVATTaxPayer** (**Citizen** *citizen)
Adds a citizen to the VAT tax list.
- void **removeIncomeTaxBuilding** (**Building** *building)
Removes a building from the income tax list.
- void **removePropertyTaxBuilding** (**Building** *building)
Removes a building from the property tax list.
- void **removeSalesTaxBuilding** (**Building** *building)
Removes a building from the sales tax list.
- void **removeVATTaxPayer** (**Citizen** *citizen)
Removes a citizen from the VAT tax list.
- void **updateTaxRate** (char cType, double rate)
Updates the tax rate for a specific tax type.

- void `collectTaxes (Building *building, char taxType)`
Collects taxes from a building based on the tax type.
- void `setTax (double rate, char taxType)`
Sets the tax rate for a specific tax type.
- void `checkImpact ()`
Checks the impact of the tax system.
- void `addTaxRate (TaxType *tax)`
Adds a new tax rate to the tax system.
- void `removeTaxRate (TaxType *taxType)`
Removes a tax rate from the tax system.

4.70.1 Detailed Description

Manages various types of taxes, including income, property, sales, and VAT.

This class allows adding and removing buildings and citizens for tax purposes, updating tax rates, and collecting taxes.

Version

1.0

Date

2024-11-04

4.70.2 Member Function Documentation

4.70.2.1 addGovernment()

```
void TaxSystem::addGovernment (
    Government * gov )
```

Adds a government to the tax system.

Parameters

<code>gov</code>	Pointer to the government object.
------------------	-----------------------------------

4.70.2.2 addIncomeTaxBuilding()

```
void TaxSystem::addIncomeTaxBuilding (
    Building * building )
```

Adds a building to the income tax list.

Parameters

<i>building</i>	Pointer to the building object.
-----------------	---------------------------------

4.70.2.3 addPropertyTaxBuilding()

```
void TaxSystem::addPropertyTaxBuilding (
```

```
    Building * building )
```

Adds a building to the property tax list.

Parameters

<i>building</i>	Pointer to the building object.
-----------------	---------------------------------

4.70.2.4 addSalesTaxBuilding()

```
void TaxSystem::addSalesTaxBuilding (
```

```
    Building * building )
```

Adds a building to the sales tax list.

Parameters

<i>building</i>	Pointer to the building object.
-----------------	---------------------------------

4.70.2.5 addTaxRate()

```
void TaxSystem::addTaxRate (
```

```
    TaxType * tax )
```

Adds a new tax rate to the tax system.

Parameters

<i>tax</i>	Pointer to the TaxType object.
------------	--

4.70.2.6 addVATTaxPayer()

```
void TaxSystem::addVATTaxPayer (
```

```
    Citizen * citizen )
```

Adds a citizen to the [VAT](#) tax list.

Parameters

<i>citizen</i>	Pointer to the citizen object.
----------------	--------------------------------

4.70.2.7 checkImpact()

```
void TaxSystem::checkImpact ( )
```

Checks the impact of the tax system.

This function is currently not implemented.

4.70.2.8 collectTaxes()

```
void TaxSystem::collectTaxes (
    Building * building,
    char taxType )
```

Collects taxes from a building based on the tax type.

Parameters

<i>building</i>	Pointer to the building object.
<i>taxType</i>	Character representing the tax type.

4.70.2.9 removeIncomeTaxBuilding()

```
void TaxSystem::removeIncomeTaxBuilding (
    Building * building )
```

Removes a building from the income tax list.

Parameters

<i>building</i>	Pointer to the building object.
-----------------	---------------------------------

4.70.2.10 removePropertyTaxBuilding()

```
void TaxSystem::removePropertyTaxBuilding (
    Building * building )
```

Removes a building from the property tax list.

Parameters

<i>building</i>	Pointer to the building object.
-----------------	---------------------------------

4.70.2.11 removeSalesTaxBuilding()

```
void TaxSystem::removeSalesTaxBuilding (
```

<code>Building</code>	* <i>building</i>)
-----------------------	---------------------

Removes a building from the sales tax list.

Parameters

<code>building</code>	Pointer to the building object.
-----------------------	---------------------------------

4.70.2.12 removeTaxRate()

```
void TaxSystem::removeTaxRate (
```

<code>TaxType</code>	* <i>taxType</i>)
----------------------	--------------------

Removes a tax rate from the tax system.

Parameters

<code>taxType</code>	Pointer to the <code>TaxType</code> object.
----------------------	---

4.70.2.13 removeVATTaxPayer()

```
void TaxSystem::removeVATTaxPayer (
```

<code>Citizen</code>	* <i>citizen</i>)
----------------------	--------------------

Removes a citizen from the `VAT` tax list.

Parameters

<code>citizen</code>	Pointer to the citizen object.
----------------------	--------------------------------

4.70.2.14 setTax()

```
void TaxSystem::setTax (
```

<code>double</code>	rate,
---------------------	-------

<code>char</code>	taxType)
-------------------	-----------

Sets the tax rate for a specific tax type.

Parameters

<code>rate</code>	New tax rate.
<code>taxType</code>	Character representing the tax type.

4.70.2.15 updateTaxRate()

```
void TaxSystem::updateTaxRate (
    char cType,
    double rate )
```

Updates the tax rate for a specific tax type.

Parameters

<i>cType</i>	Character representing the tax type.
<i>rate</i>	New tax rate.

The documentation for this class was generated from the following files:

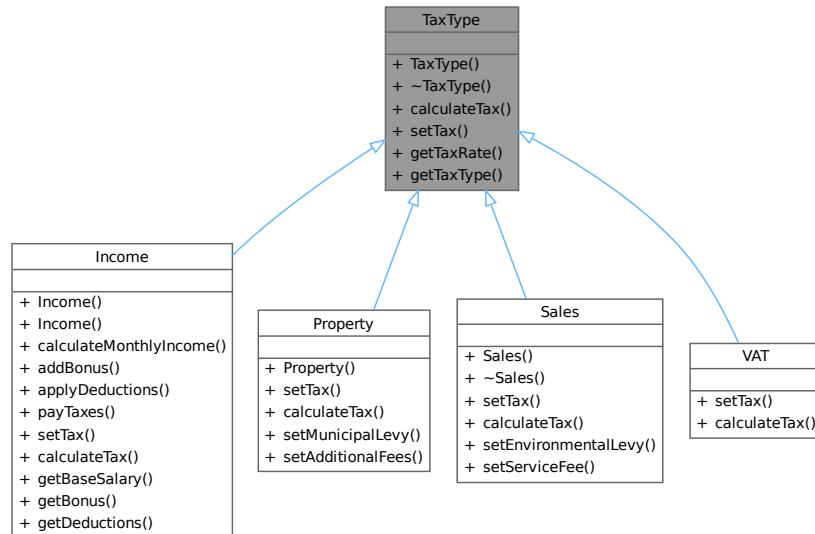
- /mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TaxSystem.cpp](#)

4.71 TaxType Class Reference

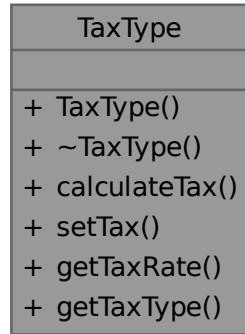
Represents a specific type of tax with a rate and type identifier.

```
#include <TaxType.h>
```

Inheritance diagram for TaxType:



Collaboration diagram for TaxType:



Public Member Functions

- **TaxType** (double rate, char type)
Constructs a new `TaxType` object.
- virtual ~**TaxType** ()
Virtual Destructor.
- virtual double **calculateTax** (double val)
Calculates the tax based on a given value.
- virtual void **setTax** (double rate)
Sets the tax rate.
- virtual double **getTaxRate** ()
Gets the current tax rate.
- char **getTaxType** ()
Gets the tax type identifier.

4.71.1 Detailed Description

Represents a specific type of tax with a rate and type identifier.

This class provides methods to calculate tax based on a given value, set and get the tax rate, and get the tax type identifier.

Version

1.0

Date

2024-11-04

4.71.2 Constructor & Destructor Documentation

4.71.2.1 TaxType()

```
TaxType::TaxType (
    double rate,
    char type )
```

Constructs a new [TaxType](#) object.

Parameters

<i>rate</i>	The tax rate.
<i>type</i>	The character representing the tax type.

4.71.3 Member Function Documentation

4.71.3.1 calculateTax()

```
double TaxType::calculateTax (
    double val ) [virtual]
```

Calculates the tax based on a given value.

Parameters

<i>val</i>	The value to calculate tax on.
------------	--------------------------------

Returns

The calculated tax.

Reimplemented in [Property](#), [Sales](#), and [Income](#).

4.71.3.2 getTaxRate()

```
double TaxType::getTaxRate ( ) [virtual]
```

Gets the current tax rate.

Returns

The current tax rate.

4.71.3.3 getTaxType()

```
char TaxType::getTaxType ( )
```

Gets the tax type identifier.

Returns

The character representing the tax type.

4.71.3.4 setTax()

```
void TaxType::setTax (
    double rate ) [virtual]
```

Sets the tax rate.

Parameters

<i>rate</i>	The new tax rate.
-------------	-------------------

Reimplemented in [VAT](#), [Income](#), [Property](#), and [Sales](#).

The documentation for this class was generated from the following files:

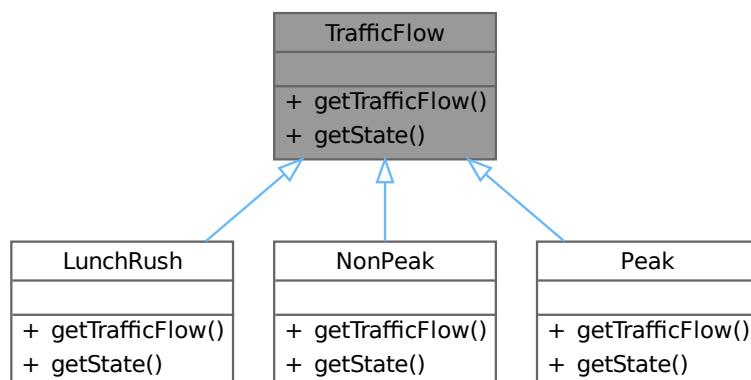
- /mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.h
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TaxType.cpp](#)

4.72 TrafficFlow Class Reference

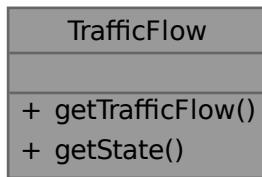
An abstract class that represents the traffic flow.

```
#include <TrafficFlow.h>
```

Inheritance diagram for TrafficFlow:



Collaboration diagram for TrafficFlow:



Public Member Functions

- virtual float [getTrafficFlow](#) ()=0
Pure virtual function to get the traffic flow.
- virtual char [getState](#) ()=0
Pure virtual function to get the state of the traffic.

4.72.1 Detailed Description

An abstract class that represents the traffic flow.

This class provides an interface for measuring traffic flow and obtaining the state of the traffic.

4.72.2 Member Function Documentation

4.72.2.1 getState()

```
virtual char TrafficFlow::getState ( ) [pure virtual]
```

Pure virtual function to get the state of the traffic.

Returns

char The current state of the traffic.

Implemented in [LunchRush](#), [NonPeak](#), and [Peak](#).

4.72.2.2 getTrafficFlow()

```
virtual float TrafficFlow::getTrafficFlow ( ) [pure virtual]
```

Pure virtual function to get the traffic flow.

Returns

float The current traffic flow.

Implemented in [LunchRush](#), [NonPeak](#), and [Peak](#).

The documentation for this class was generated from the following file:

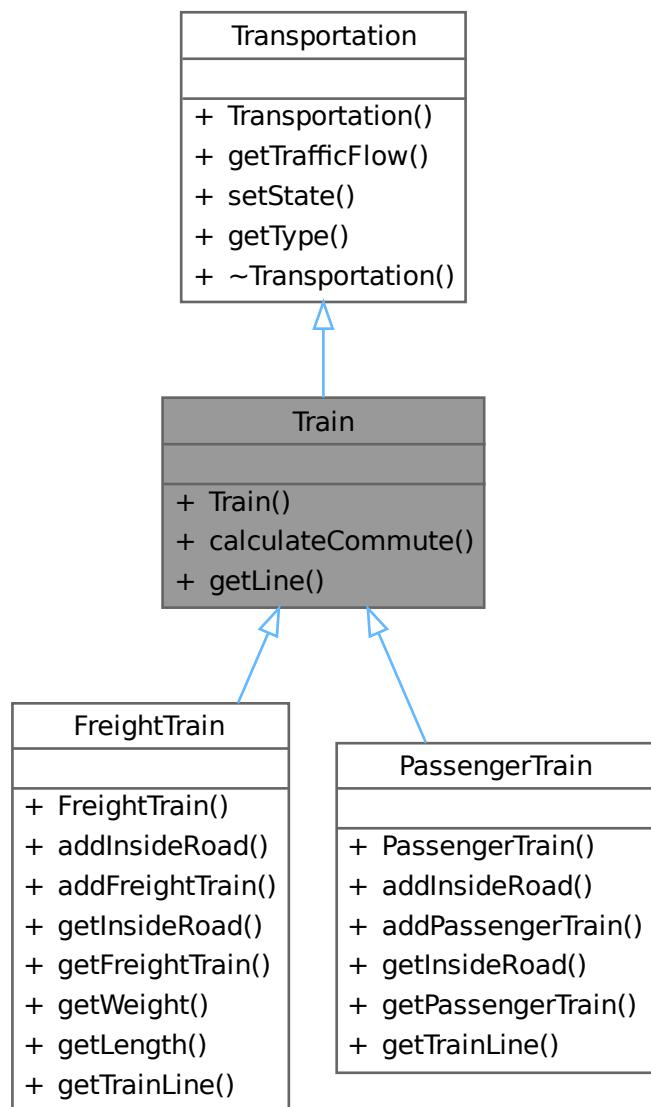
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TrafficFlow.h](#)

4.73 Train Class Reference

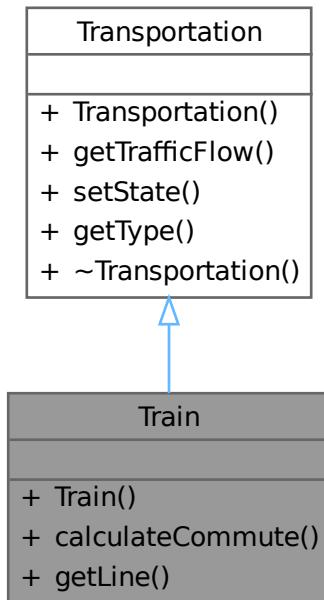
A class representing a train as a mode of transportation.

```
#include <Train.h>
```

Inheritance diagram for Train:



Collaboration diagram for Train:



Public Member Functions

- **Train** (char state, std::string line, char type)
*Construct a new **Train** object.*
- float **calculateCommute** ()
Calculate the commute time for the train.
- std::string **getLine** ()
Get the line on which the train operates.

Public Member Functions inherited from **Transportation**

- **Transportation** (char state, char type)
*Constructor for the **Transportation** class.*
- float **getTrafficFlow** ()
Gets the current traffic flow.
- bool **setState** (char state)
Sets the traffic flow state.
- char **getType** ()
Gets the type of transportation.
- **~Transportation** ()
*Destructor for the **Transportation** class.*

4.73.1 Detailed Description

A class representing a train as a mode of transportation.

The [Train](#) class inherits from the [Transportation](#) class and adds specific attributes and methods related to trains, such as the train line and commute calculation.

4.73.2 Constructor & Destructor Documentation

4.73.2.1 Train()

```
Train::Train (
    char state,
    std::string line,
    char type )
```

Construct a new [Train](#) object.

Constructs a new [Train](#) object.

Parameters

<i>state</i>	The state of the train.
<i>line</i>	The line on which the train operates.
<i>type</i>	The type of transportation.

4.73.3 Member Function Documentation

4.73.3.1 calculateCommute()

```
float Train::calculateCommute ( )
```

Calculate the commute time for the train.

Returns

The calculated commute time.

4.73.3.2 getLine()

```
std::string Train::getLine ( )
```

Get the line on which the train operates.

Gets the line on which the train operates.

Returns

The train line.

The line on which the train operates.

The documentation for this class was generated from the following files:

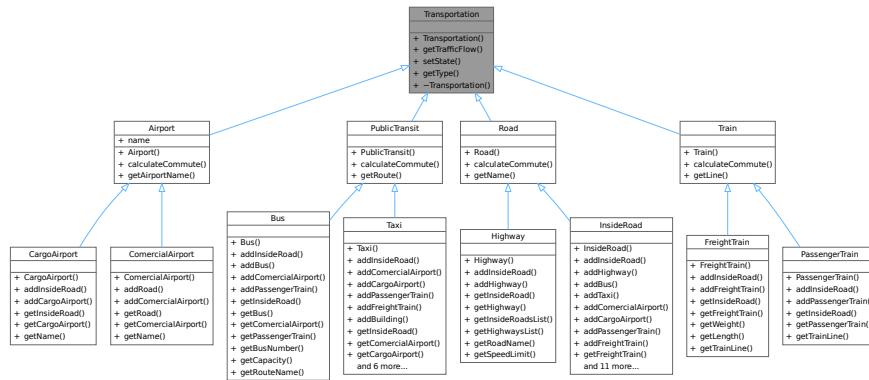
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Train.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Train.cpp](#)

4.74 Transportation Class Reference

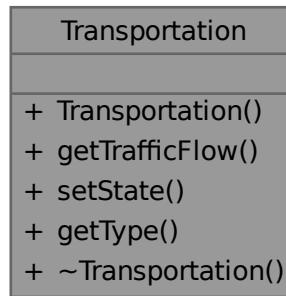
Manages traffic flow states and types of transportation.

```
#include <Transportation.h>
```

Inheritance diagram for Transportation:



Collaboration diagram for Transportation:



Public Member Functions

- **Transportation** (char state, char type)
Constructor for the `Transportation` class.
- float **getTrafficFlow** ()
Gets the current traffic flow.
- bool **setState** (char state)
Sets the traffic flow state.
- char **getType** ()
Gets the type of transportation.
- **~Transportation** ()
Destructor for the `Transportation` class.

4.74.1 Detailed Description

Manages traffic flow states and types of transportation.

The [Transportation](#) class provides methods to get and set the traffic flow state and type of transportation. It also includes a destructor to clean up resources.

4.74.2 Constructor & Destructor Documentation

4.74.2.1 [Transportation\(\)](#)

```
Transportation::Transportation (
    char state,
    char type )
```

Constructor for the [Transportation](#) class.

Constructs a new [Transportation](#) object.

Initializes the [Transportation](#) object with the given state and type.

Parameters

<i>state</i>	Initial state of the traffic flow.
<i>type</i>	Type of transportation.
<i>state</i>	The state of the transportation (P for Peak , N for NonPeak , L for LunchRush).
<i>type</i>	The type of transportation.

4.74.2.2 [~Transportation\(\)](#)

```
Transportation::~Transportation ( )
```

Destructor for the [Transportation](#) class.

Destroys the [Transportation](#) object.

Cleans up resources used by the [Transportation](#) object.

4.74.3 Member Function Documentation

4.74.3.1 [getTrafficFlow\(\)](#)

```
float Transportation::getTrafficFlow ( )
```

Gets the current traffic flow.

Gets the traffic flow based on the current state.

Returns

The current traffic flow as a float.

The current traffic flow.

4.74.3.2 getType()

```
char Transportation::getType ( )
```

Gets the type of transportation.

Returns

The type of transportation as a char.

The type of transportation.

4.74.3.3 setState()

```
bool Transportation::setState (   
    char state )
```

Sets the traffic flow state.

Sets the state of the transportation.

Updates the traffic flow state to the given state.

Parameters

state	New state of the traffic flow.
-------	--------------------------------

Returns

True if the state was successfully updated, false otherwise.

Parameters

state	The new state (P for Peak, N for NonPeak, L for LunchRush).
-------	---

Returns

True if the state was set successfully, false otherwise.

The documentation for this class was generated from the following files:

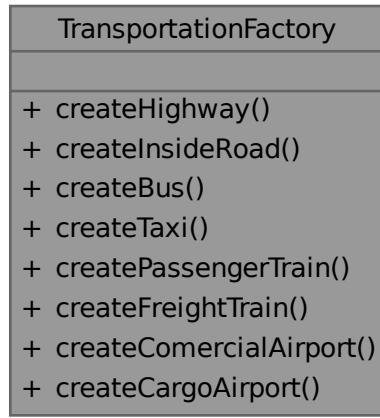
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Transportation.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Transportation.cpp](#)

4.75 TransportationFactory Class Reference

A factory class for creating different types of transportation objects.

```
#include <TransportationFactory.h>
```

Collaboration diagram for TransportationFactory:



Public Member Functions

- `Transportation * createHighway (char state, std::string roadName, float speedLimit)`
Creates a `Highway` object.
- `Transportation * createInsideRoad (char state, std::string roadName, float avgStopTime)`
Creates an `InsideRoad` object.
- `Transportation * createBus (char state, std::string route, int busNumber, int capacity)`
Creates a `Bus` object.
- `Transportation * createTaxi (char state, std::string route, std::string taxiCompany, int taxiNumber)`
Creates a `Taxi` object.
- `Transportation * createPassengerTrain (char state, std::string line)`
Creates a `PassengerTrain` object.
- `Transportation * createFreightTrain (char state, std::string line, float weight, float length)`
Creates a `FreightTrain` object.
- `Transportation * createComercialAirport (char state, std::string name)`
Creates a `ComercialAirport` object.
- `Transportation * createCargoAirport (char state, std::string name)`
Creates a `CargoAirport` object.

4.75.1 Detailed Description

A factory class for creating different types of transportation objects.

4.75.2 Member Function Documentation

4.75.2.1 createBus()

```
Transportation * TransportationFactory::createBus (
    char state,
    std::string route,
    int busNumber,
    int capacity )
```

Creates a [Bus](#) object.

Parameters

<i>state</i>	The state of the bus.
<i>route</i>	The route of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.

Returns

A pointer to the created [Bus](#) object.

Parameters

<i>state</i>	The state of the bus (P for Peak , N for NonPeak , L for LunchRush).
<i>route</i>	The route of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.

Returns

A pointer to the created [Bus](#) object, or nullptr if the parameters are invalid.

4.75.2.2 createCargoAirport()

```
Transportation * TransportationFactory::createCargoAirport (
    char state,
    std::string name )
```

Creates a [CargoAirport](#) object.

Parameters

<i>state</i>	The state of the cargo airport.
<i>name</i>	The name of the cargo airport.

Returns

A pointer to the created [CargoAirport](#) object.

Parameters

<i>state</i>	The state of the cargo airport (P for Peak , N for NonPeak , L for LunchRush).
<i>name</i>	The name of the cargo airport.

Returns

A pointer to the created [CargoAirport](#) object, or nullptr if the parameters are invalid.

4.75.2.3 createComercialAirport()

```
Transportation * TransportationFactory::createComercialAirport (
    char state,
    std::string name )
```

Creates a [ComercialAirport](#) object.

Parameters

<i>state</i>	The state of the commercial airport.
<i>name</i>	The name of the commercial airport.

Returns

A pointer to the created [ComercialAirport](#) object.

Parameters

<i>state</i>	The state of the commercial airport (P for Peak , N for NonPeak , L for LunchRush).
<i>name</i>	The name of the commercial airport.

Returns

A pointer to the created [ComercialAirport](#) object, or nullptr if the parameters are invalid.

4.75.2.4 createFreightTrain()

```
Transportation * TransportationFactory::createFreightTrain (
    char state,
    std::string line,
    float weight,
    float length )
```

Creates a [FreightTrain](#) object.

Parameters

<i>state</i>	The state of the freight train.
<i>line</i>	The line of the freight train.
<i>weight</i>	The weight of the freight.
<i>length</i>	The length of the freight train.

Returns

A pointer to the created [FreightTrain](#) object.

Parameters

<i>state</i>	The state of the freight train (P for Peak , N for NonPeak , L for LunchRush).
<i>line</i>	The line on which the freight train operates.
<i>weight</i>	The weight of the freight train.
<i>length</i>	The length of the freight train.

Returns

A pointer to the created [FreightTrain](#) object, or `nullptr` if the parameters are invalid.

4.75.2.5 createHighway()

```
Transportation * TransportationFactory::createHighway (
    char state,
    std::string roadName,
    float speedLimit )
```

Creates a [Highway](#) object.

Parameters

<i>state</i>	The state of the highway.
<i>roadName</i>	The name of the road.
<i>speedLimit</i>	The speed limit of the highway.

Returns

A pointer to the created [Highway](#) object.

Parameters

<i>state</i>	The state of the highway (P for Peak , N for NonPeak , L for LunchRush).
<i>roadName</i>	The name of the road.
<i>speedLimit</i>	The speed limit of the highway.

Returns

A pointer to the created [Highway](#) object, or nullptr if the parameters are invalid.

4.75.2.6 createInsideRoad()

```
Transportation * TransportationFactory::createInsideRoad (
    char state,
    std::string roadName,
    float avgStopTime )
```

Creates an [InsideRoad](#) object.

Parameters

<i>state</i>	The state of the inside road.
<i>roadName</i>	The name of the road.
<i>avgStopTime</i>	The average stop time on the road.

Returns

A pointer to the created [InsideRoad](#) object.

Parameters

<i>state</i>	The state of the inside road (P for Peak , N for NonPeak , L for LunchRush).
<i>roadName</i>	The name of the road.
<i>avgStopTime</i>	The average stop time on the inside road.

Returns

A pointer to the created [InsideRoad](#) object, or nullptr if the parameters are invalid.

4.75.2.7 createPassengerTrain()

```
Transportation * TransportationFactory::createPassengerTrain (
    char state,
    std::string line )
```

Creates a [PassengerTrain](#) object.

Parameters

<i>state</i>	The state of the passenger train.
<i>line</i>	The line of the passenger train.

Returns

A pointer to the created [PassengerTrain](#) object.

Parameters

<i>state</i>	The state of the passenger train (P for Peak , N for NonPeak , L for LunchRush).
<i>line</i>	The line on which the passenger train operates.

Returns

A pointer to the created [PassengerTrain](#) object, or nullptr if the parameters are invalid.

4.75.2.8 createTaxi()

```
Transportation * TransportationFactory::createTaxi (
    char state,
    std::string route,
    std::string taxiCompany,
    int taxiNumber )
```

Creates a [Taxi](#) object.

Parameters

<i>state</i>	The state of the taxi.
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The company of the taxi.
<i>taxisNumber</i>	The taxi number.

Returns

A pointer to the created [Taxi](#) object.

Parameters

<i>state</i>	The state of the taxi (P for Peak , N for NonPeak , L for LunchRush).
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The taxi company.
<i>taxisNumber</i>	The taxi number.

Returns

A pointer to the created [Taxi](#) object, or nullptr if the parameters are invalid.

The documentation for this class was generated from the following files:

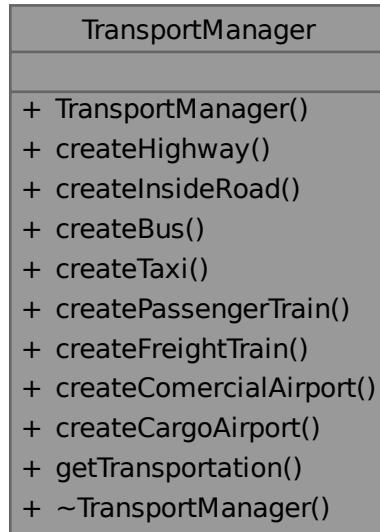
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TransportationFactory.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TransportationFactory.cpp](#)

4.76 TransportManager Class Reference

Manages various types of transportation objects.

```
#include <TransportManager.h>
```

Collaboration diagram for TransportManager:



Public Member Functions

- **TransportManager ()**
Constructor for the `TransportManager` class.
- **bool createHighway (char state, std::string roadName, float speedLimit)**
Creates a highway.
- **bool createInsideRoad (char state, std::string roadName, float avgStopTime)**
Creates an inside road.
- **bool createBus (char state, std::string route, int busNumber, int capacity)**
Creates a bus.
- **bool createTaxi (char state, std::string route, std::string taxiCompany, int taxiNumber)**
Creates a taxi.
- **bool createPassengerTrain (char state, std::string line)**
Creates a passenger train.
- **bool createFreightTrain (char state, std::string line, float weight, float length)**
Creates a freight train.
- **bool createComercialAirport (char state, std::string name)**
Creates a commercial airport.
- **bool createCargoAirport (char state, std::string name)**
Creates a cargo airport.
- **Transportation * getTransportation (size_t index)**
Gets a transportation object by index.
- **~TransportManager ()**
Destructor for the `TransportManager` class.

4.76.1 Detailed Description

Manages various types of transportation objects.

The [TransportManager](#) class uses a [TransportationFactory](#) to create and manage different types of transportation objects such as highways, roads, buses, taxis, trains, and airports.

4.76.2 Constructor & Destructor Documentation

4.76.2.1 [TransportManager\(\)](#)

```
TransportManager::TransportManager ( )
```

Constructor for the [TransportManager](#) class.

Constructs a new [TransportManager](#) object.

4.76.2.2 [~TransportManager\(\)](#)

```
TransportManager::~TransportManager ( )
```

Destructor for the [TransportManager](#) class.

Destroys the [TransportManager](#) object and cleans up resources.

4.76.3 Member Function Documentation

4.76.3.1 [createBus\(\)](#)

```
bool TransportManager::createBus (
    char state,
    std::string route,
    int busNumber,
    int capacity )
```

Creates a bus.

Creates a [Bus](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the bus operates.
<i>route</i>	The route of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.

Returns

True if the bus was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the bus.
<i>route</i>	The route of the bus.
<i>busNumber</i>	The bus number.
<i>capacity</i>	The capacity of the bus.

Returns

True if the bus was created successfully, false otherwise.

4.76.3.2 createCargoAirport()

```
bool TransportManager::createCargoAirport (
    char state,
    std::string name )
```

Creates a cargo airport.

Creates a [CargoAirport](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the airport is located.
<i>name</i>	The name of the airport.

Returns

True if the airport was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the cargo airport.
<i>name</i>	The name of the cargo airport.

Returns

True if the cargo airport was created successfully, false otherwise.

4.76.3.3 createComercialAirport()

```
bool TransportManager::createComercialAirport (
    char state,
    std::string name )
```

Creates a commercial airport.

Creates a [ComercialAirport](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the airport is located.
<i>name</i>	The name of the airport.

Returns

True if the airport was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the commercial airport.
<i>name</i>	The name of the commercial airport.

Returns

True if the commercial airport was created successfully, false otherwise.

4.76.3.4 `createFreightTrain()`

```
bool TransportManager::createFreightTrain (
    char state,
    std::string line,
    float weight,
    float length )
```

Creates a freight train.

Creates a [FreightTrain](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the train operates.
<i>line</i>	The line of the train.
<i>weight</i>	The weight of the freight.
<i>length</i>	The length of the freight.

Returns

True if the train was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the freight train.
<i>line</i>	The line of the freight train.

Parameters

<i>weight</i>	The weight of the freight.
<i>length</i>	The length of the freight train.

Returns

True if the freight train was created successfully, false otherwise.

4.76.3.5 createHighway()

```
bool TransportManager::createHighway (
    char state,
    std::string roadName,
    float speedLimit )
```

Creates a highway.

Creates a [Highway](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the highway is located.
<i>roadName</i>	The name of the highway.
<i>speedLimit</i>	The speed limit of the highway.

Returns

True if the highway was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the highway.
<i>roadName</i>	The name of the road.
<i>speedLimit</i>	The speed limit of the highway.

Returns

True if the highway was created successfully, false otherwise.

4.76.3.6 createInsideRoad()

```
bool TransportManager::createInsideRoad (
    char state,
    std::string roadName,
    float avgStopTime )
```

Creates an inside road.

Creates an [InsideRoad](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the road is located.
<i>roadName</i>	The name of the road.
<i>avgStopTime</i>	The average stop time on the road.

Returns

True if the road was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the inside road.
<i>roadName</i>	The name of the road.
<i>avgStopTime</i>	The average stop time on the road.

Returns

True if the inside road was created successfully, false otherwise.

4.76.3.7 createPassengerTrain()

```
bool TransportManager::createPassengerTrain (
    char state,
    std::string line )
```

Creates a passenger train.

Creates a [PassengerTrain](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the train operates.
<i>line</i>	The line of the train.

Returns

True if the train was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the passenger train.
<i>line</i>	The line of the passenger train.

Returns

True if the passenger train was created successfully, false otherwise.

4.76.3.8 createTaxi()

```
bool TransportManager::createTaxi (
    char state,
    std::string route,
    std::string taxiCompany,
    int taxiNumber )
```

Creates a taxi.

Creates a [Taxi](#) object and adds it to the transportation list.

Parameters

<i>state</i>	The state where the taxi operates.
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The company that owns the taxi.
<i>taxisNumber</i>	The taxi number.

Returns

True if the taxi was created successfully, false otherwise.

Parameters

<i>state</i>	The state of the taxi.
<i>route</i>	The route of the taxi.
<i>taxisCompany</i>	The company of the taxi.
<i>taxisNumber</i>	The taxi number.

Returns

True if the taxi was created successfully, false otherwise.

4.76.3.9 getTransportation()

```
Transportation * TransportManager::getTransportation (
    size_t index )
```

Gets a transportation object by index.

Parameters

<i>index</i>	The index of the transportation object.
--------------	---

Returns

A pointer to the transportation object.

Parameters

<i>index</i>	The index of the transportation object.
--------------	---

Returns

A pointer to the transportation object, or nullptr if the index is out of range.

The documentation for this class was generated from the following files:

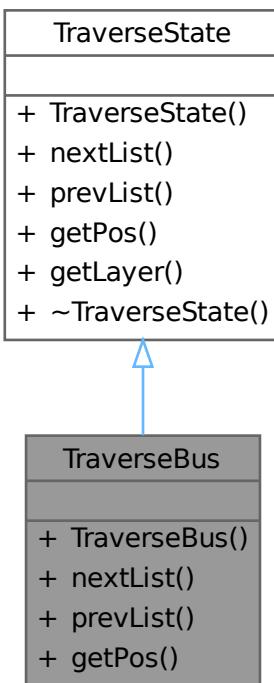
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TransportManager.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TransportManager.cpp](#)

4.77 TraverseBus Class Reference

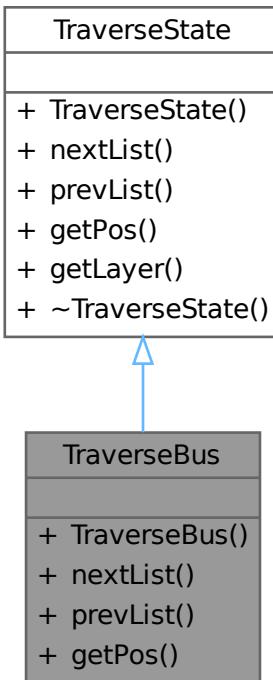
A class to manage the traversal state of a bus.

```
#include <TraverseBus.h>
```

Inheritance diagram for TraverseBus:



Collaboration diagram for TraverseBus:



Public Member Functions

- [TraverseBus \(Transportation *element\)](#)
Constructor for TraverseBus.
- [bool nextList \(\)](#)
Move to the next list.
- [bool prevList \(\)](#)
Move to the previous list.
- [Transportation * getPos \(size_t x\)](#)
Get the transportation element at the specified position.

Public Member Functions inherited from [TraverseState](#)

- [TraverseState \(Transportation *element\)](#)
Constructor for TraverseState.
- [Transportation * getLayer \(\)](#)
Gets the current Transportation layer.

4.77.1 Detailed Description

A class to manage the traversal state of a bus.

The [TraverseBus](#) class provides functionality to traverse through different lists of transportation elements, such as buses, commercial airports, and passenger trains. It maintains the current list and provides methods to navigate to the next or previous list.

4.77.2 Constructor & Destructor Documentation

4.77.2.1 TraverseBus()

```
TraverseBus::TraverseBus (
    Transportation * element )
```

Constructor for [TraverseBus](#).

Constructs a new [TraverseBus](#) object.

Initializes a new instance of the [TraverseBus](#) class with the given transportation element.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.77.3 Member Function Documentation

4.77.3.1 getPos()

```
Transportation * TraverseBus::getPos (
    size_t x ) [virtual]
```

Get the transportation element at the specified position.

Gets the transportation element at the specified position.

Retrieves the transportation element at the given position within the current list.

Parameters

<code>x</code>	The position index within the current list.
----------------	---

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.77.3.2 nextList()

```
bool TraverseBus::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Advances the current list index to the next list, if possible.

Returns

True if the list was successfully advanced, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.77.3.3 prevList()

```
bool TraverseBus::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Moves the current list index to the previous list, if possible.

Returns

True if the list was successfully moved back, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

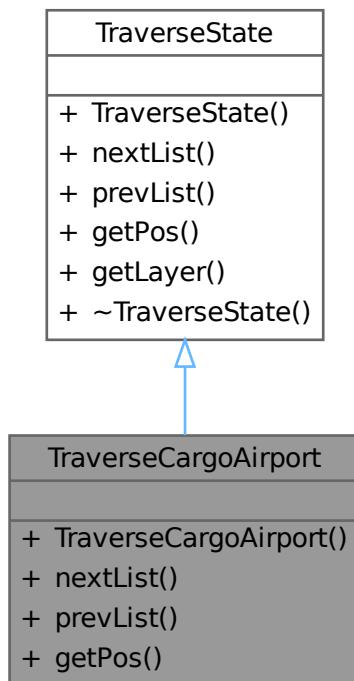
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseBus.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseBus.cpp](#)

4.78 TraverseCargoAirport Class Reference

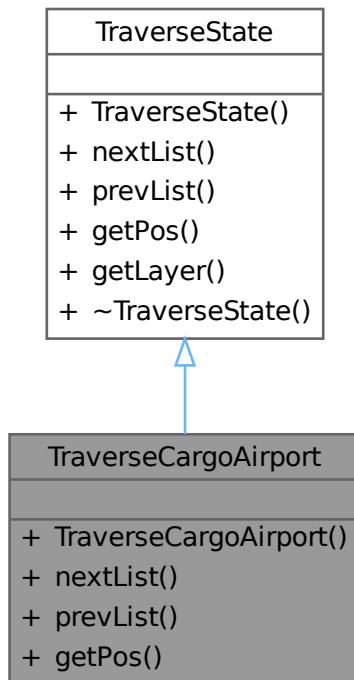
A class to traverse through a cargo airport.

```
#include <TraverseCargoAirport.h>
```

Inheritance diagram for TraverseCargoAirport:



Collaboration diagram for TraverseCargoAirport:



Public Member Functions

- `TraverseCargoAirport (Transportation *element)`
Constructor for TraverseCargoAirport.
- `bool nextList ()`
Move to the next list.
- `bool prevList ()`
Move to the previous list.
- `Transportation * getPos (size_t x)`
Get the position of a specific element in the current list.

Public Member Functions inherited from `TraverseState`

- `TraverseState (Transportation *element)`
Constructor for TraverseState.
- `Transportation * getLayer ()`
Gets the current `Transportation` layer.

4.78.1 Detailed Description

A class to traverse through a cargo airport.

The `TraverseCargoAirport` class provides methods to navigate through different lists within a cargo airport.

4.78.2 Constructor & Destructor Documentation

4.78.2.1 TraverseCargoAirport()

```
TraverseCargoAirport::TraverseCargoAirport (
    Transportation * element )
```

Constructor for [TraverseCargoAirport](#).

Constructs a new [TraverseCargoAirport](#) object.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.78.3 Member Function Documentation

4.78.3.1 getPos()

```
Transportation * TraverseCargoAirport::getPos (
    size_t x ) [virtual]
```

Get the position of a specific element in the current list.

Gets the transportation element at the specified position.

Parameters

<code>x</code>	The index of the element.
----------------	---------------------------

Returns

A pointer to the [Transportation](#) object at the specified index.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.78.3.2 nextList()

```
bool TraverseCargoAirport::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.78.3.3 prevList()

```
bool TraverseCargoAirport::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

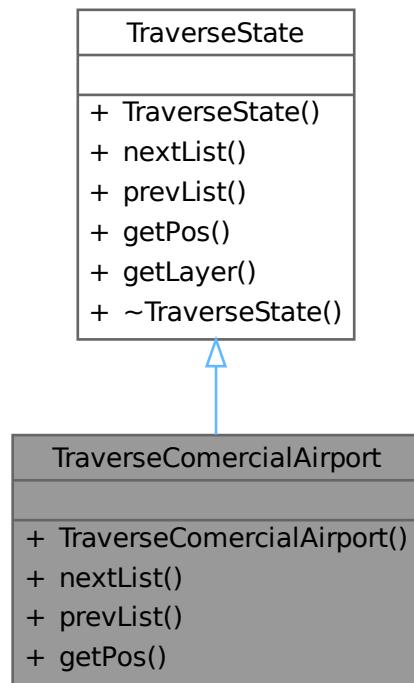
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseCargoAirport.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseCargoAirport.cpp](#)

4.79 TraverseComercialAirport Class Reference

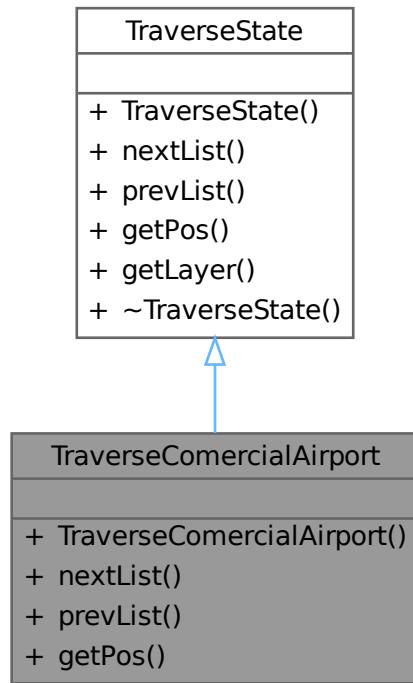
A class to traverse through a commercial airport.

```
#include <TraverseComercialAirport.h>
```

Inheritance diagram for TraverseComercialAirport:



Collaboration diagram for TraverseComercialAirport:



Public Member Functions

- `TraverseComercialAirport (Transportation *element)`
Constructor for TraverseComercialAirport.
- `bool nextList ()`
Move to the next list.
- `bool prevList ()`
Move to the previous list.
- `Transportation * getPos (size_t x)`
Get the position of a specific element.

Public Member Functions inherited from `TraverseState`

- `TraverseState (Transportation *element)`
Constructor for TraverseState.
- `Transportation * getLayer ()`
Gets the current `Transportation` layer.

4.79.1 Detailed Description

A class to traverse through a commercial airport.

The `TraverseComercialAirport` class provides methods to navigate through different lists within a commercial airport. It maintains the current position and allows moving to the next or previous list.

4.79.2 Constructor & Destructor Documentation

4.79.2.1 TraverseComercialAirport()

```
TraverseComercialAirport::TraverseComercialAirport (
    Transportation * element )
```

Constructor for [TraverseComercialAirport](#).

Constructs a new [TraverseComercialAirport](#) object.

Initializes the [TraverseComercialAirport](#) with a given [Transportation](#) element.

Parameters

<i>element</i>	A pointer to a Transportation object.
<i>element</i>	Pointer to the Transportation element.

4.79.3 Member Function Documentation

4.79.3.1 getPos()

```
Transportation * TraverseComercialAirport::getPos (
    size_t x ) [virtual]
```

Get the position of a specific element.

Gets the transportation element at the specified position.

Retrieves the [Transportation](#) object at the specified position.

Parameters

<i>x</i>	The index of the desired element.
----------	-----------------------------------

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<i>x</i>	The position of the transportation element.
----------	---

Returns

Pointer to the transportation element, or nullptr if the position is invalid.

Implements [TraverseState](#).

4.79.3.2 nextList()

```
bool TraverseComercialAirport::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Advances the current list index to the next list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.79.3.3 prevList()

```
bool TraverseComercialAirport::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Moves the current list index to the previous list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

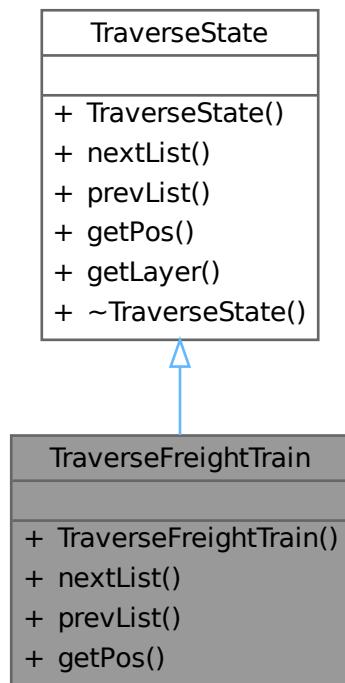
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseComercialAirport.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseComercialAirport.cpp](#)

4.80 TraverseFreightTrain Class Reference

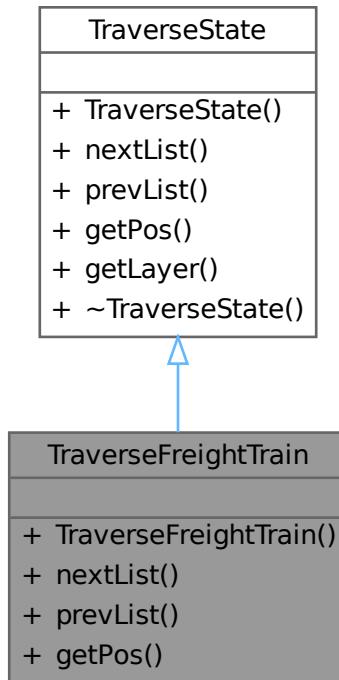
A class to traverse through [FreightTrain](#) objects.

```
#include <TraverseFreightTrain.h>
```

Inheritance diagram for TraverseFreightTrain:



Collaboration diagram for TraverseFreightTrain:



Public Member Functions

- `TraverseFreightTrain (Transportation *element)`
Constructor for TraverseFreightTrain.
- `bool nextList ()`
Move to the next list.
- `bool prevList ()`
Move to the previous list.
- `Transportation * getPos (size_t x)`
Get the position in the list.

Public Member Functions inherited from `TraverseState`

- `TraverseState (Transportation *element)`
Constructor for TraverseState.
- `Transportation * getLayer ()`
Gets the current Transportation layer.

4.80.1 Detailed Description

A class to traverse through `FreightTrain` objects.

The `TraverseFreightTrain` class provides methods to navigate through a list of `FreightTrain` objects within a `Transportation` system. It maintains the current position within the list and allows moving to the next or previous list.

4.80.2 Constructor & Destructor Documentation

4.80.2.1 TraverseFreightTrain()

```
TraverseFreightTrain::TraverseFreightTrain (   
    Transportation * element )
```

Constructor for [TraverseFreightTrain](#).

Constructs a new [TraverseFreightTrain](#) object.

Initializes a new instance of the [TraverseFreightTrain](#) class with the given [Transportation](#) element.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.80.3 Member Function Documentation

4.80.3.1 getPos()

```
Transportation * TraverseFreightTrain::getPos (   
    size_t x ) [virtual]
```

Get the position in the list.

Gets the transportation element at the specified position.

Retrieves the [Transportation](#) object at the specified position in the list.

Parameters

<code>x</code>	The position in the list.
----------------	---------------------------

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.80.3.2 nextList()

```
bool TraverseFreightTrain::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Advances the current position to the next list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.80.3.3 prevList()

```
bool TraverseFreightTrain::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Moves the current position to the previous list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

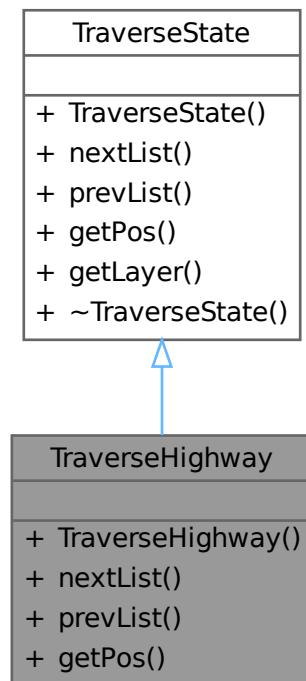
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseFreightTrain.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseFreightTrain.cpp](#)

4.81 TraverseHighway Class Reference

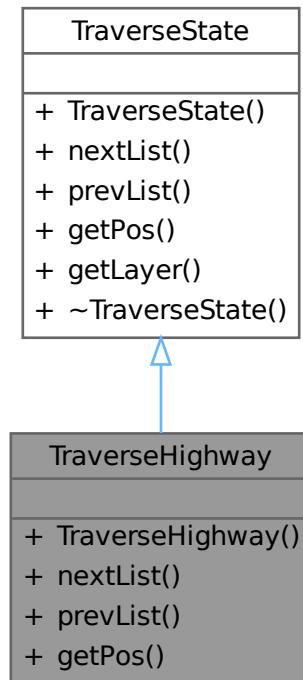
A class to traverse through highways in a transportation system.

```
#include <TraverseHighway.h>
```

Inheritance diagram for TraverseHighway:



Collaboration diagram for TraverseHighway:



Public Member Functions

- [TraverseHighway \(Transportation *element\)](#)
Constructor for TraverseHighway.
- [bool nextList \(\)](#)
Move to the next highway in the list.
- [bool prevList \(\)](#)
Move to the previous highway in the list.
- [Transportation * getPos \(size_t x\)](#)
Get the `Transportation` object at the specified position.

Public Member Functions inherited from `TraverseState`

- [TraverseState \(Transportation *element\)](#)
Constructor for TraverseState.
- [Transportation * getLayer \(\)](#)
Gets the current `Transportation` layer.

4.81.1 Detailed Description

A class to traverse through highways in a transportation system.

The `TraverseHighway` class provides methods to navigate through a list of highways, allowing movement to the next or previous highway and retrieving the current position.

4.81.2 Constructor & Destructor Documentation

4.81.2.1 TraverseHighway()

```
TraverseHighway::TraverseHighway (   
    Transportation * element )
```

Constructor for [TraverseHighway](#).

Constructs a new [TraverseHighway](#) object.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.81.3 Member Function Documentation

4.81.3.1 getPos()

```
Transportation * TraverseHighway::getPos (   
    size_t x ) [virtual]
```

Get the [Transportation](#) object at the specified position.

Gets the transportation element at the specified position.

Parameters

<code>x</code>	The position in the list.
----------------	---------------------------

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.81.3.2 nextList()

```
bool TraverseHighway::nextList ( ) [virtual]
```

Move to the next highway in the list.

Moves to the next list of transportation elements.

Returns

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.81.3.3 prevList()

```
bool TraverseHighway::prevList ( ) [virtual]
```

Move to the previous highway in the list.

Moves to the previous list of transportation elements.

Returns

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

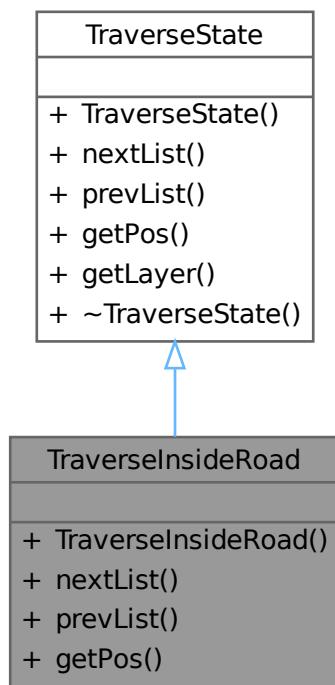
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseHighway.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseHighway.cpp](#)

4.82 TraverserInsideRoad Class Reference

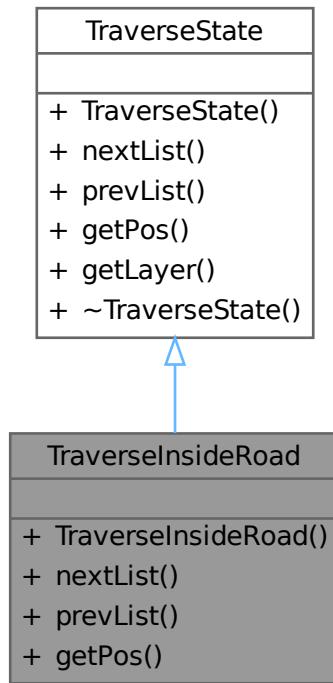
A class to traverse inside roads in a transportation system.

```
#include <TraverserInsideRoad.h>
```

Inheritance diagram for TraverseInsideRoad:



Collaboration diagram for TraverseInsideRoad:



Public Member Functions

- [TraverseInsideRoad \(Transportation *element\)](#)
Constructor for TraverseInsideRoad.
- [bool nextList \(\)](#)
Move to the next list.
- [bool prevList \(\)](#)
Move to the previous list.
- [Transportation * getPos \(size_t x\)](#)
Get the `Transportation` object at the specified position.

Public Member Functions inherited from `TraverseState`

- [TraverseState \(Transportation *element\)](#)
Constructor for TraverseState.
- [Transportation * getLayer \(\)](#)
Gets the current `Transportation` layer.

4.82.1 Detailed Description

A class to traverse inside roads in a transportation system.

The `TraverseInsideRoad` class is a state class that allows traversal of inside roads within a transportation system. It inherits from the `TraverseState` class.

4.82.2 Constructor & Destructor Documentation

4.82.2.1 TraverseInsideRoad()

```
TraverseInsideRoad::TraverseInsideRoad (   
    Transportation * element )
```

Constructor for [TraverseInsideRoad](#).

Constructs a new [TraverseInsideRoad](#) object.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.82.3 Member Function Documentation

4.82.3.1 getPos()

```
Transportation * TraverseInsideRoad::getPos (   
    size_t x ) [virtual]
```

Get the [Transportation](#) object at the specified position.

Gets the transportation element at the specified position.

Parameters

<code>x</code>	The position index.
----------------	---------------------

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.82.3.2 nextList()

```
bool TraverseInsideRoad::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.82.3.3 prevList()

```
bool TraverseInsideRoad::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

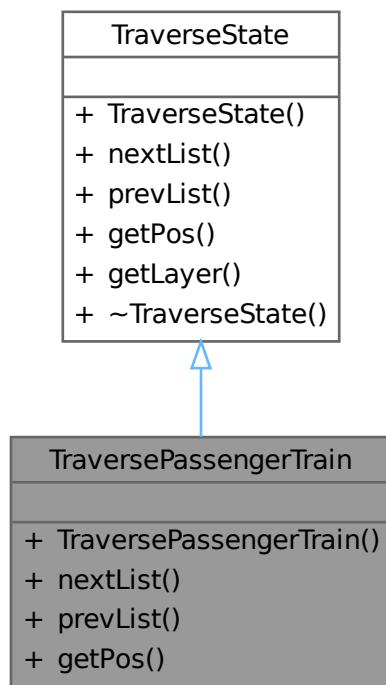
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseInsideRoad.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseInsideRoad.cpp](#)

4.83 TraversePassengerTrain Class Reference

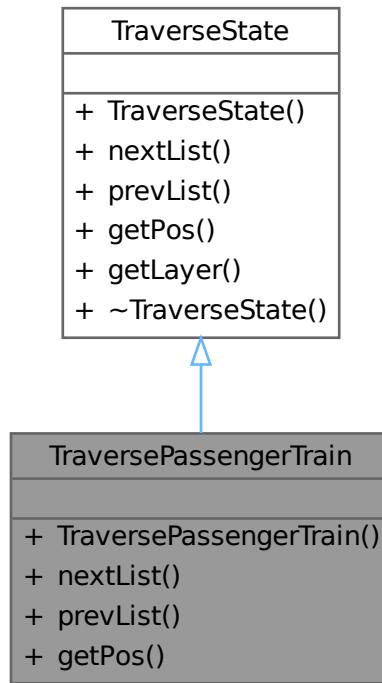
A class to traverse through a passenger train.

```
#include <TraversePassengerTrain.h>
```

Inheritance diagram for TraversePassengerTrain:



Collaboration diagram for TraversePassengerTrain:



Public Member Functions

- `TraversePassengerTrain (Transportation *element)`
Constructor for TraversePassengerTrain.
- `bool nextList ()`
Move to the next list.
- `bool prevList ()`
Move to the previous list.
- `Transportation * getPos (size_t x)`
Get the position at the specified index.

Public Member Functions inherited from `TraverseState`

- `TraverseState (Transportation *element)`
Constructor for TraverseState.
- `Transportation * getLayer ()`
Gets the current `Transportation` layer.

4.83.1 Detailed Description

A class to traverse through a passenger train.

The `TraversePassengerTrain` class provides functionality to traverse through a passenger train by maintaining the current position and bounds within the train.

4.83.2 Constructor & Destructor Documentation

4.83.2.1 TraversePassengerTrain()

```
TraversePassengerTrain::TraversePassengerTrain (   
    Transportation * element )
```

Constructor for [TraversePassengerTrain](#).

Constructs a new [TraversePassengerTrain](#) object.

Initializes a new instance of the [TraversePassengerTrain](#) class.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to the Transportation element.

4.83.3 Member Function Documentation

4.83.3.1 getPos()

```
Transportation * TraversePassengerTrain::getPos (   
    size_t x ) [virtual]
```

Get the position at the specified index.

Gets the transportation element at the specified position.

Retrieves the [Transportation](#) object at the specified index.

Parameters

<code>x</code>	The index of the position to retrieve.
----------------	--

Returns

A pointer to the [Transportation](#) object at the specified index.

Parameters

<code>x</code>	The position of the transportation element.
----------------	---

Returns

Pointer to the transportation element, or `nullptr` if the position is invalid.

Implements [TraverseState](#).

4.83.3.2 nextList()

```
bool TraversePassengerTrain::nextList ( ) [virtual]
```

Move to the next list.

Moves to the next list of transportation elements.

Advances the current position to the next list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

4.83.3.3 prevList()

```
bool TraversePassengerTrain::prevList ( ) [virtual]
```

Move to the previous list.

Moves to the previous list of transportation elements.

Moves the current position to the previous list.

Returns

true if the operation was successful, false otherwise.

True if the move was successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

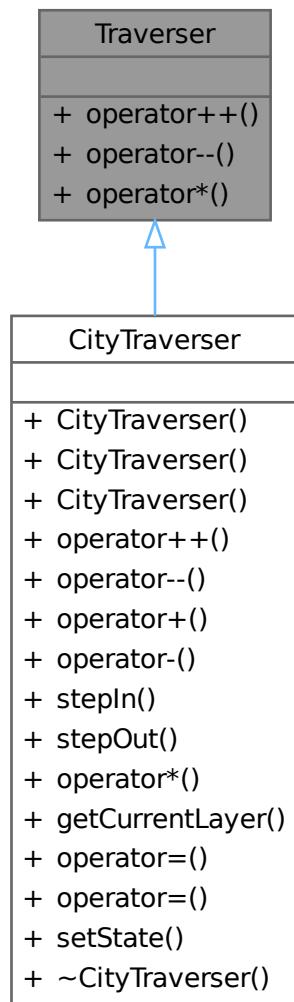
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraversePassengerTrain.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraversePassengerTrain.cpp](#)

4.84 Traverser Class Reference

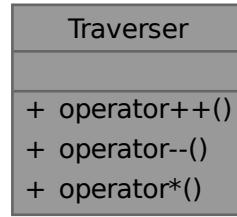
Interface for iterating over [Transportation](#) objects.

```
#include <Traverser.h>
```

Inheritance diagram for Traverser:



Collaboration diagram for Traverser:



Public Member Functions

- virtual bool `operator++ ()=0`
Increment the iterator.
- virtual bool `operator-- ()=0`
Decrement the iterator.
- virtual `Transportation * operator* ()=0`
Dereference the iterator.

4.84.1 Detailed Description

Interface for iterating over `Transportation` objects.

The `Traverser` class provides pure virtual functions for incrementing, decrementing, and dereferencing iterators over `Transportation` objects. Classes that inherit from `Traverser` must implement these functions.

4.84.2 Member Function Documentation

4.84.2.1 `operator*()`

```
virtual Transportation * Traverser::operator* ( ) [pure virtual]
```

Dereference the iterator.

Pure virtual function to dereference the iterator and access the current `Transportation` object.

Returns

Pointer to the current `Transportation` object.

Implemented in `CityTraverser`.

4.84.2.2 operator++()

```
virtual bool Traverser::operator++ ( ) [pure virtual]
```

Increment the iterator.

Pure virtual function to increment the iterator to the next [Transportation](#) object.

Returns

true if the increment was successful, false otherwise.

Implemented in [CityTraverser](#).

4.84.2.3 operator--()

```
virtual bool Traverser::operator-- ( ) [pure virtual]
```

Decrement the iterator.

Pure virtual function to decrement the iterator to the previous [Transportation](#) object.

Returns

true if the decrement was successful, false otherwise.

Implemented in [CityTraverser](#).

The documentation for this class was generated from the following file:

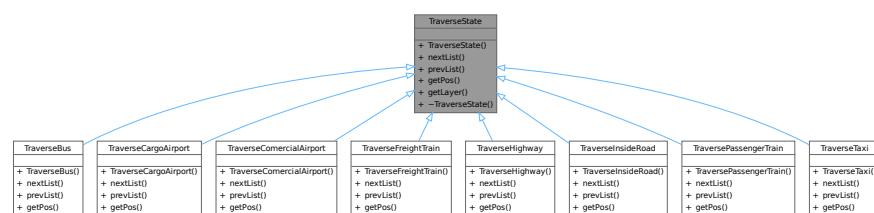
- /mnt/c/users/rudie/documents/sem2/214/project/src/[Traverser.h](#)

4.85 TraverseState Class Reference

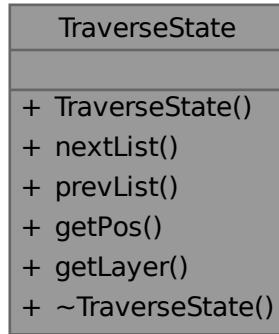
Abstract class that provides an interface for traversing through a list of [Transportation](#) elements.

```
#include <TraverseState.h>
```

Inheritance diagram for TraverseState:



Collaboration diagram for TraverseState:



Public Member Functions

- `TraverseState (Transportation *element)`
Constructor for TraverseState.
- `virtual bool nextList ()=0`
Pure virtual function to move to the next list.
- `virtual bool prevList ()=0`
Pure virtual function to move to the previous list.
- `virtual Transportation * getPos (size_t x)=0`
Pure virtual function to get the `Transportation` element at a specific position.
- `Transportation * getLayer ()`
Gets the current `Transportation` layer.

4.85.1 Detailed Description

Abstract class that provides an interface for traversing through a list of `Transportation` elements.

4.85.2 Constructor & Destructor Documentation

4.85.2.1 TraverseState()

```
TraverseState::TraverseState (
    Transportation * layer )
```

Constructor for `TraverseState`.

Construct a new `TraverseState` object.

Parameters

<i>element</i>	Pointer to a Transportation element.
<i>layer</i>	Pointer to the Transportation layer.

4.85.3 Member Function Documentation**4.85.3.1 getLayer()**

```
Transportation * TraverseState::getLayer ( )
```

Gets the current [Transportation](#) layer.

Get the [Transportation](#) layer.

Returns

Pointer to the current [Transportation](#) layer.

`Transportation*` Pointer to the [Transportation](#) layer.

4.85.3.2 getPos()

```
virtual Transportation * TraverseState::getPos (
    size_t x ) [pure virtual]
```

Pure virtual function to get the [Transportation](#) element at a specific position.

Parameters

<i>x</i>	Position index.
----------	-----------------

Returns

Pointer to the [Transportation](#) element at position *x*.

Implemented in [TraverseBus](#), [TraverseCargoAirport](#), [TraverseComercialAirport](#), [TraverseFreightTrain](#), [TraverseHighway](#), [TraverseInsideRoad](#), [TraversePassengerTrain](#), and [TraverseTaxi](#).

4.85.3.3 nextList()

```
virtual bool TraverseState::nextList ( ) [pure virtual]
```

Pure virtual function to move to the next list.

Returns

True if successful, false otherwise.

Implemented in [TraverseBus](#), [TraverseCargoAirport](#), [TraverseComercialAirport](#), [TraverseFreightTrain](#), [TraverseHighway](#), [TraverseInsideRoad](#), [TraversePassengerTrain](#), and [TraverseTaxi](#).

4.85.3.4 prevList()

```
virtual bool TraverseState::prevList ( ) [pure virtual]
```

Pure virtual function to move to the previous list.

Returns

True if successful, false otherwise.

Implemented in [TraverseBus](#), [TraverseCargoAirport](#), [TraverseComercialAirport](#), [TraverseFreightTrain](#), [TraverseHighway](#), [TraverseInsideRoad](#), [TraversePassengerTrain](#), and [TraverseTaxi](#).

The documentation for this class was generated from the following files:

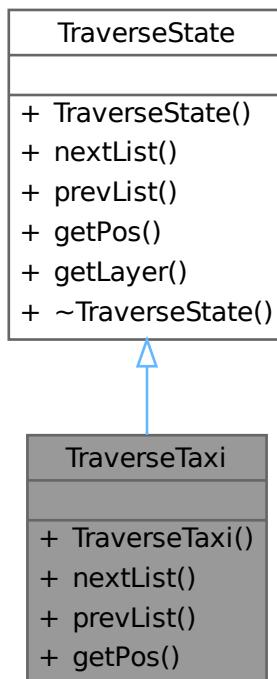
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseState.cpp](#)

4.86 TraverseTaxi Class Reference

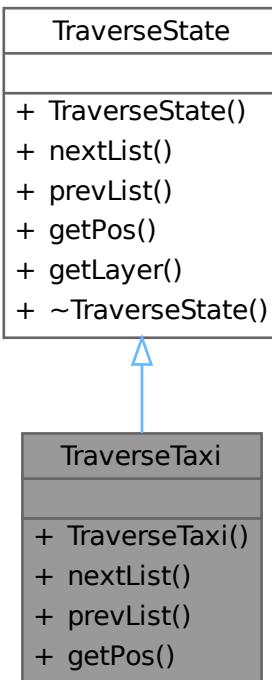
Manages the traversal state of a [Taxi](#) object.

```
#include <TraverseTaxi.h>
```

Inheritance diagram for TraverseTaxi:



Collaboration diagram for TraverseTaxi:



Public Member Functions

- [TraverseTaxi \(Transportation *element\)](#)
Constructor for TraverseTaxi.
- [bool nextList \(\)](#)
Move to the next list.
- [bool prevList \(\)](#)
Move to the previous list.
- [Transportation * getPos \(size_t x\)](#)
Get the transportation element at the specified position.

Public Member Functions inherited from [TraverseState](#)

- [TraverseState \(Transportation *element\)](#)
Constructor for TraverseState.
- [Transportation * getLayer \(\)](#)
Gets the current [Transportation](#) layer.

4.86.1 Detailed Description

Manages the traversal state of a [Taxi](#) object.

The [TraverseTaxi](#) class provides functionality to traverse through different lists of transportation elements, specifically for [Taxi](#) objects. It maintains the current list index and an upper bound for the list index.

4.86.2 Constructor & Destructor Documentation

4.86.2.1 TraverseTaxi()

```
TraverseTaxi::TraverseTaxi (
    Transportation * element )
```

Constructor for [TraverseTaxi](#).

Construct a new [TraverseTaxi](#) object.

Parameters

<code>element</code>	A pointer to a Transportation object.
<code>element</code>	Pointer to a Transportation element.

4.86.3 Member Function Documentation

4.86.3.1 getPos()

```
Transportation * TraverseTaxi::getPos (
    size_t x ) [virtual]
```

Get the transportation element at the specified position.

Get the [Transportation](#) element at a specific position.

Parameters

<code>x</code>	The position index.
----------------	---------------------

Returns

A pointer to the [Transportation](#) object at the specified position.

Parameters

<code>x</code>	Position index.
----------------	-----------------

Returns

Pointer to the [Transportation](#) element at position x.

Implements [TraverseState](#).

4.86.3.2 nextList()

```
bool TraverseTaxi::nextList ( ) [virtual]
```

Move to the next list.

Move to the next list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if successful, false otherwise.

Implements [TraverseState](#).

4.86.3.3 prevList()

```
bool TraverseTaxi::prevList ( ) [virtual]
```

Move to the previous list.

Move to the previous list of transportation elements.

Returns

True if the operation was successful, false otherwise.

True if successful, false otherwise.

Implements [TraverseState](#).

The documentation for this class was generated from the following files:

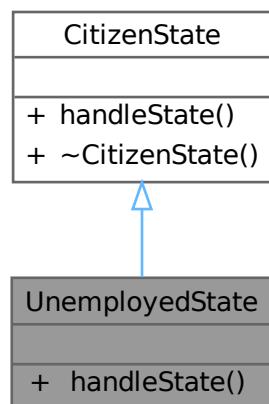
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseTaxi.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[TraverseTaxi.cpp](#)

4.87 UnemployedState Class Reference

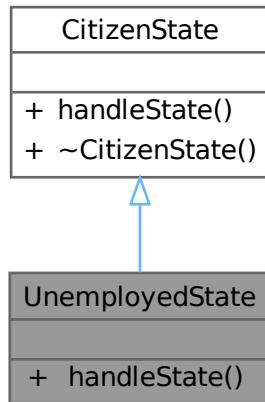
A class that represents the unemployed state of a citizen.

```
#include <UnemployedState.h>
```

Inheritance diagram for UnemployedState:



Collaboration diagram for UnemployedState:



Public Member Functions

- void `handleState` (`Citizen` &`citizen`) const override
Handles the state-specific behavior for an unemployed citizen.

Public Member Functions inherited from `CitizenState`

- virtual `~CitizenState` ()=default
Virtual destructor for the `CitizenState` class.

4.87.1 Detailed Description

A class that represents the unemployed state of a citizen.

The `UnemployedState` class is responsible for handling the behavior of a citizen when they are unemployed.

4.87.2 Member Function Documentation

4.87.2.1 `handleState()`

```
void UnemployedState::handleState (
    Citizen & citizen ) const [override], [virtual]
```

Handles the state-specific behavior for an unemployed citizen.

This function decreases the satisfaction level of a citizen due to unemployment. The satisfaction level is clamped between 0.0 and 100.0.

Parameters

<i>citizen</i>	The citizen whose state is being handled.
----------------	---

Implements [CitizenState](#).

The documentation for this class was generated from the following files:

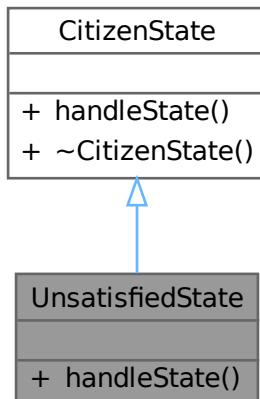
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UnemployedState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UnemployedState.cpp](#)

4.88 UnsatisfiedState Class Reference

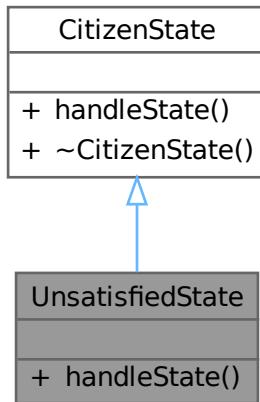
A class that represents the unsatisfied state of a citizen.

```
#include <UnsatisfiedState.h>
```

Inheritance diagram for UnsatisfiedState:



Collaboration diagram for UnsatisfiedState:



Public Member Functions

- void [handleState](#) ([Citizen](#) &citizen) const override
Handles the state-specific behavior for an unsatisfied citizen.

Public Member Functions inherited from [CitizenState](#)

- virtual ~[CitizenState](#) ()=default
Virtual destructor for the [CitizenState](#) class.

4.88.1 Detailed Description

A class that represents the unsatisfied state of a citizen.

The [UnsatisfiedState](#) class is responsible for handling the behavior of a citizen when they are unsatisfied.

4.88.2 Member Function Documentation

4.88.2.1 [handleState\(\)](#)

```
void UnsatisfiedState::handleState (
    Citizen & citizen ) const [override], [virtual]
```

Handles the state-specific behavior for an unsatisfied citizen.

This function decreases the satisfaction level of a citizen due to dissatisfaction. The satisfaction level is clamped between 0.0 and 100.0.

Parameters

<i>citizen</i>	The citizen whose state is being handled.
----------------	---

Implements [CitizenState](#).

The documentation for this class was generated from the following files:

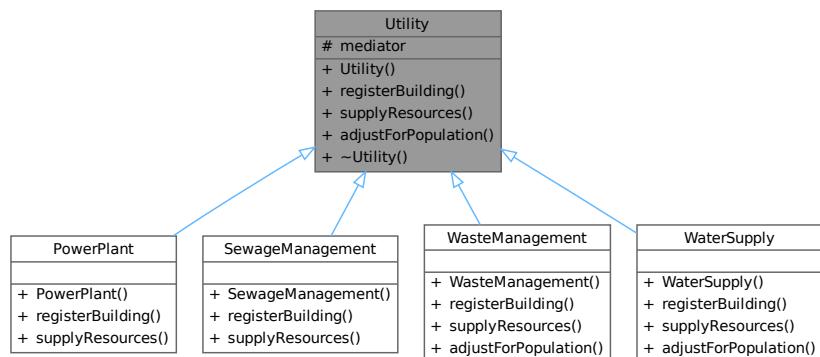
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UnsatisfiedState.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UnsatisfiedState.cpp](#)

4.89 Utility Class Reference

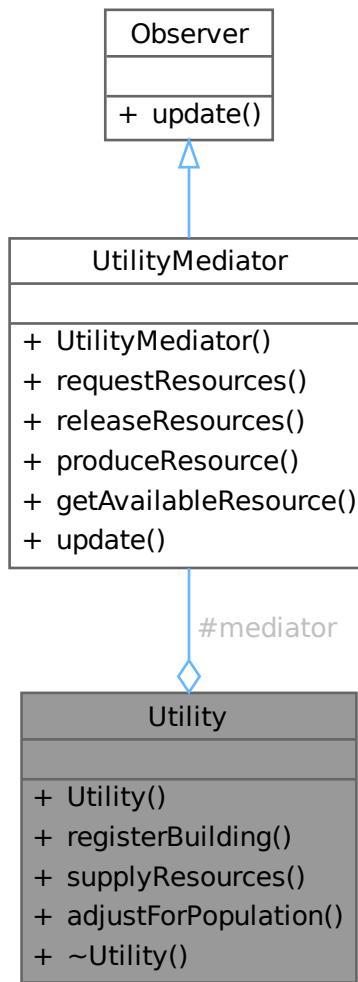
A class that represents a utility service in the city.

```
#include <Utility.h>
```

Inheritance diagram for Utility:



Collaboration diagram for Utility:



Public Member Functions

- [Utility \(UtilityMediator *mediator\)](#)
*Constructor for the **Utility** class.*
- [virtual void registerBuilding \(Building *building\)=0](#)
Registers a building with the utility service.
- [virtual void supplyResources \(Building *building\)=0](#)
Supplies resources to a building.
- [virtual void adjustForPopulation \(int newPopulation\)=0](#)
Adjusts the utility service for a new population size.
- [virtual ~Utility \(\)=default](#)
*Virtual destructor for the **Utility** class.*

Protected Attributes

- `UtilityMediator * mediator`

Reference to the mediator for managing resources.

4.89.1 Detailed Description

A class that represents a utility service in the city.

The `Utility` class is responsible for managing resources for buildings through a mediator.

4.89.2 Constructor & Destructor Documentation

4.89.2.1 Utility()

```
Utility::Utility (
```

<code>UtilityMediator * mediator</code>) [inline]
---	------------

Constructor for the `Utility` class.

Parameters

<code>mediator</code>	A pointer to the <code>UtilityMediator</code> for managing resources.
-----------------------	---

4.89.3 Member Function Documentation

4.89.3.1 adjustForPopulation()

```
virtual void Utility::adjustForPopulation (
```

<code>int newPopulation</code>) [pure virtual]
--------------------------------	------------------

Adjusts the utility service for a new population size.

Parameters

<code>newPopulation</code>	The new population size.
----------------------------	--------------------------

Implemented in `WasteManagement`, and `WaterSupply`.

4.89.3.2 registerBuilding()

```
virtual void Utility::registerBuilding (
```

<code>Building * building</code>) [pure virtual]
----------------------------------	------------------

Registers a building with the utility service.

Parameters

<i>building</i>	A pointer to the building to be registered.
-----------------	---

Implemented in [PowerPlant](#), [SewageManagement](#), [WasteManagement](#), and [WaterSupply](#).

4.89.3.3 supplyResources()

```
virtual void Utility::supplyResources (
    Building * building ) [pure virtual]
```

Supplies resources to a building.

Parameters

<i>building</i>	A pointer to the building to receive resources.
-----------------	---

Implemented in [PowerPlant](#), [SewageManagement](#), [WasteManagement](#), and [WaterSupply](#).

The documentation for this class was generated from the following file:

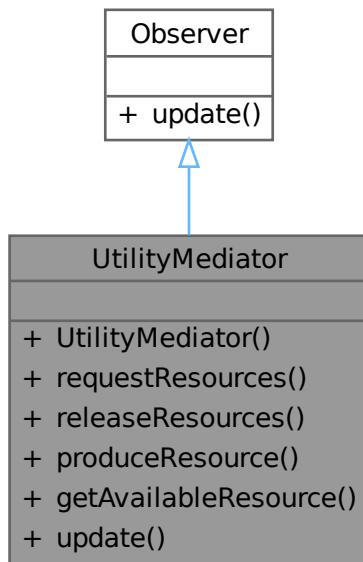
- /mnt/c/users/rudie/documents/sems/sem2/214/project/src/[Utility.h](#)

4.90 UtilityMediator Class Reference

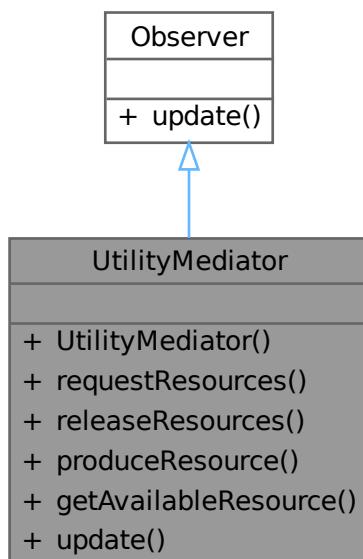
A class that manages resource distribution for utilities.

```
#include <UtilityMediator.h>
```

Inheritance diagram for UtilityMediator:



Collaboration diagram for UtilityMediator:



Public Member Functions

- **UtilityMediator ()=default**
Default constructor for the [UtilityMediator](#) class.
- **bool requestResources (ResourceType type, int amount)**
Requests resources from the mediator.
- **void releaseResources (ResourceType type, int amount)**
Releases resources back to the mediator.
- **void produceResource (ResourceType type, int amount)**
Produces resources for the mediator.
- **int getAvailableResource (ResourceType type) const**
Gets the available amount of a specific resource.
- **void update (ResourceType type, int quantity) override**
Updates the inventory with a specified quantity of a resource.

4.90.1 Detailed Description

A class that manages resource distribution for utilities.

The [UtilityMediator](#) class is responsible for managing the inventory and distribution of resources for various utilities.

4.90.2 Member Function Documentation

4.90.2.1 [getAvailableResource\(\)](#)

```
int UtilityMediator::getAvailableResource (
    ResourceType type ) const
```

Gets the available amount of a specific resource.

This function returns the amount of the specified resource available in the inventory.

Parameters

<i>type</i>	The type of resource being queried.
-------------	-------------------------------------

Returns

The amount of the specified resource available.

4.90.2.2 [produceResource\(\)](#)

```
void UtilityMediator::produceResource (
    ResourceType type,
    int amount )
```

Produces resources for the mediator.

This function adds the specified amount of resources to the inventory.

Parameters

<i>type</i>	The type of resource being produced.
<i>amount</i>	The amount of resource being produced.

4.90.2.3 releaseResources()

```
void UtilityMediator::releaseResources (
    ResourceType type,
    int amount )
```

Releases resources back to the mediator.

This function adds the specified amount of resources back to the inventory.

Parameters

<i>type</i>	The type of resource being released.
<i>amount</i>	The amount of resource being released.

4.90.2.4 requestResources()

```
bool UtilityMediator::requestResources (
    ResourceType type,
    int amount )
```

Requests resources from the mediator.

This function checks if the requested amount of resources is available and, if so, deducts it from the inventory.

Parameters

<i>type</i>	The type of resource being requested.
<i>amount</i>	The amount of resource being requested.

Returns

true if the request is fulfilled, false otherwise.

4.90.2.5 update()

```
void UtilityMediator::update (
    ResourceType type,
    int quantity ) [override], [virtual]
```

Updates the inventory with a specified quantity of a resource.

This function adds the specified quantity of a resource to the inventory.

Parameters

<i>type</i>	The type of resource being updated.
<i>quantity</i>	The quantity of the resource being added.

Implements [Observer](#).

The documentation for this class was generated from the following files:

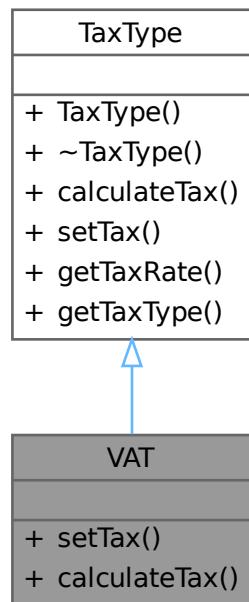
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UtilityMediator.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[UtilityMediator.cpp](#)

4.91 VAT Class Reference

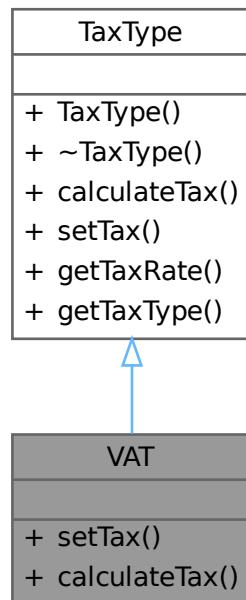
A class that represents Value Added Tax ([VAT](#)).

```
#include <VAT.h>
```

Inheritance diagram for VAT:



Collaboration diagram for VAT:



Public Member Functions

- void `setTax` (double rate)

Sets the VAT rate.

- void `calculateTax` ()

Calculates the VAT.

Public Member Functions inherited from [TaxType](#)

- `TaxType` (double rate, char type)

Constructs a new TaxType object.

- virtual `~TaxType` ()

Virtual Destructor.

- virtual double `calculateTax` (double val)

Calculates the tax based on a given value.

- virtual double `getTaxRate` ()

Gets the current tax rate.

- char `getTaxType` ()

Gets the tax type identifier.

4.91.1 Detailed Description

A class that represents Value Added Tax ([VAT](#)).

The [VAT](#) class is responsible for handling the calculation and setting of [VAT](#).

4.91.2 Member Function Documentation

4.91.2.1 calculateTax()

```
void VAT::calculateTax ( )
```

Calculates the [VAT](#).

This function calculates the [VAT](#) based on the current rate and applicable amount.

4.91.2.2 setTax()

```
void VAT::setTax (
    double rate ) [virtual]
```

Sets the [VAT](#) rate.

This function sets the [VAT](#) rate to the specified value.

Parameters

<i>rate</i>	The VAT rate to be set.
-------------	---

Reimplemented from [TaxType](#).

The documentation for this class was generated from the following files:

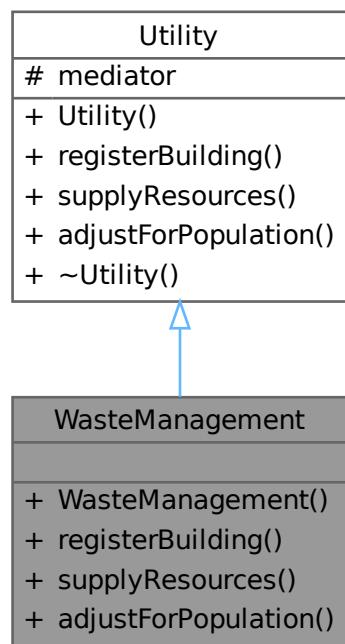
- /mnt/c/users/rudie/documents/sem2/214/project/src/[VAT.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[VAT.cpp](#)

4.92 WasteManagement Class Reference

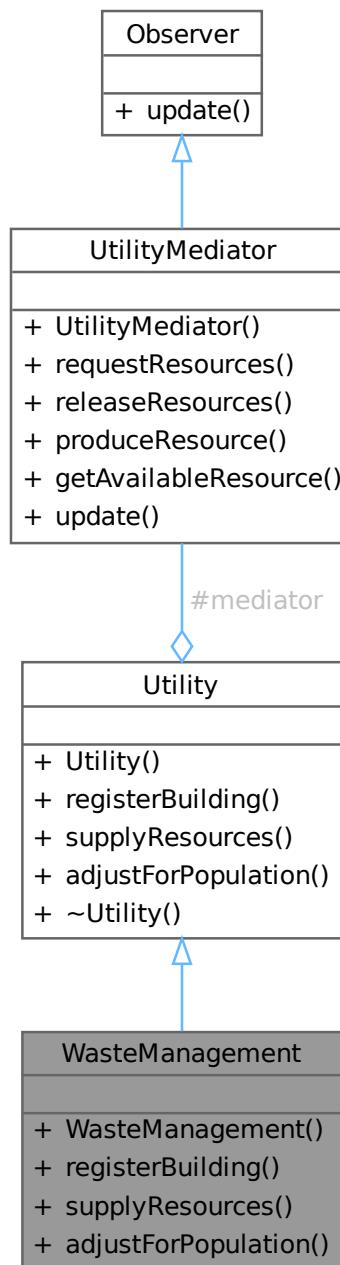
A class that represents waste management services in the city.

```
#include <WasteManagement.h>
```

Inheritance diagram for WasteManagement:



Collaboration diagram for WasteManagement:



Public Member Functions

- [WasteManagement \(UtilityMediator *mediator\)](#)
Constructor for the [WasteManagement](#) class.
- void [registerBuilding \(Building *building\) override](#)
Registers a building with the waste management service.
- void [supplyResources \(Building *building\) override](#)

Supplies waste management services to a building.

- void **adjustForPopulation** (int newPopulation) override

Adjusts waste management services based on the new population size.

Public Member Functions inherited from [Utility](#)

- **Utility** ([UtilityMediator](#) *mediator)

Constructor for the [Utility](#) class.

- virtual ~**Utility** ()=default

Virtual destructor for the [Utility](#) class.

Additional Inherited Members

Protected Attributes inherited from [Utility](#)

- [UtilityMediator](#) * **mediator**

Reference to the mediator for managing resources.

4.92.1 Detailed Description

A class that represents waste management services in the city.

The [WasteManagement](#) class is responsible for handling waste management services for buildings through a mediator.

4.92.2 Constructor & Destructor Documentation

4.92.2.1 [WasteManagement\(\)](#)

```
WasteManagement::WasteManagement (
    UtilityMediator * mediator )
```

Constructor for the [WasteManagement](#) class.

Parameters

<code>mediator</code>	A pointer to the UtilityMediator for managing resources.
-----------------------	--

4.92.3 Member Function Documentation

4.92.3.1 [adjustForPopulation\(\)](#)

```
void WasteManagement::adjustForPopulation (
    int newPopulation ) [override], [virtual]
```

Adjusts waste management services based on the new population size.

Parameters

<i>newPopulation</i>	The new population size.
----------------------	--------------------------

Implements [Utility](#).

4.92.3.2 registerBuilding()

```
void WasteManagement::registerBuilding (
    Building * building ) [override], [virtual]
```

Registers a building with the waste management service.

Parameters

<i>building</i>	A pointer to the building to be registered.
-----------------	---

Implements [Utility](#).

4.92.3.3 supplyResources()

```
void WasteManagement::supplyResources (
    Building * building ) [override], [virtual]
```

Supplies waste management services to a building.

Parameters

<i>building</i>	A pointer to the building to receive waste management services.
-----------------	---

Implements [Utility](#).

The documentation for this class was generated from the following files:

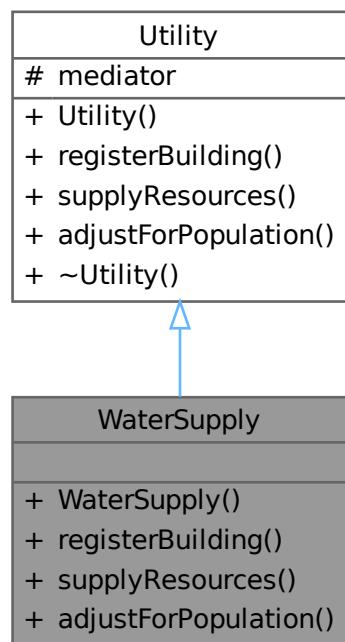
- /mnt/c/users/rudie/documents/sem2/214/project/src/[WasteManagement.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[WasteManagement.cpp](#)

4.93 WaterSupply Class Reference

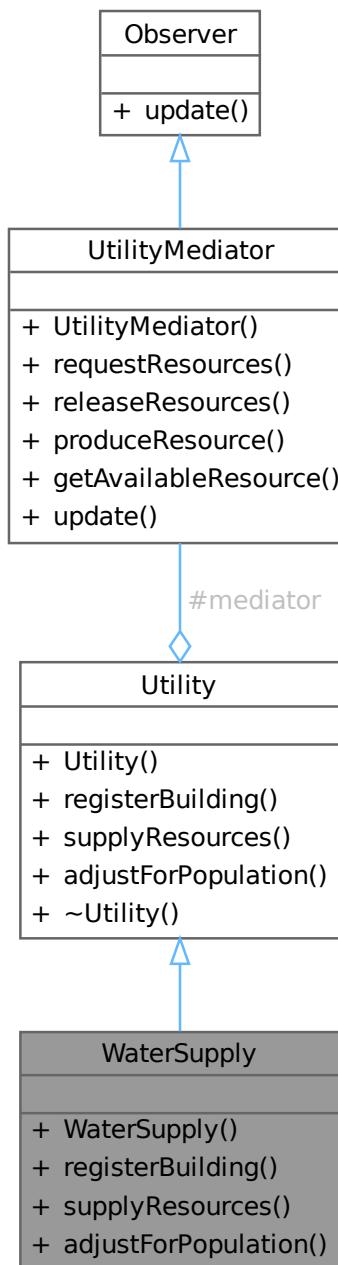
A class that represents water supply services in the city.

```
#include <WaterSupply.h>
```

Inheritance diagram for WaterSupply:



Collaboration diagram for WaterSupply:



Public Member Functions

- **WaterSupply (UtilityMediator *mediator)**
Constructor for the WaterSupply class.
- **void registerBuilding (Building *building) override**
Registers a building with the water supply service.
- **void supplyResources (Building *building) override**

Supplies water to a building.

- void [adjustForPopulation](#) (int newPopulation) override
Adjusts water supply services based on the new population size.

Public Member Functions inherited from [Utility](#)

- [Utility](#) ([UtilityMediator](#) *mediator)
Constructor for the [Utility](#) class.
- virtual ~[Utility](#) ()=default
Virtual destructor for the [Utility](#) class.

Additional Inherited Members

Protected Attributes inherited from [Utility](#)

- [UtilityMediator](#) * **mediator**
Reference to the mediator for managing resources.

4.93.1 Detailed Description

A class that represents water supply services in the city.

The [WaterSupply](#) class is responsible for handling water supply services for buildings through a mediator.

4.93.2 Constructor & Destructor Documentation

4.93.2.1 [WaterSupply\(\)](#)

```
WaterSupply::WaterSupply (
    UtilityMediator * mediator )
```

Constructor for the [WaterSupply](#) class.

Parameters

mediator	A pointer to the UtilityMediator for managing resources.
--------------------------	--

4.93.3 Member Function Documentation

4.93.3.1 [adjustForPopulation\(\)](#)

```
void WaterSupply::adjustForPopulation (
    int newPopulation ) [override], [virtual]
```

Adjusts water supply services based on the new population size.

Parameters

<i>newPopulation</i>	The new population size.
----------------------	--------------------------

Implements [Utility](#).

4.93.3.2 registerBuilding()

```
void WaterSupply::registerBuilding (
    Building * building ) [override], [virtual]
```

Registers a building with the water supply service.

Parameters

<i>building</i>	A pointer to the building to be registered.
-----------------	---

Implements [Utility](#).

4.93.3.3 supplyResources()

```
void WaterSupply::supplyResources (
    Building * building ) [override], [virtual]
```

Supplies water to a building.

Parameters

<i>building</i>	A pointer to the building to receive water.
-----------------	---

Implements [Utility](#).

The documentation for this class was generated from the following files:

- /mnt/c/users/rudie/documents/sem2/214/project/src/[WaterSupply.h](#)
- /mnt/c/users/rudie/documents/sem2/214/project/src/[WaterSupply.cpp](#)

Chapter 5

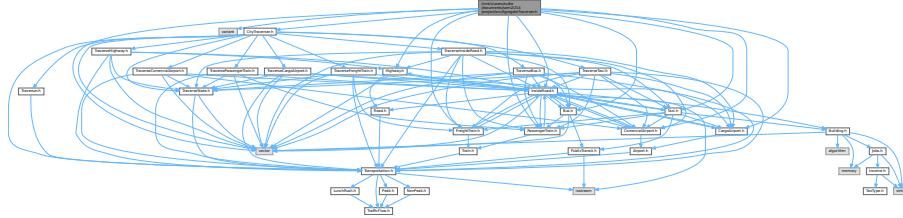
File Documentation

5.1 /mnt/c/users/rudie/documents/sem2/214/project/src/AggregateTraverser.h File Reference

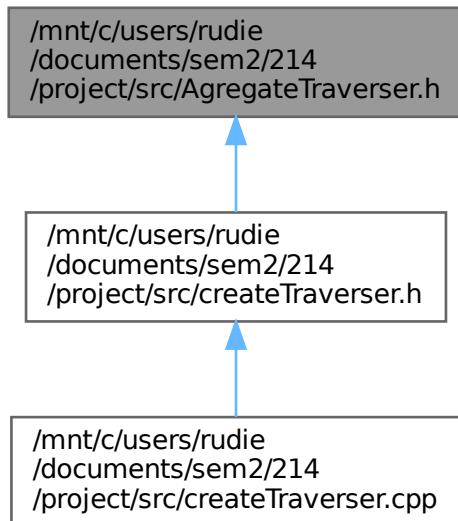
Defines the [AggregateTraverser](#) class and its interface for creating [CityTraverser](#) objects.

```
#include <vector>
#include <variant>
#include "Highway.h"
#include "InsideRoad.h"
#include "Bus.h"
#include "Taxi.h"
#include "FreightTrain.h"
#include "PassengerTrain.h"
#include "ComercialAirport.h"
#include "CargoAirport.h"
#include "CityTraverser.h"
```

Include dependency graph for AggregateTraverser.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [AggregateTraverser](#)
Abstract base class for creating [CityTraverser](#) objects.

5.1.1 Detailed Description

Defines the [AggregateTraverser](#) class and its interface for creating [CityTraverser](#) objects.

This file contains the declaration of the [AggregateTraverser](#) class, which provides an interface for creating [CityTraverser](#) objects. The [AggregateTraverser](#) class is an abstract base class with pure virtual functions that must be implemented by derived classes.

The file also includes necessary headers for various transportation types and the [CityTraverser](#) class.

Author

Samvit_Prakash_u23525119

Date

2024-11-04

5.2 AggregateTraverser.h

[Go to the documentation of this file.](#)

```

00001
00015 #ifndef AGREGATETRAVERSER_H
00016 #define AGREGATETRAVERSER_H
00017
00018 #include <vector>
00019 #include <variant>
00020
00021 #include "Highway.h"
00022 #include "InsideRoad.h"
00023 #include "Bus.h"
00024 #include "Taxi.h"
00025 #include "FreightTrain.h"
00026 #include "PassengerTrain.h"
00027 #include "ComercialAirport.h"
00028 #include "CargoAirport.h"
00029
00030 #include "CityTraverser.h"
00031
00040 class AggregateTraverser {
00041     public:
00046     virtual CityTraverser* createCityTraverser() = 0;
00047
00053     virtual CityTraverser* createCityTraverser(Transportation *t) = 0;
00054
00060     virtual CityTraverser* createCityTraverser(CityTraverser *t) = 0;
00061 };
00062
00063 #endif // AGREGATETRAVERSER_H

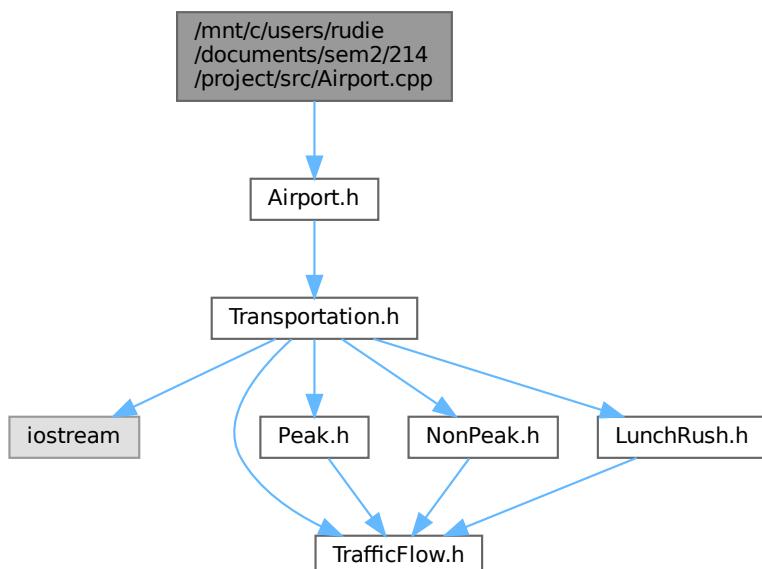
```

5.3 /mnt/c/users/rudie/documents/sem2/214/project/src/Airport.cpp File Reference

Implementation of the [Airport](#) class.

```
#include "Airport.h"
```

Include dependency graph for Airport.cpp:



5.3.1 Detailed Description

Implementation of the [Airport](#) class.

This file contains the implementation of the [Airport](#) class, which is a derived class from the [Transportation](#) class. It includes the constructor and the method to get the airport name.

Version

0.1

Date

2024-11-04

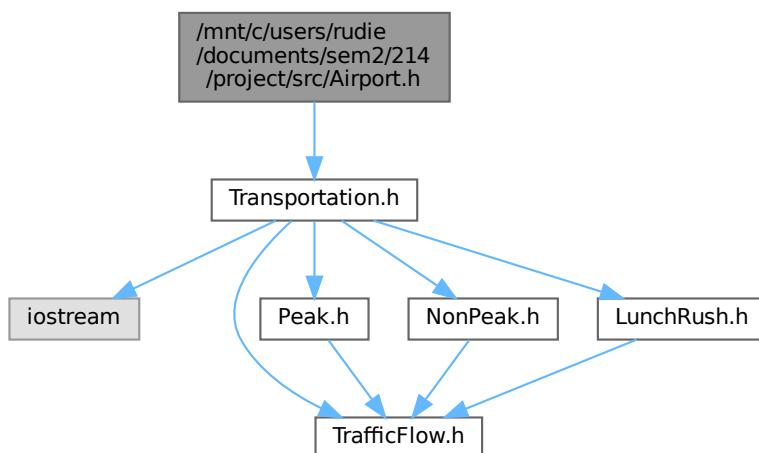
Note

This file is part of a project for managing different types of transportation.

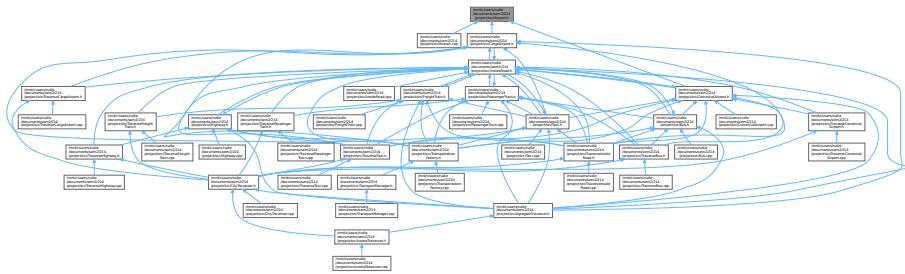
5.4 /mnt/c/users/rudie/documents/sem2/214/project/src/Airport.h File Reference

Defines the [Airport](#) class which inherits from the [Transportation](#) class.

```
#include "Transportation.h"  
Include dependency graph for Airport.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Airport](#)

Represents an airport as a type of transportation.

5.4.1 Detailed Description

Defines the [Airport](#) class which inherits from the [Transportation](#) class.

Author

Samvit_Prakash_u23525119

Date

2024-11-04

5.5 Airport.h

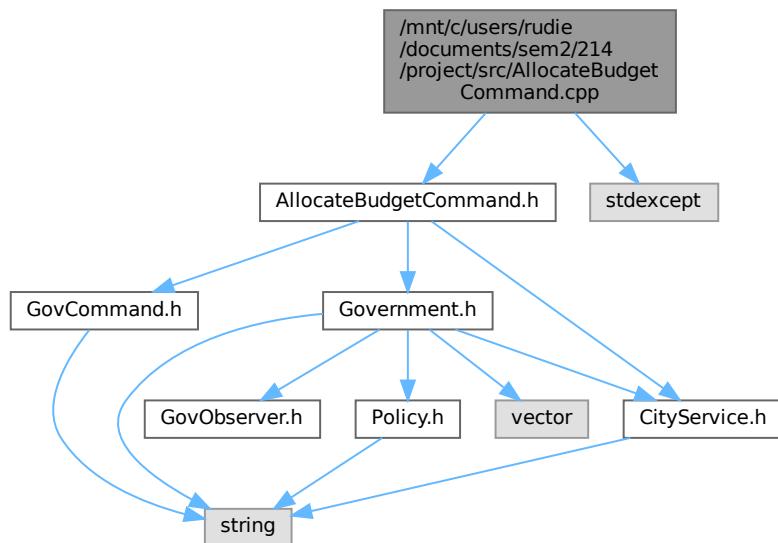
[Go to the documentation of this file.](#)

```
00001
00009 #ifndef AIRPORT_H
00010 #define AIRPORT_H
00011
00012 #include "Transportation.h"
00013
00018 class Airport : public Transportation {
00019     public:
00023         std::string name;
00024
00025     public:
00032         Airport(char state, std::string name, char type);
00033
00038         float calculateCommute();
00039
00044         std::string getAirportName();
00045     };
00046
00047 #endif
```

5.6 /mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.cpp File Reference

Implementation of the [AllocateBudgetCommand](#) class.

```
#include "AllocateBudgetCommand.h"
#include <stdexcept>
Include dependency graph for AllocateBudgetCommand.cpp:
```



5.6.1 Detailed Description

Implementation of the [AllocateBudgetCommand](#) class.

This file contains the implementation of the [AllocateBudgetCommand](#) class, which is used to allocate a budget to a city service. It includes methods to execute, undo, and validate the budget allocation, as well as getters and setters for the allocation amount and previous allocation amount.

Version

0.1

Date

2024-11-04

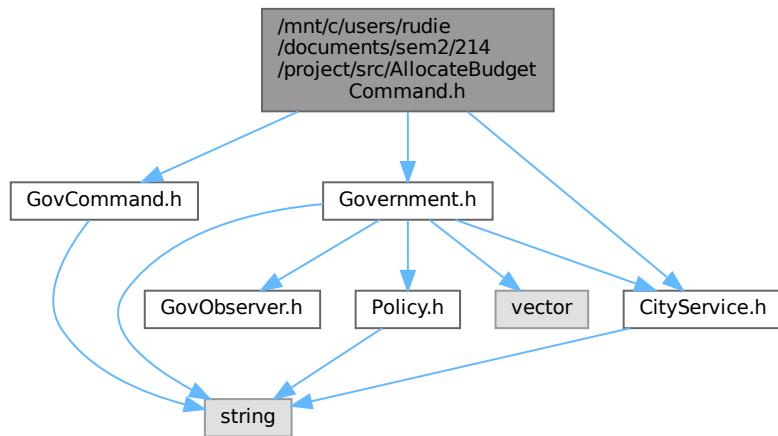
Note

This file is part of a project for managing city budgets and services.

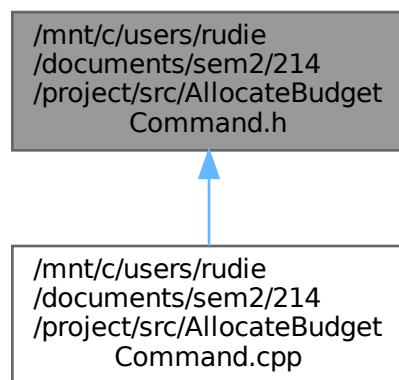
5.7 /mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.h File Reference

Definition of the [AllocateBudgetCommand](#) class.

```
#include "GovCommand.h"
#include "Government.h"
#include "CityService.h"
Include dependency graph for AllocateBudgetCommand.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [AllocateBudgetCommand](#)

Represents a command to allocate a budget to a city service.

5.7.1 Detailed Description

Definition of the [AllocateBudgetCommand](#) class.

This file contains the definition of the [AllocateBudgetCommand](#) class, which is used to allocate a budget to a city service. It includes methods to execute, undo, and validate the budget allocation, as well as getters and setters for the allocation amount and previous allocation amount.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city budgets and services.

5.8 AllocateBudgetCommand.h

[Go to the documentation of this file.](#)

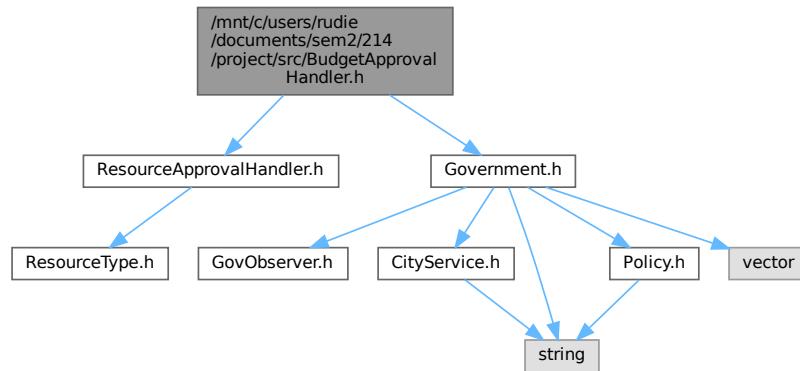
```
00001  
00016 #ifndef ALLOCATEBUDGETCOMMAND_H  
00017 #define ALLOCATEBUDGETCOMMAND_H  
00018  
00019 #include "GovCommand.h"  
00020 #include "Government.h"  
00021 #include "CityService.h"  
00022  
00031 class AllocateBudgetCommand : public GovCommand {  
00032  
00033 private:  
00034     Government* government;  
00035     CityService& service;  
00036     double amount;  
00037     double prevAllocation;  
00038  
00039 public:  
00047     AllocateBudgetCommand(Government* gov, CityService& srv, double amt);  
00048  
00054     void execute() override;  
00055  
00061     void undo() override;  
00062  
00068     double getAmount() const;  
00069  
00076     void setAmount(double amt);  
00077  
00083     double getPrevAllocation() const;  
00084  
00091     void setPrevAllocation(double prevAmt);  
00092  
00101     bool validateAllocation() const;  
00102  
00110     void executeWithValidation();  
00111  
00117     std::string getName() const override;  
00118  
00124     std::string getDescription() const override;  
00125  
00132     bool canExecute() const override;  
00133  
00139     double returnVal() override;  
00140 };  
00141  
00142 #endif // ALLOCATEBUDGETCOMMAND_H
```

5.9 /mnt/c/users/rudie/documents/sem2/214/project/src/BudgetApprovalHandler.h File Reference

Definition of the [BudgetApprovalHandler](#) class.

```
#include "ResourceApprovalHandler.h"
#include "Government.h"
```

Include dependency graph for BudgetApprovalHandler.h:



Classes

- class [BudgetApprovalHandler](#)
Handles budget approval requests.

5.9.1 Detailed Description

Definition of the [BudgetApprovalHandler](#) class.

This file contains the definition of the [BudgetApprovalHandler](#) class, which is responsible for handling budget approval requests. It includes methods to handle requests and allocate budget to city services.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city budgets and services.

5.10 BudgetApprovalHandler.h

[Go to the documentation of this file.](#)

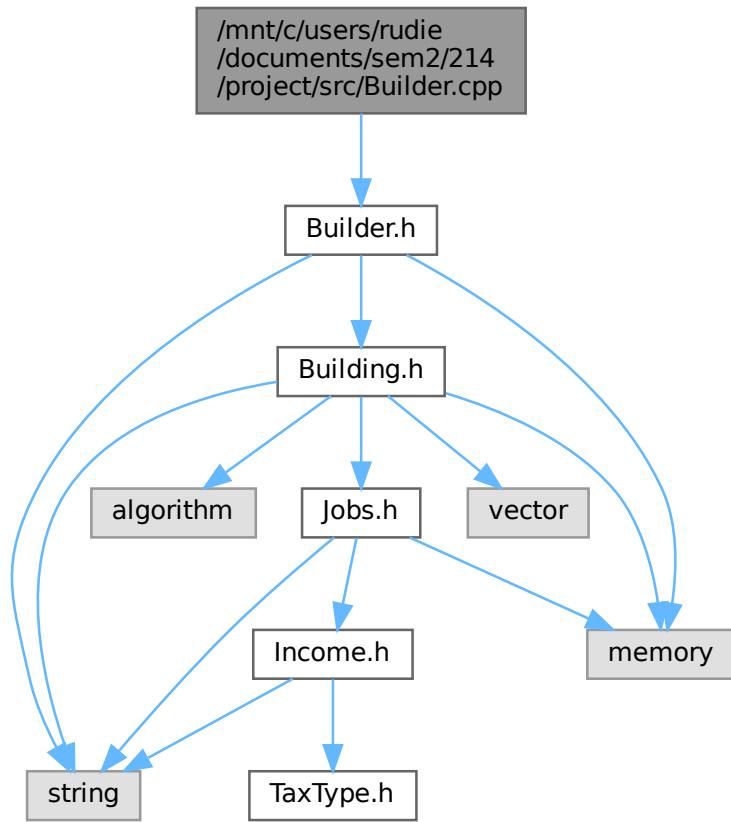
```
00001
00015 #ifndef BUDGETAPPROVALHANDLER_H
00016 #define BUDGETAPPROVALHANDLER_H
00017
00018 #include "ResourceApprovalHandler.h"
00019 #include "Government.h"
00020
00028 class BudgetApprovalHandler : public ResourceApprovalHandler {
00029 private:
00030     Government* government;
00031     int resourceCost;
00032
00033 public:
00040     BudgetApprovalHandler(Government* gov, int cost) : government(gov), resourceCost(cost) {}
00041
00052     bool handleRequest(ResourceType type, int quantity) override {
00053         double totalCost = resourceCost * quantity;
00054         CityService service("Resource Allocation", totalCost); // Placeholder CityService instance
00055
00056         if (government->getBudget() >= totalCost) {
00057             government->allocateBudget(service, totalCost);
00058             return ResourceApprovalHandler::handleRequest(type, quantity);
00059         }
00060         return false;
00061     }
00062 };
00063
00064 #endif // BUDGETAPPROVALHANDLER_H
```

5.11 /mnt/c/users/rudie/documents/sem2/214/project/src/Builder.cpp File Reference

Implementation of the [Builder](#) class.

```
#include "Builder.h"
```

Include dependency graph for Builder.cpp:



5.11.1 Detailed Description

Implementation of the [Builder](#) class.

This file contains the implementation of the [Builder](#) class, which is a base class for all builders. It includes methods to set various properties of a building such as name, area, floors, capacity, citizen satisfaction, economic growth, and resource consumption.

Version

0.1

Date

2024-11-04

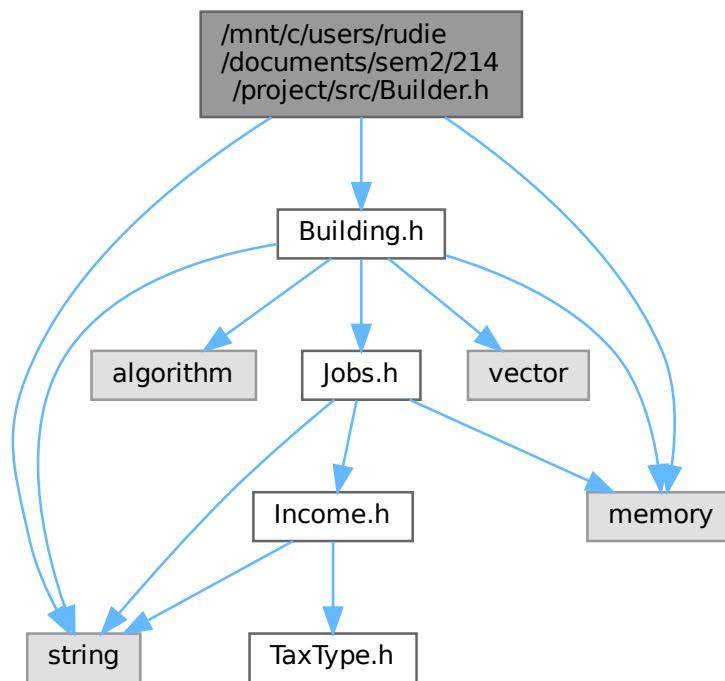
Note

This file is part of a project for managing city buildings and their properties.

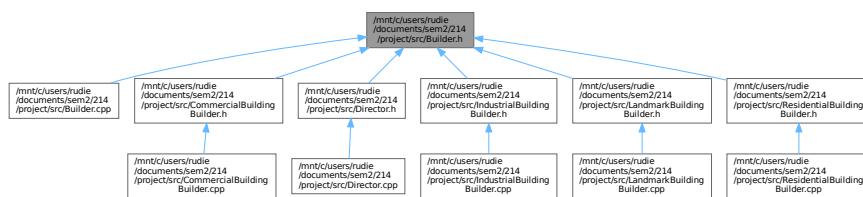
5.12 /mnt/c/users/rudie/documents/sem2/214/project/src/Builder.h File Reference

Definition of the [Builder](#) class.

```
#include <string>
#include "Building.h"
#include <memory>
Include dependency graph for Builder.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Builder](#)

Base class for all builders.

5.12.1 Detailed Description

Definition of the [Builder](#) class.

This file contains the definition of the [Builder](#) class, which is a base class for all builders. It includes methods to set various properties of a building such as name, area, floors, capacity, citizen satisfaction, economic growth, and resource consumption.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city buildings and their properties.

5.13 Builder.h

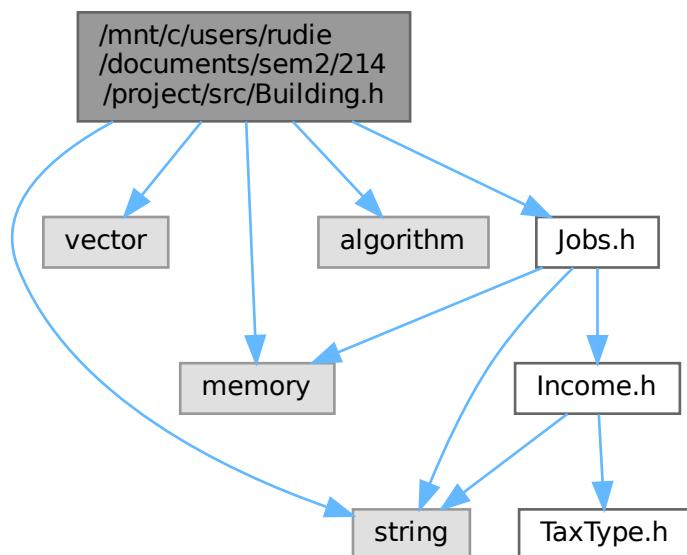
[Go to the documentation of this file.](#)

```
00001
00015 #ifndef BUILDER_H
00016 #define BUILDER_H
00017
00018 #include <string>
00019 #include "Building.h"
00020 #include <memory>
00021
00022 using namespace std;
00023
00031 class Builder {
00032
00033 protected:
00034     string name;
00035     float area = 0.0f;
00036     int floors = 0;
00037     int capacity = 0;
00038     float citizenSatisfaction = 0.0f;
00039     float economicGrowth = 0.0f;
00040     float resourceConsumption = 0.0f;
00041
00042 public:
00049     Builder& setName(string name);
00050
00057     Builder& setArea(float area);
00058
00065     Builder& setFloors(int floors);
00066
00073     Builder& setCapacity(int capacity);
00074
00081     Builder& setCitizenSatisfaction(float citizenSatisfaction);
00082
00089     Builder& setEconomicGrowth(float economicGrowth);
00090
00097     Builder& setResourceConsumption(float resourceConsumption);
00098
00104     virtual std::unique_ptr<Building> build() = 0;
00105
00109     virtual ~Builder() = default;
00110 };
00111
00112 #endif // BUILDER_H
```

5.14 /mnt/c/users/rudie/documents/sem2/214/project/src/Building.h File Reference

Definition of the [Building](#) class.

```
#include <string>
#include <vector>
#include <memory>
#include <algorithm>
#include "Jobs.h"
Include dependency graph for Building.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Building](#)
Represents a building with various properties and job management capabilities.

5.14.1 Detailed Description

Definition of the [Building](#) class.

This file contains the definition of the [Building](#) class, which represents a building with various properties such as name, area, floors, capacity, citizen satisfaction, economic growth, and resource consumption. It includes methods to manage jobs within the building.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city buildings and their properties.

5.15 Building.h

[Go to the documentation of this file.](#)

```
00001
00015 #ifndef BUILDING_H
00016 #define BUILDING_H
00017
00018 #include <string>
00019 #include <vector>
00020 #include <memory>
00021 #include <algorithm>
00022 #include "Jobs.h"
00023
00031 class Building {
00032 protected:
00033     std::string name;
00034     float area;
00035     int floors;
00036     int capacity;
00037     float citizenSatisfaction;
00038     float economicGrowth;
00039     float resourceConsumption;
00040     std::vector<std::shared_ptr<Jobs>> jobs;
00041
00042 public:
00043     Building(const std::string& name, float area, int floors, int capacity,
00044             float satisfactionImpact, float growthImpact, float consumption);
00045
00046     Building(int Builder);
00047
00048     void setName(const std::string& name);
00049
00050     std::string getName() const;
00051
00052     virtual void construct() = 0;
00053
00054     virtual ~Building() = default;
00055
00056     float getSatisfaction() const;
00057
00058     float getEconomicGrowth() const;
00059
00060     float getResourceConsumption() const;
00061
00062     virtual void updateImpacts() = 0;
00063
00064     virtual std::string getType() const = 0;
00065
00066     void addJob(std::shared_ptr<Jobs> job);
```

```

00127
00128     void listJobs() const;
00129
00130     bool hireEmployee(const std::string& jobTitle);
00131
00132     void releaseEmployee(const std::string& jobTitle);
00133
00134     std::shared_ptr<Jobs> getAvailableJob();
00135
00136     void displayJobInfo(const std::string& jobTitle) const;
00137
00138     const std::vector<std::shared_ptr<Jobs>> getJobs() const;
00139
00140     virtual double payTaxes(TaxType* taxType) = 0;
00141
00142     virtual void undoCollectTaxes() = 0;
00143
00144 private:
00145     virtual void calculateSatisfactionImpact() = 0;
00146
00147     virtual void calculateEconomicImpact() = 0;
00148
00149     virtual void calculateResourceConsumption() = 0;
00150
00151 };
00152
00153 #endif // BUILDING_H

```

5.16 /mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.cpp File Reference

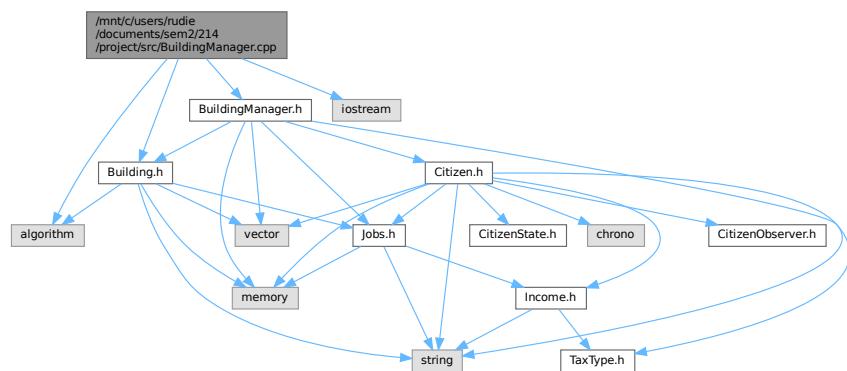
Implementation of the [BuildingManager](#) class.

```

#include "BuildingManager.h"
#include "Building.h"
#include <iostream>
#include <algorithm>

```

Include dependency graph for BuildingManager.cpp:



5.16.1 Detailed Description

Implementation of the [BuildingManager](#) class.

This file contains the implementation of the [BuildingManager](#) class, which manages buildings and citizens. It includes methods to add buildings and citizens, assign jobs to citizens, release citizens from jobs, list all jobs in a building, and find available jobs across all buildings.

Version

0.1

Date

2024-11-04

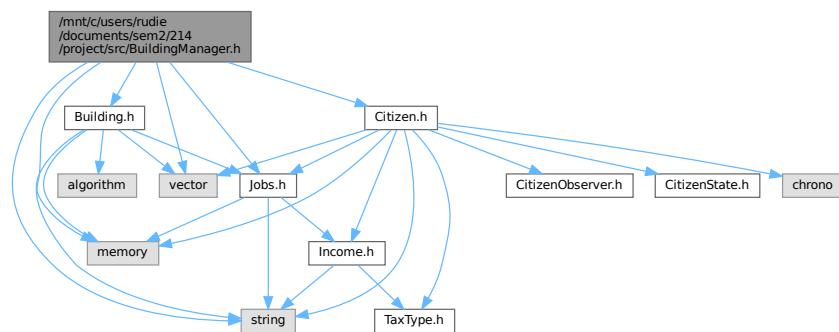
Note

This file is part of a project for managing city buildings and their properties.

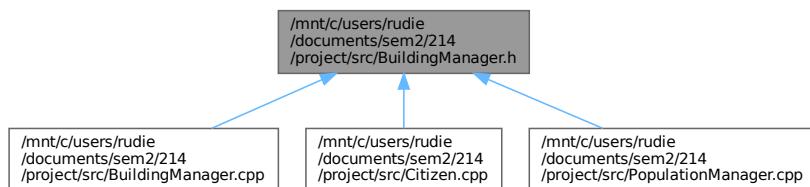
5.17 /mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.h File Reference

Definition of the [BuildingManager](#) class.

```
#include <vector>
#include <string>
#include "Building.h"
#include "Citizen.h"
#include "Jobs.h"
#include <memory>
Include dependency graph for BuildingManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [BuildingManager](#)
Manages buildings and citizens.

5.17.1 Detailed Description

Definition of the [BuildingManager](#) class.

This file contains the definition of the [BuildingManager](#) class, which manages buildings and citizens. It includes methods to add buildings and citizens, assign jobs to citizens, release citizens from jobs, list all jobs in a building, and find available jobs across all buildings.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city buildings and their properties.

5.18 BuildingManager.h

[Go to the documentation of this file.](#)

```

00001
00015 #ifndef BUILDINGMANAGER_H
00016 #define BUILDINGMANAGER_H
00017
00018 #include <vector>
00019 #include <string>
00020 #include "Building.h"
00021 #include "Citizen.h"
00022 #include "Jobs.h"
00023 #include <memory>
00024
00032 class BuildingManager {
00033 private:
00034     std::vector<std::shared_ptr<Building>> buildings;
00035     std::vector<Citizen*> citizens;
00036
00037 public:
00043     BuildingManager(const std::vector<std::shared_ptr<Building>>& buildingList);
00044
00050     void addBuilding(Building* building);
00051
00057     std::vector<std::shared_ptr<Building>> getBuildings();
00058
00064     void addCitizen(Citizen* citizen);
00065
00075     bool assignJobToCitizen(const std::string& jobTitle, Citizen* citizen, Building* building);
00076
00082     void releaseCitizenFromJob(Citizen* citizen);
00083
00089     void listAllJobsInBuilding(const Building* building) const;
00090
00096     std::shared_ptr<Jobs> findAvailableJob();
00097 };
00098
00099 #endif // BUILDINGMANAGER_H

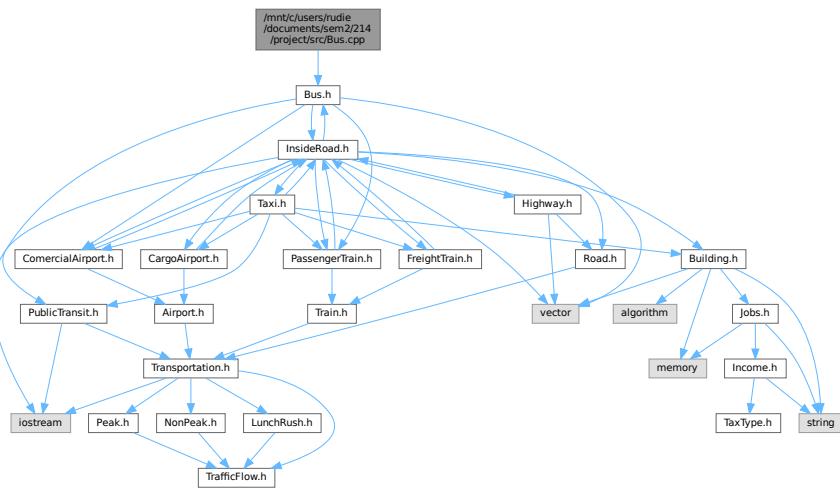
```

5.19 /mnt/c/users/rudie/documents/sem2/214/project/src/Bus.cpp File Reference

Implementation of the [Bus](#) class.

```
#include "Bus.h"
```

Include dependency graph for Bus.cpp:



5.19.1 Detailed Description

Implementation of the [Bus](#) class.

This file contains the implementation of the [Bus](#) class, which is a derived class from [PublicTransit](#). It includes methods to add and retrieve inside roads, buses, commercial airports, and passenger trains. It also includes methods to get the bus number, capacity, and route name.

Version

0.1

Date

2024-11-04

Note

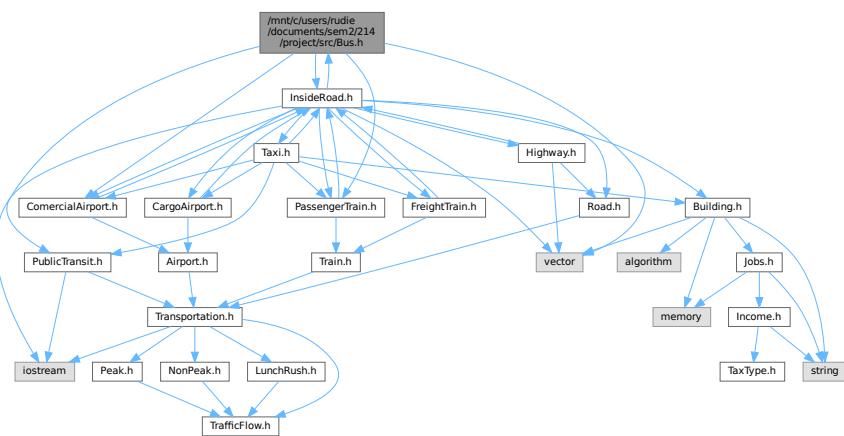
This file is part of a project for managing different types of public transportation.

5.20 /mnt/c/users/rudie/documents/sem2/214/project/src/Bus.h File Reference

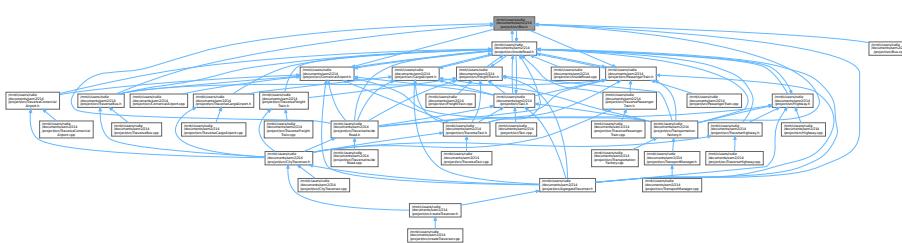
Header file for the [Bus](#) class.

```
#include <vector>
#include "PublicTransit.h"
#include "InsideRoad.h"
#include "ComercialAirport.h"
#include "PassengerTrain.h"
```

Include dependency graph for Bus.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Bus](#)
Represents a bus in the public transit system.

5.20.1 Detailed Description

Header file for the [Bus](#) class.

This file contains the declaration of the [Bus](#) class, which is a type of [PublicTransit](#). The [Bus](#) class manages information about buses, including their number, capacity, and connections to other transportation modes such as [InsideRoad](#), [ComercialAirport](#), and [PassengerTrain](#).

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing different types of public transportation.

5.21 Bus.h

[Go to the documentation of this file.](#)

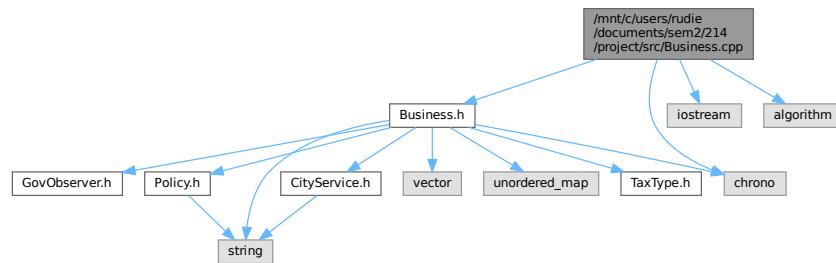
```
00001
00015 #ifndef BUS_H
00016 #define BUS_H
00017
00018 #include <vector>
00019
00020 #include "PublicTransit.h"
00021 #include "InsideRoad.h"
00022 #include "ComercialAirport.h"
00023 #include "PassengerTrain.h"
00024
00025 class InsideRoad;
00026 class ComercialAirport;
00027 class PassengerTrain;
00028
00029 class Bus : public PublicTransit {
00030     private:
00031         int busNumber;
00032         int capacity;
00033         std::vector<InsideRoad*> insideRoads;
00034         std::vector<Bus*> buses;
00035         std::vector<ComercialAirport*> comercialAirports;
00036         std::vector<PassengerTrain*> passengerTrains;
00037
00038     public:
00039         Bus(char state, std::string route, int busNumber, int capacity);
00040
00041         bool addInsideRoad(InsideRoad *insideRoad);
00042
00043         bool addBus(Bus *bus);
00044
00045         bool addComercialAirport(ComercialAirport *comercialAirport);
00046
00047         bool addPassengerTrain(PassengerTrain *passengerTrain);
00048
00049         InsideRoad *getInsideRoad(std::size_t x);
00050
00051         Bus *getBus(std::size_t x);
00052
00053         ComercialAirport *getComercialAirport(std::size_t x);
00054
00055         PassengerTrain *getPassengerTrain(std::size_t x);
00056
00057         int getBusNumber();
00058
00059         int getCapacity();
00060
00061         std::string getRouteName();
00062
00063     };
00064
00065 #endif // BUS_H
```

5.22 /mnt/c/users/rudie/documents/sem2/214/project/src/Business.cpp File Reference

Implementation of the [Business](#) class.

```
#include "Business.h"
#include <iostream>
#include <algorithm>
#include <chrono>
```

Include dependency graph for Business.cpp:



5.22.1 Detailed Description

Implementation of the [Business](#) class.

This file contains the implementation of the [Business](#) class, which manages business operations such as updating tax rates, policies, and services, processing tax payments, and printing business details.

Version

0.1

Date

2024-11-04

Note

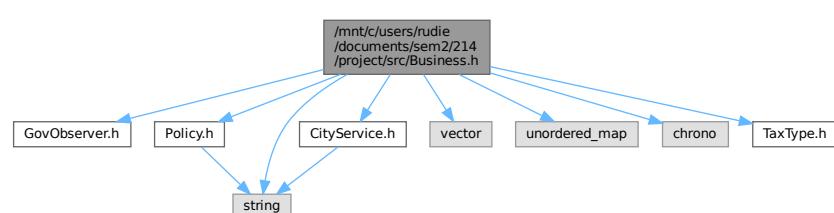
This file is part of a project for managing city businesses and their properties.

5.23 /mnt/c/users/rudie/documents/sem2/214/project/src/Business.h File Reference

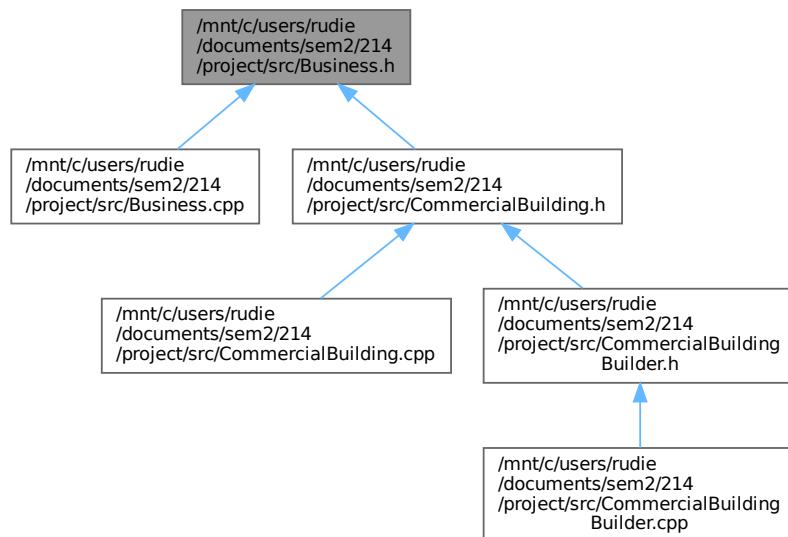
Header file for the [Business](#) class.

```
#include "GovObserver.h"
#include "Policy.h"
#include "CityService.h"
#include <vector>
#include <string>
#include <unordered_map>
#include <chrono>
#include "TaxType.h"
```

Include dependency graph for Business.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Business](#)

Represents a business that observes government policies and updates its state accordingly.

5.23.1 Detailed Description

Header file for the [Business](#) class.

This file contains the declaration of the [Business](#) class, which manages business operations such as updating tax rates, policies, and services, processing tax payments, and printing business details.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city businesses and their properties.

5.24 Business.h

[Go to the documentation of this file.](#)

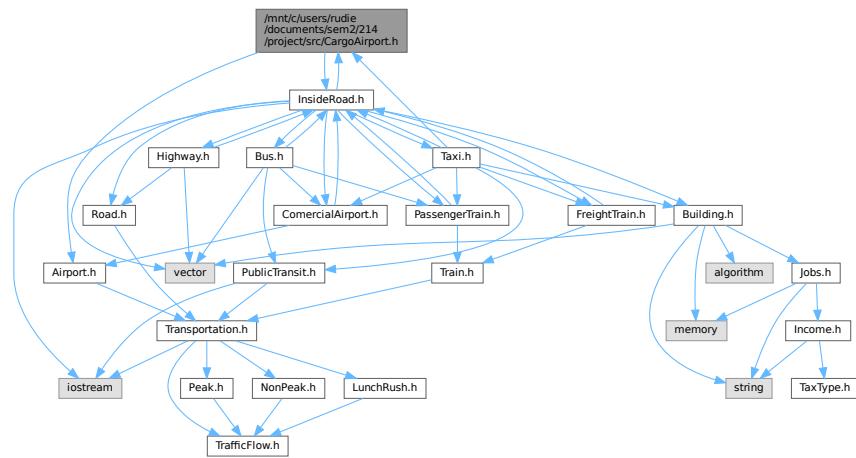
```
00001
00014 #ifndef BUSINESS_H
00015 #define BUSINESS_H
00016
00017 #include "GovObserver.h"
00018 #include "Policy.h"
00019 #include "CityService.h"
00020 #include <vector>
00021 #include <string>
00022 #include <unordered_map>
00023 #include <chrono>
00024 #include "TaxType.h"
00025
00033 class Business : public GovObserver {
00034
00035 private:
00036     double revenue;
00037     double taxRate;
00038     std::vector<std::string> services;
00039     std::vector<Policy> policies;
00040     std::unordered_map<char, std::chrono::steady_clock::time_point> lastTaxPayments;
00041     std::chrono::seconds taxCooldownPeriod;
00042
00043 public:
00050     Business(double initialRevenue, double initialTaxRate);
00051
00057     void updateTaxRate(double rate) override;
00058
00064     void updatePolicy(Policy policy) override;
00065
00071     void updateServices(CityService service) override;
00072
00078     void payTax(double amount);
00079
00085     void addService(const std::string& serviceName);
00086
00092     void removeService(const std::string& serviceName);
00093
00099     void addPolicy(const Policy& policy);
00100
00106     void removePolicy(const Policy& policy);
00107
00113     double calculateTax() const;
00114
00118     void printDetails() const;
00119
00126     double payTaxes(TaxType* taxType);
00127
00133     void setTaxCooldownPeriod(int seconds);
00134 };
00135
00136 #endif // BUSINESS_H
```

5.25 /mnt/c/users/rudie/documents/sem2/214/project/src/CargoAirport.h File Reference

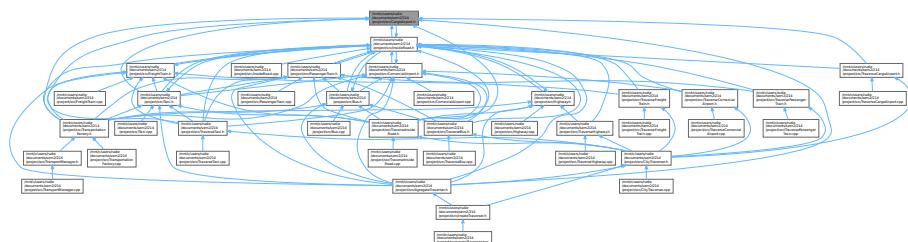
Header file for the [CargoAirport](#) class.

```
#include "Airport.h"
#include "InsideRoad.h"
```

Include dependency graph for CargoAirport.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CargoAirport](#)
Represents a cargo airport.

5.25.1 Detailed Description

Header file for the [CargoAirport](#) class.

This file contains the definition of the [CargoAirport](#) class, which inherits from the [Airport](#) class. The [CargoAirport](#) class manages a collection of [InsideRoad](#) and [CargoAirport](#) objects.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing different types of transportation.

5.26 CargoAirport.h

[Go to the documentation of this file.](#)

```

00014 #ifndef CARGOAIRPORT_H
00015 #define CARGOAIRPORT_H
00016
00017 #include "Airport.h"
00018 #include "InsideRoad.h"
00019
00020 class InsideRoad;
00021
00022 class CargoAirport : public Airport {
00023     private:
00024         std::vector<InsideRoad*> insideRoads;
00025         std::vector<CargoAirport*> cargoAirports;
00026
00027     public:
00028         CargoAirport(char state, std::string name);
00029
00030         bool addInsideRoad(InsideRoad *insideRoad);
00031
00032         bool addCargoAirport(CargoAirport *cargoAirport);
00033
00034         InsideRoad* getInsideRoad(std::size_t index);
00035
00036         CargoAirport* getCargoAirport(std::size_t index);
00037
00038         std::string getName();
00039     };
00040
00041
00042 #endif // CARGOAIRPORT_H

```

5.27 /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.cpp File Reference

Implementation of the [Citizen](#) class.

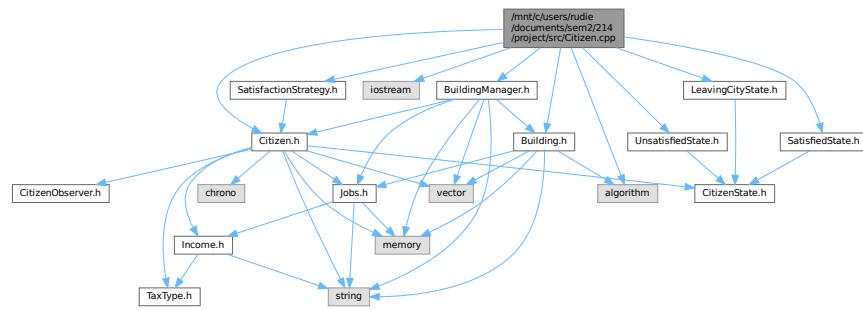
```

#include "Citizen.h"
#include <algorithm>
#include <iostream>
#include "BuildingManager.h"
#include "SatisfiedState.h"
#include "UnsatisfiedState.h"
#include "LeavingCityState.h"
#include "SatisfactionStrategy.h"

```

```
#include "Building.h"
```

Include dependency graph for Citizen.cpp:



Functions

- template<typename T >
T **minMax** (T value, T min, T max)
Utility function to constrain a value within a specified range.

5.27.1 Detailed Description

Implementation of the [Citizen](#) class.

This file contains the implementation of the [Citizen](#) class, which manages citizen attributes and behaviors such as satisfaction, tax payments, job search, and relationship status.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city citizens and their properties.

5.27.2 Function Documentation

5.27.2.1 **minMax()**

```
template<typename T >
T minMax (
    T value,
    T min,
    T max )
```

[Utility](#) function to constrain a value within a specified range.

Template Parameters

<i>T</i>	The type of the value.
----------	------------------------

Parameters

<i>value</i>	The value to constrain.
<i>min</i>	The minimum allowable value.
<i>max</i>	The maximum allowable value.

Returns

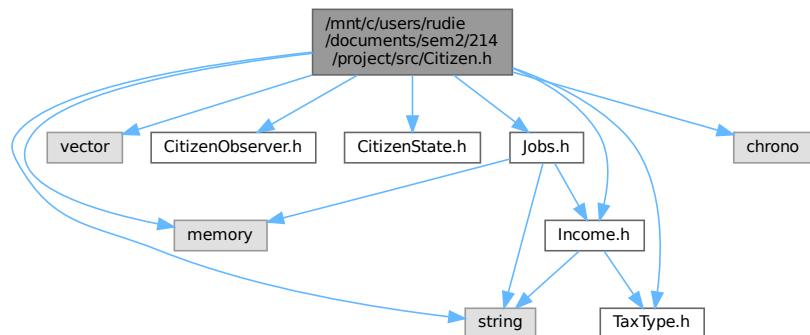
T The constrained value.

5.28 /mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.h File Reference

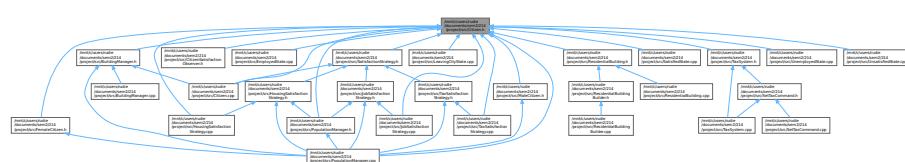
Header file for the [Citizen](#) class.

```
#include <string>
#include <memory>
#include <vector>
#include "CitizenObserver.h"
#include "CitizenState.h"
#include "Income.h"
#include "Jobs.h"
#include "TaxType.h"
#include <chrono>
```

Include dependency graph for Citizen.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Citizen](#)

Manages citizen attributes and behaviors.

5.28.1 Detailed Description

Header file for the [Citizen](#) class.

This file contains the declaration of the [Citizen](#) class, which manages citizen attributes and behaviors such as satisfaction, tax payments, job search, and relationship status.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city citizens and their properties.

5.29 Citizen.h

[Go to the documentation of this file.](#)

```
00001
00014 #ifndef CITIZEN_H
00015 #define CITIZEN_H
00016
00017 #include <string>
00018 #include <memory>
00019 #include <vector>
00020 #include "CitizenObserver.h"
00021 #include "CitizenState.h"
00022 #include "Income.h"
00023 #include "Jobs.h"
00024 #include "TaxType.h"
00025 #include <chrono>
00026
00027 class BuildingManager;
00028 class SatisfactionStrategy;
00029 class Building;
00030
00037 class Citizen {
00038 private:
00039     float taxRate;
00040     std::string name;
00041     int age;
00042     float satisfaction = 50.0f;
00043     bool maritalStatus = false;
00044     bool health = true;
00045     double bankBalance = 0.0;
00046     std::string relationshipStatus = "Single";
00047     int marriageDuration = 0;
00048     CitizenState* currentState = nullptr;
00049     std::vector<CitizenObserver*> observers;
00050     std::vector<std::shared_ptr<SatisfactionStrategy>> satisfactionStrategies;
00051     std::shared_ptr<Income> income;
00052     std::shared_ptr<Jobs> job;
00053     std::string jobTitle = "Unemployed";
00054     bool taxCooldown;
00055     std::chrono::steady_clock::time_point lastTaxPayment;
00056     static constexpr std::chrono::seconds taxCooldownPeriod{5};
```

```
00057 public:
00058     bool employed = false;
00060
00067     Citizen(const std::string& name, int age);
00068
00072     virtual ~Citizen();
00073
00079     virtual std::shared_ptr<Citizen> clone() const = 0;
00080
00081     // Observer pattern methods
00082     void addObserver(CitizenObserver* observer);
00083
00087     void detachAllObservers();
00088
00092     void removeObserver(CitizenObserver* observer);
00093
00099     void notifyObservers();
00100
00104     void updateSatisfaction();
00105
00106     // State and satisfaction updates
00107     void setState(CitizenState* newState);
00108
00112     void updateRelationshipStatus();
00113
00117     void incrementMarriageDuration();
00118
00119     void resetMarriageDuration();
00120
00124     void addSatisfactionStrategy(std::shared_ptr<SatisfactionStrategy> strategy);
00125
00129     void removeSatisfactionStrategy();
00130
00134     void setSatisfactionLevel(double level) { satisfaction = level; }
00135
00139     void depositMonthlyIncome();
00140
00144     void searchAndApplyForJob(BuildingManager& manager, Building* building, std::string jobtitle);
00145
00149     void checkAndUpdateState();
00150
00154     void setJobTitle(const std::string& jobTitle) { this->jobTitle = jobTitle; }
00155
00159     std::string getJob() const { return jobTitle; }
00160
00164     void setEmployed(bool status) { employed = status; }
00165
00169     bool isEmployed() const { return employed; }
00170
00174     float getTaxRate() const;
00175
00179     void setTaxRate(float rate);
00180
00184     void payTaxes(TaxType* taxType);
00185
00189     double getBankBalance() const;
00190
00194     void setBankBalance(double balance);
00195
00199     void increaseBankBalance(double amount);
```

```

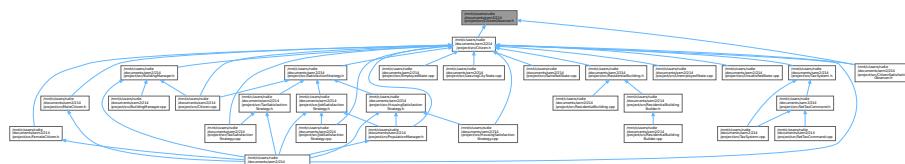
00318     void subtractBankBalance(double amount);
00324
00325
00326     // Tax cooldown management
00327     void setTaxCooldown(bool status);
00328
00329     bool getTaxCooldown() const;
00330
00331     bool isOnCooldown() const;
00332
00333     std::shared_ptr<Jobs> getJobObj() { return job; }
00334
00335     void setJob(std::shared_ptr<Jobs> job) { this->job = job; }
00336
00337     void unsetJob() { this->job = nullptr; }
00338 };
00339
00340 #endif // CITIZEN_H

```

5.30 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenObserver.h File Reference

Header file for the [CitizenObserver](#) class.

This graph shows which files directly or indirectly include this file:



Classes

- class [CitizenObserver](#)
Interface for observers of [Citizen](#) objects.

5.30.1 Detailed Description

Header file for the [CitizenObserver](#) class.

This file contains the declaration of the [CitizenObserver](#) class, which defines the interface for observers that want to be notified of changes to [Citizen](#) objects.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city citizens and their properties.

5.31 CitizenObserver.h

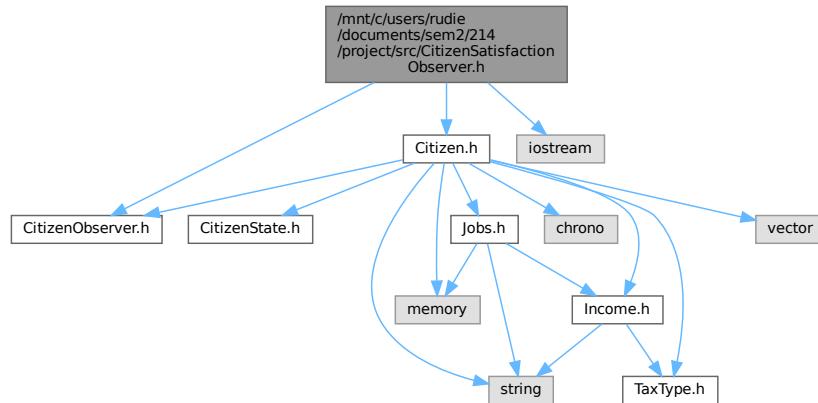
[Go to the documentation of this file.](#)

```
00001
00014 #ifndef CITIZEN_OBSERVER_H
00015 #define CITIZEN_OBSERVER_H
00016
00017 class Citizen; // Forward declaration
00018
00025 class CitizenObserver {
00026 public:
00030     virtual ~CitizenObserver() = default;
00031
00037     virtual void update(Citizen* citizen) = 0;
00038 };
00039
00040 #endif // CITIZEN_OBSERVER_H
```

5.32 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenSatisfactionObserver.h File Reference

Header file for the [CitizenSatisfactionObserver](#) class.

```
#include "CitizenObserver.h"
#include "Citizen.h"
#include <iostream>
Include dependency graph for CitizenSatisfactionObserver.h:
```



Classes

- class [CitizenSatisfactionObserver](#)

Observes changes in citizen satisfaction and updates their state accordingly.

5.32.1 Detailed Description

Header file for the [CitizenSatisfactionObserver](#) class.

This file contains the declaration of the [CitizenSatisfactionObserver](#) class, which observes changes in citizen satisfaction and updates their state accordingly.

Version

0.1

Date

2024-11-04

Note

This file is part of a project for managing city citizens and their properties.

5.33 CitizenSatisfactionObserver.h

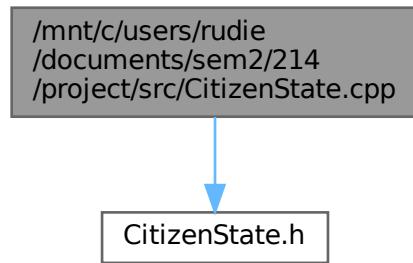
[Go to the documentation of this file.](#)

```
00001  
00014 #ifndef CITIZEN_SATISFACTION_OBSERVER_H  
00015 #define CITIZEN_SATISFACTION_OBSERVER_H  
00016  
00017 #include "CitizenObserver.h"  
00018 #include "Citizen.h"  
00019 #include <iostream>  
00020  
00028 class CitizenSatisfactionObserver : public CitizenObserver {  
00029 public:  
00035     void update(Citizen* citizen) override {  
00036         // Adjust satisfaction and update state based on conditions  
00037         citizen->updateSatisfaction(); // Update satisfaction  
00038         citizen->checkAndUpdateState(); // Check and update state if necessary  
00039  
00040         std::cout << citizen->getName() << "'s satisfaction updated to "  
00041             << citizen->getSatisfactionLevel() << "%, state updated.\n";  
00042     }  
00043 };  
00044  
00045 #endif // CITIZEN_SATISFACTION_OBSERVER_H
```

5.34 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.cpp File Reference

Implementation of the [CitizenState](#) class.

```
#include "CitizenState.h"
Include dependency graph for CitizenState.cpp:
```



5.34.1 Detailed Description

Implementation of the [CitizenState](#) class.

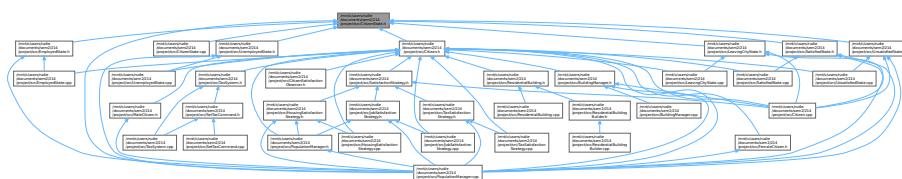
Date

2024-11-04 This file contains the implementation of the [CitizenState](#) class which handles the state of a citizen.

5.35 /mnt/c/users/rudie/documents/sem2/214/project/src/CitizenState.h File Reference

Declaration of the [CitizenState](#) class.

This graph shows which files directly or indirectly include this file:



Classes

- class [CitizenState](#)

Abstract class representing the state of a citizen.

5.35.1 Detailed Description

Declaration of the [CitizenState](#) class.

This file contains the declaration of the [CitizenState](#) class which handles the state of a citizen.

Date

2024-11-04

Version

1.0

5.36 CitizenState.h

[Go to the documentation of this file.](#)

```
00001
00010 #ifndef CITIZENSTATE_H
00011 #define CITIZENSTATE_H
00012
00013 class Citizen;
00014
00018 class CitizenState {
00019 public:
00027     virtual void handleState(Citizen& citizen) const = 0;
00028
00032     virtual ~CitizenState() = default;
00033 };
00034
00035 #endif // CITIZENSTATE_H
```

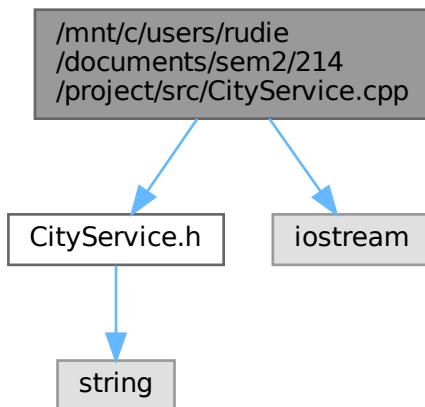
5.37 /mnt/c/users/rudie/documents/sem2/214/project/src/CityService.cpp File Reference

Implementation of the [CityService](#) class.

```
#include "CityService.h"
```

```
#include <iostream>
```

Include dependency graph for CityService.cpp:



5.37.1 Detailed Description

Implementation of the [CityService](#) class.

This file contains the implementation of the [CityService](#) class which manages city services.

Date

2024-11-04

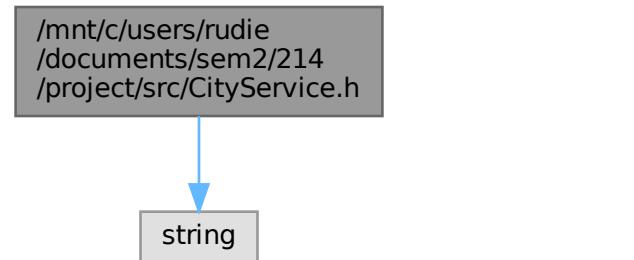
Version

1.0

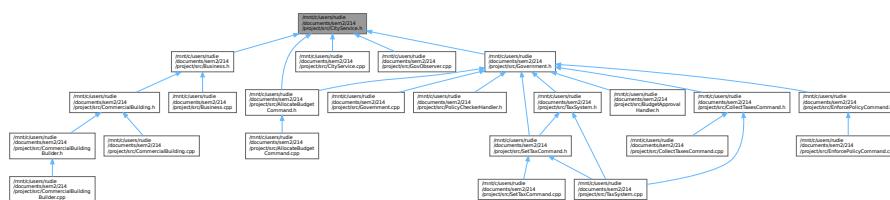
5.38 /mnt/c/users/rudie/documents/sem2/214/project/src/CityService.h File Reference

Declaration of the [CityService](#) class.

```
#include <string>
Include dependency graph for CityService.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CityService](#)

Class representing a city service.

5.38.1 Detailed Description

Declaration of the [CityService](#) class.

This file contains the declaration of the [CityService](#) class which manages city services.

Date

2024-11-04

Version

1.0

5.39 CityService.h

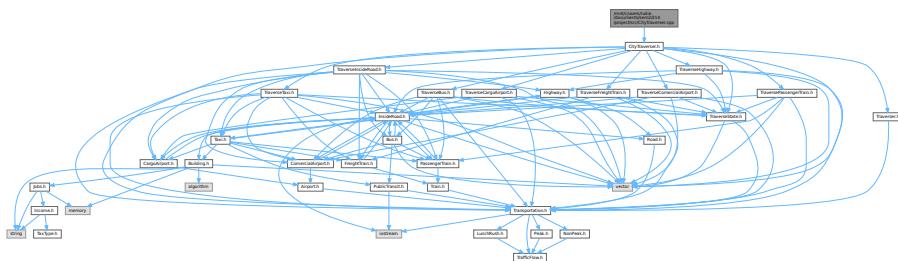
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef CITYSERVICE_H
00011 #define CITYSERVICE_H
00012
00013 #include <string>
00014
00018 class CityService {
00019
00020 private:
00021     std::string serviceName;
00022     double budgetAllocated;
00023
00024 public:
00030     CityService();
00031
00040     CityService(const std::string& name, double initialBudget);
00041
00049     void updateBudget(double amount);
00050
00056     void provideService();
00057
00063     double getBudgetAllocated() const;
00064
00070     void setServiceName(const std::string& name);
00071
00077     std::string getServiceName() const;
00078
00082     void printDetails() const;
00083
00090     bool isWithinBudget(double amount) const;
00091
00098     void allocateAdditionalBudget(double amount, double limit);
00099
00105     void reduceBudget(double amount);
00106 };
00107
00108 #endif // CITYSERVICE_H
```

5.40 /mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.cpp File Reference

Implementation of the [CityTraverser](#) class.

```
#include "CityTraverser.h"
Include dependency graph for CityTraverser.cpp:
```



5.40.1 Detailed Description

Implementation of the [CityTraverser](#) class.

This file contains the implementation of the [CityTraverser](#) class which traverses through different city transportation layers.

Date

2024-11-04

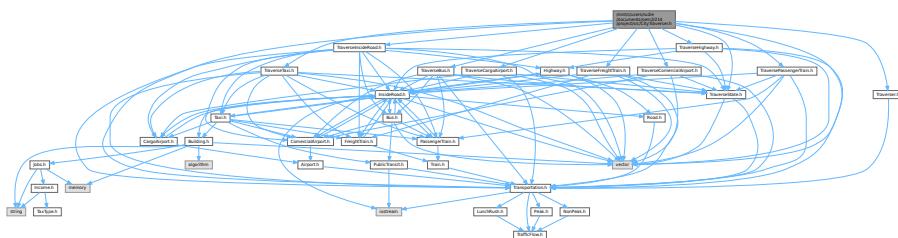
Version

1.0

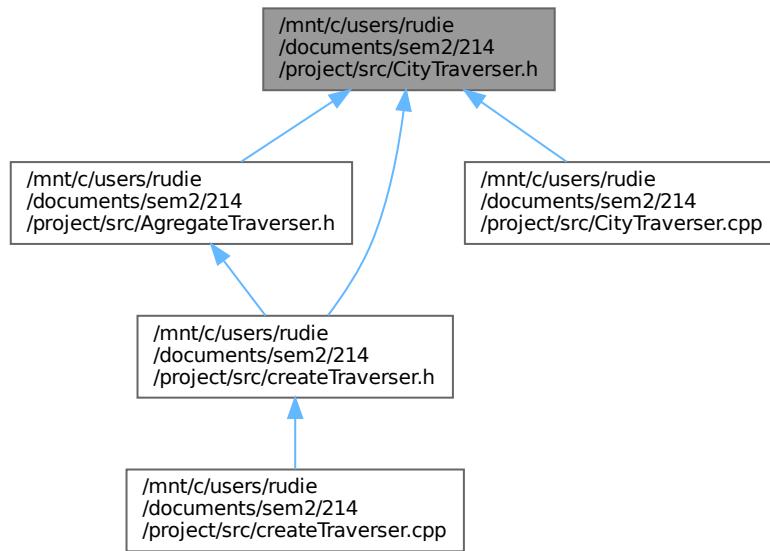
5.41 /mnt/c/users/rudie/documents/sem2/214/project/src/CityTraverser.h File Reference

Header file for the [CityTraverser](#) class, a concrete iterator for traversing transportation elements in a city.

```
#include <vector>
#include "Traverser.h"
#include "Transportation.h"
#include "TraverseState.h"
#include "TraverseHighway.h"
#include "TraverseInsideRoad.h"
#include "TraverseBus.h"
#include "TraverseTaxi.h"
#include "TraverseFreightTrain.h"
#include "TraversePassengerTrain.h"
#include "TraverseCargoAirport.h"
#include "TraverseComercialAirport.h"
Include dependency graph for CityTraverser.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CityTraverser](#)
A concrete iterator for traversing transportation elements in a city.

5.41.1 Detailed Description

Header file for the [CityTraverser](#) class, a concrete iterator for traversing transportation elements in a city.

This file contains the definition of the [CityTraverser](#) class, which is responsible for iterating over various transportation elements within a city. It provides methods for moving forward and backward through the elements, stepping into and out of layers, and accessing the current element.

Note

This class is part of the iterator pattern implementation.

Author

Samvit_Prakash_u23525119

Date

2024-11-04

5.42 CityTraverser.h

[Go to the documentation of this file.](#)

```

00001
00014 #ifndef CITYTRAVERSER_h
00015 #define CITYTRAVERSER_h
00016
00017 #include <vector>
00018
00019 #include "Traverser.h"
00020 #include "Transportation.h"
00021 #include "TraverseState.h"
00022 #include "TraverseHighway.h"
00023 #include "TraverseInsideRoad.h"
00024 #include "TraverseBus.h"
00025 #include "TraverseTaxi.h"
00026 #include "TraverseFreightTrain.h"
00027 #include "TraversePassengerTrain.h"
00028 #include "TraverseCargoAirport.h"
00029 #include "TraverseComercialAirport.h"
00030
00039 class CityTraverser: public Traverser {
00040     private:
00041         TraverseState *state;
00042         int currentElement;
00043         Transportation* Element;
00044         std::vector<Transportation*> prevLayers;
00045         Transportation* currentLayer;
00046         char dataType;
00047
00048     public:
00052         CityTraverser();
00053
00058         CityTraverser(Transportation* t);
00059
00064         CityTraverser(CityTraverser* t);
00065
00070         bool operator++();
00071
00076         bool operator--();
00077
00082         bool operator+( );
00083
00088         bool operator-( );
00089
00094         bool stepIn();
00095
00100         bool stepOut();
00101
00106         Transportation* operator*();
00107
00112         Transportation* getCurrentLayer();
00113
00119         bool operator=(Transportation* t);
00120
00126         bool operator=(CityTraverser* t);
00127
00132         bool setState();
00133
00137         ~CityTraverser();
00138 };
00139
00140 #endif

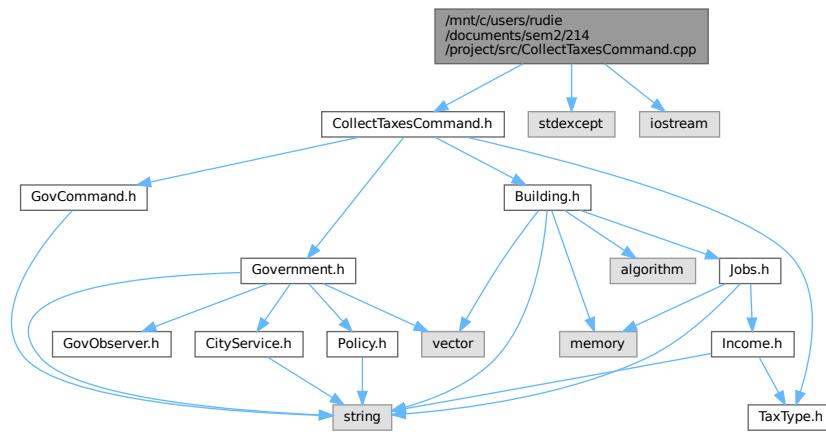
```

5.43 /mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.cpp File Reference

Implementation of the [CollectTaxesCommand](#) class.

```
#include "CollectTaxesCommand.h"
#include <stdexcept>
```

```
#include <iostream>
Include dependency graph for CollectTaxesCommand.cpp:
```



5.43.1 Detailed Description

Implementation of the [CollectTaxesCommand](#) class.

This file contains the implementation of the [CollectTaxesCommand](#) class which handles the collection of taxes.

Version

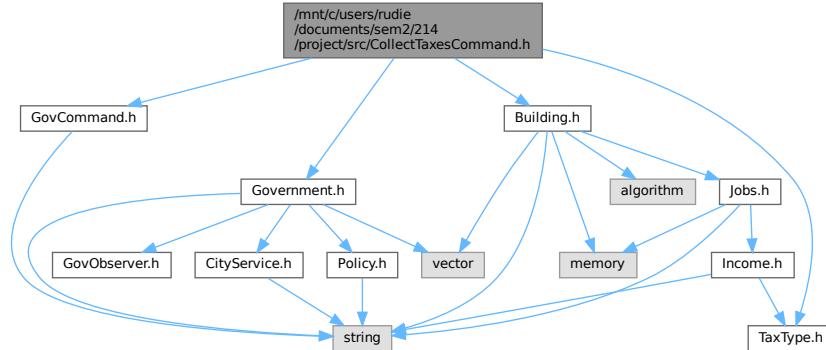
1.0

5.44 /mnt/c/users/rudie/documents/sem2/214/project/src/CollectTaxesCommand.h File Reference

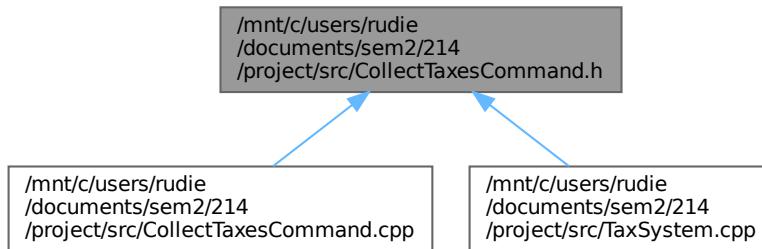
Declaration of the [CollectTaxesCommand](#) class.

```
#include "GovCommand.h"
#include "Government.h"
#include "Building.h"
#include "TaxType.h"
```

Include dependency graph for CollectTaxesCommand.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CollectTaxesCommand](#)
CollectTaxesCommand class.

5.44.1 Detailed Description

Declaration of the [CollectTaxesCommand](#) class.

This file contains the declaration of the [CollectTaxesCommand](#) class which handles the collection of taxes.

Version

1.0

5.45 CollectTaxesCommand.h

[Go to the documentation of this file.](#)

```

00001
00010 #ifndef COLLECTTAXESCOMMAND_H
00011 #define COLLECTTAXESCOMMAND_H
00012
00013 #include "GovCommand.h"
00014 #include "Government.h"
00015 #include "Building.h"
00016 #include "TaxType.h"
00017
00023 class CollectTaxesCommand : public GovCommand {
00024
00025 private:
00026     Government* government;
00027     double collectedTaxes;
00028
00029     std::shared_ptr<Building> building;
00030     double taxesCollected;
00031     double taxRate;
00032     TaxType* taxType;
00033
00034 public:
00044     CollectTaxesCommand(Government* gov, std::shared_ptr<Building> building, TaxType* taxType);
00045
00049     void execute() override;
00050
00054     void undo() override;
00055

```

```

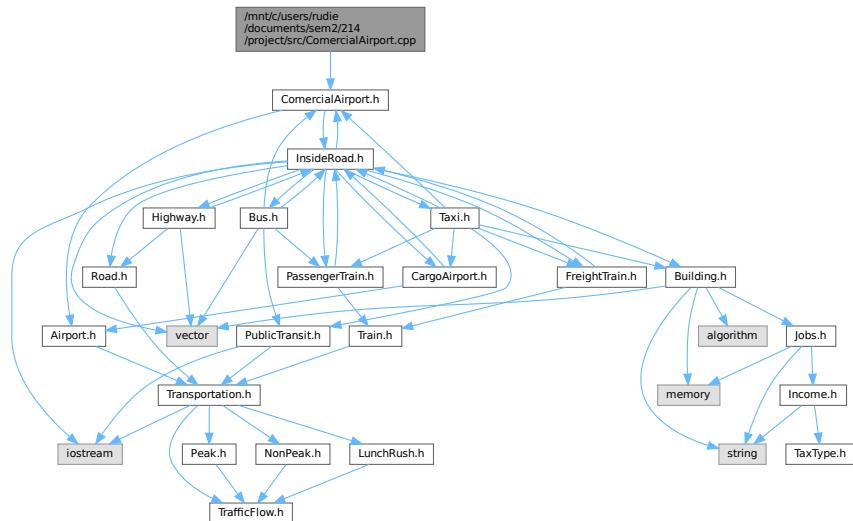
00061     double getCollectedTaxes() const;
00062
00069     void setCollectedTaxes(double amount);
00070
00078     bool validateCollection() const;
00079
00087     void executeWithValidation();
00088
00094     std::string getName() const override;
00095
00101     std::string getDescription() const override;
00102
00108     bool canExecute() const override;
00109
00115     double getTaxesCollected();
00116
00122     double returnVal();
00123 };
00124
00125 #endif // COLLECTTAXESCOMMAND_H

```

5.46 /mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.cpp File Reference

Implementation of the [ComercialAirport](#) class.

```
#include "ComercialAirport.h"
Include dependency graph for ComercialAirport.cpp:
```



5.46.1 Detailed Description

Implementation of the [ComercialAirport](#) class.

This file contains the implementation of the [ComercialAirport](#) class which manages commercial airports.

Version

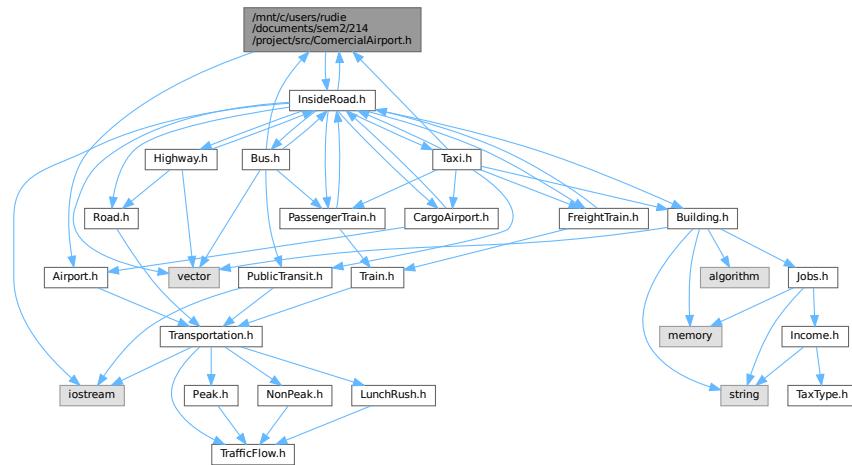
1.0

5.47 /mnt/c/users/rudie/documents/sem2/214/project/src/ComercialAirport.h File Reference

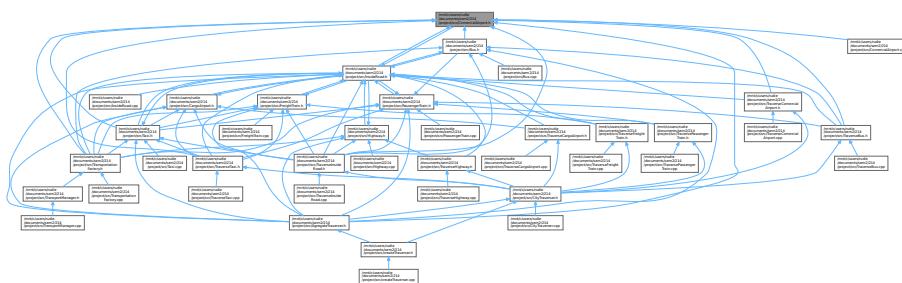
Defines the [ComercialAirport](#) class which inherits from the [Airport](#) class.

```
#include "Airport.h"
#include "InsideRoad.h"
```

Include dependency graph for ComercialAirport.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ComercialAirport](#)
- Represents a commercial airport.*

5.47.1 Detailed Description

Defines the [ComercialAirport](#) class which inherits from the [Airport](#) class.

This file contains the declaration of the [ComercialAirport](#) class, which represents a commercial airport. It includes methods to manage roads and other commercial airports associated with this airport.

Version

1.0

Date

04/10/2024

5.48 ComercialAirport.h

[Go to the documentation of this file.](#)

```

00001
00013 #ifndef COMERCIALAIRPORT_H
00014 #define COMERCIALAIRPORT_H
00015
00016 #include "Airport.h"
00017 #include "InsideRoad.h"
00018
00019 class InsideRoad;
00020
00029 class ComercialAirport : public Airport {
00030     private:
00031         std::vector<InsideRoad*> roads;
00032         std::vector<ComercialAirport*> comercialAirports;
00033
00034     public:
00041         ComercialAirport(char state, std::string name);
00042
00049         bool addRoad(InsideRoad *road);
00050
00057         bool addComercialAirport(ComercialAirport *comercialAirport);
00058
00065         InsideRoad* getRoad(std::size_t index);
00066
00073         ComercialAirport* getComercialAirport(std::size_t index);
00074
00080         std::string getName();
00081     };
00082
00083 #endif // COMERCIALAIRPORT_H

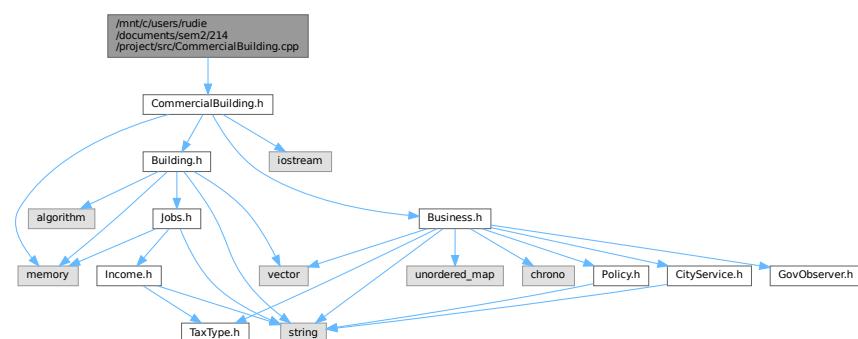
```

5.49 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.cpp File Reference

Implementation of the [CommercialBuilding](#) class.

```
#include "CommercialBuilding.h"
```

Include dependency graph for CommercialBuilding.cpp:



5.49.1 Detailed Description

Implementation of the [CommercialBuilding](#) class.

This file contains the implementation of the [CommercialBuilding](#) class which manages commercial buildings.

Date

2024-11-04

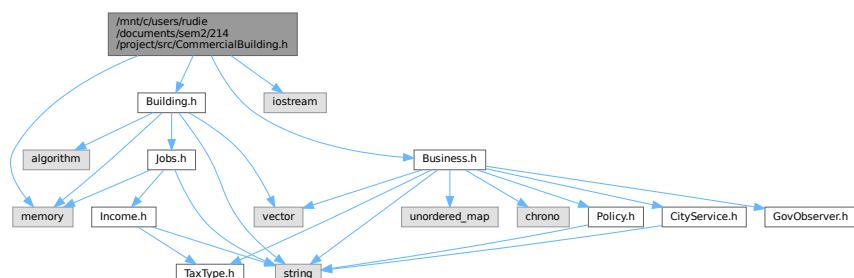
Version

1.0

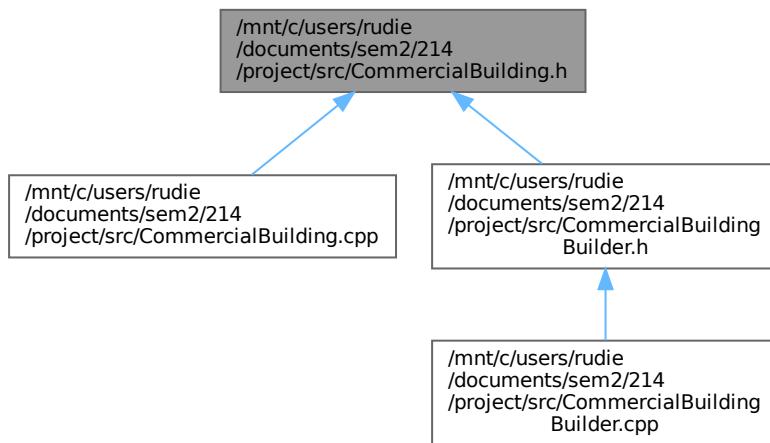
5.50 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuilding.h File Reference

Declaration of the [CommercialBuilding](#) class.

```
#include "Building.h"
#include "Business.h"
#include <memory>
#include <iostream>
Include dependency graph for CommercialBuilding.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CommercialBuilding](#)
Represents a commercial building.

5.50.1 Detailed Description

Declaration of the [CommercialBuilding](#) class.

This file contains the declaration of the [CommercialBuilding](#) class which manages commercial buildings.

Date

2024-11-04

Version

1.0

5.51 CommercialBuilding.h

[Go to the documentation of this file.](#)

```
00001  
00010 #ifndef COMMERCIALBUILDING_H  
00011 #define COMMERCIALBUILDING_H  
00012  
00013 #include "Building.h"  
00014 #include "Business.h"  
00015 #include <memory>  
00016 #include <iostream>  
00017  
00018 using namespace std;  
00019
```

```

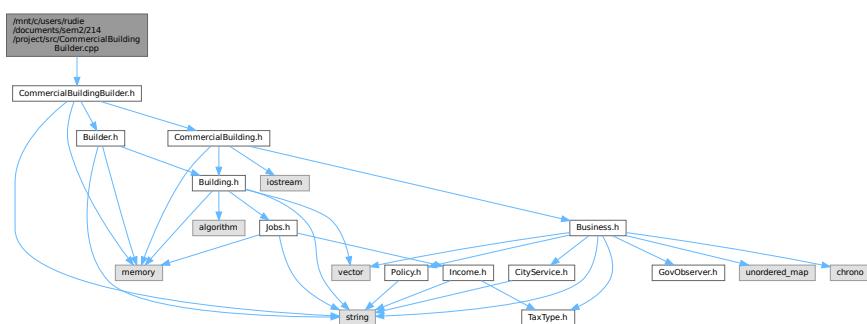
00027 class CommercialBuilding : public Building {
00028 public:
00044     CommercialBuilding(const std::string& name, float area, int floors, int capacity,
00045                         float citizenSatisfaction, float economicGrowth, float resourceConsumption,
00046                         int businessUnits, float customerTraffic);
00047
00051     ~CommercialBuilding(); // Destructor
00052
00058     std::string getType() const override;
00059
00065     void updateImpacts() override;
00066
00072     void updateCustomer(int traffic);
00073
00077     void construct() override;
00078
00085     double payTaxes(TaxType* taxType);
00086
00090     void undoCollectTaxes();
00091
00097     void setBusiness(Business* business);
00098
00099 private:
00100     int businessUnits;
00101     float customerTraffic;
00102     Business* business;
00103     std::string bType;
00104
00105 protected:
00109     void calculateEconomicImpact();
00110
00114     void calculateResourceConsumption();
00115
00119     void calculateSatisfactionImpact();
00120 };
00121
00122 #endif // COMMERCIALBUILDING_H

```

5.52 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.cpp File Reference

Implementation of the [CommercialBuildingBuilder](#) class.

```
#include "CommercialBuildingBuilder.h"
Include dependency graph for CommercialBuildingBuilder.cpp:
```



5.52.1 Detailed Description

Implementation of the [CommercialBuildingBuilder](#) class.

This file contains the implementation of the [CommercialBuildingBuilder](#) class which builds commercial buildings.

Date

2024-11-04

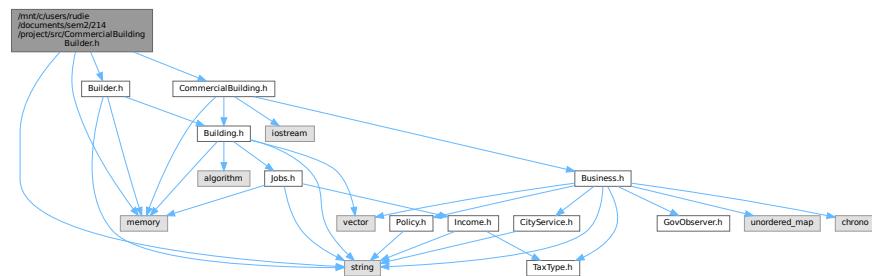
Version

1.0

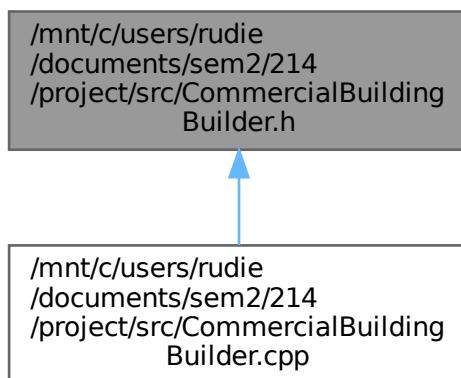
5.53 /mnt/c/users/rudie/documents/sem2/214/project/src/CommercialBuildingBuilder.h File Reference

Declaration of the [CommercialBuildingBuilder](#) class.

```
#include "Builder.h"
#include "CommercialBuilding.h"
#include <string>
#include <memory>
Include dependency graph for CommercialBuildingBuilder.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CommercialBuildingBuilder](#)

Builder class for constructing [CommercialBuilding](#) objects.

5.53.1 Detailed Description

Declaration of the [CommercialBuildingBuilder](#) class.

This file contains the declaration of the [CommercialBuildingBuilder](#) class which builds commercial buildings.

Date

2024-11-04

Version

1.0

5.54 CommercialBuildingBuilder.h

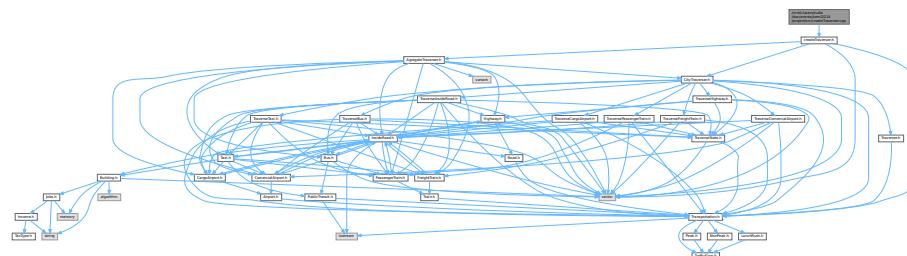
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef COMMERCIALBUILDINGBUILDER_H
00011 #define COMMERCIALBUILDINGBUILDER_H
00012
00013 #include "Builder.h"
00014 #include "CommercialBuilding.h"
00015 #include <string>
00016 #include <memory>
00017
00018 using namespace std;
00019
00020 class CommercialBuildingBuilder : public Builder {
00021
00022 private:
00023     int businessUnits = 0;
00024     float customerTraffic = 0.0f;
00025
00026 public:
00027     CommercialBuildingBuilder& setBusinessUnits(int units);
00028
00029     int getBusinessUnits();
00030
00031     float getCustomerTraffic();
00032
00033     CommercialBuildingBuilder& setCustomerTraffic(float customerTraffic);
00034
00035     CommercialBuildingBuilder();
00036
00037     std::unique_ptr<Building> build() override;
00038
00039 };
00040
00041 #endif // COMMERCIALBUILDINGBUILDER_H
```

5.55 /mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.cpp File Reference

Implementation of the [CreateTraverser](#) class for creating [CityTraverser](#) objects.

```
#include "createTraverser.h"
Include dependency graph for createTraverser.cpp:
```



5.55.1 Detailed Description

Implementation of the [CreateTraverser](#) class for creating [CityTraverser](#) objects.

This file contains the implementation of the [CreateTraverser](#) class, which provides methods to create instances of [CityTraverser](#) with different parameters.

Version

1.0

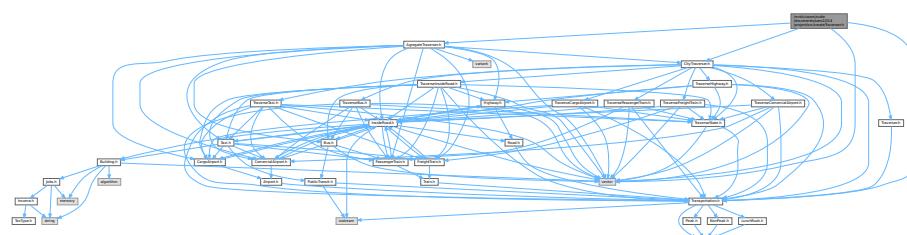
Date

2024-11-04

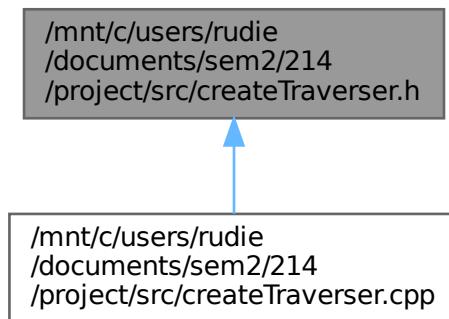
5.56 /mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.h File Reference

Header file for the [CreateTraverser](#) class.

```
#include <vector>
#include "AggregateTraverser.h"
#include "Transportation.h"
#include "CityTraverser.h"
Include dependency graph for createTraverser.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CreateTraverser](#)
A class to create [CityTraverser](#) instances.

5.56.1 Detailed Description

Header file for the [CreateTraverser](#) class.

This file contains the definition of the [CreateTraverser](#) class, which is a subclass of [AggregateTraverser](#). The [CreateTraverser](#) class provides methods to create instances of [CityTraverser](#).

Author

Samvit_Prakash_u23525119

Date

2024-11-04

5.57 createTraverser.h

[Go to the documentation of this file.](#)

```
00001  
00011 #ifndef CREATETRAVERSER_H  
00012 #define CREATETRAVERSER_H  
00013  
00014 #include <vector>  
00015 #include "AggregateTraverser.h"  
00016 #include "Transportation.h"  
00017 #include "CityTraverser.h"  
00018  
00019 class AggregateTraverser;  
00020 class Transportation;  
00021 class CityTraverser;
```

```

00022
00029 class CreateTraverser : public AggregateTraverser {
00030     public:
00035     CityTraverser* createCityTraverser();
00036
00042     CityTraverser* createCityTraverser(Transportation *t);
00043
00049     CityTraverser* createCityTraverser(CityTraverser *t);
00050 }
00051
00052 #endif // CREATETRAVERSER_H

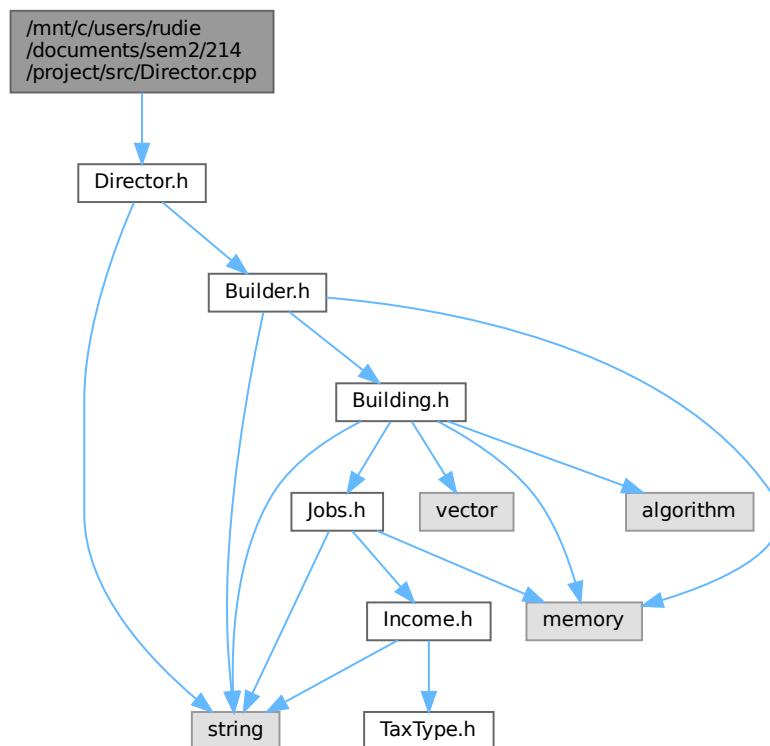
```

5.58 /mnt/c/users/rudie/documents/sem2/214/project/src/Director.cpp File Reference

Implementation of the [Director](#) class for constructing buildings.

```
#include "Director.h"
```

Include dependency graph for Director.cpp:



5.58.1 Detailed Description

Implementation of the [Director](#) class for constructing buildings.

This file contains the implementation of the [Director](#) class, which provides methods to construct buildings of different sizes using a [Builder](#) object.

Version

1.0

Date

2024-11-04

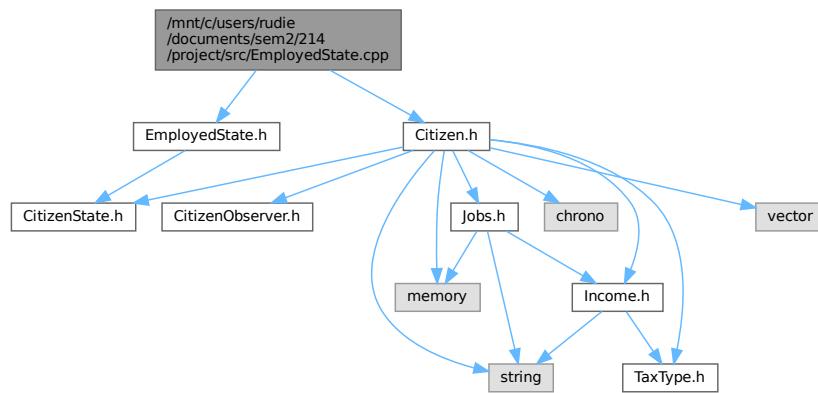
5.59 Director.h

```
00001 #ifndef DIRECTOR_H
00002 #define DIRECTOR_H
00003
00004 #include "Builder.h"
00005 #include <string>
00006 using namespace std;
00007
00017 class Director {
00018
00019 private:
00020     Builder* builder;
00022 public:
00030     void setBuilder(Builder& builder);
00031
00039     std::unique_ptr<Building> constructSmallBuilding();
00040
00048     std::unique_ptr<Building> constructMediumBuilding();
00049
00057     std::unique_ptr<Building> constructLargeBuilding();
00058 };
00059
00060 #endif // DIRECTOR_H
```

5.60 /mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.cpp File Reference

Implementation of the [EmployedState](#) class for handling the employed state of a [Citizen](#).

```
#include "EmployedState.h"
#include "Citizen.h"
Include dependency graph for EmployedState.cpp:
```



Functions

- double **clamp** (double value, double min, double max)

Clamps a value between a minimum and maximum.

5.60.1 Detailed Description

Implementation of the [EmployedState](#) class for handling the employed state of a [Citizen](#).

This file contains the implementation of the [EmployedState](#) class, which provides methods to handle the state of a [Citizen](#) when they are employed.

Version

1.0

Date

2024-11-04

5.60.2 Function Documentation

5.60.2.1 clamp()

```
double clamp (
    double value,
    double min,
    double max )
```

Clamps a value between a minimum and maximum.

This function ensures that the value stays within the specified range.

Parameters

<i>value</i>	The value to be clamped.
<i>min</i>	The minimum value.
<i>max</i>	The maximum value.

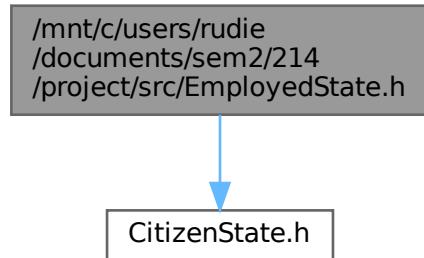
Returns

double The clamped value.

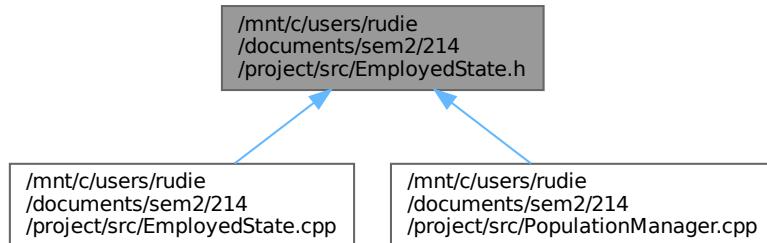
5.61 /mnt/c/users/rudie/documents/sem2/214/project/src/EmployedState.h File Reference

Declaration of the [EmployedState](#) class for handling the employed state of a [Citizen](#).

```
#include "CitizenState.h"
Include dependency graph for EmployedState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [EmployedState](#)
The [EmployedState](#) class for handling the employed state of a [Citizen](#).

5.61.1 Detailed Description

Declaration of the [EmployedState](#) class for handling the employed state of a [Citizen](#).

This file contains the declaration of the [EmployedState](#) class, which provides methods to handle the state of a [Citizen](#) when they are employed.

Version

1.0

Date

2024-11-04

5.62 EmployedState.h

[Go to the documentation of this file.](#)

```

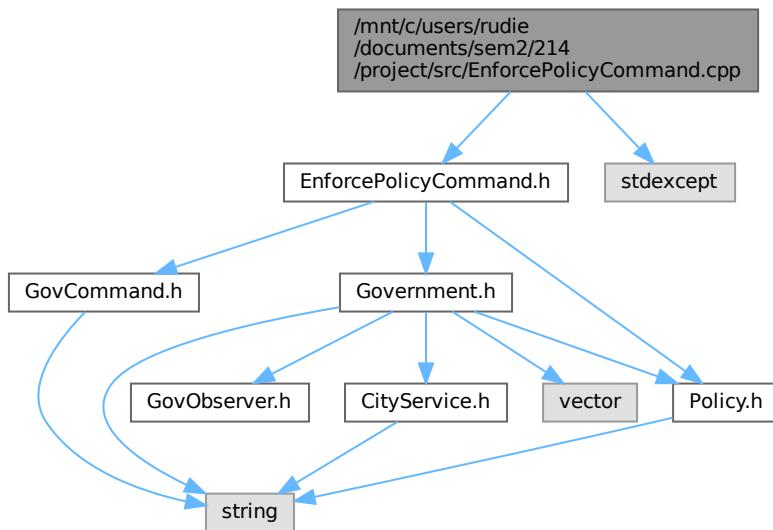
00001
00012 #ifndef EMPLOYEDSTATE_H
00013 #define EMPLOYEDSTATE_H
00014
00015 #include "CitizenState.h"
00016
00023 class EmployedState : public CitizenState {
00024 public:
00032     void handleState(Citizen& citizen) const override;
00033 };
00034
00035 #endif // EMPLOYEDSTATE_H

```

5.63 /mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.cpp File Reference

Implementation of the [EnforcePolicyCommand](#) class for enforcing policies in the government.

```
#include "EnforcePolicyCommand.h"
#include <stdexcept>
Include dependency graph for EnforcePolicyCommand.cpp:
```



5.63.1 Detailed Description

Implementation of the [EnforcePolicyCommand](#) class for enforcing policies in the government.

This file contains the implementation of the [EnforcePolicyCommand](#) class, which provides methods to enforce, undo, and validate policies in the government.

Version

1.0

Date

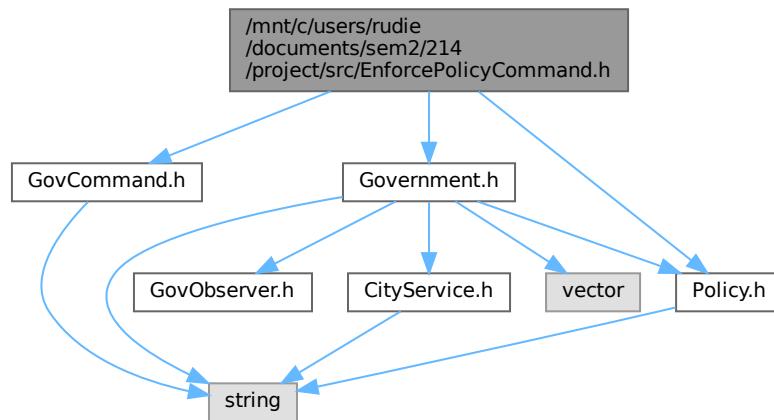
2024-11-04

5.64 /mnt/c/users/rudie/documents/sem2/214/project/src/EnforcePolicyCommand.h File Reference

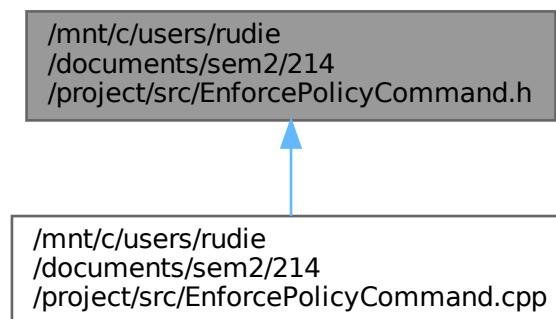
Declaration of the [EnforcePolicyCommand](#) class for enforcing policies in the government.

```
#include "GovCommand.h"
#include "Government.h"
#include "Policy.h"
```

Include dependency graph for EnforcePolicyCommand.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [EnforcePolicyCommand](#)

The [EnforcePolicyCommand](#) class for enforcing policies in the government.

5.64.1 Detailed Description

Declaration of the [EnforcePolicyCommand](#) class for enforcing policies in the government.

This file contains the declaration of the [EnforcePolicyCommand](#) class, which provides methods to enforce, undo, and validate policies in the government.

Version

1.0

Date

2024-11-04

5.65 EnforcePolicyCommand.h

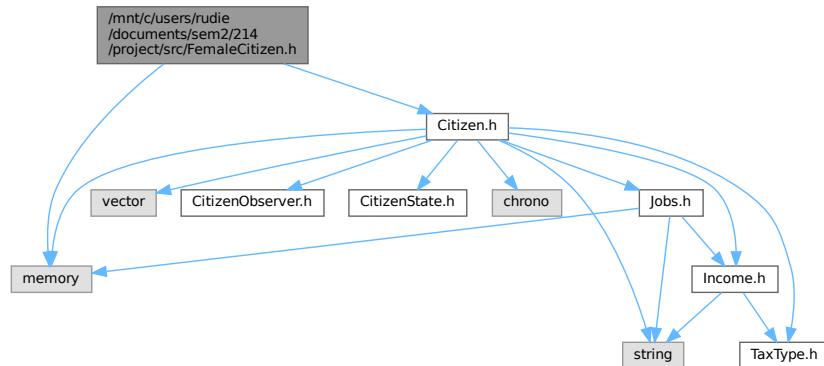
[Go to the documentation of this file.](#)

```
00001 #ifndef ENFORCEPOLICYCOMMAND_H
00002 #define ENFORCEPOLICYCOMMAND_H
00003
00004 #include "GovCommand.h"
00005 #include "Government.h"
00006 #include "Policy.h"
00007
00025 class EnforcePolicyCommand : public GovCommand {
00026
00027 private:
00028     Government* government;
00029     Policy policy;
00030     bool isEnforced;
00032 public:
00041     EnforcePolicyCommand(Government* gov, Policy pol);
00042
00048     void execute() override;
00049
00055     void undo() override;
00056
00062     bool isPolicyEnforced() const;
00063
00069     void setPolicyEnforced(bool enforced);
00070
00078     bool validateEnforcement() const;
00079
00085     void executeWithValidation();
00086
00092     std::string getName() const override;
00093
00099     std::string getDescription() const override;
00100
00106     bool canExecute() const override;
00107
00113     double returnVal() override;
00114 };
00115
00116 #endif // ENFORCEPOLICYCOMMAND_H
```

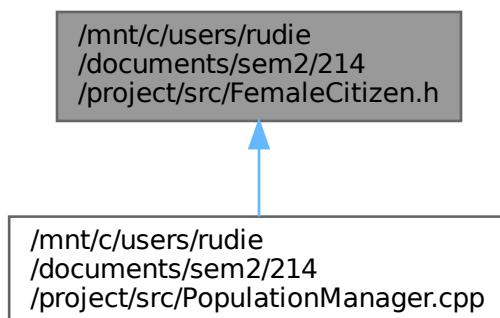
5.66 /mnt/c/users/rudie/documents/sem2/214/project/src/FemaleCitizen.h File Reference

Definition of the [FemaleCitizen](#) class.

```
#include "Citizen.h"
#include <memory>
Include dependency graph for FemaleCitizen.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [FemaleCitizen](#)
A class representing a female citizen.

5.66.1 Detailed Description

Definition of the [FemaleCitizen](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [FemaleCitizen](#) class, which inherits from the [Citizen](#) class. The [FemaleCitizen](#) class represents a female citizen with specific attributes and behaviors.

5.67 FemaleCitizen.h

[Go to the documentation of this file.](#)

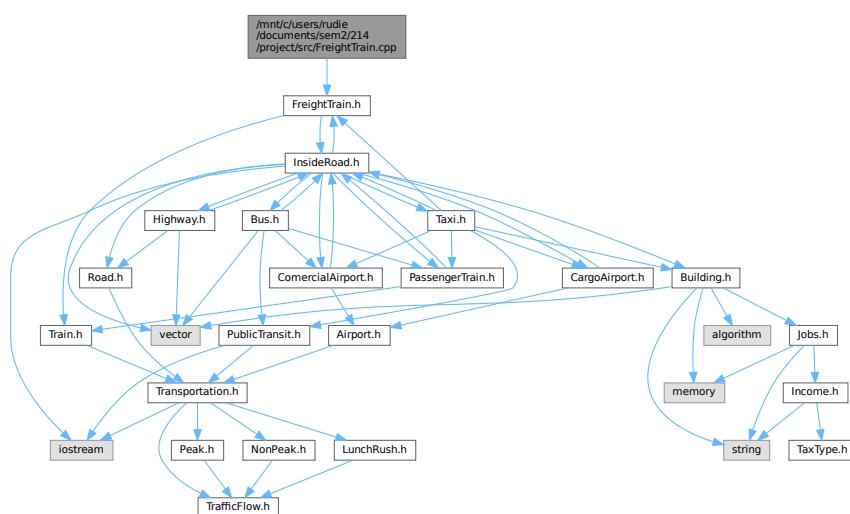
```
00001
00010 #ifndef FEMALE_CITIZEN_H
00011 #define FEMALE_CITIZEN_H
00012
00013 #include "Citizen.h"
00014 #include <memory>
00015
00023 class FemaleCitizen : public Citizen {
00024 public:
00030     FemaleCitizen(const std::string& name, int age) : Citizen(name, age) {}
00031
00036     std::shared_ptr<Citizen> clone() const override {
00037         return std::make_shared<FemaleCitizen>(*this);
00038     }
00039 };
00040
00041 #endif // FEMALE_CITIZEN_H
```

5.68 /mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.cpp File Reference

Implementation of the [FreightTrain](#) class.

```
#include "FreightTrain.h"
```

Include dependency graph for FreightTrain.cpp:



5.68.1 Detailed Description

Implementation of the [FreightTrain](#) class.

Version

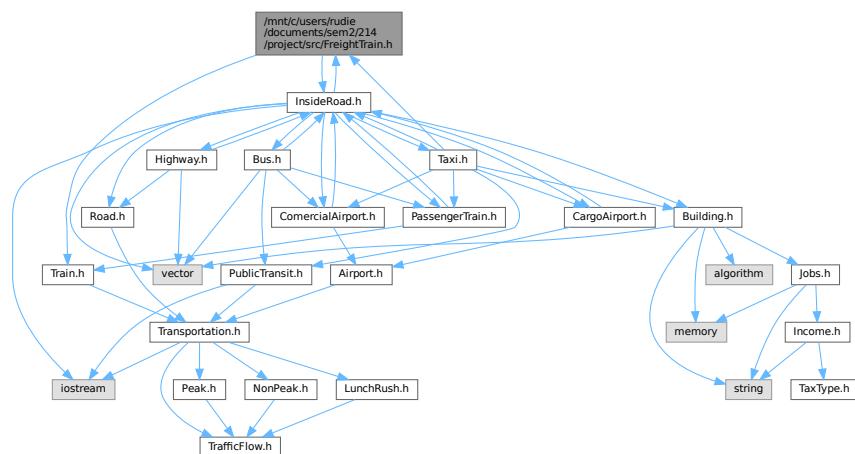
1.0

This file contains the implementation of the [FreightTrain](#) class, which inherits from the [Train](#) class. The [FreightTrain](#) class represents a freight train with specific attributes and behaviors.

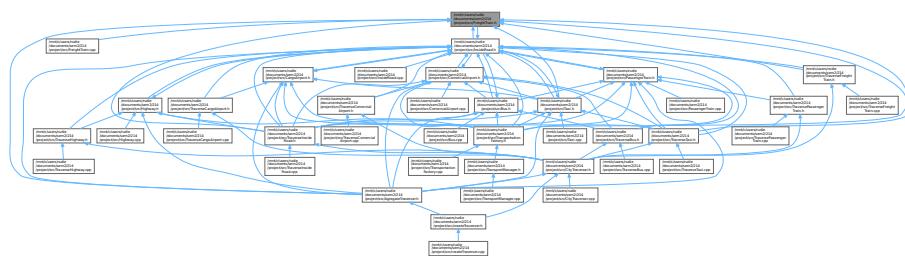
5.69 /mnt/c/users/rudie/documents/sem2/214/project/src/FreightTrain.h File Reference

Header file for the [FreightTrain](#) class.

```
#include "Train.h"
#include "InsideRoad.h"
Include dependency graph for FreightTrain.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [FreightTrain](#)

A class representing a freight train.

5.69.1 Detailed Description

Header file for the [FreightTrain](#) class.

This file contains the definition of the [FreightTrain](#) class, which inherits from the [Train](#) class. The [FreightTrain](#) class represents a freight train with specific attributes such as weight and length, and it can manage a collection of [InsideRoad](#) and [FreightTrain](#) objects.

Author

Samvit_Prakash_u23525119

Date

04/10/2024

5.70 FreightTrain.h

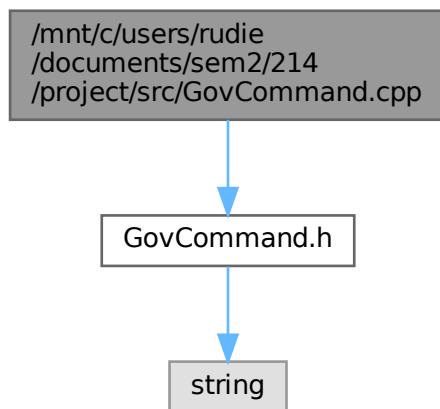
[Go to the documentation of this file.](#)

```
00001
00012 #ifndef FREIGHTTRAIN_H
00013 #define FREIGHTTRAIN_H
00014
00015 #include "Train.h"
00016 #include "InsideRoad.h"
00017
00018 class InsideRoad;
00019
00028 class FreightTrain : public Train {
00029     private:
00030         float weight;
00031         float length;
00032
00033         std::vector<InsideRoad*> insideRoads;
00034         std::vector<FreightTrain*> freightTrains;
00035
00036     public:
00045         FreightTrain(char state, std::string line, float weight, float length);
00046
00053         bool addInsideRoad(InsideRoad *insideRoad);
00054
00061         bool addFreightTrain(FreightTrain *freightTrain);
00062
00069         InsideRoad* getInsideRoad(size_t index);
00070
00077         FreightTrain* getFreightTrain(size_t index);
00078
00084         float getWeight();
00085
00091         float getLength();
00092
00098         std::string getTrainLine();
00099     };
00101 #endif // FREIGHTTRAIN_H
```

5.71 /mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.cpp File Reference

Implementation of the [GovCommand](#) class.

```
#include "GovCommand.h"  
Include dependency graph for GovCommand.cpp:
```



5.71.1 Detailed Description

Implementation of the [GovCommand](#) class.

Version

1.0

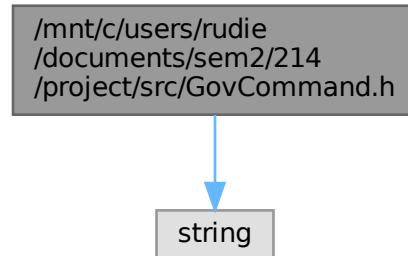
Date

2024-11-04 This file contains the implementation of the [GovCommand](#) class, which serves as a base class for various government-related commands. Derived classes should override the methods to provide specific command execution logic.

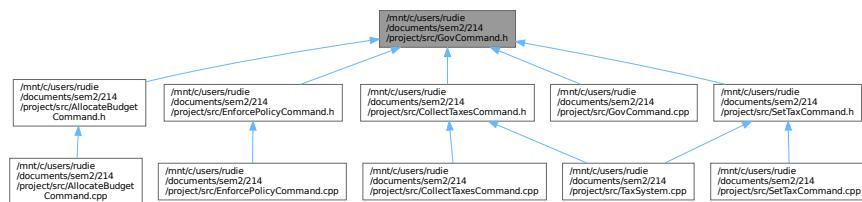
5.72 /mnt/c/users/rudie/documents/sem2/214/project/src/GovCommand.h File Reference

Definition of the [GovCommand](#) class.

```
#include <string>
Include dependency graph for GovCommand.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [GovCommand](#)
Abstract base class for government commands.

5.72.1 Detailed Description

Definition of the [GovCommand](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [GovCommand](#) class, which serves as a base class for various government-related commands. Derived classes should override the methods to provide specific command execution logic.

5.73 GovCommand.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef GOVCOMMAND_H
00012 #define GOVCOMMAND_H
00013
00014 #include <string>
00015
00023 class GovCommand {
00024 public:
00030     virtual void execute() = 0;
00031
00037     virtual void undo() = 0;
00038
00045     virtual double returnVal() = 0;
00046
00053     virtual std::string getName() const = 0;
00054
00061     virtual std::string getDescription() const = 0;
00062
00069     virtual bool canExecute() const = 0;
00070
00076     virtual ~GovCommand() = default;
00077 };
00078
00079 #endif // GOVCOMMAND_H

```

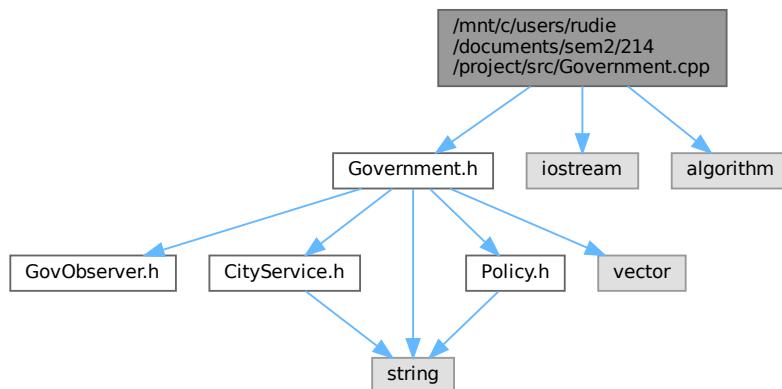
5.74 /mnt/c/users/rudie/documents/sem2/214/project/src/ ↵ Government.cpp File Reference

Implementation of the [Government](#) class.

```

#include "Government.h"
#include <iostream>
#include <algorithm>
Include dependency graph for Government.cpp:

```



5.74.1 Detailed Description

Implementation of the [Government](#) class.

Version

1.0

Date

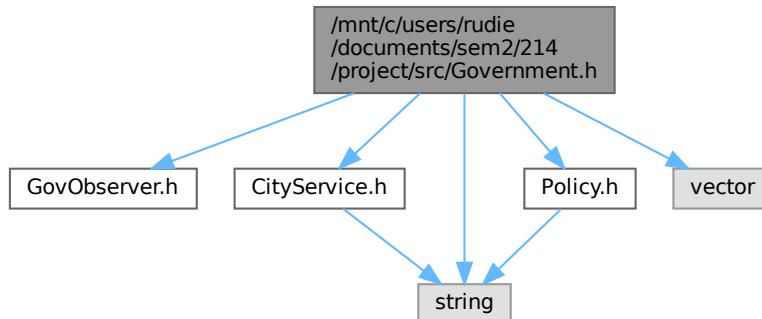
2024-11-04 This file contains the implementation of the [Government](#) class, which manages various government-related operations such as setting tax rates, allocating budgets, and notifying observers.

5.75 /mnt/c/users/rudie/documents/sem2/214/project/src/Government.h File Reference

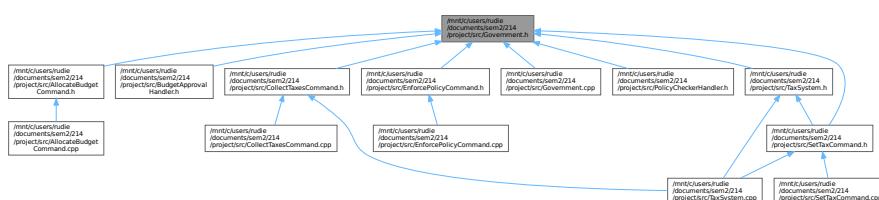
Definition of the [Government](#) class.

```
#include "GovObserver.h"
#include "CityService.h"
#include "Policy.h"
#include <vector>
#include <string>
```

Include dependency graph for Government.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Government](#)

Manages various government-related operations.

5.75.1 Detailed Description

Definition of the [Government](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [Government](#) class, which manages various government-related operations such as setting tax rates, allocating budgets, and notifying observers.

5.76 Government.h

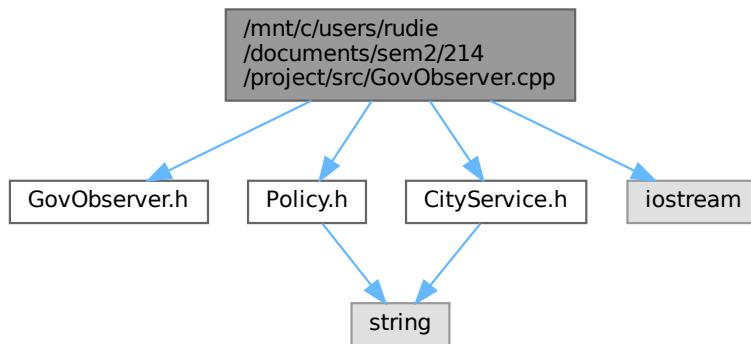
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef GOVERNMENT_H
00011 #define GOVERNMENT_H
00012
00013 #include "GovObserver.h"
00014 #include "CityService.h"
00015 #include "Policy.h"
00016 #include <vector>
00017 #include <string>
00018
00026 class Government {
00027
00028 private:
00029     double taxRate;
00030     double budget;
00031     std::string governmentName;
00032     std::vector<GovObserver*> observers;
00033
00034 public:
00039     Government(std::string name);
00040
00045     void setTax(double rate);
00046
00051     double getTaxRate() const;
00052
00056     void notifyCitizen();
00057
00061     void notifyBusinesses();
00062
00066     void notifyServices();
00067
00073     void allocateBudget(CityService& service, double amount);
00074
00080     void revertBudgetAllocation(CityService& service, double amount);
00081
00086     void enforcePolicy(Policy policy);
00087
00092     void update(int newPopulation);
00093
00098     double collectTaxes();
00099
00104     void refundTaxes(double amount);
00105
00110     void addTaxesToBudget(double amount);
00111
00116     void registerObserver(GovObserver* observer);
00117
00122     void unregisterObserver(GovObserver* observer);
00123
00127     void notifyObservers();
00128
00133     double getBudget() const;
00134 };
00135
00136 #endif // GOVERNMENT_H
```

5.77 /mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.cpp File Reference

Implementation of the [GovObserver](#) class.

```
#include "GovObserver.h"
#include "Policy.h"
#include "CityService.h"
#include <iostream>
Include dependency graph for GovObserver.cpp:
```



5.77.1 Detailed Description

Implementation of the [GovObserver](#) class.

Version

1.0

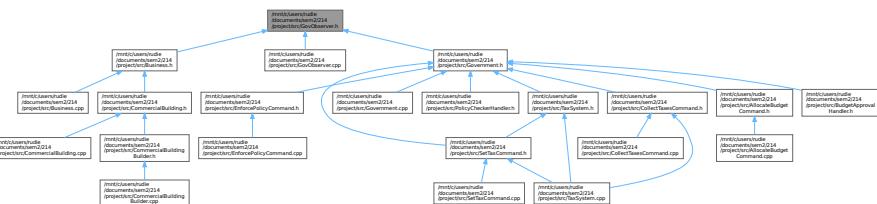
Date

2024-11-04 This file contains the implementation of the [GovObserver](#) class, which provides methods to update the tax rate, policy, and services for the observer.

5.78 /mnt/c/users/rudie/documents/sem2/214/project/src/GovObserver.h File Reference

Definition of the [GovObserver](#) class.

This graph shows which files directly or indirectly include this file:



Classes

- class [GovObserver](#)

Abstract base class for observers that monitor government changes.

5.78.1 Detailed Description

Definition of the [GovObserver](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [GovObserver](#) class, which provides methods to update the tax rate, policy, and services for the observer.

5.79 GovObserver.h

[Go to the documentation of this file.](#)

```
00001
00010 #ifndef GOVOSERVER_H
00011 #define GOVOSERVER_H
00012
00013 class Policy; // Forward declaration
00014 class CityService; // Forward declaration
00015
00022 class GovObserver {
00023 public:
00028     virtual void updateTaxRate(double rate) = 0;
00029
00034     virtual void updatePolicy(Policy policy) = 0;
00035
00040     virtual void updateServices(CityService services) = 0;
00041
00047     virtual ~GovObserver() = default;
00048 };
00049
00050 #endif // GOVOSERVER_H
```

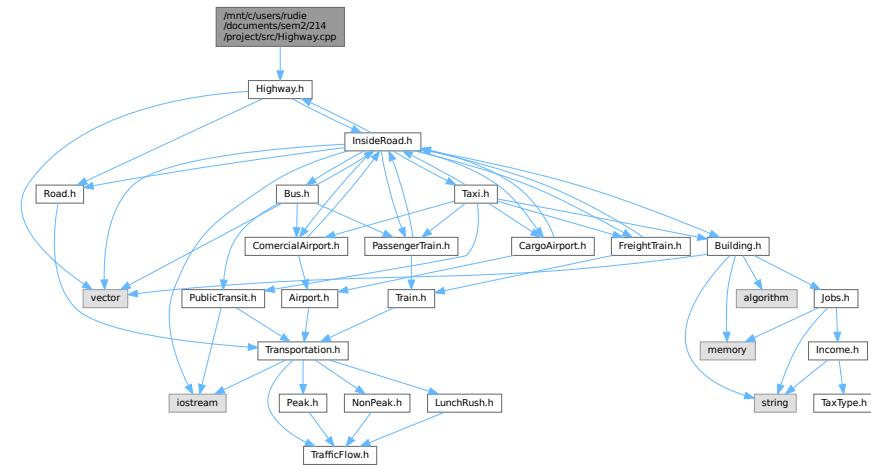
5.80 /mnt/c/users/rudie/documents/sem2/214/project/src/Highway.cpp

File Reference

Implementation of the [Highway](#) class.

```
#include "Highway.h"
```

Include dependency graph for Highway.cpp:



5.80.1 Detailed Description

Implementation of the [Highway](#) class.

Version

1.0

Date

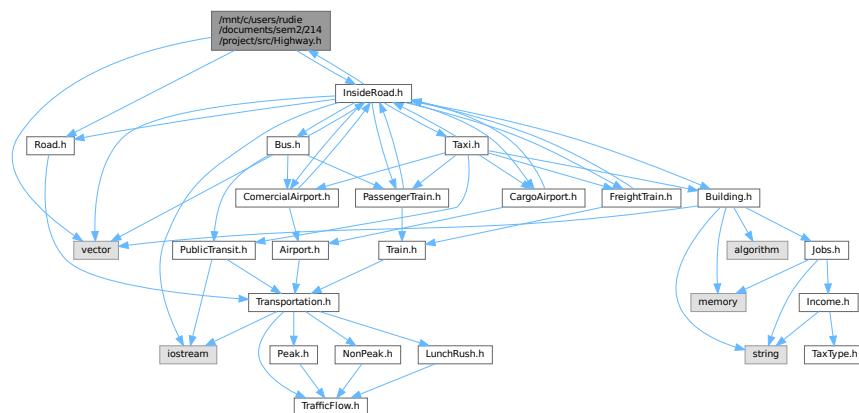
2024-11-04 This file contains the implementation of the [Highway](#) class, which inherits from the [Road](#) class. The [Highway](#) class represents a highway with specific attributes and behaviors.

5.81 /mnt/c/users/rudie/documents/sem2/214/project/src/Highway.h File Reference

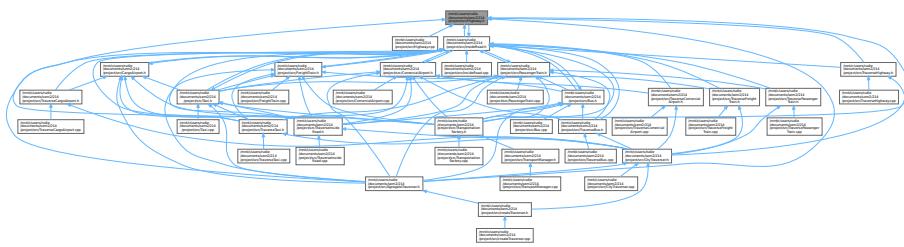
Header file for the [Highway](#) class.

```
#include <vector>
#include "Road.h"
```

```
#include "InsideRoad.h"
Include dependency graph for Highway.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Highway](#)

Represents a highway with a speed limit.

5.81.1 Detailed Description

Header file for the [Highway](#) class.

This file contains the definition of the [Highway](#) class, which inherits from the [Road](#) class. The [Highway](#) class represents a highway with a speed limit and can contain multiple inside roads and other highways.

Date

2024-11-04

Author

Samvit_Prakash_u23525119

Date

04/10/2024

5.82 Highway.h

[Go to the documentation of this file.](#)

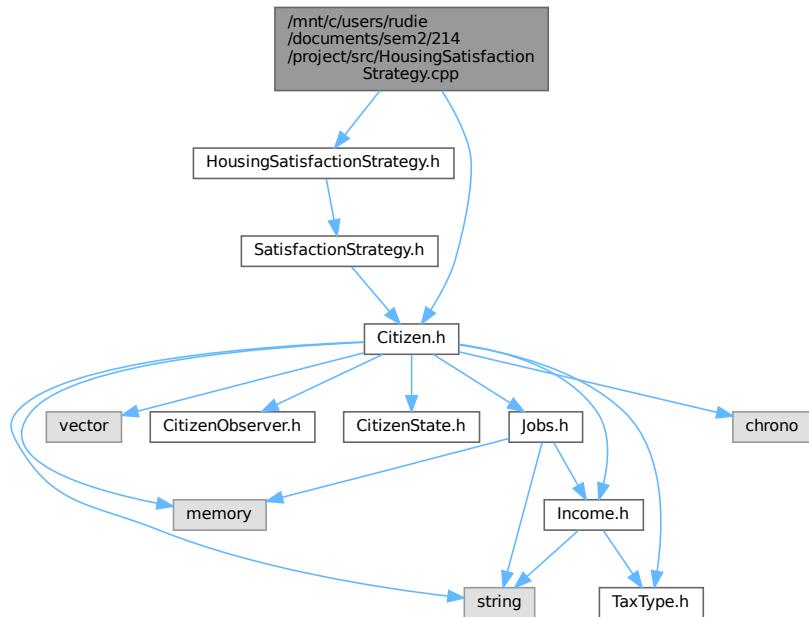
```
00001
00011 #ifndef HIGHWAY_H
00012 #define HIGHWAY_H
00013
00014 #include <vector>
00015
00016 #include "Road.h"
00017 #include "InsideRoad.h"
00018
00019 class InsideRoad;
00020
00028 class Highway : public Road {
00029     private:
00030         float speedLimit;
00031         std::vector<InsideRoad*> insideRoads;
00032         std::vector<Highway*> highways;
00033
00034     public:
00042         Highway(char state, std::string roadName, float speedLimit);
00043
00050         bool addInsideRoad(InsideRoad *insideRoad);
00051
00058         bool addHighway(Highway *highway);
00059
00066         InsideRoad *getInsideRoad(std::size_t x);
00067
00074         Highway *getHighway(std::size_t x);
00075
00081         std::vector<InsideRoad*> getInsideRoadsList();
00082
00088         std::vector<Highway*> getHighwaysList();
00089
00095         std::string getRoadName();
00096
00102         float getSpeedLimit();
00103 };
00104
00105 #endif // HIGHWAY_H
```

5.83 /mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.cpp File Reference

Implementation of the [HousingSatisfactionStrategy](#) class.

```
#include "HousingSatisfactionStrategy.h"
#include "Citizen.h"
```

Include dependency graph for `HousingSatisfactionStrategy.cpp`:



5.83.1 Detailed Description

Implementation of the [HousingSatisfactionStrategy](#) class.

Version

1.0

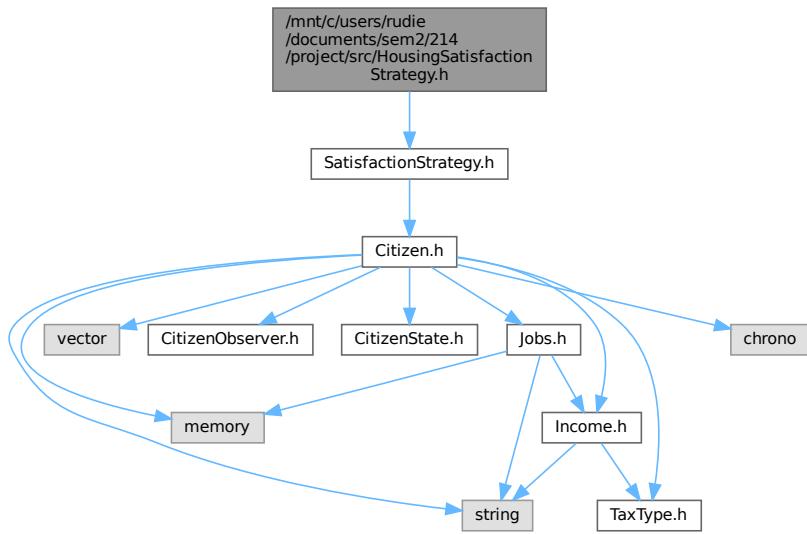
Date

2024-11-04 This file contains the implementation of the [HousingSatisfactionStrategy](#) class, which provides methods to calculate and update citizen satisfaction based on housing conditions.

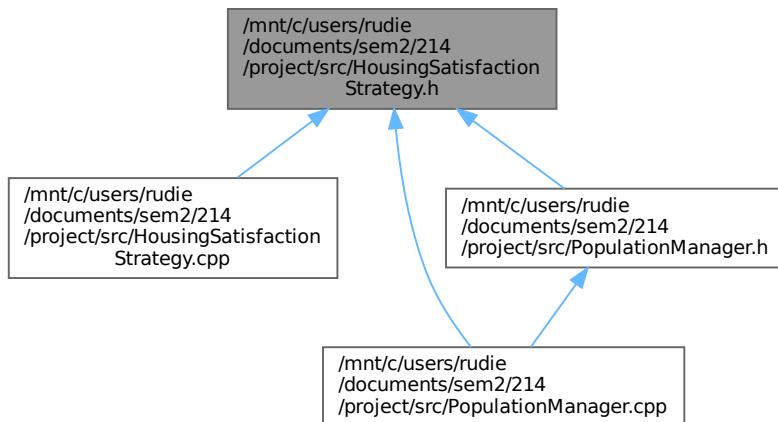
5.84 /mnt/c/users/rudie/documents/sem2/214/project/src/HousingSatisfactionStrategy.h File Reference

Definition of the [HousingSatisfactionStrategy](#) class.

```
#include "SatisfactionStrategy.h"
Include dependency graph for HousingSatisfactionStrategy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [HousingSatisfactionStrategy](#)
Strategy for calculating and updating citizen satisfaction based on housing conditions.

5.84.1 Detailed Description

Definition of the [HousingSatisfactionStrategy](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [HousingSatisfactionStrategy](#) class, which provides methods to calculate and update citizen satisfaction based on housing conditions.

5.85 HousingSatisfactionStrategy.h

[Go to the documentation of this file.](#)

```

00001
00010 #ifndef HOUSINGSATISFACTIONSTRATEGY_H
00011 #define HOUSINGSATISFACTIONSTRATEGY_H
00012
00013 #include "SatisfactionStrategy.h"
00014
00022 class HousingSatisfactionStrategy : public SatisfactionStrategy {
00023 public:
00029     float calculateSatisfaction(const Citizen& citizen) override;
00030
00035     void updateForHousingChange(Citizen& citizen) override;
00036
00043     void updateForJobChange(Citizen& citizen) override {};
00044
00051     void updateForTaxChange(Citizen& citizen) override {};
00052 };
00053
00054 #endif // HOUSINGSATISFACTIONSTRATEGY_H

```

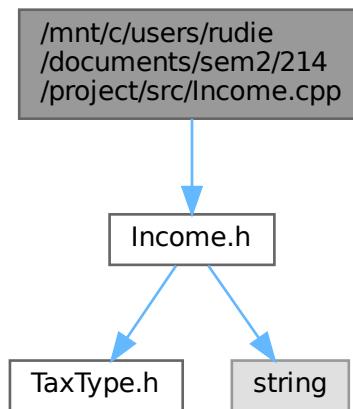
5.86 /mnt/c/users/rudie/documents/sem2/214/project/src/Income.cpp

File Reference

Implementation of the [Income](#) class.

```
#include "Income.h"
```

Include dependency graph for Income.cpp:



5.86.1 Detailed Description

Implementation of the [Income](#) class.

Version

1.0

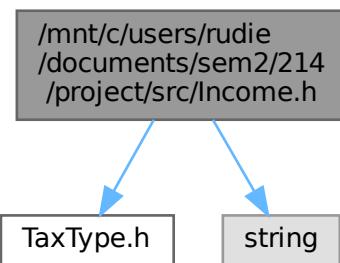
Date

2024-11-04 This file contains the implementation of the [Income](#) class, which provides methods to calculate monthly income, add bonuses, apply deductions, and calculate taxes.

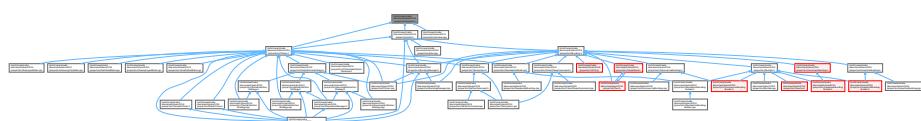
5.87 /mnt/c/users/rudie/documents/sem2/214/project/src/Income.h File Reference

Definition of the [Income](#) class.

```
#include "TaxType.h"
#include <string>
Include dependency graph for Income.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Income](#)

Manages income-related operations.

5.87.1 Detailed Description

Definition of the [Income](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [Income](#) class, which provides methods to calculate monthly income, add bonuses, apply deductions, and calculate taxes.

5.88 Income.h

[Go to the documentation of this file.](#)

```

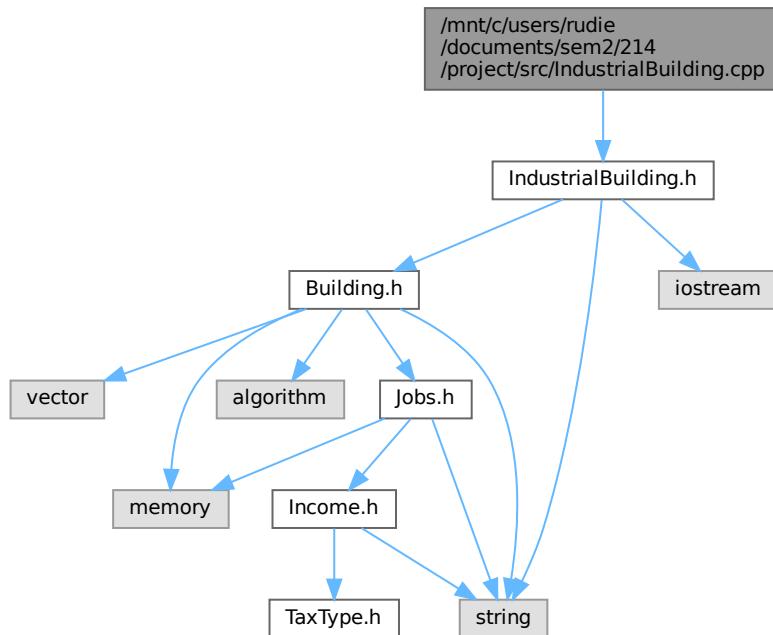
00001
00010 #ifndef INCOME_H
00011 #define INCOME_H
00012
00013 #include "TaxType.h"
00014 #include <string>
00015
00023 class Income : public TaxType {
00024
00025 private:
00026     double incomeTax;
00027     char cType;
00028     double baseSalary;
00029     double bonus;
00030     double deductions;
00031
00032 public:
00033     Income(double salary, double rate, const char category)
00034         : TaxType(rate, category), baseSalary(salary), bonus(0.0), deductions(0.0) {}
00041
00046     Income(double salary) : Income(salary, 0.0, 'A') {}
00047
00052     double calculateMonthlyIncome() const;
00053
00058     void addBonus(double amount);
00059
00064     void applyDeductions(double amount);
00065
00071     double payTaxes(TaxType& taxType);
00072
00077     void setTax(double rate) override;
00078
00084     double calculateTax(double val) override;
00085
00086     // Getters
00091     double getBaseSalary() const { return baseSalary; }
00092
00097     double getBonus() const { return bonus; }
00098
00103     double getDeductions() const { return deductions; }
00104 };
00105
00106 #endif // INCOME_H

```

5.89 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.cpp File Reference

Implementation of the [IndustrialBuilding](#) class.

```
#include "IndustrialBuilding.h"
Include dependency graph for IndustrialBuilding.cpp:
```



5.89.1 Detailed Description

Implementation of the [IndustrialBuilding](#) class.

Version

1.0

Date

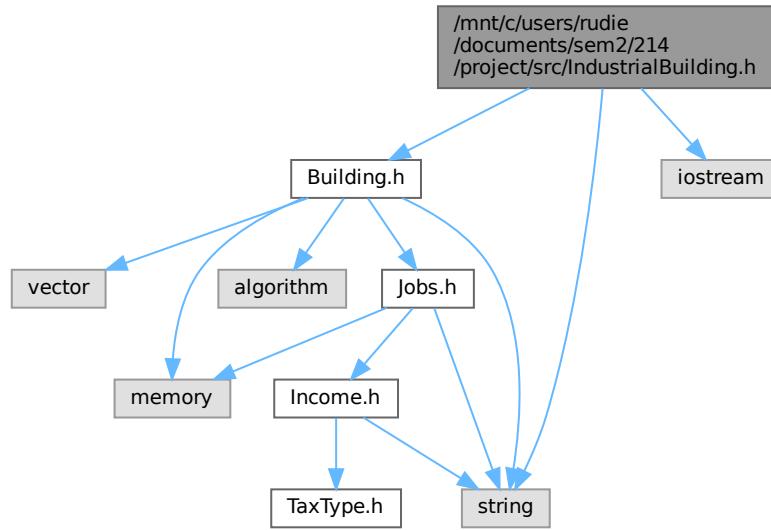
2024-11-04 This file contains the implementation of the [IndustrialBuilding](#) class, which represents an industrial building with specific attributes and behaviors.

5.90 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.h File Reference

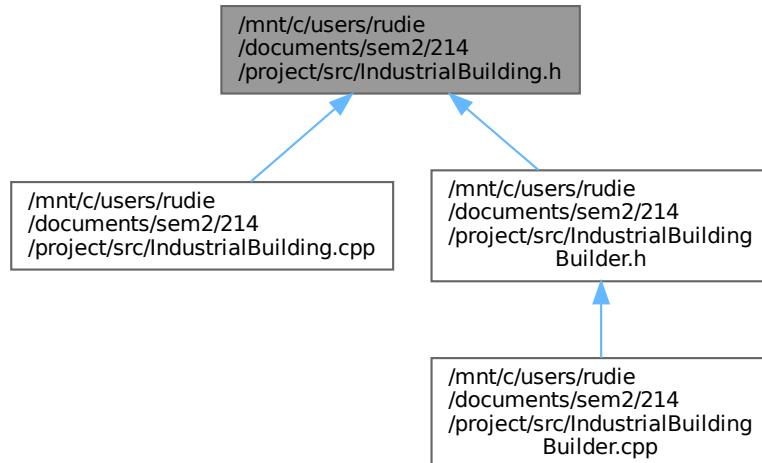
Definition of the [IndustrialBuilding](#) class.

```
#include "Building.h"
#include <string>
```

```
#include <iostream>
Include dependency graph for IndustrialBuilding.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [IndustrialBuilding](#)
Represents an industrial building with specific attributes and behaviors.

5.90.1 Detailed Description

Definition of the [IndustrialBuilding](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [IndustrialBuilding](#) class, which represents an industrial building with specific attributes and behaviors.

5.91 IndustrialBuilding.h

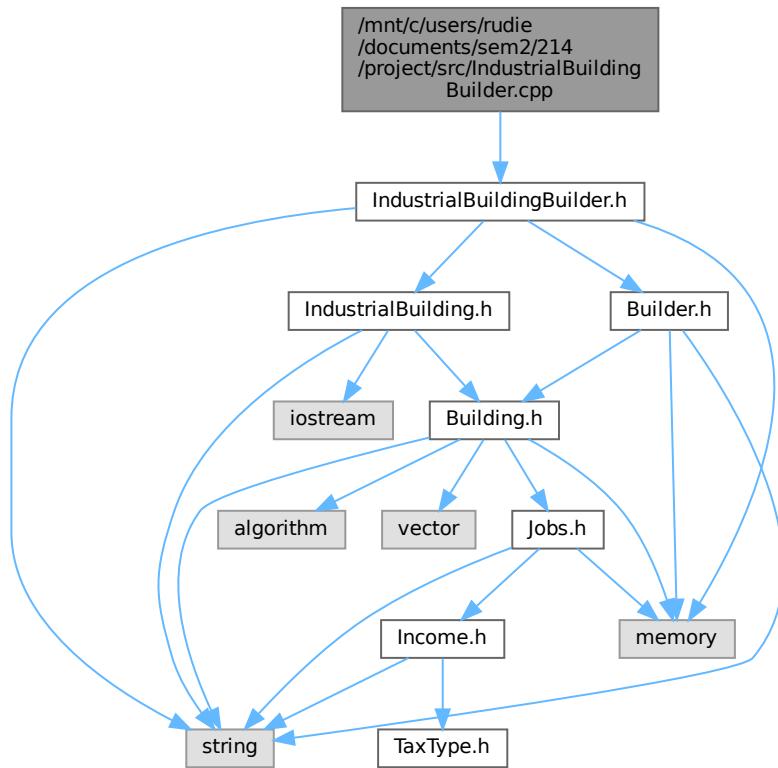
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef INDUSTRIALBUILDING_H
00011 #define INDUSTRIALBUILDING_H
00012
00013 #include "Building.h"
00014 #include <string>
00015 #include <iostream>
00016
00017 using namespace std;
00018
00026 class IndustrialBuilding : public Building {
00027
00028 private:
00029     float pollutionLevel;
00030     float productionCapacity;
00031     string bType;
00032
00033 public:
00046     IndustrialBuilding(const std::string& name, float area, int floors, int capacity,
00047                         float citizenSatisfaction, float economicGrowth, float resourceConsumption,
00048                         float pollutionLevel, float productionCapacity);
00049
00054     string getType() const override;
00055
00059     void construct() override;
00060
00064     void updateImpacts() override;
00065
00070     void upgradeTech(float techLevel);
00071
00077     double payTaxes(TaxType* taxType);
00078
00082     void undoCollectTaxes();
00083
00084 protected:
00088     void calculateEconomicImpact();
00089
00093     void calculateResourceConsumption();
00094
00098     void calculateSatisfactionImpact();
00099 };
00100
00101 #endif // INDUSTRIALBUILDING_H
```

5.92 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.cpp File Reference

Implementation of the [IndustrialBuildingBuilder](#) class.

```
#include "IndustrialBuildingBuilder.h"
Include dependency graph for IndustrialBuildingBuilder.cpp:
```



5.92.1 Detailed Description

Implementation of the [IndustrialBuildingBuilder](#) class.

Version

1.0

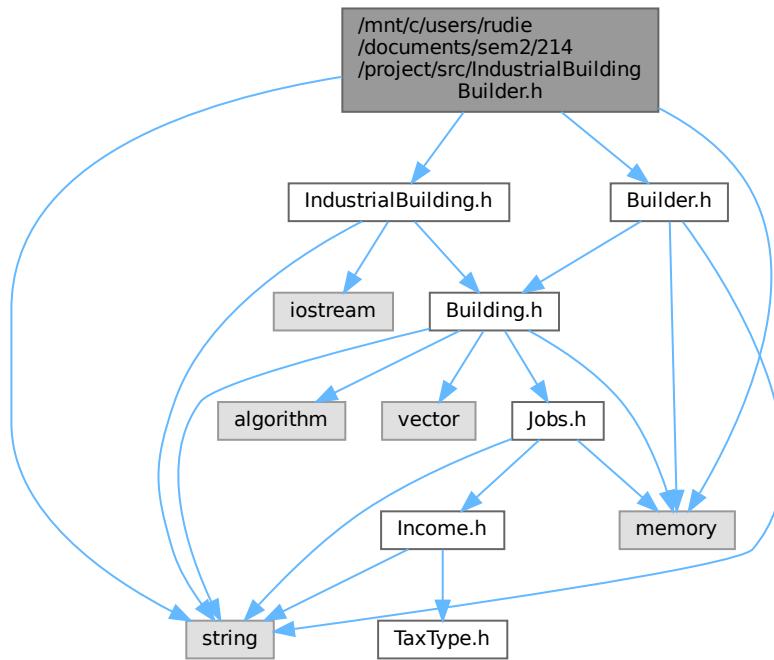
Date

2024-11-04 This file contains the implementation of the [IndustrialBuildingBuilder](#) class, which provides methods to set attributes and build an [IndustrialBuilding](#) object.

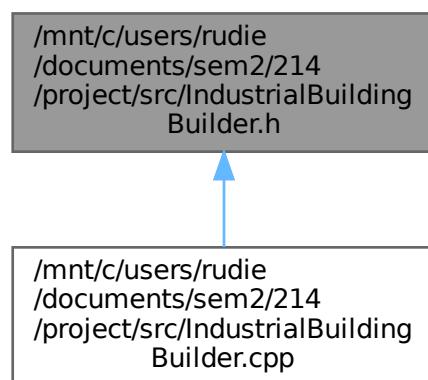
5.93 /mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingBuilder.h File Reference

Definition of the [IndustrialBuildingBuilder](#) class.

```
#include "Builder.h"
#include "IndustrialBuilding.h"
#include <memory>
#include <string>
Include dependency graph for IndustrialBuildingBuilder.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [IndustrialBuildingBuilder](#)

Builder class for constructing *IndustrialBuilding* objects.

5.93.1 Detailed Description

Definition of the *IndustrialBuildingBuilder* class.

Version

1.0

Date

2024-11-04 This file contains the definition of the *IndustrialBuildingBuilder* class, which provides methods to set attributes and build an *IndustrialBuilding* object.

5.94 IndustrialBuildingBuilder.h

[Go to the documentation of this file.](#)

```
00001
00010 #ifndef INDUSTRIALBUILDINGBUILDER_H
00011 #define INDUSTRIALBUILDINGBUILDER_H
00012
00013 #include "Builder.h"
00014 #include "IndustrialBuilding.h"
00015 #include <memory>
00016 #include <string>
00017
00018 using namespace std;
00019
00026 class IndustrialBuildingBuilder : public Builder {
00027
00028 private:
00029     float pollutionLevel = 0.0f;
00030     float productionCapacity = 0.0f;
00031
00032 public:
00033     IndustrialBuildingBuilder();
00034
00043     IndustrialBuildingBuilder& setPollutionLevel(float level);
00044
00050     IndustrialBuildingBuilder& setProductionCapacity(float capacity);
00051
00056     float getPollutionLevel();
00057
00062     float getProductionCapacity();
00063
00068     std::unique_ptr<Building> build() override;
00069 };
00070
00071 #endif // INDUSTRIALBUILDINGBUILDER_H
```

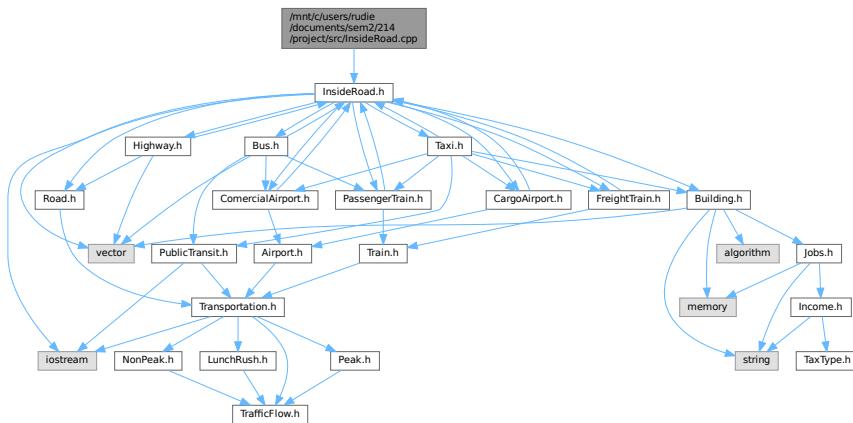
5.95 /mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.cpp

File Reference

Implementation of the *InsideRoad* class.

```
#include "InsideRoad.h"
```

Include dependency graph for InsideRoad.cpp:



5.95.1 Detailed Description

Implementation of the [InsideRoad](#) class.

Version

1.0

Date

2024-11-04 This file contains the implementation of the [InsideRoad](#) class, which represents an inside road with specific attributes and behaviors.

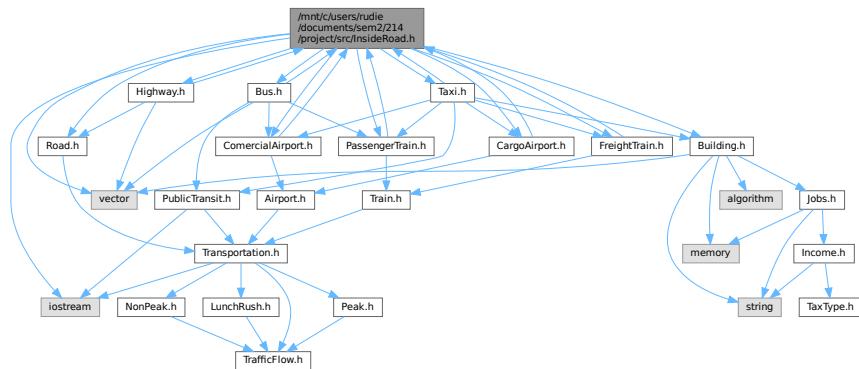
5.96 /mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.h File Reference

Header file for the [InsideRoad](#) class.

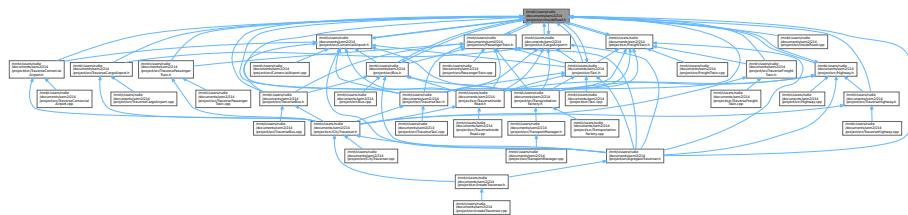
```
#include <vector>
#include <iostream>
#include "Road.h"
#include "Highway.h"
#include "Bus.h"
#include "Taxi.h"
#include "ComercialAirport.h"
#include "CargoAirport.h"
#include "PassengerTrain.h"
#include "FreightTrain.h"
```

```
#include "Building.h"
```

Include dependency graph for InsideRoad.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [InsideRoad](#)
Represents an inside road that can contain various transportation entities.

5.96.1 Detailed Description

Header file for the [InsideRoad](#) class.

Version

1.0

Date

2024-11-04 This file contains the declaration of the [InsideRoad](#) class, which represents a type of road that can contain various transportation entities such as buses, taxis, airports, and trains.

04/10/2024

5.97 InsideRoad.h

[Go to the documentation of this file.](#)

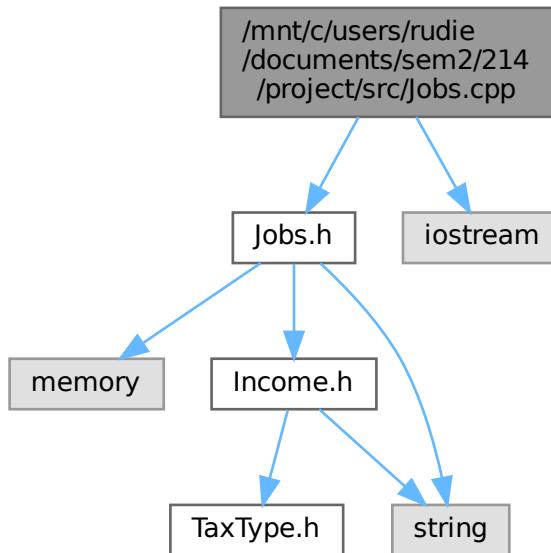
```
00001
00012 #ifndef INSIDEROAD_H
00013 #define INSIDEROAD_H
00014
00015 #include <vector>
00016 #include <iostream>
00017
00018 #include "Road.h"
00019 #include "Highway.h"
00020 #include "Bus.h"
00021 #include "Taxi.h"
00022 #include "ComercialAirport.h"
00023 #include "CargoAirport.h"
00024 #include "PassengerTrain.h"
00025 #include "FreightTrain.h"
00026 #include "Building.h"
00027
00028 class Highway;
00029 class Bus;
00030 class Taxi;
00031 class ComercialAirport;
00032 class CargoAirport;
00033 class PassengerTrain;
00034 class FreightTrain;
00035 class Building;
00036
00044 class InsideRoad : public Road {
00045     private:
00046         float avgStopTime;
00047         std::vector<InsideRoad*> insideRoads;
00048         std::vector<Highway*> highways;
00049         std::vector<Bus*> buses;
00050         std::vector<Taxi*> taxis;
00051         std::vector<ComercialAirport*> comercialAirports;
00052         std::vector<CargoAirport*> cargoAirports;
00053         std::vector<PassengerTrain*> passengerTrains;
00054         std::vector<FreightTrain*> freightTrains;
00055         std::vector<Building*> buildings;
00056
00057     public:
00064         InsideRoad(char state, std::string roadName, float avgStopTime);
00065
00071         bool addInsideRoad(InsideRoad *insideRoad);
00072
00078         bool addHighway(Highway *highway);
00079
00085         bool addBus(Bus *bus);
00086
00092         bool addTaxi(Taxi *taxi);
00093
00099         bool addComercialAirport(ComercialAirport *comercialAirport);
00100
00106         bool addCargoAirport(CargoAirport *cargoAirport);
00107
00113         bool addPassengerTrain(PassengerTrain *passengerTrain);
00114
00120         bool addFreightTrain(FreightTrain *freightTrain);
00121
00127         FreightTrain* getFreightTrain(std::size_t x);
00128
00137         bool addBuilding(Building *building);
00138
00144         PassengerTrain* getPassengerTrain(std::size_t x);
00145
00151         Highway* getHighway(std::size_t x);
00152
00158         InsideRoad* getInsideRoad(std::size_t x);
00159
00165         Bus* getBus(std::size_t x);
00166
00172         Taxi* getTaxi(std::size_t x);
00173
00179         ComercialAirport* getComercialAirport(std::size_t x);
00180
00186         CargoAirport* getCargoAirport(std::size_t x);
00187
00193         Building *getBuilding(std::size_t x);
00194
00199         float getAvgStopTime();
00200
00205         std::string getRoadName();
00206 };
```

```
00207  
00208 #endif // INSIDEROAD_H
```

5.98 /mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.cpp File Reference

Implementation of the [Jobs](#) class.

```
#include "Jobs.h"  
#include <iostream>  
Include dependency graph for Jobs.cpp:
```



5.98.1 Detailed Description

Implementation of the [Jobs](#) class.

Version

1.0

Date

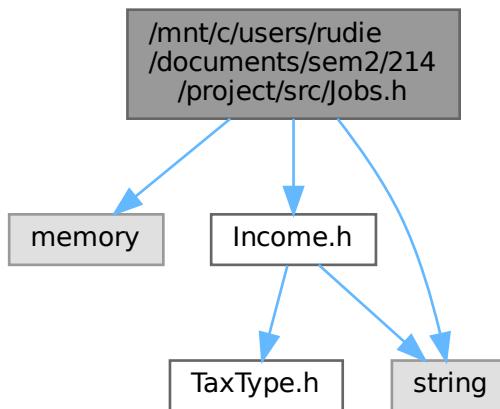
2024-11-04 This file contains the implementation of the [Jobs](#) class, which manages job-related operations such as hiring and releasing employees, and displaying job information.

5.99 /mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.h File Reference

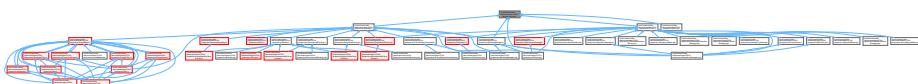
Definition of the [Jobs](#) class.

```
#include <memory>
#include "Income.h"
#include <string>
```

Include dependency graph for Jobs.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Jobs](#)

Manages job-related operations.

5.99.1 Detailed Description

Definition of the [Jobs](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [Jobs](#) class, which manages job-related operations such as hiring and releasing employees, and displaying job information.

5.100 Jobs.h

[Go to the documentation of this file.](#)

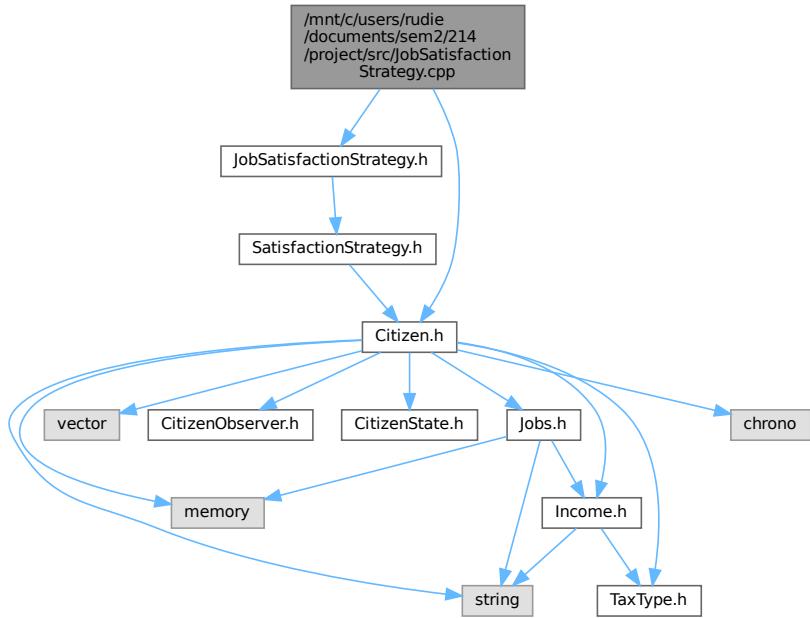
```
00001
00010 #ifndef JOBS_H
00011 #define JOBS_H
00012
00013 #include <memory>
00014 #include "Income.h"
00015 #include <string>
00016 #include <memory>
00017
00025 class Jobs {
00026 private:
00027     std::string title;
00028     bool occupied = false;
00029     std::shared_ptr<Income> income;
00030
00031 public:
00032     Jobs(const std::string& jobTitle, double salary);
00033
00034     // Getters
00035     std::string getTitle() const { return title; }
00036
00037     std::shared_ptr<Income> getIncome() const { return income; }
00038
00039     bool isOccupied() const { return occupied; }
00040
00041     void hireEmployee();
00042
00043     void releaseEmployee();
00044
00045     void displayJobInfo() const;
00046
00047     void unOcuppy();
00048
00049 };
00050 #endif // JOBS_H
```

5.101 /mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.cpp File Reference

Implementation of the [JobSatisfactionStrategy](#) class.

```
#include "JobSatisfactionStrategy.h"
#include "Citizen.h"
```

Include dependency graph for JobSatisfactionStrategy.cpp:



5.101.1 Detailed Description

Implementation of the [JobSatisfactionStrategy](#) class.

Version

1.0

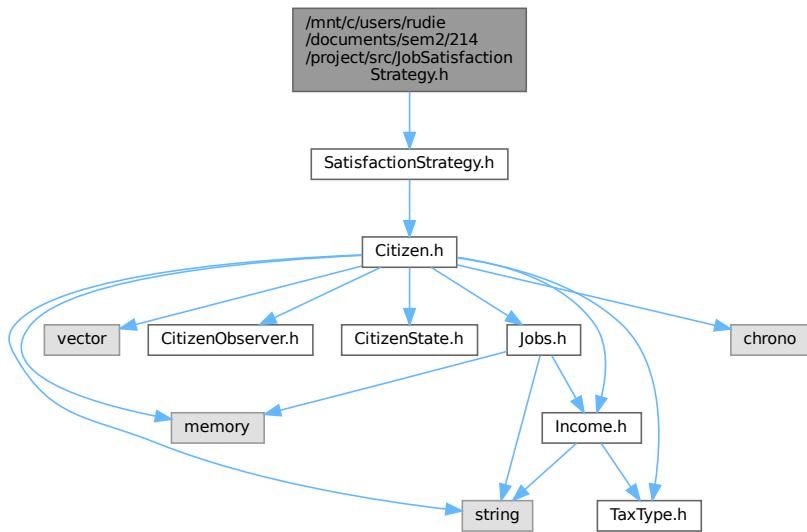
Date

2024-11-04 This file contains the implementation of the [JobSatisfactionStrategy](#) class, which provides methods to calculate and update citizen satisfaction based on job conditions.

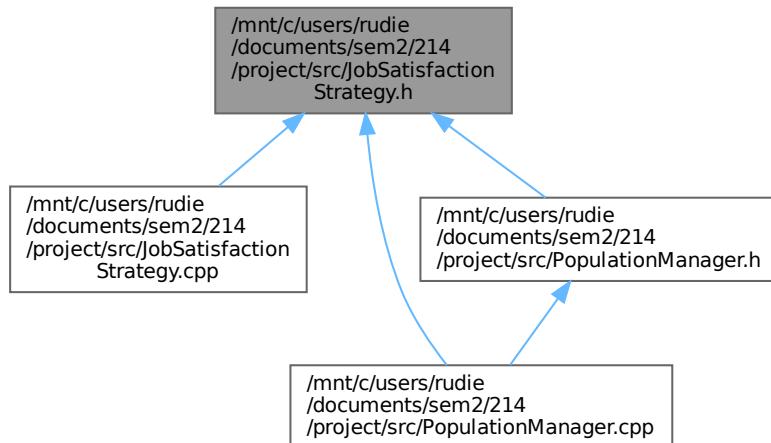
5.102 /mnt/c/users/rudie/documents/sem2/214/project/src/JobSatisfactionStrategy.h File Reference

Definition of the [JobSatisfactionStrategy](#) class.

```
#include "SatisfactionStrategy.h"
Include dependency graph for JobSatisfactionStrategy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [JobSatisfactionStrategy](#)

Strategy for calculating and updating citizen satisfaction based on job conditions.

5.102.1 Detailed Description

Definition of the [JobSatisfactionStrategy](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [JobSatisfactionStrategy](#) class, which provides methods to calculate and update citizen satisfaction based on job conditions.

5.103 JobSatisfactionStrategy.h

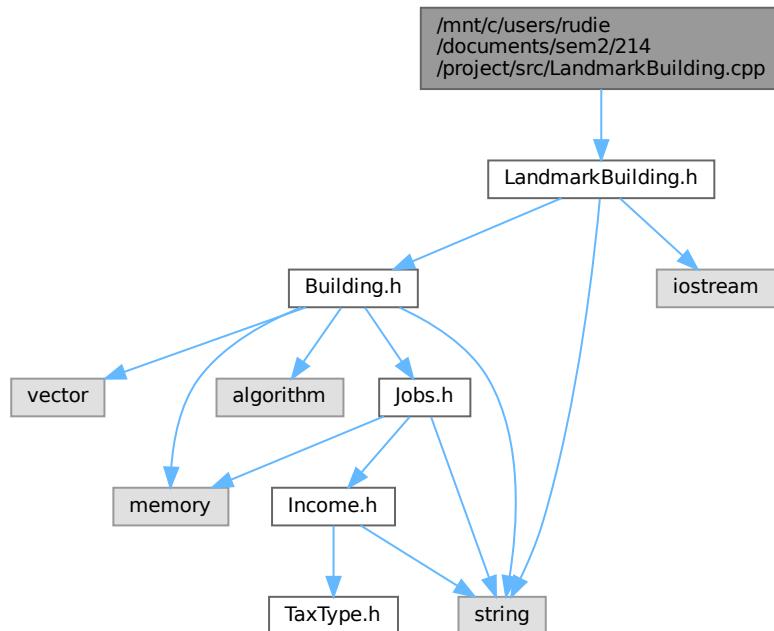
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef JOBSATISFACTIONSTRATEGY_H
00011 #define JOBSATISFACTIONSTRATEGY_H
00012
00013 #include "SatisfactionStrategy.h"
00014
00022 class JobSatisfactionStrategy : public SatisfactionStrategy {
00023 public:
00029     float calculateSatisfaction(const Citizen& citizen) override;
00030
00035     void updateForJobChange(Citizen& citizen) override;
00036
00043     void updateForHousingChange(Citizen& citizen) override {}
00044
00051     void updateForTaxChange(Citizen& citizen) override {}
00052 };
00053
00054 #endif // JOBSATISFACTIONSTRATEGY_H
```

5.104 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.cpp File Reference

Implementation of the [LandmarkBuilding](#) class.

```
#include "LandmarkBuilding.h"
Include dependency graph for LandmarkBuilding.cpp:
```



5.104.1 Detailed Description

Implementation of the [LandmarkBuilding](#) class.

Version

1.0

Date

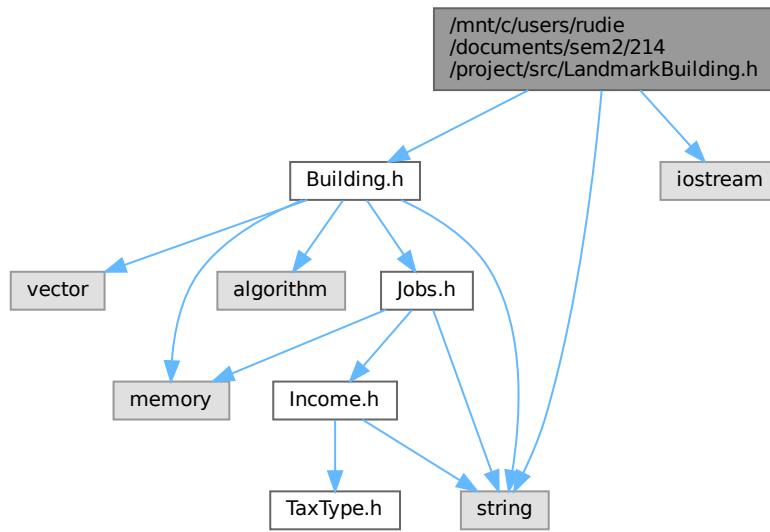
2024-11-04 This file contains the implementation of the [LandmarkBuilding](#) class, which represents a landmark building with specific attributes and behaviors.

5.105 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuilding.h File Reference

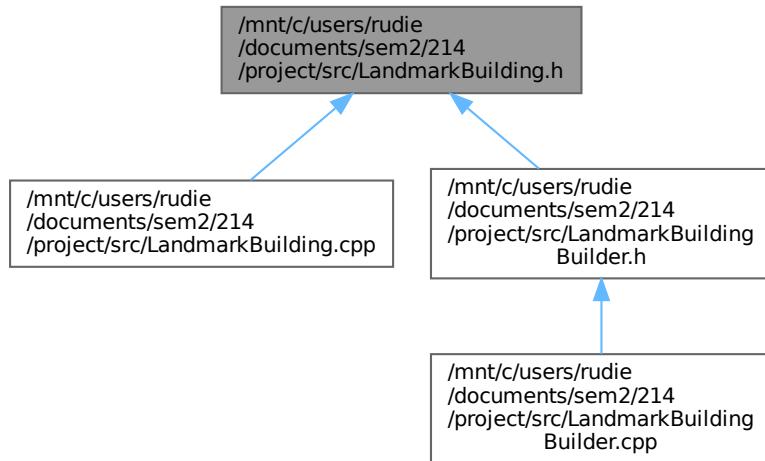
Definition of the [LandmarkBuilding](#) class.

```
#include "Building.h"
#include <string>
```

```
#include <iostream>
Include dependency graph for LandmarkBuilding.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [LandmarkBuilding](#)

Represents a landmark building with specific attributes and behaviors.

5.105.1 Detailed Description

Definition of the [LandmarkBuilding](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [LandmarkBuilding](#) class, which represents a landmark building with specific attributes and behaviors.

5.106 LandmarkBuilding.h

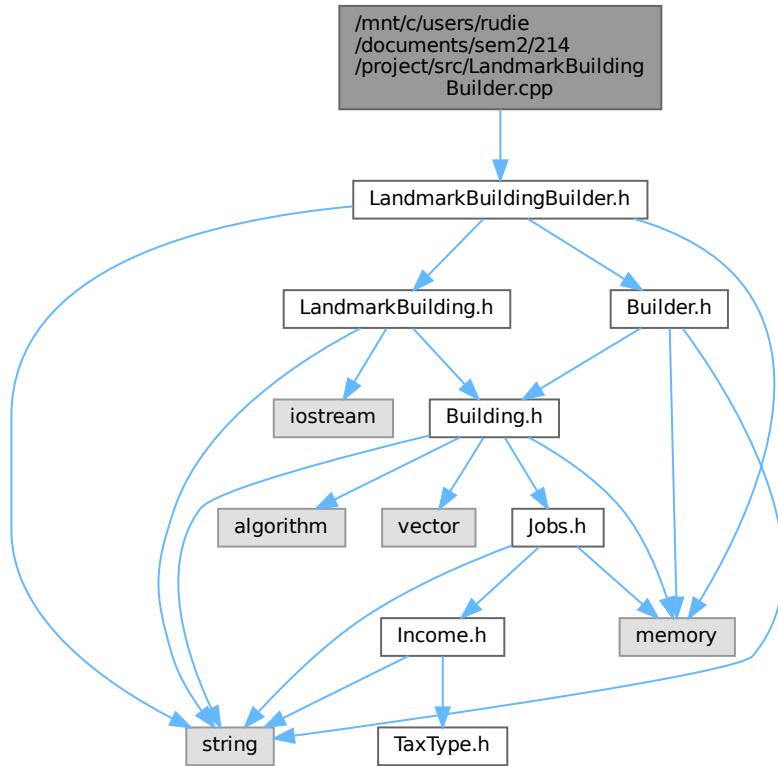
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef LANDMARKBUILDING_H
00011 #define LANDMARKBUILDING_H
00012
00013 #include "Building.h"
00014 #include <string>
00015 #include <iostream>
00016
00017 using namespace std;
00018
00026 class LandmarkBuilding : public Building {
00027
00028 private:
00029     int visitorCapacity;
00030     float culturalValue;
00031     bool isHistoric;
00032     string bType;
00033
00034 public:
00048     LandmarkBuilding(const std::string& name, float area, int floors, int capacity,
00049                         float citizenSatisfactionImpact, float economicGrowthImpact,
00050                         float resourceConsumption, int visitorCapacity, float culturalValue, bool
00051                         isHistoric);
00056     string getType() const override;
00057
00061     void updateImpacts() override;
00062
00066     void construct() override;
00067
00072     void hostEvent(int visitors);
00073
00079     double payTaxes(TaxType* taxType) override;
00080
00084     void undoCollectTaxes() override;
00085
00086 protected:
00090     void calculateEconomicImpact();
00091
00095     void calculateResourceConsumption();
00096
00100     void calculateSatisfactionImpact();
00101 };
00102
00103 #endif // LANDMARKBUILDING_H
```

5.107 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.cpp File Reference

Implementation of the [LandmarkBuildingBuilder](#) class.

```
#include "LandmarkBuildingBuilder.h"
Include dependency graph for LandmarkBuildingBuilder.cpp:
```



5.107.1 Detailed Description

Implementation of the [LandmarkBuildingBuilder](#) class.

Version

1.0

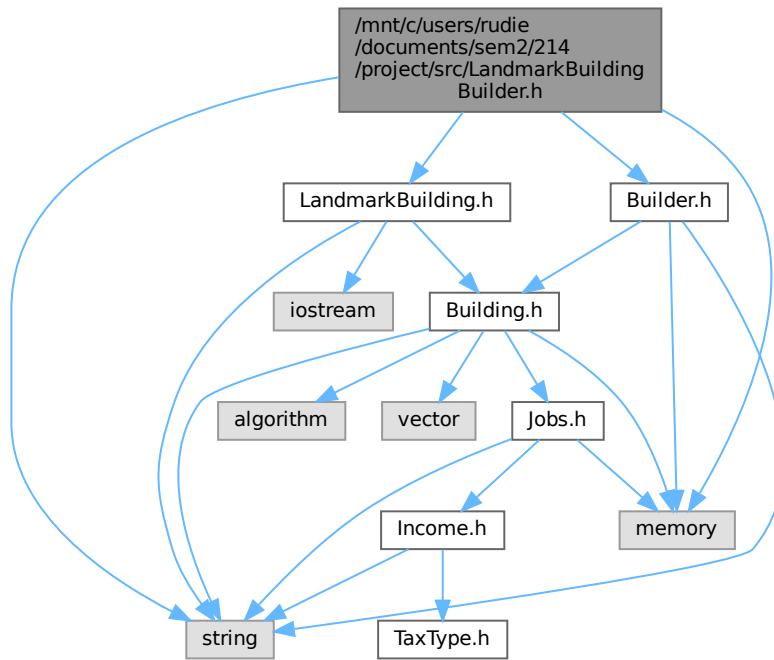
Date

2024-11-04 This file contains the implementation of the [LandmarkBuildingBuilder](#) class, which provides methods to set attributes and build a [LandmarkBuilding](#) object.

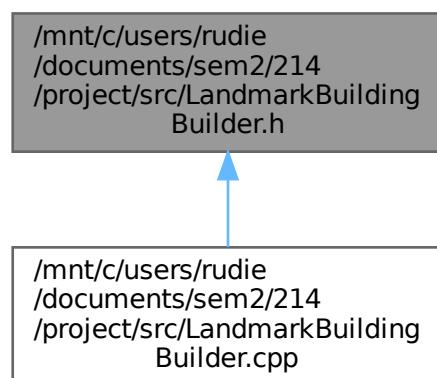
5.108 /mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.h File Reference

Definition of the [LandmarkBuildingBuilder](#) class.

```
#include "Builder.h"
#include "LandmarkBuilding.h"
#include <string>
#include <memory>
Include dependency graph for LandmarkBuildingBuilder.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [LandmarkBuildingBuilder](#)

Builder class for constructing *LandmarkBuilding* objects.

5.108.1 Detailed Description

Definition of the [LandmarkBuildingBuilder](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [LandmarkBuildingBuilder](#) class, which provides methods to set attributes and build a [LandmarkBuilding](#) object.

5.109 LandmarkBuildingBuilder.h

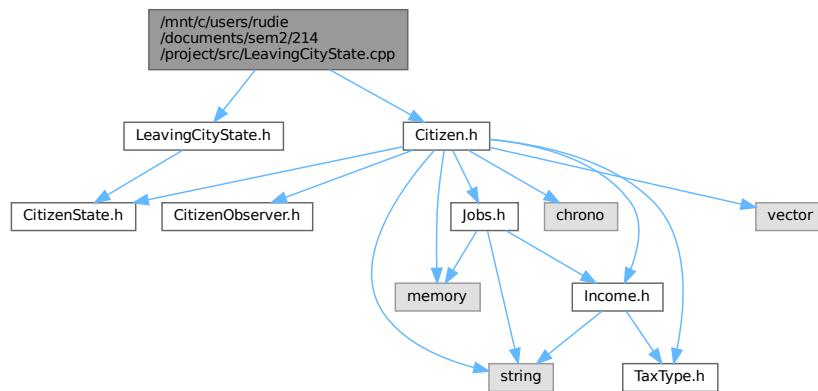
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef LANDMARKBUILDINGBUILDER_H
00011 #define LANDMARKBUILDINGBUILDER_H
00012
00013 #include "Builder.h"
00014 #include "LandmarkBuilding.h"
00015 #include <string>
00016 #include <memory>
00017
00018 using namespace std;
00019
00026 class LandmarkBuildingBuilder : public Builder {
00027
00028 private:
00029     int visitorCapacity = 0;
00030     float culturalValue = 0.0f;
00031     bool isHistoric = false;
00032
00033 public:
00037     LandmarkBuildingBuilder();
00038
00043     int getVisitorCapacity();
00044
00050     LandmarkBuildingBuilder& setVisitorCapacity(int visitorCapacity);
00051
00056     float getCulturalValue();
00057
00063     LandmarkBuildingBuilder& setCulturalValue(float culturalValue);
00064
00069     bool getIsHistoric();
00070
00076     LandmarkBuildingBuilder& setIsHistoric(bool isHistoric);
00077
00082     std::unique_ptr<Building> build() override;
00083 };
00084
00085 #endif // LANDMARKBUILDINGBUILDER_H
```

5.110 /mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.cpp File Reference

Implementation of the [LeavingCityState](#) class.

```
#include "LeavingCityState.h"
#include "Citizen.h"
Include dependency graph for LeavingCityState.cpp:
```



5.110.1 Detailed Description

Implementation of the [LeavingCityState](#) class.

Version

1.0

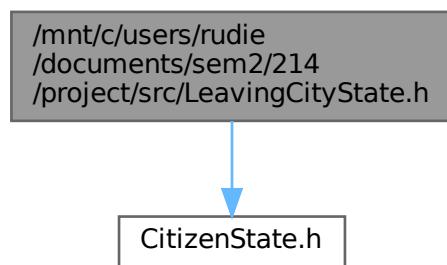
Date

2024-11-04 This file contains the implementation of the [LeavingCityState](#) class, which handles the state of a citizen preparing to leave the city.

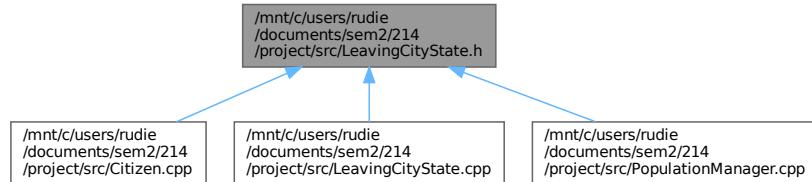
5.111 /mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityState.h File Reference

Definition of the [LeavingCityState](#) class.

```
#include "CitizenState.h"
Include dependency graph for LeavingCityState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [LeavingCityState](#)
Handles the state of a citizen preparing to leave the city.

5.111.1 Detailed Description

Definition of the [LeavingCityState](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [LeavingCityState](#) class, which handles the state of a citizen preparing to leave the city.

5.112 LeavingCityState.h

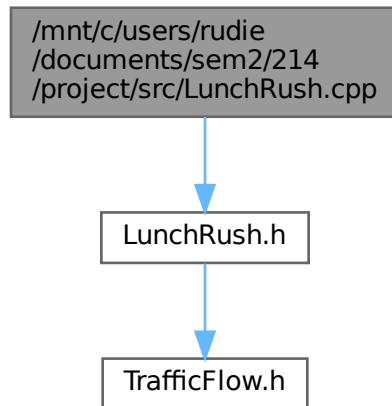
[Go to the documentation of this file.](#)

```
00001  
00010 #ifndef LEAVINGCITYSTATE_H  
00011 #define LEAVINGCITYSTATE_H  
00012  
00013 #include "CitizenState.h"  
00014  
00022 class LeavingCityState : public CitizenState {  
00023 public:  
00028     void handleState(Citizen& citizen) const override;  
00029 };  
00030  
00031 #endif // LEAVINGCITYSTATE_H
```

5.113 /mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.cpp File Reference

Implementation of the [LunchRush](#) class.

```
#include "LunchRush.h"  
Include dependency graph for LunchRush.cpp:
```



5.113.1 Detailed Description

Implementation of the [LunchRush](#) class.

Version

1.0

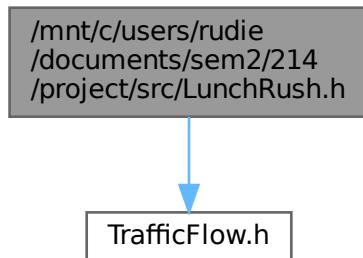
Date

2024-11-04 This file contains the implementation of the [LunchRush](#) class, which provides methods to get the traffic flow and state during the lunch rush period.

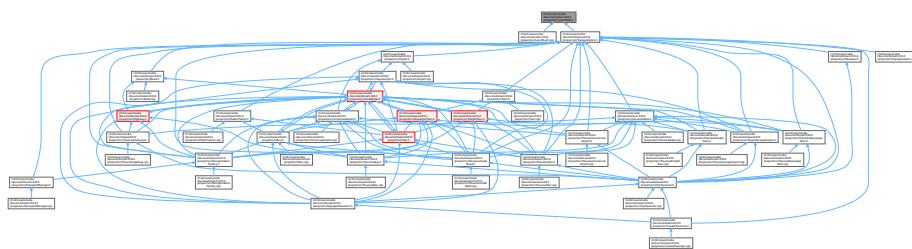
5.114 /mnt/c/users/rudie/documents/sem2/214/project/src/LunchRush.h File Reference

Header file for the [LunchRush](#) class.

```
#include "TrafficFlow.h"  
Include dependency graph for LunchRush.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [LunchRush](#)
A class representing traffic flow during lunch hours.

5.114.1 Detailed Description

Header file for the [LunchRush](#) class.

Version

1.0

Date

2024-11-04 This file contains the declaration of the [LunchRush](#) class, which is derived from the [TrafficFlow](#) class. The [LunchRush](#) class represents a specific state of traffic flow during lunch hours.

04/10/2024

5.115 LunchRush.h

[Go to the documentation of this file.](#)

```

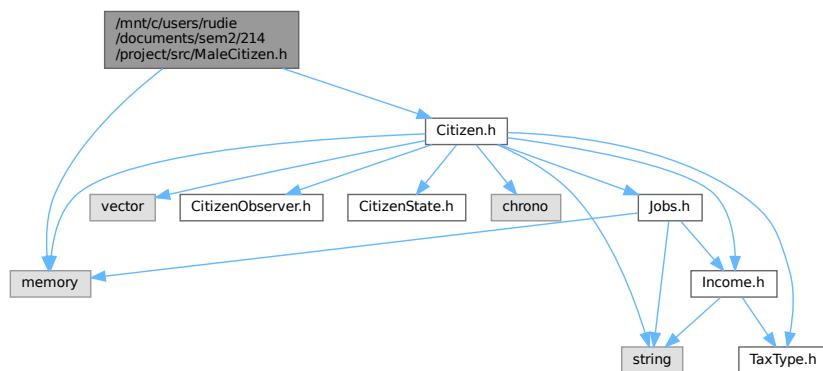
00001
00012 #ifndef LUNCHRUSH_H
00013 #define LUNCHRUSH_H
00014
00015 #include "TrafficFlow.h"
00016
00024 class LunchRush : public TrafficFlow {
00025     private:
00026         char state = 'L';
00027         float trafficFlow = 0.5;
00028
00029     public:
00034         float getTrafficFlow();
00035
00040         char getState();
00041     };
00042
00043 #endif // LUNCHRUSH_H

```

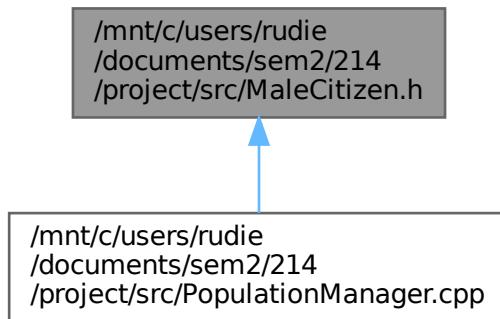
5.116 /mnt/c/users/rudie/documents/sem2/214/project/src/MaleCitizen.h File Reference

Definition of the [MaleCitizen](#) class.

```
#include "Citizen.h"
#include <memory>
Include dependency graph for MaleCitizen.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [MaleCitizen](#)

A class representing a male citizen.

5.116.1 Detailed Description

Definition of the [MaleCitizen](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [MaleCitizen](#) class, which inherits from the [Citizen](#) class. The [MaleCitizen](#) class represents a male citizen with specific attributes and behaviors.

5.117 MaleCitizen.h

[Go to the documentation of this file.](#)

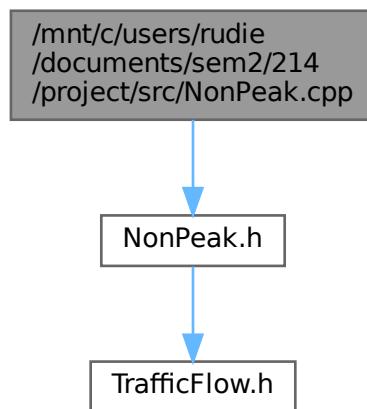
```
00001
00010 #ifndef MALE_CITIZEN_H
00011 #define MALE_CITIZEN_H
00012
00013 #include "Citizen.h"
00014 #include <memory>
00015
00023 class MaleCitizen : public Citizen {
00024 public:
00030     MaleCitizen(const std::string& name, int age) : Citizen(name, age) {}
00031
00036     std::shared_ptr<Citizen> clone() const override {
00037         return std::make_shared<MaleCitizen>(*this);
00038     }
00039 };
00040
00041 #endif // MALE_CITIZEN_H
```

5.118 /mnt/c/users/rudie/documents/sem2/214/project/src/NonPeak.cpp File Reference

Implementation of the [NonPeak](#) class.

```
#include "NonPeak.h"
```

Include dependency graph for NonPeak.cpp:



5.118.1 Detailed Description

Implementation of the [NonPeak](#) class.

Version

1.0

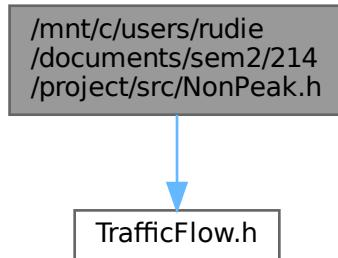
Date

2024-11-04 This file contains the implementation of the [NonPeak](#) class, which provides methods to get the traffic flow and state during non-peak hours.

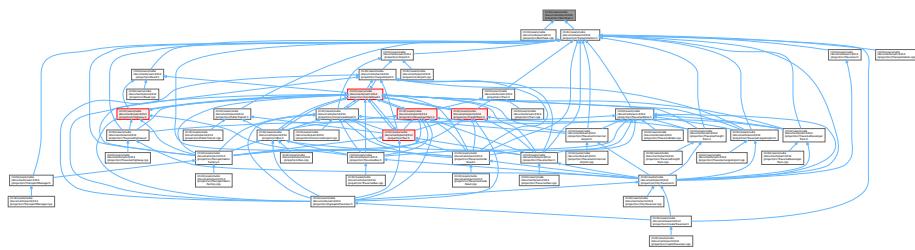
5.119 /mnt/c/users/rudie/documents/sem2/214/project/src/NonPeak.h File Reference

Header file for the [NonPeak](#) class.

```
#include "TrafficFlow.h"  
Include dependency graph for NonPeak.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [NonPeak](#)
A class to represent traffic flow during non-peak hours.

5.119.1 Detailed Description

Header file for the [NonPeak](#) class.

Version

1.0

This file contains the definition of the [NonPeak](#) class, which is a subclass of [TrafficFlow](#). The [NonPeak](#) class represents traffic flow during non-peak hours.

Date

2024-11-04

5.120 NonPeak.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef NONPEAK_H
00012 #define NONPEAK_H
00013
00014 #include "TrafficFlow.h"
00015
00023 class NonPeak : public TrafficFlow {
00024     private:
00025         float trafficFlow = 0.2;
00026         char state = 'N';
00027
00028     public:
00034         float getTrafficFlow();
00035
00041         char getState();
00042
00043 };
00044
00045 #endif // NONPEAK_H

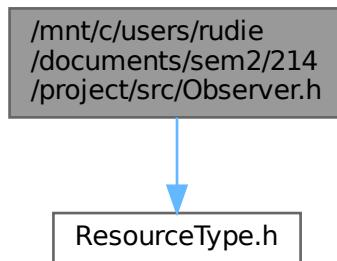
```

5.121 /mnt/c/users/rudie/documents/sem2/214/project/src/Observer.h File Reference

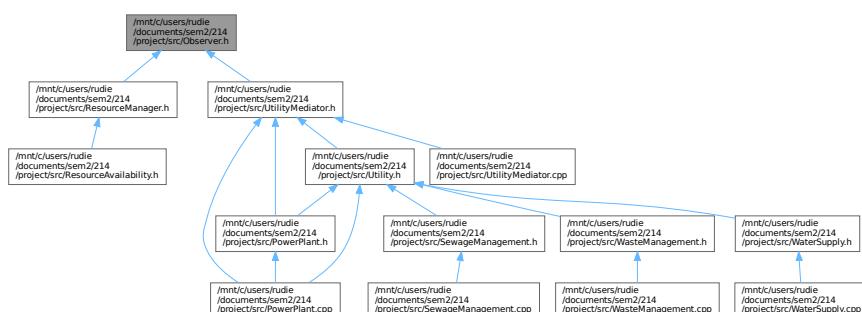
Definition of the [Observer](#) class.

```
#include "ResourceType.h"
```

Include dependency graph for Observer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Observer**

Interface for objects that need to be notified of changes in resource types and quantities.

5.121.1 Detailed Description

Definition of the **Observer** class.

Version

1.0

This file contains the definition of the **Observer** class, which provides an interface for objects that need to be notified of changes in resource types and quantities.

5.122 Observer.h

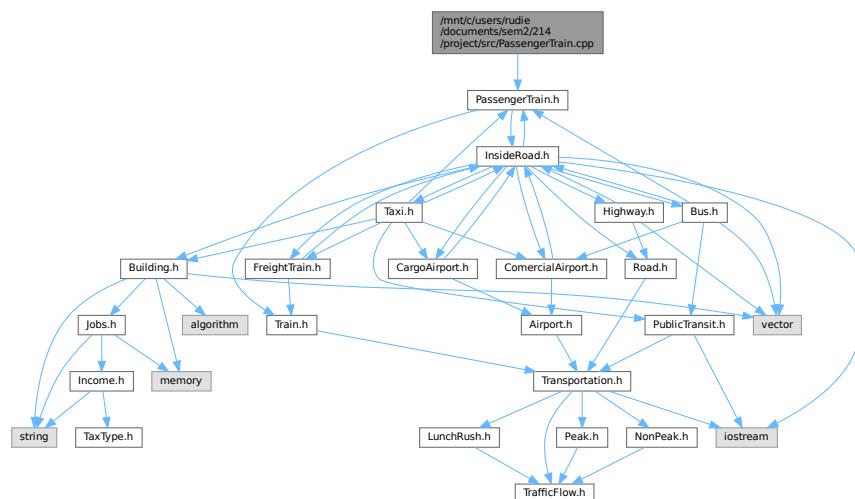
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef OBSERVER_H
00011 #define OBSERVER_H
00012
00013 #include "ResourceType.h"
00014
00022 class Observer {
00023 public:
00029     virtual void update(ResourceType type, int quantity) = 0;
00030 };
00031
00032 #endif // OBSERVER_H
```

5.123 /mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.cpp File Reference

Implementation of the **PassengerTrain** class.

```
#include "PassengerTrain.h"
Include dependency graph for PassengerTrain.cpp:
```



5.123.1 Detailed Description

Implementation of the [PassengerTrain](#) class.

Version

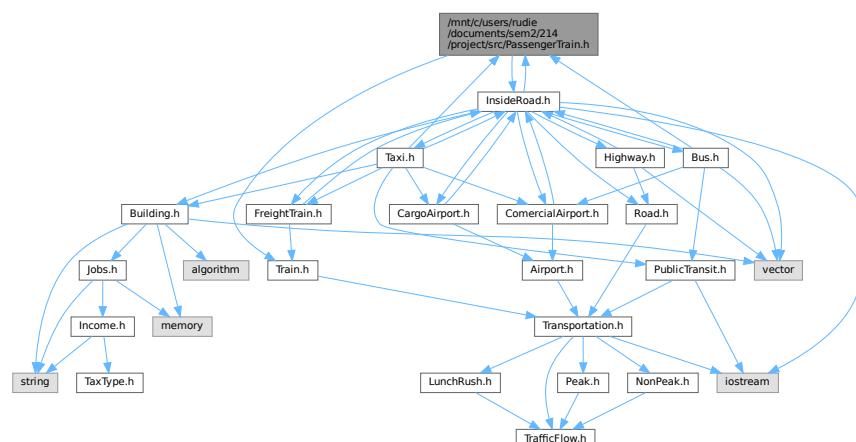
1.0

This file contains the implementation of the [PassengerTrain](#) class, which represents a passenger train with specific attributes and behaviors.

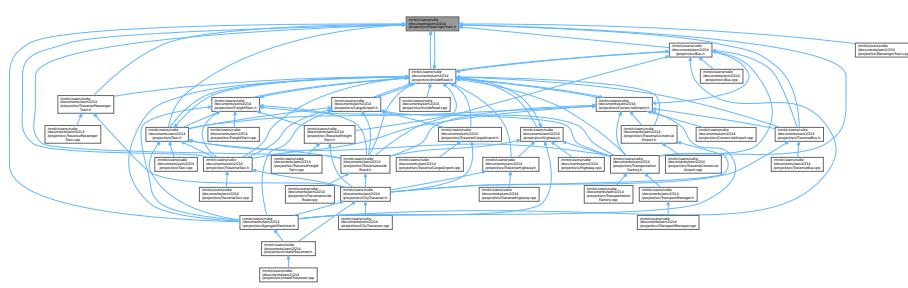
5.124 /mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.h File Reference

Header file for the [PassengerTrain](#) class.

```
#include "Train.h"
#include "InsideRoad.h"
Include dependency graph for PassengerTrain.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PassengerTrain](#)
A class representing a passenger train.

5.124.1 Detailed Description

Header file for the [PassengerTrain](#) class.

Version

1.0

This file contains the declaration of the [PassengerTrain](#) class, which inherits from the [Train](#) class. The [PassengerTrain](#) class manages a collection of [InsideRoad](#) and [PassengerTrain](#) objects.

Date

04/10/2024

5.125 PassengerTrain.h

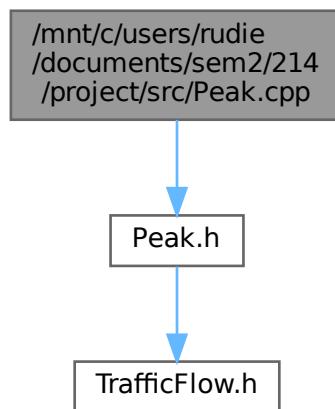
[Go to the documentation of this file.](#)

```
00001
00011 #ifndef PASSANGERTRAIN_H
00012 #define PASSANGERTRAIN_H
00013
00014 #include "Train.h"
00015 #include "InsideRoad.h"
00016
00017 class InsideRoad;
00018
00025 class PassengerTrain : public Train {
00026     private:
00027         std::vector<InsideRoad*> insideRoads;
00028         std::vector<PassengerTrain*> passengerTrains;
00029
00030     public:
00031         PassengerTrain(char state, std::string line);
00032
00033         bool addInsideRoad(InsideRoad *insideRoad);
00034
00035         bool addPassengerTrain(PassengerTrain *passengerTrain);
00036
00037         InsideRoad *getInsideRoad(std::size_t x);
00038
00039         PassengerTrain *getPassengerTrain(std::size_t x);
00040
00041         std::string getTrainLine();
00042
00043     };
00044
00045 #endif // PASSANGERTRAIN_H
```

5.126 /mnt/c/users/rudie/documents/sem2/214/project/src/Peak.cpp File Reference

Implementation of the [Peak](#) class.

```
#include "Peak.h"  
Include dependency graph for Peak.cpp:
```



5.126.1 Detailed Description

Implementation of the [Peak](#) class.

Version

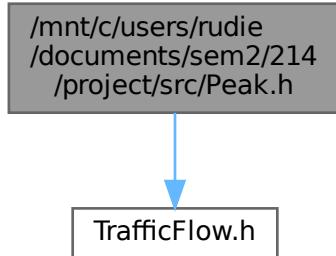
1.0

This file contains the implementation of the [Peak](#) class, which provides methods to get the traffic flow and state during peak hours.

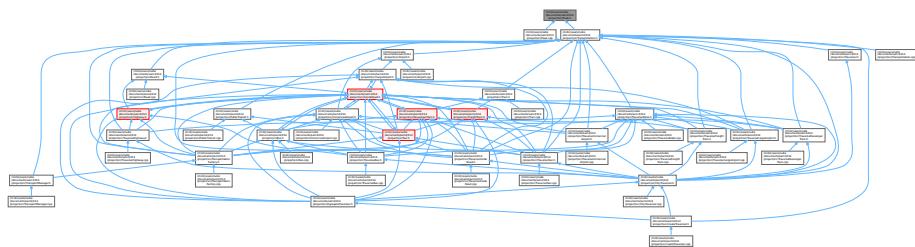
5.127 /mnt/c/users/rudie/documents/sem2/214/project/src/Peak.h File Reference

Header file for the [Peak](#) class, which inherits from [TrafficFlow](#).

```
#include "TrafficFlow.h"  
Include dependency graph for Peak.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Peak](#)

A class representing peak traffic flow.

5.127.1 Detailed Description

Header file for the [Peak](#) class, which inherits from [TrafficFlow](#).

Version

1.0

This file contains the declaration of the [Peak](#) class, which represents a specific type of traffic flow with a predefined state and traffic flow value.

Date

04/10/2024

5.128 Peak.h

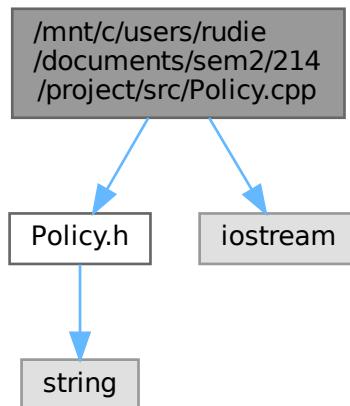
[Go to the documentation of this file.](#)

```
00001
00011 #ifndef PEAK_H
00012 #define PEAK_H
00013
00014 #include "TrafficFlow.h"
00015
00023 class Peak : public TrafficFlow {
00024     private:
00025         char state = 'P';
00026         float trafficFlow = 0.8;
00027
00028     public:
00034         float getTrafficFlow();
00035
00041         char getState();
00042 };
00043
00044 #endif // PEAK_H
```

5.129 /mnt/c/users/rudie/documents/sem2/214/project/src/Policy.cpp File Reference

Implementation of the [Policy](#) class.

```
#include "Policy.h"
#include <iostream>
Include dependency graph for Policy.cpp:
```



5.129.1 Detailed Description

Implementation of the [Policy](#) class.

Version

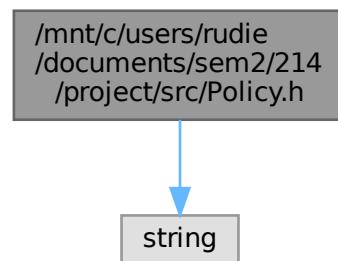
1.0

This file contains the implementation of the [Policy](#) class, which represents a policy with specific attributes and behaviors.

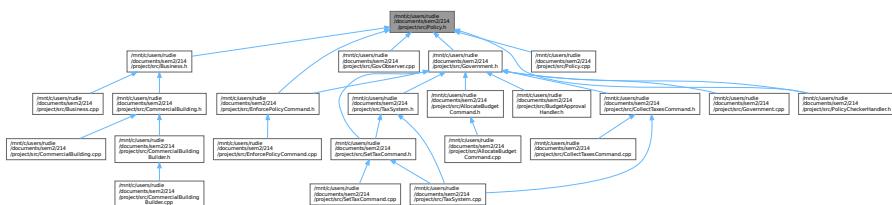
5.130 /mnt/c/users/rudie/documents/sem2/214/project/src/Policy.h File Reference

Definition of the [Policy](#) class.

```
#include <string>
Include dependency graph for Policy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Policy](#)

Represents a policy with specific attributes and behaviors.

5.130.1 Detailed Description

Definition of the [Policy](#) class.

Version

1.0

This file contains the definition of the [Policy](#) class, which represents a policy with specific attributes and behaviors.

5.131 Policy.h

[Go to the documentation of this file.](#)

```

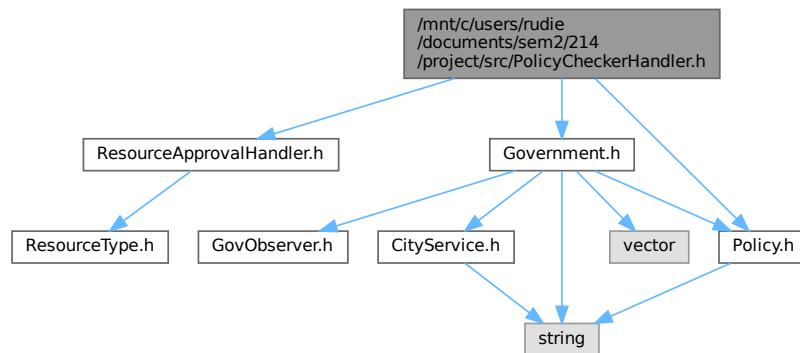
00001
00010 #ifndef POLICY_H
00011 #define POLICY_H
00012
00013 #include <string>
00014
00022 class Policy {
00023
00024 private:
00025     std::string policyName;
00026     std::string impactLevel;
00027
00028 public:
00032     Policy();
00033
00039     Policy(const std::string& name, const std::string& impact);
00040
00046     void implement();
00047
00053     void revoke();
00054
00059     std::string getPolicyName() const;
00060
00065     std::string getImpactLevel() const;
00066 };
00067
00068 #endif // POLICY_H

```

5.132 /mnt/c/users/rudie/documents/sem2/214/project/src/PolicyCheckerHandler.h File Reference

Definition of the [PolicyCheckerHandler](#) class.

```
#include "ResourceApprovalHandler.h"
#include "Government.h"
#include "Policy.h"
Include dependency graph for PolicyCheckerHandler.h:
```



Classes

- class [PolicyCheckerHandler](#)

Handles policy enforcement checks for resource approval requests.

5.132.1 Detailed Description

Definition of the [PolicyCheckerHandler](#) class.

Version

1.0

This file contains the definition of the [PolicyCheckerHandler](#) class, which handles policy enforcement checks for resource approval requests.

5.133 PolicyCheckerHandler.h

[Go to the documentation of this file.](#)

```

00001
00010 #ifndef POLICYCHECKHANDLER_H
00011 #define POLICYCHECKHANDLER_H
00012
00013 #include "ResourceApprovalHandler.h"
00014 #include "Government.h"
00015 #include "Policy.h"
00016
00024 class PolicyCheckerHandler : public ResourceApprovalHandler {
00025 private:
00026     Government* government;
00027     bool policyEnforced;
00028
00029 public:
00034     PolicyCheckerHandler(Government* gov) : government(gov), policyEnforced(false) {}
00035
00042     bool handleRequest(ResourceType type, int quantity) override {
00043         Policy policy(type, quantity);
00044         government->enforcePolicy(policy);
00045         if (isPolicyEnforced()) {
00046             return ResourceApprovalHandler::handleRequest(type, quantity);
00047         }
00048         return false;
00049     }
00050
00055     bool isPolicyEnforced() const {
00056         return policyEnforced;
00057     }
00058
00063     void setPolicyEnforced(bool enforced) {
00064         policyEnforced = enforced;
00065     }
00066 };
00067
00068 #endif // POLICYCHECKHANDLER_H

```

5.134 /mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.cpp File Reference

Implementation of the [PopulationManager](#) class.

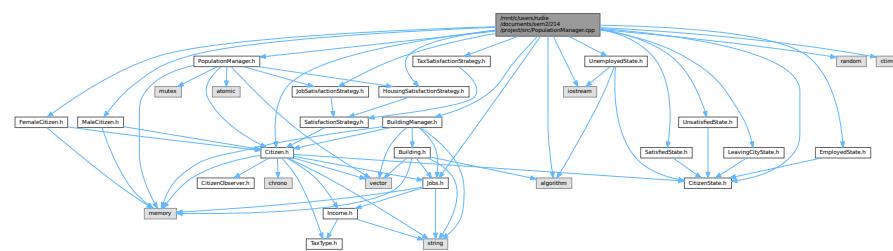
```

#include "PopulationManager.h"
#include "Citizen.h"
#include "CitizenState.h"
#include "EmployedState.h"
#include "UnemployedState.h"
#include "SatisfiedState.h"
#include "UnsatisfiedState.h"
#include "LeavingCityState.h"

```

```
#include "MaleCitizen.h"
#include "FemaleCitizen.h"
#include "JobSatisfactionStrategy.h"
#include "HousingSatisfactionStrategy.h"
#include "TaxSatisfactionStrategy.h"
#include "BuildingManager.h"
#include "Jobs.h"
#include <iostream>
#include <algorithm>
#include <random>
#include <ctime>
```

Include dependency graph for PopulationManager.cpp:



5.134.1 Detailed Description

Implementation of the [PopulationManager](#) class.

Version

1.0

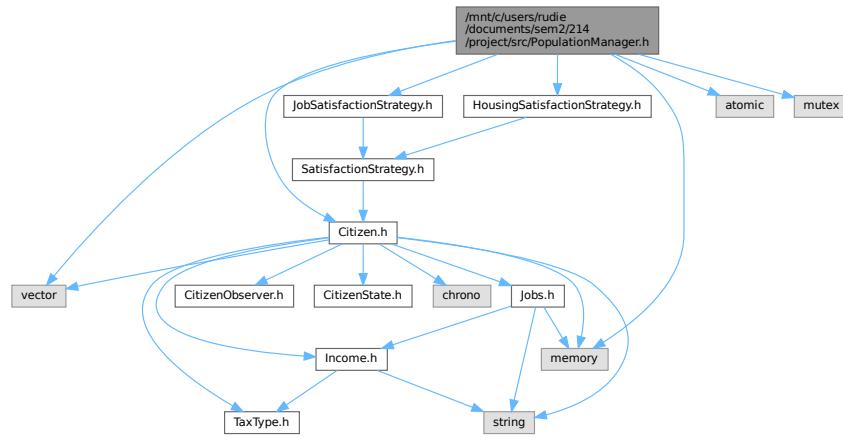
This file contains the implementation of the [PopulationManager](#) class, which manages the population of citizens, including adding, removing, and updating citizens.

5.135 /mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.h File Reference

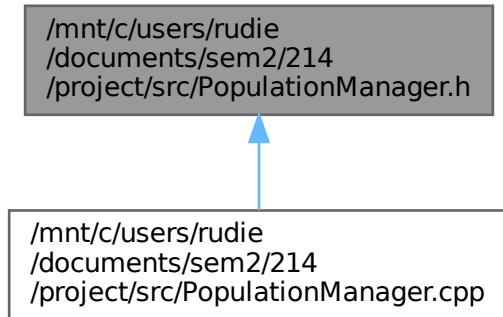
Definition of the [PopulationManager](#) class.

```
#include <vector>
#include <memory>
#include <atomic>
#include <mutex>
#include "Citizen.h"
#include "JobSatisfactionStrategy.h"
```

```
#include "HousingSatisfactionStrategy.h"
Include dependency graph for PopulationManager.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PopulationManager](#)
Manages the population of citizens.

5.135.1 Detailed Description

Definition of the [PopulationManager](#) class.

Version

1.0

This file contains the definition of the [PopulationManager](#) class, which manages the population of citizens, including adding, removing, and updating citizens.

5.136 PopulationManager.h

[Go to the documentation of this file.](#)

```

00001
00010 #ifndef POPULATIONMANAGER_H
00011 #define POPULATIONMANAGER_H
00012
00013 #include <vector>
00014 #include <memory>
00015 #include <atomic>
00016 #include <mutex>
00017 #include "Citizen.h"
00018 #include "JobSatisfactionStrategy.h"
00019 #include "HousingSatisfactionStrategy.h"
00020
00021 class BuildingManager;
00022
00030 class PopulationManager {
00031 private:
00032     std::vector<std::shared_ptr<Citizen>> citizens;
00033     std::atomic<bool> programRunning;
00034     std::mutex citizenMutex;
00035     std::shared_ptr<JobSatisfactionStrategy> jobSatisfaction;
00036     std::shared_ptr<HousingSatisfactionStrategy> housingSatisfaction;
00037
00038 public:
00044     PopulationManager();
00045
00050     void findJobsForUnemployedCitizens(BuildingManager& buildingManager);
00051
00055     void updateCitizensAge();
00056
00061     void addCitizen(std::shared_ptr<Citizen> citizen);
00062
00066     void checkCitizenStates();
00067
00071     void simulatePopulationGrowth();
00072
00076     void removeLeavingCitizens();
00077
00081     void manageRelationships();
00082
00087     int getPopulation();
00088
00093     const std::vector<std::shared_ptr<Citizen>>& getCitizens() const;
00094
00098     void updateCitizensSatisfaction();
00099
00100 private:
00104     void addRandomCitizen();
00105
00109     void removeRandomCitizen();
00110 };
00111
00112 #endif // POPULATIONMANAGER_H

```

5.137 /mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.cpp File Reference

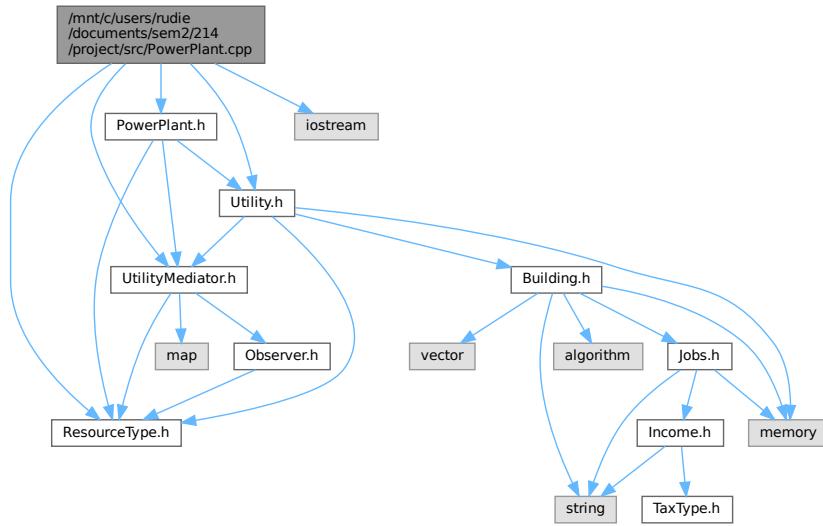
Implementation of the [PowerPlant](#) class.

```

#include "PowerPlant.h"
#include "Utility.h"
#include "UtilityMediator.h"
#include "ResourceType.h"
#include <iostream>

```

Include dependency graph for PowerPlant.cpp:



5.137.1 Detailed Description

Implementation of the [PowerPlant](#) class.

Version

1.0

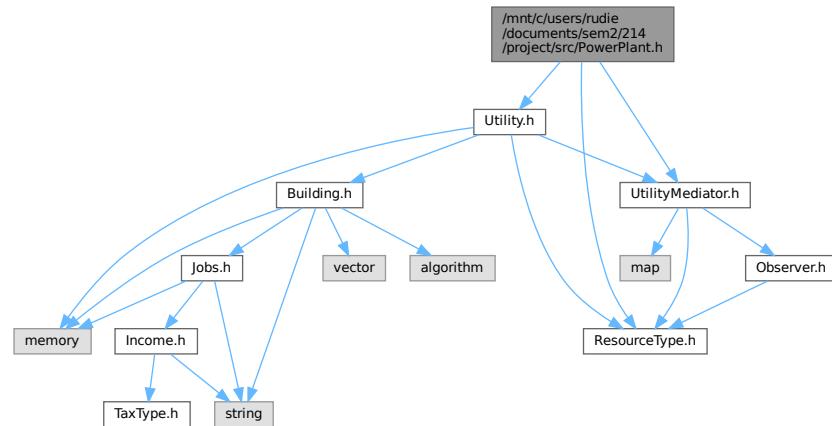
This file contains the implementation of the [PowerPlant](#) class, which represents a power plant that supplies power to buildings through a mediator.

5.138 /mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.h File Reference

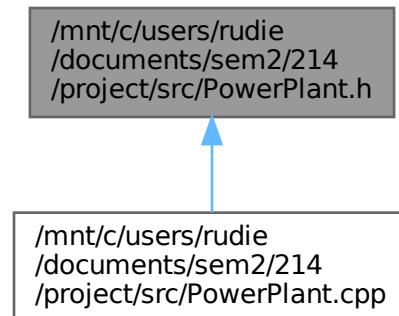
Definition of the [PowerPlant](#) class.

```
#include "Utility.h"
#include "UtilityMediator.h"
```

```
#include "ResourceType.h"  
Include dependency graph for PowerPlant.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **PowerPlant**
Represents a power plant that supplies power to buildings.

5.138.1 Detailed Description

Definition of the [PowerPlant](#) class.

Version

1.0

This file contains the definition of the [PowerPlant](#) class, which represents a power plant that supplies power to buildings through a mediator.

5.139 PowerPlant.h

[Go to the documentation of this file.](#)

```

00001
00010 #ifndef __POWERPLANT_H
00011 #define __POWERPLANT_H
00012
00013 #include "Utility.h"
00014 #include "UtilityMediator.h"
00015 #include "ResourceType.h"
00016
00017 class UtilityMediator;
00018
00026 class PowerPlant : public Utility {
00027     private:
00028         UtilityMediator* mediator;
00029
00030     public:
00035         PowerPlant(UtilityMediator* mediator);
00036
00041         void registerBuilding(Building* building) override;
00042
00047         void supplyResources(Building* building) override;
00048
00049         //void generateElectricity(Building *building);
00050
00051         //void adjustForCitizen(Citizen* citizen) override;
00052         //PowerPlant();
00053 };
00054
00055 #endif // __POWERPLANT_H

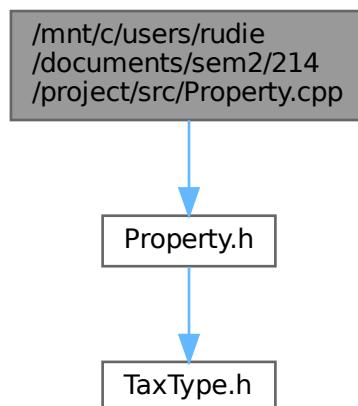
```

5.140 /mnt/c/users/rudie/documents/sem2/214/project/src/Property.cpp File Reference

Implementation of the [Property](#) class.

#include "Property.h"

Include dependency graph for Property.cpp:



5.140.1 Detailed Description

Implementation of the [Property](#) class.

Version

1.0

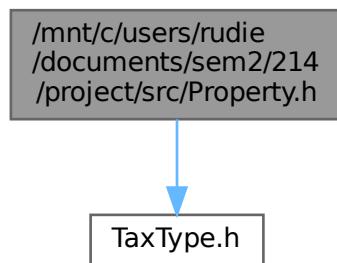
Date

2024-11-04 This file contains the implementation of the [Property](#) class, which represents a property with specific attributes and behaviors related to property tax calculation.

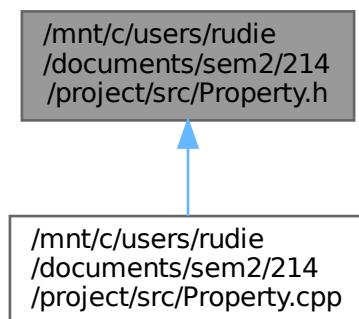
5.141 /mnt/c/users/rudie/documents/sem2/214/project/src/Property.h File Reference

Definition of the [Property](#) class.

```
#include "TaxType.h"  
Include dependency graph for Property.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Property](#)

Represents a property with specific attributes and behaviors related to property tax calculation.

5.141.1 Detailed Description

Definition of the [Property](#) class.

Version

1.0

Date

2024-11-04 This file contains the definition of the [Property](#) class, which represents a property with specific attributes and behaviors related to property tax calculation.

5.142 Property.h

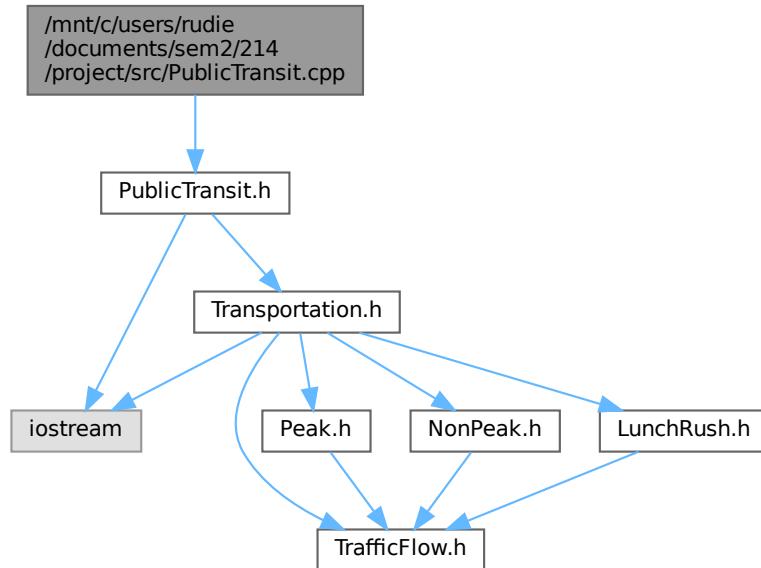
[Go to the documentation of this file.](#)

```
00001
00010 #ifndef PROPERTY_H
00011 #define PROPERTY_H
00012
00013 #include "TaxType.h"
00014
00022 class Property : public TaxType {
00023 private:
00024     double propertyTax;
00025     double municipalLevy;
00026     double additionalFees;
00027
00028 public:
00035     Property(double rate, double levy, double fees);
00036
00041     void setTax(double rate) override;
00042
00048     double calculateTax(double PropertyValue) override;
00049
00054     void setMunicipalLevy(double levy);
00055
00060     void setAdditionalFees(double fees);
00061 };
00062
00063 #endif // PROPERTY_H
```

5.143 /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.cpp File Reference

Implementation of the [PublicTransit](#) class.

```
#include "PublicTransit.h"
Include dependency graph for PublicTransit.cpp:
```



5.143.1 Detailed Description

Implementation of the [PublicTransit](#) class.

Version

1.0

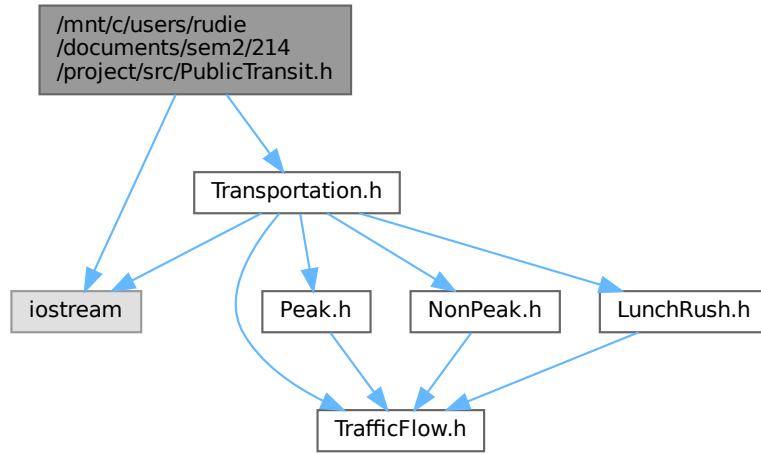
Date

2024-11-04 This file contains the implementation of the [PublicTransit](#) class, which represents a public transit system with specific attributes and behaviors.

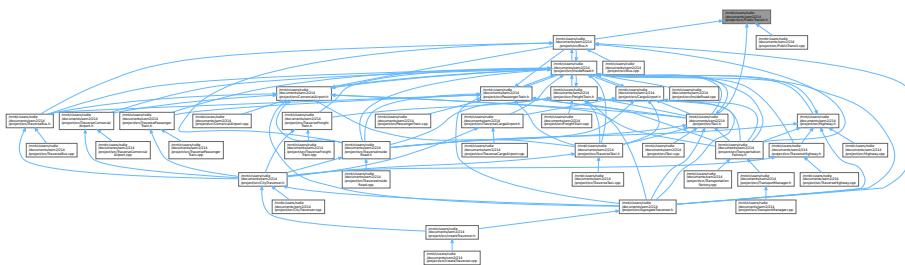
5.144 /mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.h File Reference

Header file for the [PublicTransit](#) class.

```
#include <iostream>
#include "Transportation.h"
Include dependency graph for PublicTransit.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [PublicTransit](#)
A class representing public transit transportation.

5.144.1 Detailed Description

Header file for the [PublicTransit](#) class.

Version

1.0

This file contains the declaration of the [PublicTransit](#) class, which is a subclass of the [Transportation](#) class. It represents a public transit system with a specific route.

Date

2024-11-04

5.145 PublicTransit.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef PUBLICTRANSIT_H
00013 #define PUBLICTRANSIT_H
00014
00015 #include <iostream>
00016 #include "Transportation.h"
00017
00025 class PublicTransit : public Transportation {
00026     private:
00027         std::string route;
00028
00029     public:
00037         PublicTransit(char state, std::string route, char type);
00038
00044         float calculateCommute();
00045
00051         std::string getRoute();
00052     };
00053
00054 #endif // PUBLICTRANSIT_H

```

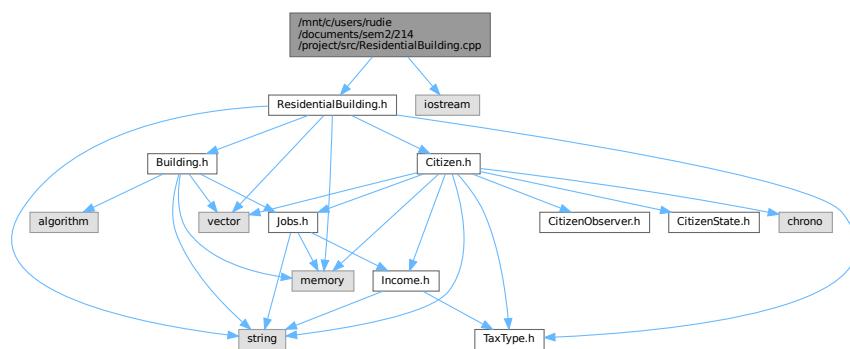
5.146 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.cpp File Reference

Implementation of the [ResidentialBuilding](#) class.

```
#include "ResidentialBuilding.h"
```

```
#include <iostream>
```

Include dependency graph for ResidentialBuilding.cpp:



5.146.1 Detailed Description

Implementation of the [ResidentialBuilding](#) class.

Version

1.0

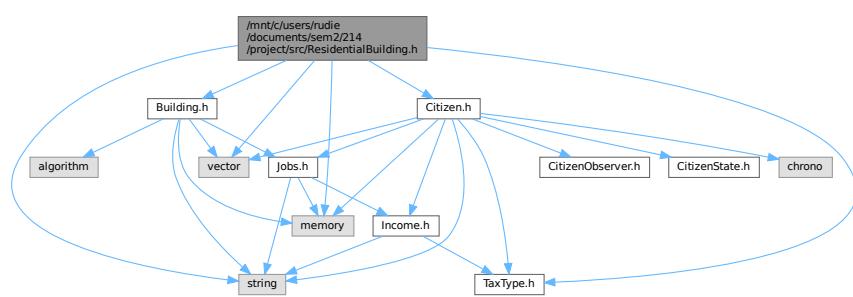
This file contains the implementation of the [ResidentialBuilding](#) class, which represents a residential building with specific attributes and behaviors.

5.147 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.h File Reference

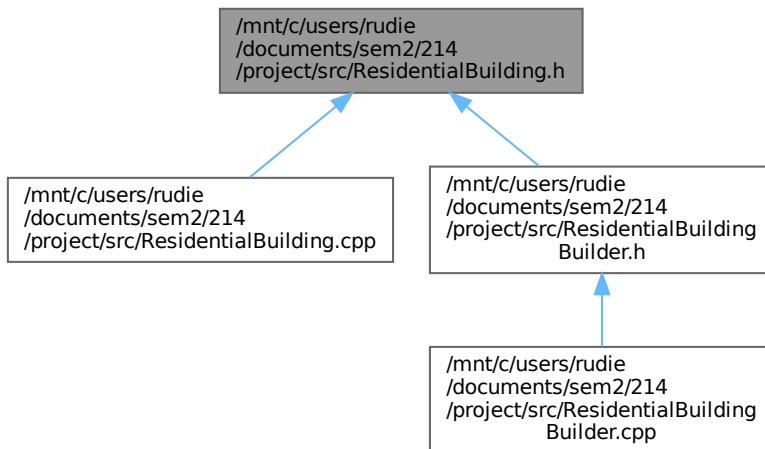
Header file for the [ResidentialBuilding](#) class.

```
#include <string>
#include <vector>
#include <memory>
#include "Building.h"
#include "Citizen.h"
#include "TaxType.h"
```

Include dependency graph for ResidentialBuilding.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ResidentialBuilding](#)

Represents a residential building in the simulation.

5.147.1 Detailed Description

Header file for the [ResidentialBuilding](#) class.

This file contains the definition of the [ResidentialBuilding](#) class, which represents a residential building in the simulation. It inherits from the [Building](#) class and includes additional attributes and methods specific to residential buildings.

Version

1.0

Date

2024-11-04

5.148 ResidentialBuilding.h

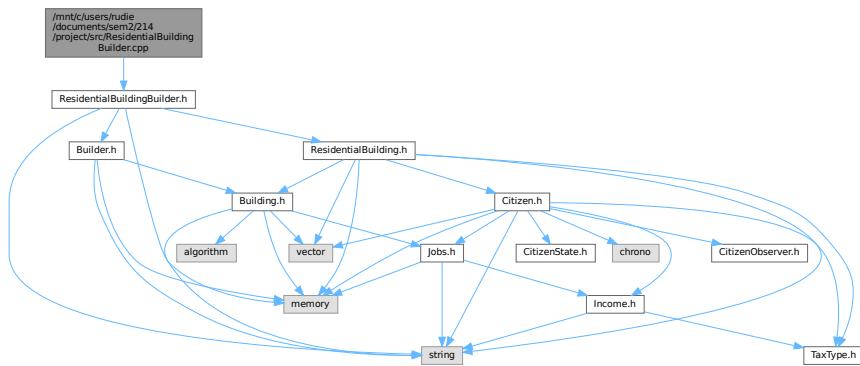
[Go to the documentation of this file.](#)

```
00001  
00014 #ifndef RESIDENTIALBUILDING_H  
00015 #define RESIDENTIALBUILDING_H  
00016  
00017 #include <string>  
00018 #include <vector>  
00019 #include <memory>  
00020 #include "Building.h"  
00021 #include "Citizen.h"  
00022 #include "TaxType.h"  
00023  
00033 class ResidentialBuilding : public Building {  
00034 public:  
00048     ResidentialBuilding(const std::string& name, float area, int floors,  
00049                             int capacity, float citizenSatisfaction,  
00050                             float economicGrowth, float resourceConsumption,  
00051                             int residentialUnits, float comfortLevel);  
00052  
00058     std::string getType() const override;  
00059  
00063     void updateImpacts() override;  
00064  
00070     void upgradeComfort(float comfort);  
00071  
00075     void construct() override;  
00076  
00083     double payTaxes(TaxType* taxType);  
00084  
00090     double calculatePropertyTax();  
00091  
00097     void addResidents(Citizen* citizen);  
00098  
00102     void undoCollectTaxes();  
00103  
00109     int getResidentialUnits() const;  
00110  
00111 private:  
00112     int residentialUnits;  
00113     float comfortLevel;  
00114     std::string bType;  
00115     double totalTaxCollected;  
00116     double propertyTax;  
00117     double totalPropertyTaxCollected;  
00118     double totalIncomeTaxCollected;  
00119  
00120 protected:  
00124     void calculateEconomicImpact() override;  
00125  
00129     void calculateResourceConsumption() override;  
00130  
00134     void calculateSatisfactionImpact() override;  
00135  
00136     std::vector<Citizen*> residents;  
00137 };  
00138  
00139 #endif // RESIDENTIALBUILDING_H
```

5.149 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.cpp File Reference

Implementation file for the [ResidentialBuildingBuilder](#) class.

```
#include "ResidentialBuildingBuilder.h"
Include dependency graph for ResidentialBuildingBuilder.cpp:
```



5.149.1 Detailed Description

Implementation file for the [ResidentialBuildingBuilder](#) class.

This file contains the implementation of the [ResidentialBuildingBuilder](#) class, which is used to construct [ResidentialBuilding](#) objects with specific attributes.

Version

1.0

Date

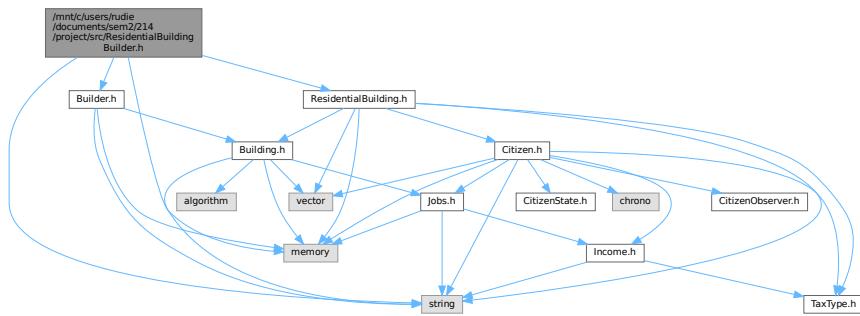
2024-11-04

5.150 /mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.h File Reference

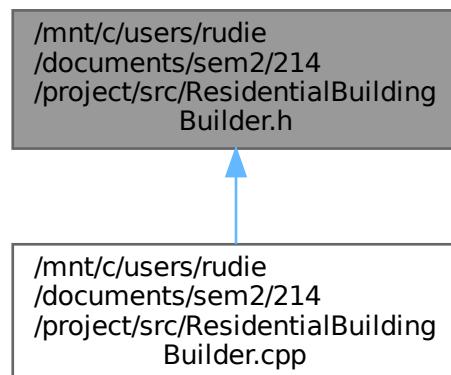
Header file for the [ResidentialBuildingBuilder](#) class.

```
#include "Builder.h"
#include "ResidentialBuilding.h"
#include <string>
```

```
#include <memory>
Include dependency graph for ResidentialBuildingBuilder.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ResidentialBuildingBuilder](#)
Builder class for constructing *ResidentialBuilding* objects.

5.150.1 Detailed Description

Header file for the `ResidentialBuildingBuilder` class.

This file contains the definition of the `ResidentialBuildingBuilder` class, which is used to construct `ResidentialBuilding` objects with specific attributes.

Version

1.0

Date

2024-11-04

5.151 ResidentialBuildingBuilder.h

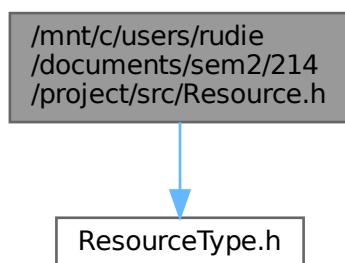
[Go to the documentation of this file.](#)

```
00001
00012 #ifndef RESIDENTIALBUILDINGBUILDER_H
00013 #define RESIDENTIALBUILDINGBUILDER_H
00014
00015 #include "Builder.h"
00016 #include "ResidentialBuilding.h"
00017 #include <string>
00018 #include <memory>
00019
00020 using namespace std;
00021
00029 class ResidentialBuildingBuilder : public Builder {
00030
00031 private:
00032     int residentialUnit = 0;
00033     float comfort = 0.0f;
00034
00035 public:
00039     ResidentialBuildingBuilder();
00040
00047     ResidentialBuildingBuilder& setResidentialUnit(int unit);
00048
00055     ResidentialBuildingBuilder& setComfort(float comfort);
00056
00062     int getResidentialUnit();
00063
00069     float getComfort();
00070
00076     std::unique_ptr<Building> build() override;
00077 };
00078
00079 #endif // RESIDENTIALBUILDINGBUILDER_H
```

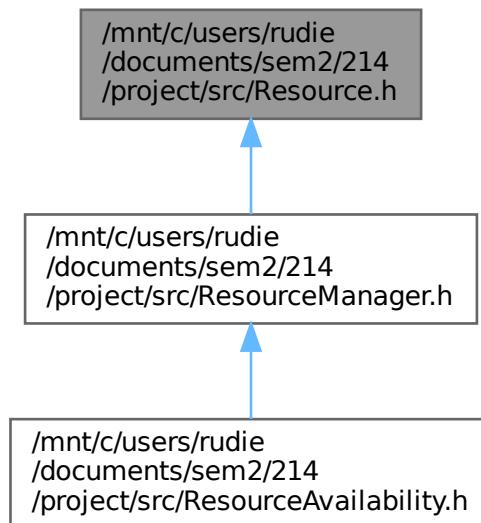
5.152 /mnt/c/users/rudie/documents/sem2/214/project/src/Resource.h File Reference

Header file for the [Resource](#) class.

```
#include "ResourceType.h"
Include dependency graph for Resource.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Resource](#)
Represents a resource with a specific type and quantity.

5.152.1 Detailed Description

Header file for the [Resource](#) class.

This file contains the definition of the [Resource](#) class, which represents a resource with a specific type and quantity. It provides methods to allocate and release the resource.

Version

1.0

Date

2024-11-04

5.153 Resource.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef RESOURCE_H
00013 #define RESOURCE_H
00014
00015 #include "ResourceType.h"
00016
00023 class Resource {
00024 private:
00025     ResourceType type;
00026     int quantity;
00027
00028 public:
00035     Resource(ResourceType resourceType, int initialQuantity)
00036         : type(resourceType), quantity(initialQuantity) {}
00037
00044     bool allocate(int amount) {
00045         if (quantity >= amount) {
00046             quantity -= amount;
00047             return true;
00048         }
00049         return false;
00050     }
00051
00057     void release(int amount) { quantity += amount; }
00058
00064     ResourceType getType() const { return type; }
00065
00071     int getQuantity() const { return quantity; }
00072 };
00073
00074 #endif // RESOURCE_H

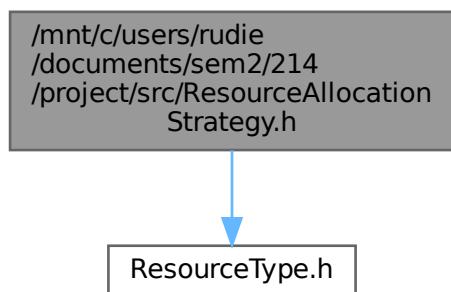
```

5.154 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAllocationStrategy.h File Reference

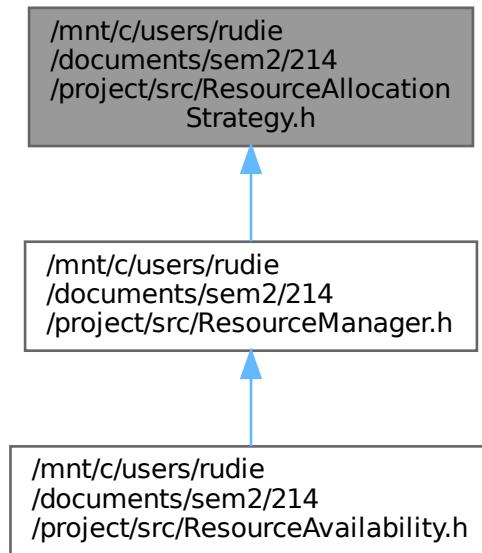
Header file for resource allocation strategy classes.

```
#include "ResourceType.h"
```

Include dependency graph for ResourceAllocationStrategy.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ResourceAllocationStrategy](#)
Interface for resource allocation strategies.
- class [PriorityDistributionStrategy](#)
Strategy for priority-based resource allocation.
- class [EqualDistributionStrategy](#)
Strategy for equal distribution of resources.

5.154.1 Detailed Description

Header file for resource allocation strategy classes.

This file contains the definition of the [ResourceAllocationStrategy](#) interface and its derived classes, which provide different strategies for allocating resources.

Version

1.0

Date

2024-11-04

5.155 ResourceAllocationStrategy.h

[Go to the documentation of this file.](#)

```

00012 #ifndef RESOURCEALLOCATIONSTRATEGY_H
00013 #define RESOURCEALLOCATIONSTRATEGY_H
00014
00015 #include "ResourceType.h"
00016
00023 class ResourceAllocationStrategy {
00024 public:
00032     virtual bool allocate(ResourceType type, int quantity) = 0;
00033 };
00034
00042 class PriorityDistributionStrategy : public ResourceAllocationStrategy {
00043 public:
00051     bool allocate(ResourceType type, int quantity) override {
00052         // Logic for priority-based allocation
00053         return true;
00054     }
00055 };
00056
00064 class EqualDistributionStrategy : public ResourceAllocationStrategy {
00065 public:
00073     bool allocate(ResourceType type, int quantity) override {
00074         // Logic for equal distribution
00075         return true;
00076     }
00077 };
00078
00079 #endif // RESOURCEALLOCATIONSTRATEGY_H

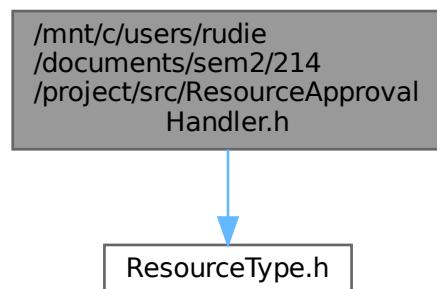
```

5.156 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceApprovalHandler.h File Reference

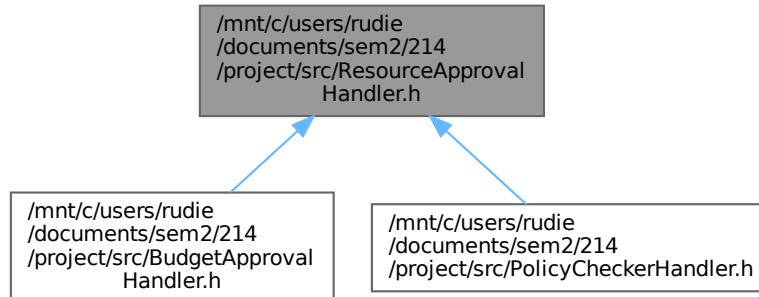
Header file for resource approval handler classes.

```
#include "ResourceType.h"
```

Include dependency graph for ResourceApprovalHandler.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ResourceApprovalHandler](#)
Base class for handling resource approval requests.
- class [BudgetApprovalHandler](#)
Handles budget approval requests.

5.156.1 Detailed Description

Header file for resource approval handler classes.

This file contains the definition of the [ResourceApprovalHandler](#) class and its derived classes, which provide a chain of responsibility pattern for approving resource requests.

Version

1.0

Date

2024-11-04

5.157 ResourceApprovalHandler.h

[Go to the documentation of this file.](#)

```

00001
00013 #ifndef RESOURCEAPPROVALHANDLER_H
00014 #define RESOURCEAPPROVALHANDLER_H
00015
00016 #include "ResourceType.h"
00017
00025 class ResourceApprovalHandler {
00026 protected:
00027     ResourceApprovalHandler* nextHandler;
00028
00029 public:
00033     ResourceApprovalHandler() : nextHandler(nullptr) {}
  
```

```

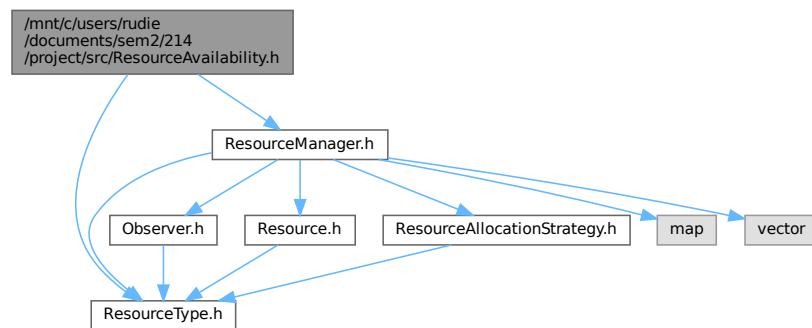
00034     virtual ~ResourceApprovalHandler() {}
00038
00039
00045     void setNextHandler(ResourceApprovalHandler* handler) {
00046         nextHandler = handler;
00047     }
00048
00056     virtual bool handleRequest(ResourceType type, int quantity) {
00057         if (nextHandler) return nextHandler->handleRequest(type, quantity);
00058         return false;
00059     }
00060 };
00061
00068 class BudgetApprovalHandler : public ResourceApprovalHandler {
00069 public:
00077     bool handleRequest(ResourceType type, int quantity) override {
00078         if (type == ResourceType::Budget) {
00079             // Check if budget allows for this request
00080             return ResourceApprovalHandler::handleRequest(type, quantity);
00081         }
00082         return false;
00083     }
00084 };
00085
00086 #endif // RESOURCEAPPROVALHANDLER_H

```

5.158 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAvailability.h File Reference

Header file for the [ResourceAvailability](#) class.

```
#include "ResourceType.h"
#include "ResourceManager.h"
Include dependency graph for ResourceAvailability.h:
```



Classes

- class [ResourceAvailability](#)
Checks the availability of a specific resource type and quantity.

5.158.1 Detailed Description

Header file for the [ResourceAvailability](#) class.

This file contains the definition of the [ResourceAvailability](#) class, which checks the availability of a specific resource type and quantity.

Version

1.0

Date

2024-11-04

5.159 ResourceAvailability.h

[Go to the documentation of this file.](#)

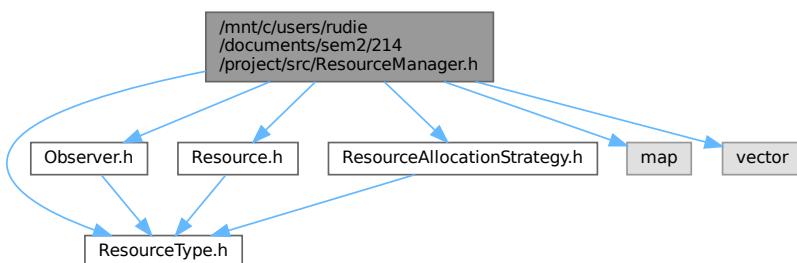
```
00001
00012 #ifndef RESOURCEAVAILABILITY_H
00013 #define RESOURCEAVAILABILITY_H
00014
00015 #include "ResourceType.h"
00016 #include "ResourceManager.h"
00017
00025 class ResourceAvailability {
00026 private:
00027     ResourceManager* resourceManager;
00028
00029 public:
00035     ResourceAvailability(ResourceManager* manager) : resourceManager(manager) {}
00036
00044     bool checkAvailability(ResourceType type, int quantity) const {
00045         Resource* resource = resourceManager->getResource(type);
00046         if (resource) {
00047             return resource->getQuantity() >= quantity;
00048         }
00049         return false; // Resource type not found
00050     }
00051 };
00052
00053 #endif // RESOURCEAVAILABILITY_H
```

5.160 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceManager.h File Reference

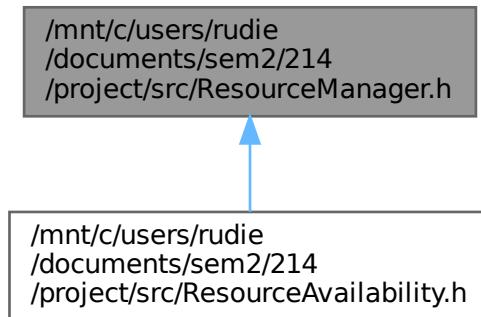
Header file for the [ResourceManager](#) class.

```
#include "ResourceType.h"
#include "Observer.h"
#include "Resource.h"
#include "ResourceAllocationStrategy.h"
#include <map>
#include <vector>
```

Include dependency graph for ResourceManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ResourceManager](#)

Manages resources, handles resource allocation and release, and notifies observers.

5.160.1 Detailed Description

Header file for the [ResourceManager](#) class.

This file contains the definition of the [ResourceManager](#) class, which manages resources, handles resource allocation and release, and notifies observers about resource changes.

Version

1.0

Date

2024-11-04

5.161 ResourceManager.h

[Go to the documentation of this file.](#)

```
00001\n00012 #ifndef RESOURCEMANAGER_H\n00013 #define RESOURCEMANAGER_H\n00014\n00015 #include "ResourceType.h"\n00016 #include "Observer.h"\n00017 #include "Resource.h"\n00018 #include "ResourceAllocationStrategy.h"\n00019 #include <map>\n00020 #include <vector>\n00021\n00029 class ResourceManager {\n
```

```

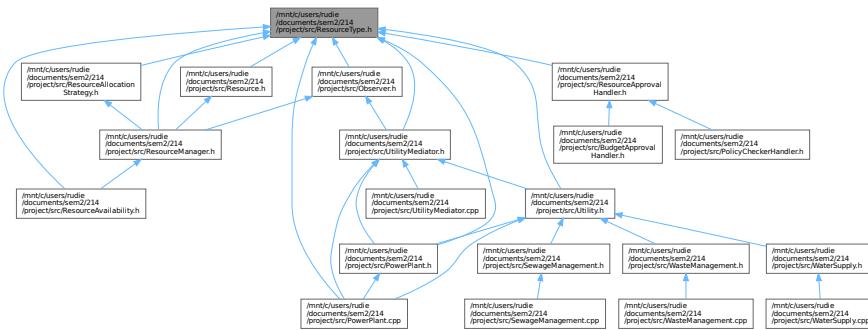
00030 private:
00031     int budget;
00032     std::map<ResourceType, Resource*> resources;
00033     std::map<ResourceType, int> resourceCosts;
00034     ResourceAllocationStrategy* allocationStrategy;
00035     std::vector<Observer*> observers;
00036
00037 public:
00043     ResourceManager(int initialBudget) : budget(initialBudget) {}
00044
00052     bool allocateResources(ResourceType type, int quantity) {
00053         if (resources.count(type) && budget >= resourceCosts[type] * quantity &&
00054             resources[type]->allocate(quantity)) {
00055             budget -= resourceCosts[type] * quantity;
00056             notifyObservers(type, quantity);
00057             return true;
00058         }
00059     }
00060
00067     void releaseResources(ResourceType type, int quantity) {
00068         if (resources.count(type)) {
00069             resources[type]->release(quantity);
00070             notifyObservers(type, -quantity); // Negative to indicate release
00071         }
00072     }
00073
00079     void setAllocationStrategy(ResourceAllocationStrategy* strategy) {
00080         allocationStrategy = strategy;
00081     }
00082
00088     int getBudget() const { return budget; }
00089
00095     void addObserver(Observer* observer) {
00096         observers.push_back(observer);
00097     }
00098
00105     Resource* getResource(ResourceType type) const {
00106         auto it = resources.find(type);
00107         return (it != resources.end()) ? it->second : nullptr;
00108     }
00109
00116     void addResource(ResourceType type, Resource* resource) {
00117         if (resources.count(type) == 0) {
00118             resources[type] = resource;
00119         }
00120     }
00121
00122 private:
00129     void notifyObservers(ResourceType type, int quantity) {
00130         for (auto obs : observers) {
00131             obs->update(type, quantity);
00132         }
00133     }
00134 };
00135
00136 #endif // RESOURCEMANAGER_H

```

5.162 /mnt/c/users/rudie/documents/sem2/214/project/src/ResourceType.h File Reference

Header file for the ResourceType enumeration.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum class **ResourceType** {
 Water , Power , Materials , Budget ,
 Waste , Sewage , Recycling }

Enumeration of different resource types.

5.162.1 Detailed Description

Header file for the ResourceType enumeration.

This file contains the definition of the ResourceType enumeration, which represents different types of resources that can be managed.

Version

1.0

Date

2024-11-04

5.162.2 Enumeration Type Documentation

5.162.2.1 ResourceType

```
enum class ResourceType [strong]
```

Enumeration of different resource types.

The ResourceType enumeration represents different types of resources that can be managed.

Enumerator

Water	Water resource.
Power	Power resource.
Materials	Materials resource (e.g., wood, steel, concrete).
Budget	Budget resource.
Waste	Waste resource.
Sewage	Sewage resource.

5.163 ResourceType.h

[Go to the documentation of this file.](#)

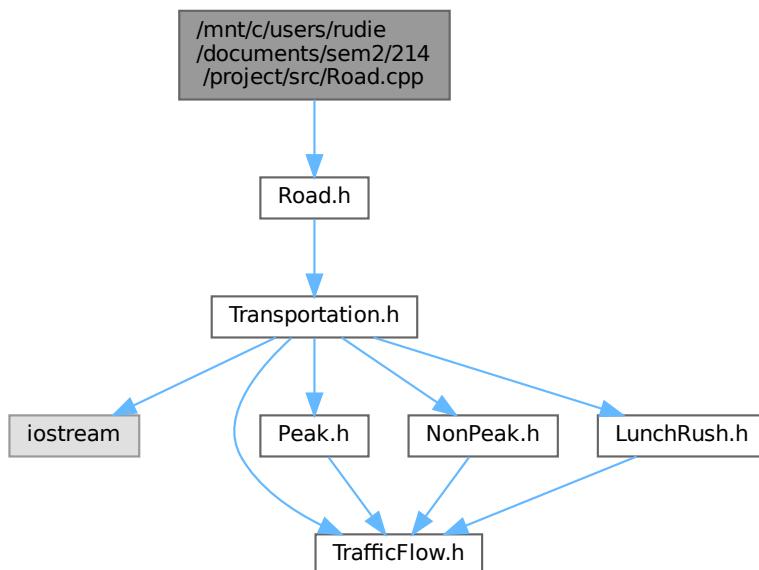
```
00001
00012 #ifndef RESOURCETYPE_H
00013 #define RESOURCETYPE_H
00014
00021 enum class ResourceType {
00022     Water,
00023     Power,
00024     Materials,
00025     Budget,
00026     Waste,
00027     Sewage,
00028     Recycling
00029 };
00030
00031 #endif // RESOURCETYPE_H
```

5.164 /mnt/c/users/rudie/documents/sem2/214/project/src/Road.cpp File Reference

Implementation file for the [Road](#) class.

```
#include "Road.h"
```

Include dependency graph for Road.cpp:



5.164.1 Detailed Description

Implementation file for the [Road](#) class.

This file contains the implementation of the [Road](#) class, which represents a road in the transportation system.

Version

1.0

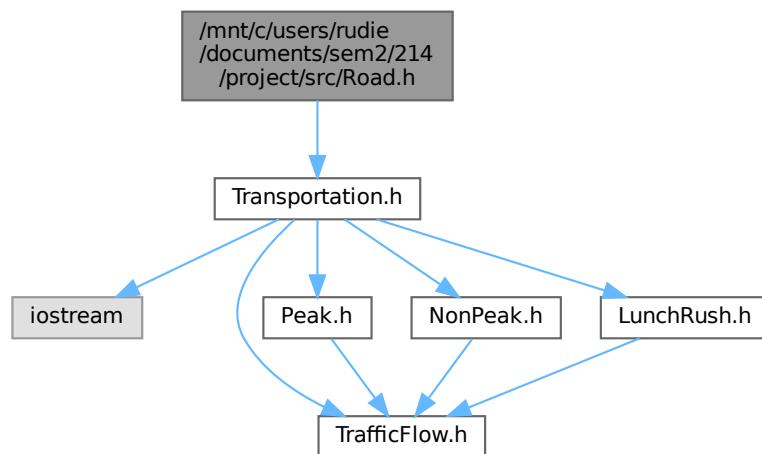
Date

2024-11-04

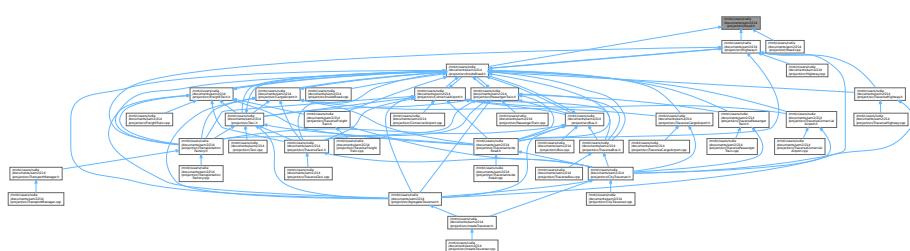
5.165 /mnt/c/users/rudie/documents/sem2/214/project/src/Road.h File Reference

Defines the [Road](#) class which inherits from the [Transportation](#) class.

```
#include "Transportation.h"
Include dependency graph for Road.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Road](#)

A class representing a road as a type of transportation.

5.165.1 Detailed Description

Defines the [Road](#) class which inherits from the [Transportation](#) class.

This file contains the declaration of the [Road](#) class, which represents a road as a type of transportation. It includes methods to calculate the commute and get the name of the road.

Version

1.0

Date

2024-11-04

5.166 Road.h

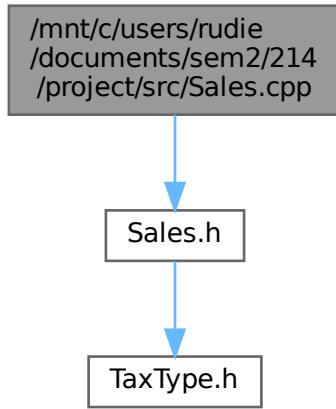
[Go to the documentation of this file.](#)

```
00001
00013 #ifndef ROAD_H
00014 #define ROAD_H
00015
00016 #include "Transportation.h"
00017
00025 class Road : public Transportation {
00026 private:
00027     std::string roadName;
00028
00029 public:
00037     Road(char state, std::string roadName, char type);
00038
00044     float calculateCommute();
00045
00051     std::string getName();
00052 }
00053
00054 #endif // ROAD_H
```

5.167 /mnt/c/users/rudie/documents/sem2/214/project/src/Sales.cpp File Reference

Implementation file for the [Sales](#) class.

```
#include "Sales.h"  
Include dependency graph for Sales.cpp:
```



5.167.1 Detailed Description

Implementation file for the [Sales](#) class.

This file contains the implementation of the [Sales](#) class, which represents a type of tax that includes a base sales tax, an environmental levy, and a service fee.

Version

1.0

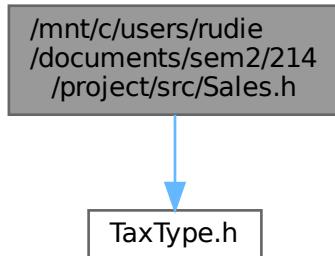
Date

2024-11-04

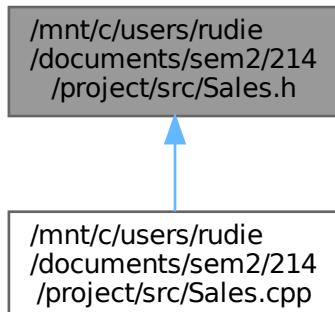
5.168 /mnt/c/users/rudie/documents/sem2/214/project/src/Sales.h File Reference

Header file for the [Sales](#) class.

```
#include "TaxType.h"
Include dependency graph for Sales.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Sales](#)

Represents a type of tax that includes a base sales tax, an environmental levy, and a service fee.

5.168.1 Detailed Description

Header file for the [Sales](#) class.

This file contains the definition of the [Sales](#) class, which represents a type of tax that includes a base sales tax, an environmental levy, and a service fee.

Version

1.0

Date

2024-11-04

5.169 Sales.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef SALES_H
00013 #define SALES_H
00014
00015 #include "TaxType.h"
00016
00024 class Sales : public TaxType {
00025 private:
00026     double salesTax;
00027     double environmentalLevy;
00028     double serviceFee;
00029
00030 public:
00038     Sales(double rate, double levy, double fee);
00039
00043     virtual ~Sales() override;
00044
00050     void setTax(double rate) override;
00051
00058     double calculateTax(double saleAmount) override;
00059
00065     void setEnvironmentalLevy(double levy);
00066
00072     void setServiceFee(double fee);
00073 };
00074
00075 #endif // SALES_H

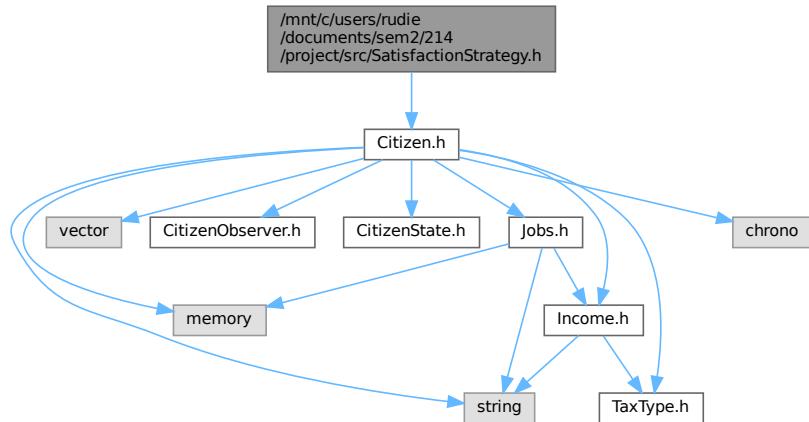
```

5.170 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfactionStrategy.h File Reference

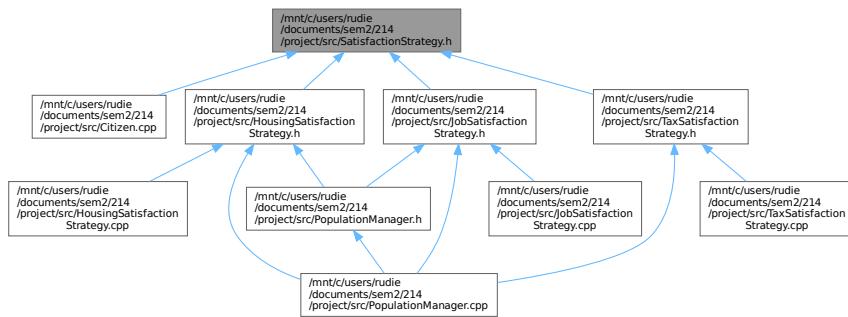
Header file for the [SatisfactionStrategy](#) class.

#include "Citizen.h"

Include dependency graph for SatisfactionStrategy.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SatisfactionStrategy](#)
Interface for calculating and updating citizen satisfaction.

5.170.1 Detailed Description

Header file for the [SatisfactionStrategy](#) class.

This file contains the definition of the [SatisfactionStrategy](#) class, which provides an interface for calculating and updating citizen satisfaction based on various factors such as job, housing, and tax changes.

Version

1.0

Date

2024-11-04

5.171 SatisfactionStrategy.h

[Go to the documentation of this file.](#)

```

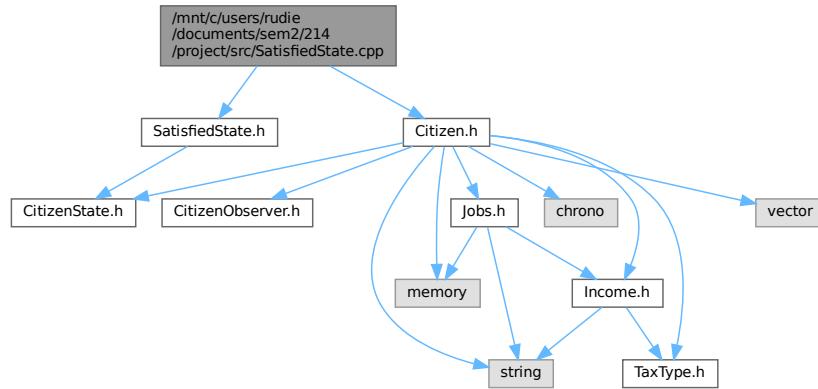
00001
00013 #ifndef SATISFACTIONSTRATEGY_H
00014 #define SATISFACTIONSTRATEGY_H
00015
00016 #include "Citizen.h"
00017
00025 class SatisfactionStrategy {
00026 public:
00030     virtual ~SatisfactionStrategy() = default;
00031
00038     virtual float calculateSatisfaction(const Citizen& citizen) = 0;
00039
00045     virtual void updateForJobChange(Citizen& citizen) = 0;
00046
00052     virtual void updateForHousingChange(Citizen& citizen) = 0;
00053
00059     virtual void updateForTaxChange(Citizen& citizen) = 0;
00060 };
00061
00062 #endif // SATISFACTIONSTRATEGY_H
  
```

5.172 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.cpp File Reference

Implementation file for the [SatisfiedState](#) class.

```
#include "SatisfiedState.h"
#include "Citizen.h"
```

Include dependency graph for SatisfiedState.cpp:



5.172.1 Detailed Description

Implementation file for the [SatisfiedState](#) class.

This file contains the implementation of the [SatisfiedState](#) class, which represents a state where a citizen is satisfied. It provides methods to handle the state and update the citizen's satisfaction level.

Version

1.0

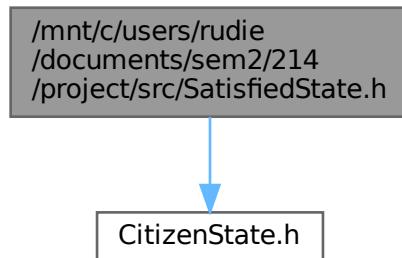
Date

2024-11-04

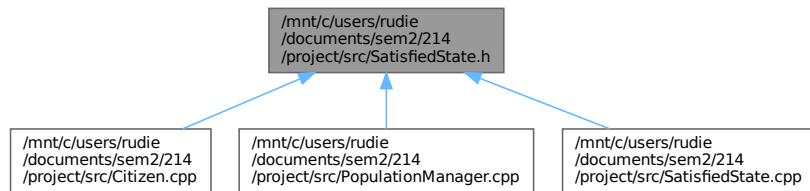
5.173 /mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.h File Reference

Header file for the [SatisfiedState](#) class.

```
#include "CitizenState.h"
Include dependency graph for SatisfiedState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [SatisfiedState](#)
Represents a state where a citizen is satisfied.

5.173.1 Detailed Description

Header file for the [SatisfiedState](#) class.

This file contains the definition of the [SatisfiedState](#) class, which represents a state where a citizen is satisfied. It provides methods to handle the state and update the citizen's satisfaction level.

Version

1.0

Date

2024-11-04

5.174 SatisfiedState.h

[Go to the documentation of this file.](#)

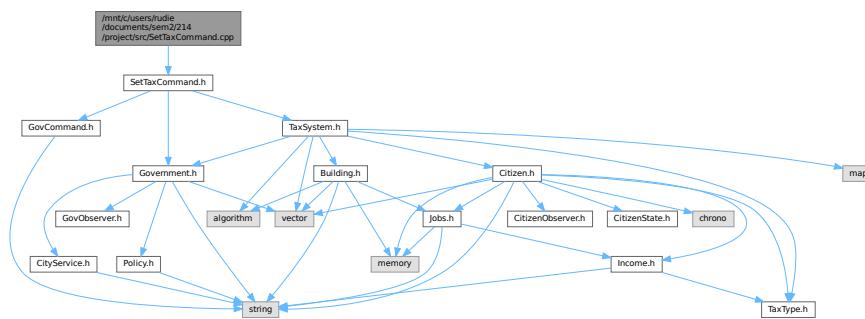
```
00001
00013 #ifndef SATISFIEDSTATE_H
00014 #define SATISFIEDSTATE_H
00015
00016 #include "CitizenState.h"
00017
00025 class SatisfiedState : public CitizenState {
00026 public:
00032     void handleState(Citizen& citizen) const override;
00033 };
00034
00035 #endif // SATISFIEDSTATE_H
```

5.175 /mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.cpp File Reference

Implementation of the [SetTaxCommand](#) class.

```
#include "SetTaxCommand.h"
```

Include dependency graph for SetTaxCommand.cpp:



5.175.1 Detailed Description

Implementation of the [SetTaxCommand](#) class.

This file contains the implementation of the [SetTaxCommand](#) class, which is used to set the tax rate in the government.

Version

1.0

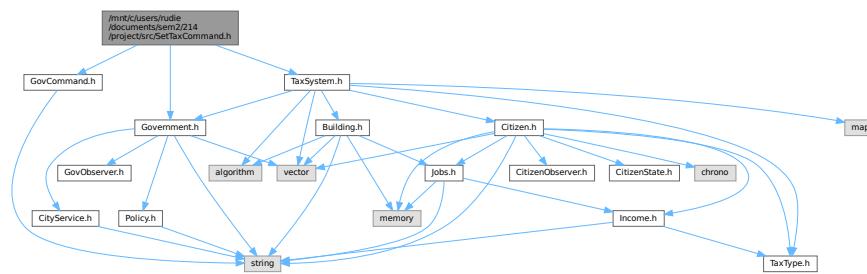
Date

2024-11-04

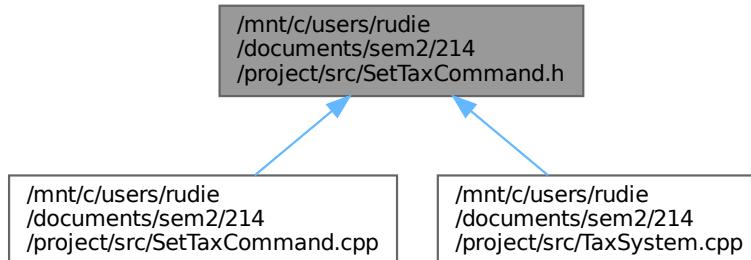
5.176 /mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxCommand.h File Reference

Declaration of the [SetTaxCommand](#) class.

```
#include "GovCommand.h"
#include "Government.h"
#include "TaxSystem.h"
Include dependency graph for SetTaxCommand.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [SetTaxCommand](#)
A command to set the tax rate in the government.

5.176.1 Detailed Description

Declaration of the [SetTaxCommand](#) class.

This file contains the declaration of the [SetTaxCommand](#) class, which is used to set the tax rate in the government.

Version

1.0

Date

2024-11-04

5.177 SetTaxCommand.h

[Go to the documentation of this file.](#)

```

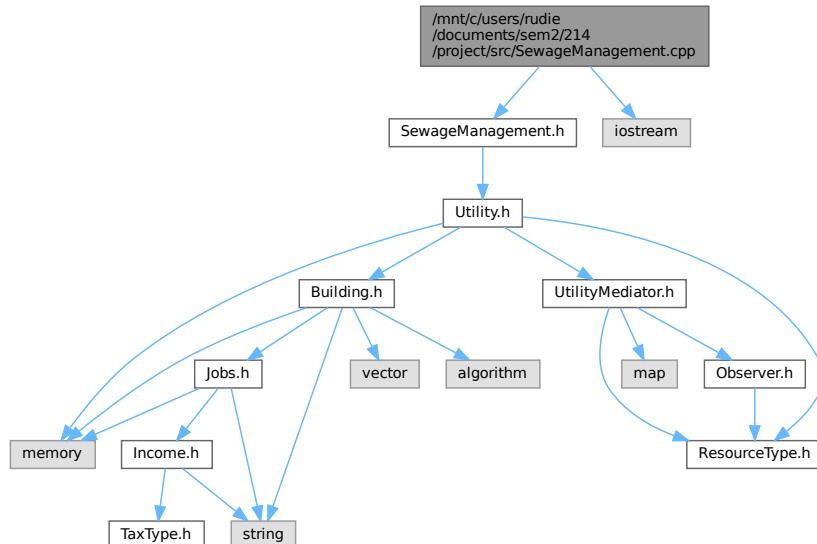
00001 #ifndef SETTAXCOMMAND_H
00002 #define SETTAXCOMMAND_H
00003
00015 #include "GovCommand.h"
00016 #include "Government.h"
00017 #include "TaxSystem.h"
00018
00025 class SetTaxCommand : public GovCommand {
00026
00027 private:
00028     Government* government;
00029     double taxRate;
00030     double previousTaxRate;
00031
00032 public:
00041     SetTaxCommand(Government* gov, double rate);
00042
00053     SetTaxCommand(Government* gov, TaxSystem* taxSys, double rate, char category);
00054
00060     void execute() override;
00061
00067     void undo() override;
00068
00074     std::string getName() const override;
00075
00081     std::string getDescription() const override;
00082
00088     double returnVal() override;
00089
00095     bool canExecute() const override;
00096 };
00097
00098 #endif // SETTAXCOMMAND_H

```

5.178 /mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.cpp File Reference

Implementation of the [SewageManagement](#) class.

```
#include "SewageManagement.h"
#include <iostream>
Include dependency graph for SewageManagement.cpp:
```



5.178.1 Detailed Description

Implementation of the [SewageManagement](#) class.

This file contains the implementation of the [SewageManagement](#) class, which manages sewage services for buildings.

Version

1.0

Date

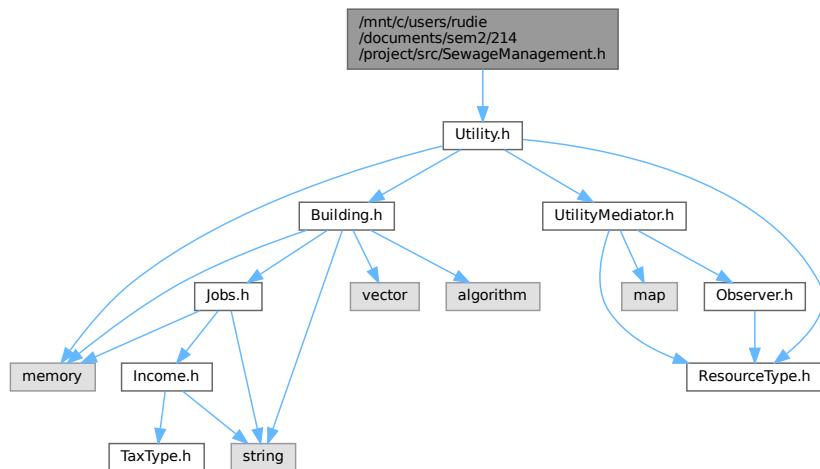
2024-11-04

5.179 /mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.h File Reference

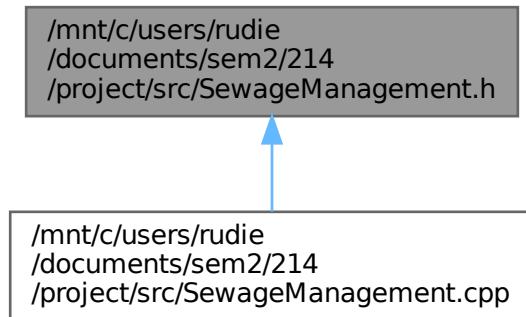
Declaration of the [SewageManagement](#) class.

```
#include "Utility.h"
```

Include dependency graph for SewageManagement.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SewageManagement](#)
A class to manage sewage services for buildings.

5.179.1 Detailed Description

Declaration of the [SewageManagement](#) class.

This file contains the declaration of the [SewageManagement](#) class, which manages sewage services for buildings.

Version

1.0

Date

2024-11-04

5.180 SewageManagement.h

[Go to the documentation of this file.](#)

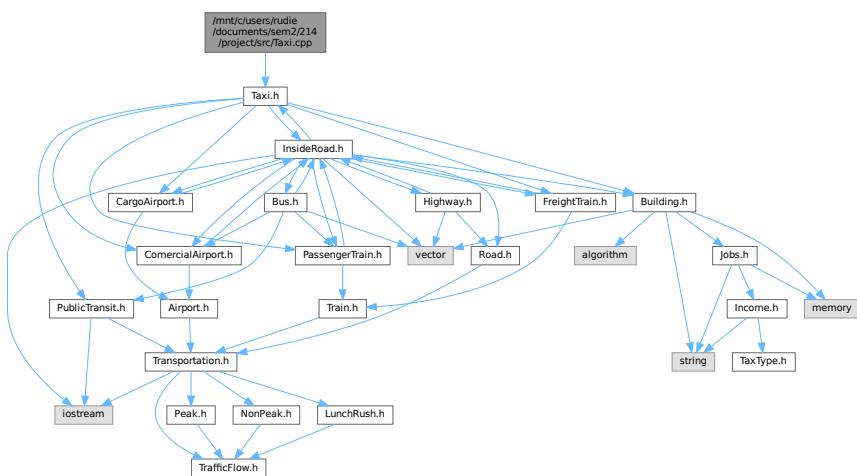
```
00001 #ifndef SEWAGEMANAGEMENT_H
00002 #define SEWAGEMANAGEMENT_H
00003
00015 #include "Utility.h"
00016
00023 class SewageManagement : public Utility {
00024
00025 public:
00033     SewageManagement(UtilityMediator* mediator);
00034
00040     void registerBuilding(Building* building) override;
00041
00049     void supplyResources(Building* building) override;
00050
00051     // /**
00052     // * @brief Adjusts sewage management based on a citizen's requirements.
00053     // *
00054     // * @param citizen Pointer to the Citizen object.
00055     // */
00056     // void adjustForCitizen(Citizen* citizen) override;
00057 };
00058
00059 #endif // SEWAGEMANAGEMENT_H
```

5.181 /mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.cpp File Reference

Implementation of the `Taxi` class.

```
#include "Taxi.h"
```

Include dependency graph for `Taxi.cpp`:



5.181.1 Detailed Description

Implementation of the `Taxi` class.

This file contains the implementation of the `Taxi` class, which represents a taxi service in the public transit system.

Version

1.0

Date

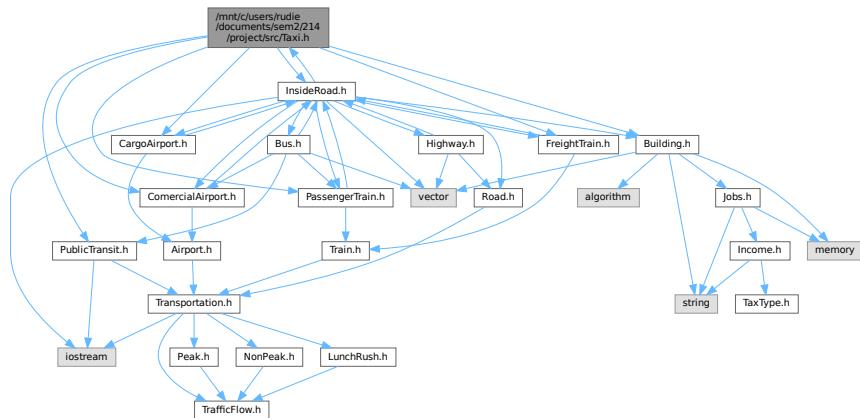
2024-11-04

5.182 /mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.h File Reference

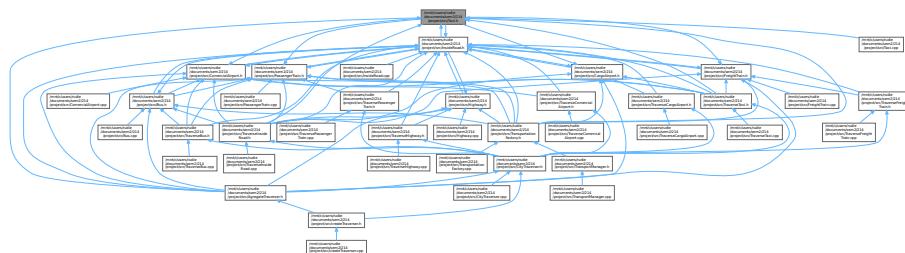
Header file for the `Taxi` class, which represents a taxi in a public transit system.

```
#include "PublicTransit.h"
#include "InsideRoad.h"
#include "ComercialAirport.h"
#include "CargoAirport.h"
#include "FreightTrain.h"
```

```
#include "PassengerTrain.h"
#include "Building.h"
Include dependency graph for Taxi.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Taxi](#)

A class representing a taxi in a public transit system.

5.182.1 Detailed Description

Header file for the [Taxi](#) class, which represents a taxi in a public transit system.

This file contains the declaration of the [Taxi](#) class, which represents a taxi service in the public transit system.

Version

1.0

Date

2024-11-04

5.183 Taxi.h

[Go to the documentation of this file.](#)

```

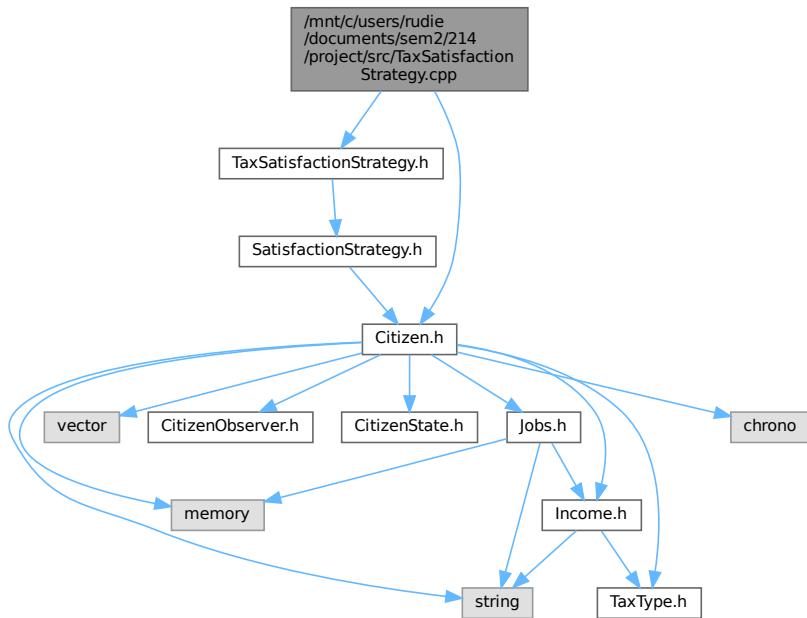
00001
00012 #ifndef TAXI_H
00013 #define TAXI_H
00014
00015 #include "PublicTransit.h"
00016 #include "InsideRoad.h"
00017 #include "ComercialAirport.h"
00018 #include "CargoAirport.h"
00019 #include "FreightTrain.h"
00020 #include "PassengerTrain.h"
00021 #include "Building.h"
00022
00023 class InsideRoad;
00024 class ComercialAirport;
00025 class CargoAirport;
00026 class PassengerTrain;
00027 class FreightTrain;
00028 class Building;
00029
00030 class Taxi : public PublicTransit {
00031     private:
00032         std::string taxiCompany;
00033         int taxiNumber;
00034
00035         std::vector<InsideRoad*> insideRoads;
00036         std::vector<ComercialAirport*> comercialAirports;
00037         std::vector<CargoAirport*> cargoAirports;
00038         std::vector<PassengerTrain*> passengerTrains;
00039         std::vector<FreightTrain*> freightTrains;
00040         std::vector<Building*> buildings;
00041
00042     public:
00043         Taxi(char state, std::string route, std::string taxiCompany, int taxiNumber);
00044
00045         bool addInsideRoad(InsideRoad *insideRoad);
00046
00047         bool addComercialAirport(ComercialAirport *comercialAirport);
00048
00049         bool addCargoAirport(CargoAirport *cargoAirport);
00050
00051         bool addPassengerTrain(PassengerTrain *passengerTrain);
00052
00053         bool addFreightTrain(FreightTrain *freightTrain);
00054
00055         bool addBuilding(Building *building);
00056
00057         InsideRoad *getInsideRoad(std::size_t x);
00058
00059         ComercialAirport *getComercialAirport(std::size_t x);
00060
00061         CargoAirport *getCargoAirport(std::size_t x);
00062
00063         PassengerTrain *getPassengerTrain(std::size_t x);
00064
00065         FreightTrain *getFreightTrain(std::size_t x);
00066
00067         Building *getBuilding(std::size_t x);
00068
00069         std::string getRouteName();
00070
00071         std::string getTaxiCompany();
00072
00073         int getTaxiNumber();
00074     };
00075 #endif // TAXI_H

```

5.184 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxiSatisfactionStrategy.cpp File Reference

Implementation of the [TaxiSatisfactionStrategy](#) class.

```
#include "TaxSatisfactionStrategy.h"
#include "Citizen.h"
Include dependency graph for TaxSatisfactionStrategy.cpp:
```



5.184.1 Detailed Description

Implementation of the [TaxSatisfactionStrategy](#) class.

This file contains the implementation of the [TaxSatisfactionStrategy](#) class, which calculates and updates the satisfaction level of a citizen based on tax rates.

Version

1.0

Date

2024-11-04

5.185 TaxSatisfactionStrategy.h

```

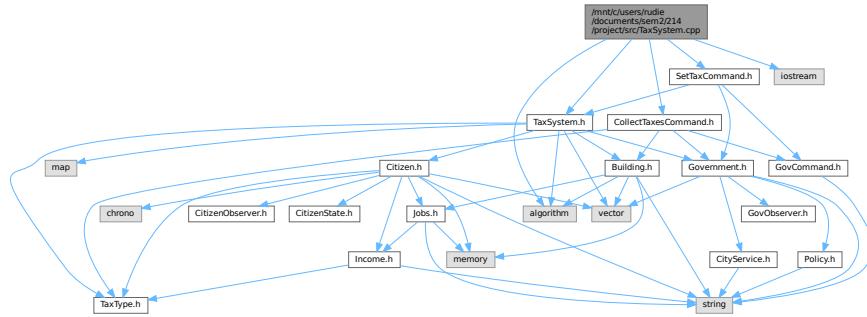
00001 #ifndef TAXSATISFACTIONSTRATEGY_H
00002 #define TAXSATISFACTIONSTRATEGY_H
00003
00004 #include "SatisfactionStrategy.h"
00005
00016 class TaxSatisfactionStrategy : public SatisfactionStrategy {
00017 public:
00024     float calculateSatisfaction(const Citizen& citizen) override;
00025
00033     void updateForJobChange(Citizen& citizen) override {}
00034
00042     void updateForHousingChange(Citizen& citizen) override {}
00043
00049     void updateForTaxChange(Citizen& citizen) override;
00050 };
00051
00052 #endif // TAXSATISFACTIONSTRATEGY_H
  
```

5.186 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.cpp File Reference

Implementation of the `TaxSystem` class.

```
#include "TaxSystem.h"
#include "CollectTaxesCommand.h"
#include "SetTaxCommand.h"
#include <iostream>
#include <algorithm>
Include dependency graph for TaxSystem.cpp:
```

Include dependency graph for TaxSystem.cpp:



5.186.1 Detailed Description

Implementation of the TaxSystem class.

This file contains the implementation of the `TaxSystem` class, which manages various types of taxes, including income, property, sales, and VAT. It allows adding and removing buildings and citizens for tax purposes, updating tax rates, and collecting taxes.

Version

1.0

Date

2024-11-04

5.187 TaxSystem.h

```
00001 // TaxSystem.h
00002
00003 #ifndef TAXSYSTEM_H
00004 #define TAXSYSTEM_H
00005
00006 #include <vector>
00007 #include <map>
00008 #include <algorithm>
00009 #include "Building.h"
00010 #include "Citizen.h"
00011 #include "TaxType.h"
00012 #include "Government.h"
00013
00024 class TaxSystem {
```

```

00025 public:
00031     void addGovernment(Government* gov);
00032
00038     void addIncomeTaxBuilding(Building* building);
00039
00045     void addPropertyTaxBuilding(Building* building);
00046
00052     void addSalesTaxBuilding(Building* building);
00053
00059     void addVATTaxPayer(Citizen* citizen);
00060
00066     void removeIncomeTaxBuilding(Building* building);
00067
00073     void removePropertyTaxBuilding(Building* building);
00074
00080     void removeSalesTaxBuilding(Building* building);
00081
00087     void removeVATTaxPayer(Citizen* citizen);
00088
00095     void updateTaxRate(char cType, double rate);
00096
00103     void collectTaxes(Building* building, char taxType);
00104
00111     void setTax(double rate, char taxType);
00112
00118     void checkImpact();
00119
00125     void addTaxRate(TaxType* tax);
00126
00132     void removeTaxRate(TaxType* taxType);
00133
00134 private:
00135     Government* government;
00136     std::vector<Building*> incomeTaxbuildings;
00137     std::vector<Building*> propertyTaxbuildings;
00138     std::vector<Building*> salesTaxbuildings;
00139     std::vector<Citizen*> vatTaxpayers;
00140     std::map<char, TaxType*> taxRates;
00141     float totalTaxCollected = 0.0;
00142 };
00143
00144 #endif // TAXSYSTEM_H

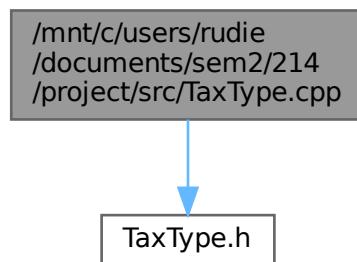
```

5.188 /mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.cpp File Reference

Implementation of the [TaxType](#) class.

```
#include "TaxType.h"
```

Include dependency graph for TaxType.cpp:



5.188.1 Detailed Description

Implementation of the [TaxType](#) class.

This file contains the implementation of the [TaxType](#) class, which represents a specific type of tax with a rate and type identifier.

Version

1.0

Date

2024-11-04

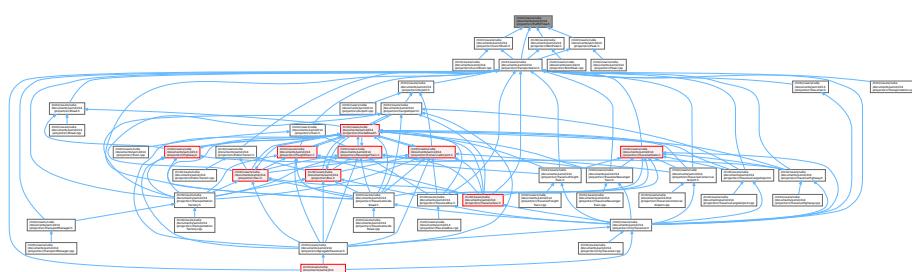
5.189 TaxType.h

```
00001 #ifndef TAXTYPE_H
00002 #define TAXTYPE_H
00003
00014 class TaxType {
00015
00016 private:
00017     double taxRate;
00018     char cType;
00019
00020 public:
00021     TaxType(double rate, char type);
00022
00032     virtual ~TaxType() {} // Virtual Destructor
00033
00040     virtual double calculateTax(double val);
00041
00047     virtual void setTax(double rate);
00048
00054     virtual double getTaxRate();
00055
00061     char getTaxType();
00062 };
00063
00064 #endif // TAXTYPE_H
```

5.190 /mnt/c/users/rudie/documents/sem2/214/project/src/TrafficFlow.h File Reference

Defines the [TrafficFlow](#) interface for traffic flow measurement.

This graph shows which files directly or indirectly include this file:



Classes

- class [TrafficFlow](#)

An abstract class that represents the traffic flow.

5.190.1 Detailed Description

Defines the [TrafficFlow](#) interface for traffic flow measurement.

This file contains the definition of the [TrafficFlow](#) interface, which provides methods for measuring traffic flow and obtaining the state of the traffic.

Version

1.0

Date

2024-11-04

5.191 TrafficFlow.h

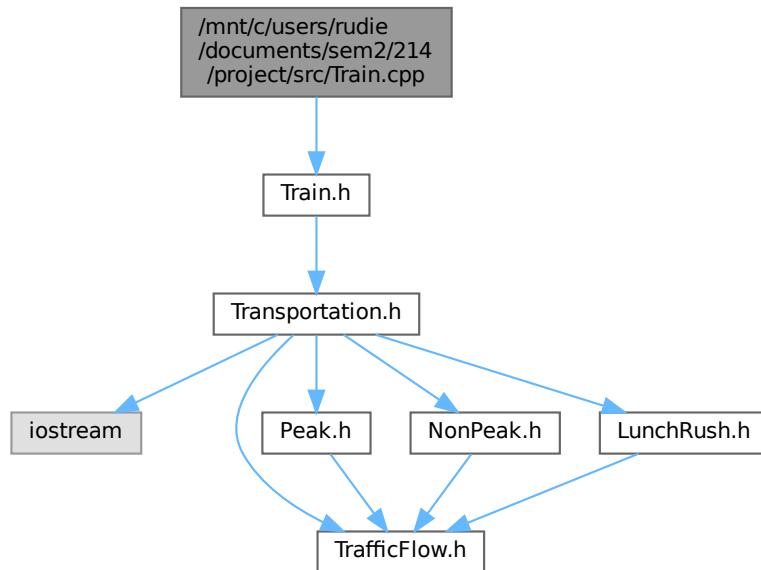
[Go to the documentation of this file.](#)

```
00001
00013 #ifndef TRAFFICFLOW_H
00014 #define TRAFFICFLOW_H
00015
00022 class TrafficFlow {
00023     public:
00029         virtual float getTrafficFlow() = 0;
00030
00036         virtual char getState() = 0;
00037 };
00038
00039 #endif // TRAFFICFLOW_H
```

5.192 /mnt/c/users/rudie/documents/sem2/214/project/src/Train.cpp File Reference

Implementation of the [Train](#) class.

```
#include "Train.h"
Include dependency graph for Train.cpp:
```



5.192.1 Detailed Description

Implementation of the [Train](#) class.

This file contains the implementation of the [Train](#) class, which represents a train in the transportation system.

Version

1.0

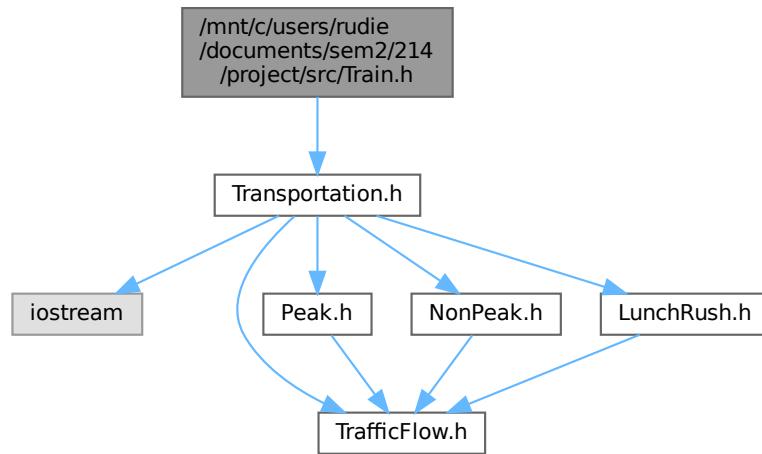
Date

2024-11-04

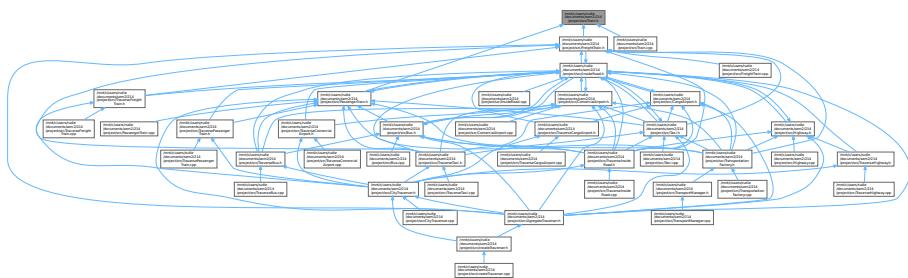
5.193 /mnt/c/users/rudie/documents/sem2/214/project/src/Train.h File Reference

Header file for the [Train](#) class.

```
#include "Transportation.h"
Include dependency graph for Train.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Train](#)
A class representing a train as a mode of transportation.

5.193.1 Detailed Description

Header file for the [Train](#) class.

This file contains the declaration of the [Train](#) class, which is a derived class of [Transportation](#). It includes methods to calculate the commute and get the train line.

Version

1.0

Date

2024-11-04

5.194 Train.h

[Go to the documentation of this file.](#)

```

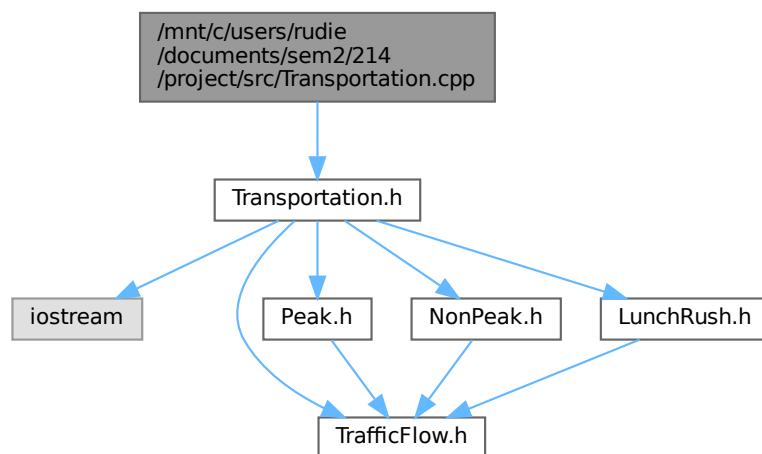
00001
00012 #ifndef TRAIN_H
00013 #define TRAIN_H
00014
00015 #include "Transportation.h"
00016
00024 class Train : public Transportation {
00025     private:
00026         std::string line;
00027
00028     public:
00036         Train(char state, std::string line, char type);
00037
00043         float calculateCommute();
00044
00050         std::string getLine();
00051     };
00052
00053 #endif // TRAIN_H

```

5.195 /mnt/c/users/rudie/documents/sem2/214/project/src/← Transportation.cpp File Reference

Implementation of the [Transportation](#) class.

```
#include "Transportation.h"
Include dependency graph for Transportation.cpp:
```



5.195.1 Detailed Description

Implementation of the [Transportation](#) class.

This file contains the implementation of the [Transportation](#) class, which represents a mode of transportation with different states such as [Peak](#), [NonPeak](#), and [LunchRush](#).

Version

1.0

Date

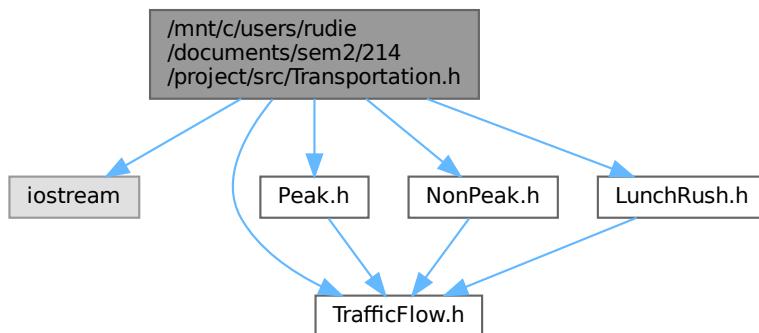
2024-11-04

5.196 /mnt/c/users/rudie/documents/sem2/214/project/src/Transportation.h File Reference

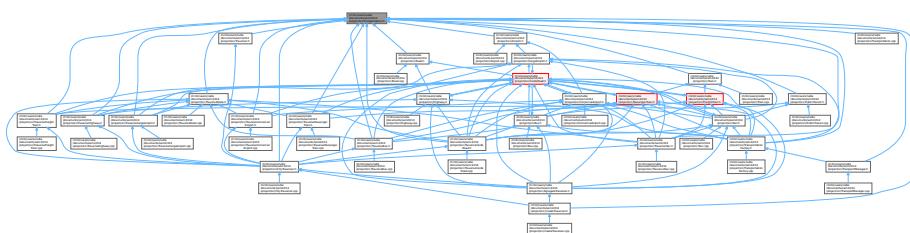
Header file for the [Transportation](#) class.

```
#include <iostream>
#include "TrafficFlow.h"
#include "Peak.h"
#include "NonPeak.h"
#include "LunchRush.h"
```

Include dependency graph for Transportation.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Transportation](#)
Manages traffic flow states and types of transportation.

5.196.1 Detailed Description

Header file for the [Transportation](#) class.

This file contains the declaration of the [Transportation](#) class, which is used to manage traffic flow states and types of transportation.

Version

1.0

Date

2024-11-04

5.197 Transportation.h

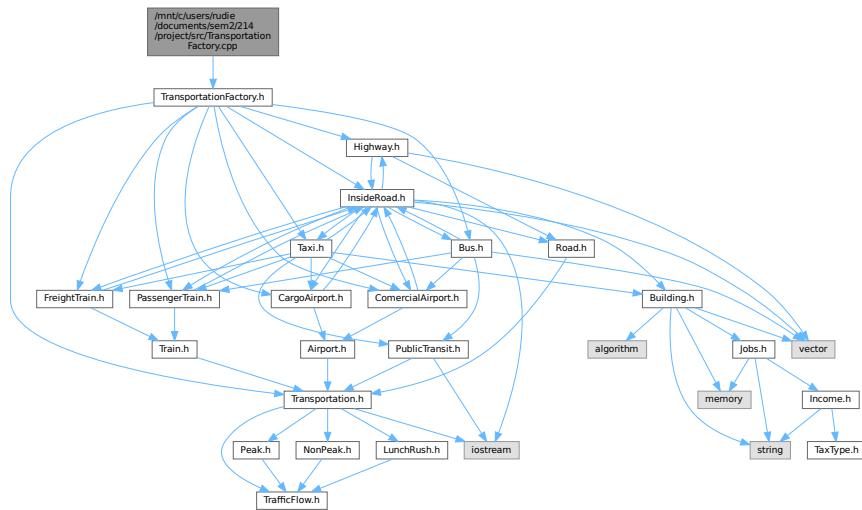
[Go to the documentation of this file.](#)

```
00001
00012 #ifndef TRANSPORTATION_H
00013 #define TRANSPORTATION_H
00014
00015 #include <iostream>
00016
00017 #include "TrafficFlow.h"
00018 #include "Peak.h"
00019 #include "NonPeak.h"
00020 #include "LunchRush.h"
00021
00029 class Transportation {
00030     private:
00031         TrafficFlow *state;
00032         char type;
00033     public:
00042         Transportation(char state, char type);
00043
00049         float getTrafficFlow();
00050
00059         bool setState(char state);
00060
00066         char getType();
00067
00073         ~Transportation();
00074 };
00075
00076 #endif // TRANSPORTATION_H
```

5.198 /mnt/c/users/rudie/documents/sem2/214/project/src/← TransportationFactory.cpp File Reference

Implementation of the [TransportationFactory](#) class.

```
#include "TransportationFactory.h"
Include dependency graph for TransportationFactory.cpp:
```



5.198.1 Detailed Description

Implementation of the [TransportationFactory](#) class.

This file contains the implementation of the [TransportationFactory](#) class, which provides factory methods to create various types of transportation objects.

Version

1.0

Date

2024-11-04

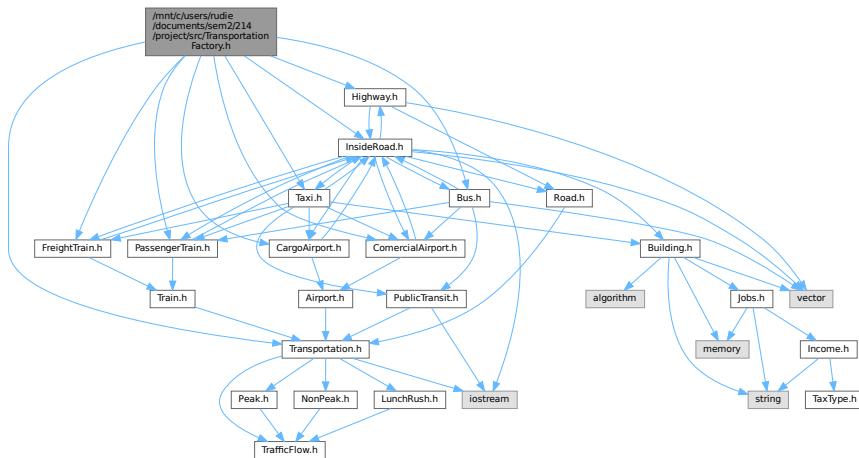
5.199 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportationFactory.h File Reference

Header file for the [TransportationFactory](#) class.

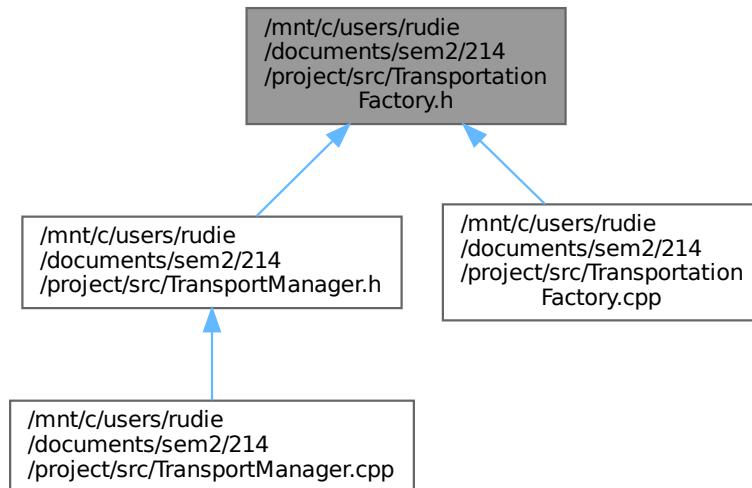
```
#include "Transportation.h"
#include "Highway.h"
#include "InsideRoad.h"
#include "Bus.h"
#include "Taxi.h"
#include "PassengerTrain.h"
#include "FreightTrain.h"
#include "ComercialAirport.h"
```

```
#include "CargoAirport.h"
```

Include dependency graph for `TransportationFactory.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class [TransportationFactory](#)

A factory class for creating different types of transportation objects.

5.199.1 Detailed Description

Header file for the [TransportationFactory](#) class.

This file contains the declaration of the [TransportationFactory](#) class, which is responsible for creating various types of transportation objects.

Version

1.0

Date

2024-11-04

5.200 TransportationFactory.h

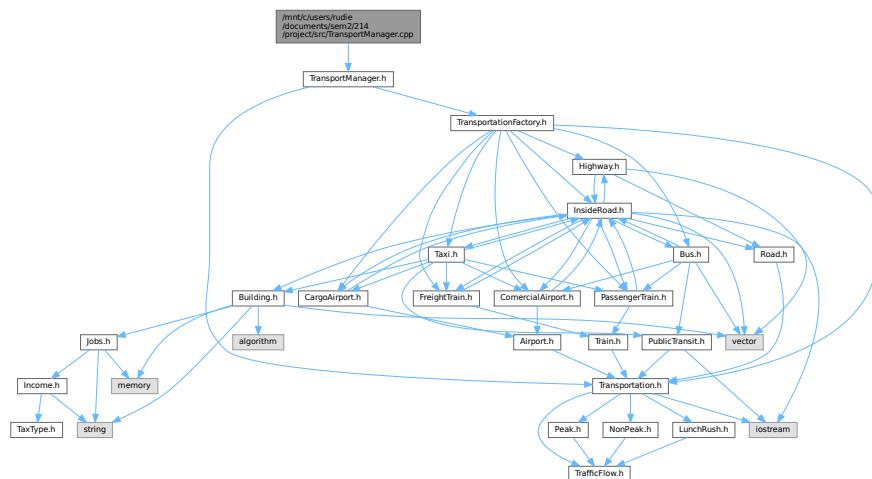
[Go to the documentation of this file.](#)

```
00001
00011 #ifndef TRANSPORTATIONFACTORY_H
00012 #define TRANSPORTATIONFACTORY_H
00013
00014 #include "Transportation.h"
00015 #include "Highway.h"
00016 #include "InsideRoad.h"
00017 #include "Bus.h"
00018 #include "Taxi.h"
00019 #include "PassengerTrain.h"
00020 #include "FreightTrain.h"
00021 #include "ComercialAirport.h"
00022 #include "CargoAirport.h"
00023
00028 class TransportationFactory {
00029     public:
00037         Transportation* createHighway(char state, std::string roadName, float speedLimit);
00038
00046         Transportation* createInsideRoad(char state, std::string roadName, float avgStopTime);
00047
00056         Transportation* createBus(char state, std::string route, int busNumber, int capacity);
00057
00066         Transportation* createTaxi(char state, std::string route, std::string taxiCompany, int
taxiNumber);
00067
00074         Transportation* createPassengerTrain(char state, std::string line);
00075
00084         Transportation* createFreightTrain(char state, std::string line, float weight, float length);
00085
00092         Transportation* createComercialAirport(char state, std::string name);
00093
00100         Transportation* createCargoAirport(char state, std::string name);
00101 };
00102
00103 #endif // TRANSPORTATIONFACTORY_H
```

5.201 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.cpp File Reference

Implementation of the [TransportManager](#) class.

```
#include "TransportManager.h"
Include dependency graph for TransportManager.cpp:
```



5.201.1 Detailed Description

Implementation of the [TransportManager](#) class.

This file contains the implementation of the [TransportManager](#) class, which manages the creation and storage of various transportation objects using the [TransportationFactory](#).

Version

1.0

Date

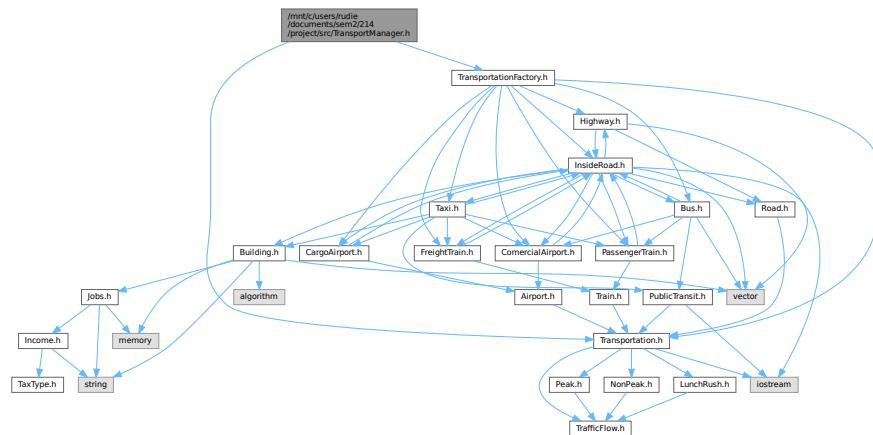
2024-11-04

5.202 /mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.h File Reference

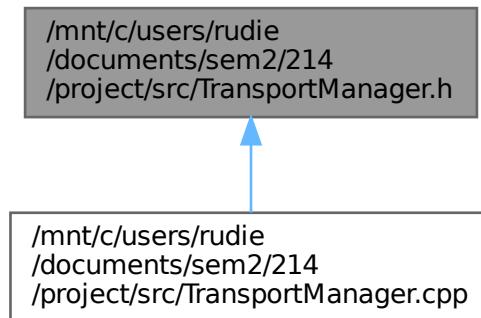
Header file for the [TransportManager](#) class.

```
#include "Transportation.h"
#include "TransportationFactory.h"
```

Include dependency graph for TransportManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **TransportManager**
Manages various types of transportation objects.

5.202.1 Detailed Description

Header file for the `TransportManager` class.

This file contains the definition of the `TransportManager` class, which is responsible for managing various types of transportation objects.

Version

1.0

Date

2024-11-04

5.203 TransportManager.h

[Go to the documentation of this file.](#)

```

00011 #ifndef TRANSPORTMANAGER_H
00012 #define TRANSPORTMANAGER_H
00013
00014 #include "Transportation.h"
00015 #include "TransportationFactory.h"
00016
00023 class TransportManager {
00024     private:
00025         TransportationFactory* factory;
00026         std::vector<Transportation*> transportations;
00027
00028     public:
00029         TransportManager();
00033
00042         bool createHighway(char state, std::string roadName, float speedLimit);
00043
00052         bool createInsideRoad(char state, std::string roadName, float avgStopTime);
00053
00063         bool createBus(char state, std::string route, int busNumber, int capacity);
00064
00074         bool createTaxi(char state, std::string route, std::string taxiCompany, int taxiNumber);
00075
00083         bool createPassengerTrain(char state, std::string line);
00084
00094         bool createFreightTrain(char state, std::string line, float weight, float length);
00095
00103         bool createComercialAirport(char state, std::string name);
00104
00112         bool createCargoAirport(char state, std::string name);
00113
00120         Transportation* getTransportation(size_t index);
00121
00125     ~TransportManager();
00127
00128 #endif // TRANSPORTMANAGER_H

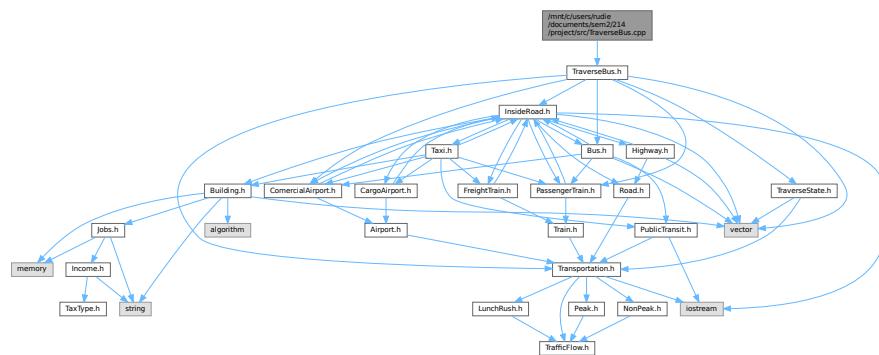
```

5.204 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus File Reference

Implementation of the [TraverseBus](#) class.

```
#include "TraverseBus.h"
```

Include dependency graph for TraverseBus.cpp:



5.204.1 Detailed Description

Implementation of the [TraverseBus](#) class.

This file contains the implementation of the [TraverseBus](#) class, which provides methods to traverse through different lists of transportation elements related to buses.

Version

1.0

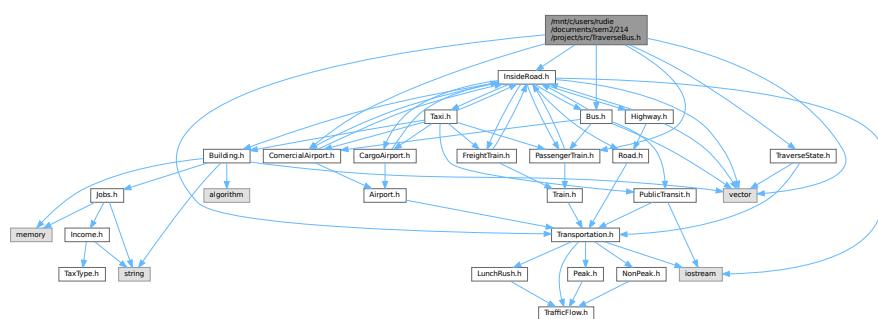
Date

2024-11-04

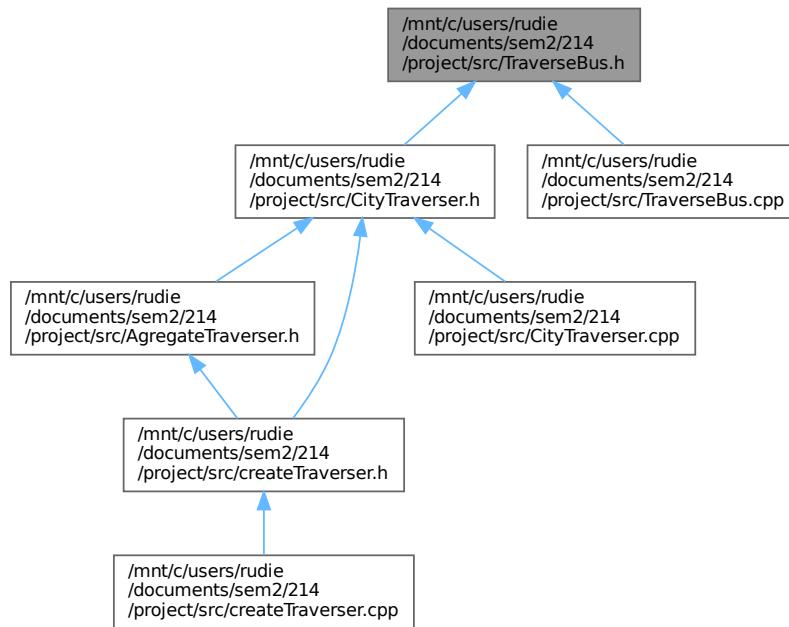
5.205 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseBus.h File Reference

Header file for the [TraverseBus](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "Bus.h"
#include "ComercialAirport.h"
#include "PassengerTrain.h"
Include dependency graph for TraverseBus.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseBus](#)
A class to manage the traversal state of a bus.

5.205.1 Detailed Description

Header file for the [TraverseBus](#) class.

This file contains the definition of the [TraverseBus](#) class, which is a concrete implementation of the [TraverseState](#) interface. The [TraverseBus](#) class is responsible for managing the traversal state of a bus within a transportation system.

Version

1.0

Date

2024-11-04

5.206 TraverseBus.h

[Go to the documentation of this file.](#)

```

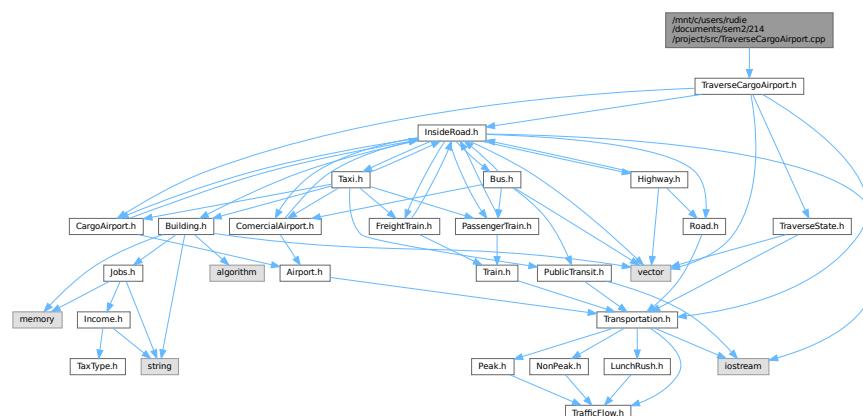
00001
00014 #ifndef TRAVERSEBUS_H
00015 #define TRAVERSEBUS_H
00016
00017 #include <vector>
00018
00019 #include "TraverseState.h"
00020 #include "Transportation.h"
00021 #include "InsideRoad.h"
00022 #include "Bus.h"
00023 #include "ComercialAirport.h"
00024 #include "PassengerTrain.h"
00025
00026 class Transportation;
00027 class InsideRoad;
00028 class Bus;
00029 class ComercialAirport;
00030 class PassengerTrain;
00031
00041 class TraverseBus: public TraverseState{
00042     private:
00043         int currentList = 0;
00044         int upperBound = 3;
00045
00046     public:
00055     TraverseBus(Transportation *element);
00056
00064     bool nextList();
00065
00073     bool prevList();
00074
00084     Transportation* getPos(size_t x);
00085
00086 };
00087
00088 #endif // TRAVERSEBUS_H

```

5.207 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.cpp File Reference

Implementation of the [TraverseCargoAirport](#) class.

```
#include "TraverseCargoAirport.h"
Include dependency graph for TraverseCargoAirport.cpp:
```



5.207.1 Detailed Description

Implementation of the [TraverseCargoAirport](#) class.

This file contains the implementation of the [TraverseCargoAirport](#) class, which provides methods to traverse through different lists of transportation elements related to cargo airports.

Version

1.0

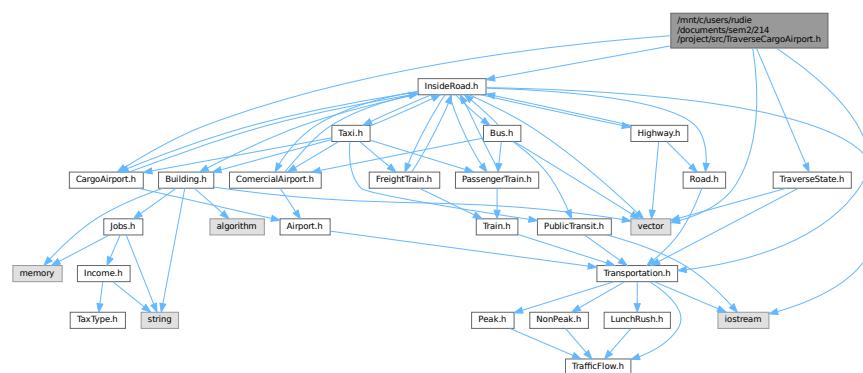
Date

2024-11-04

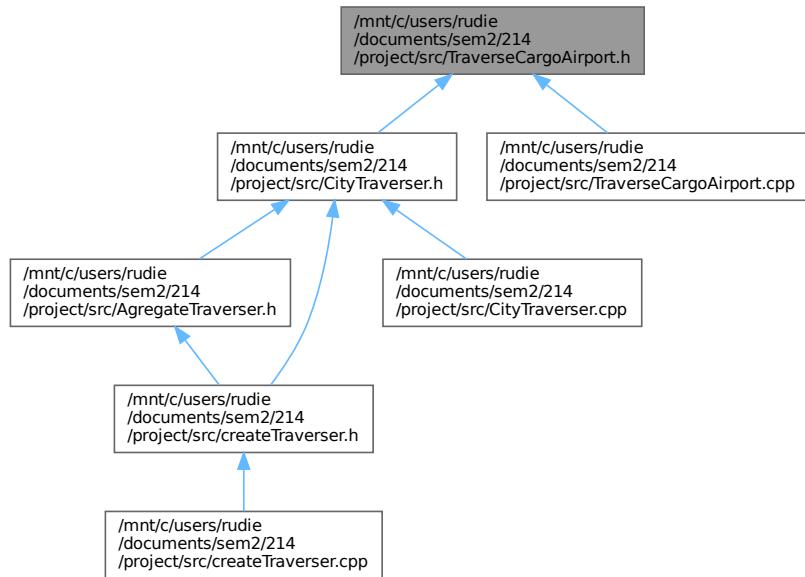
5.208 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseCargoAirport.h File Reference

Header file for the [TraverseCargoAirport](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "CargoAirport.h"
Include dependency graph for TraverseCargoAirport.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseCargoAirport](#)
A class to traverse through a cargo airport.

5.208.1 Detailed Description

Header file for the [TraverseCargoAirport](#) class.

This file contains the definition of the [TraverseCargoAirport](#) class, which is a subclass of [TraverseState](#). It provides functionality to traverse through a cargo airport using different lists.

Version

1.0

Date

2024-11-04

5.209 TraverseCargoAirport.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef TRAVERSECARGOAIRPORT_H
00013 #define TRAVERSECARGOAIRPORT_H
00014
00015 #include <vector>
00016
00017 #include "TraverseState.h"
00018 #include "Transportation.h"
00019 #include "InsideRoad.h"
00020 #include "CargoAirport.h"
00021
00022 class Transportation;
00023 class InsideRoad;
00024 class CargoAirport;
00025
00032 class TraverseCargoAirport: public TraverseState{
00033     private:
00034         int currentList = 0;
00035         int upperBound = 1;
00037     public:
00042         TraverseCargoAirport(Transportation *element);
00043
00048         bool nextList();
00049
00054         bool prevList();
00055
00061         Transportation* getPos(size_t x);
00062     };
00063
00064 #endif // TRAVERSECARGOAIRPORT_H

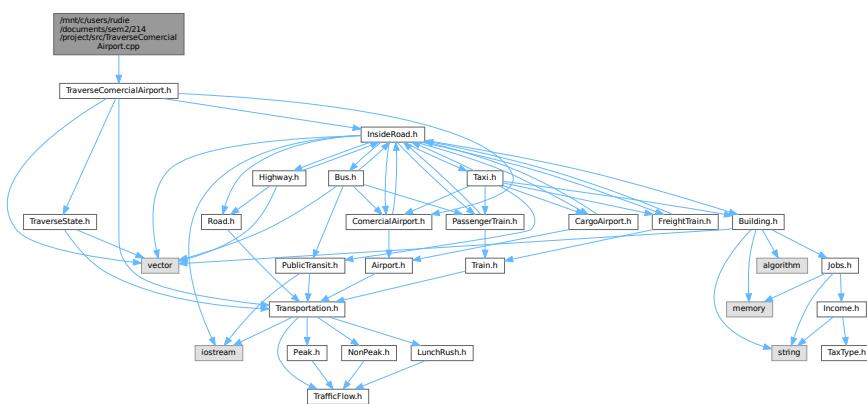
```

5.210 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.cpp File Reference

Implementation of the [TraverseComercialAirport](#) class.

```
#include "TraverseComercialAirport.h"
```

Include dependency graph for TraverseComercialAirport.cpp:



5.210.1 Detailed Description

Implementation of the [TraverseComercialAirport](#) class.

This file contains the implementation of the [TraverseComercialAirport](#) class, which provides methods to traverse through different lists of transportation elements related to commercial airports.

Version

1.0

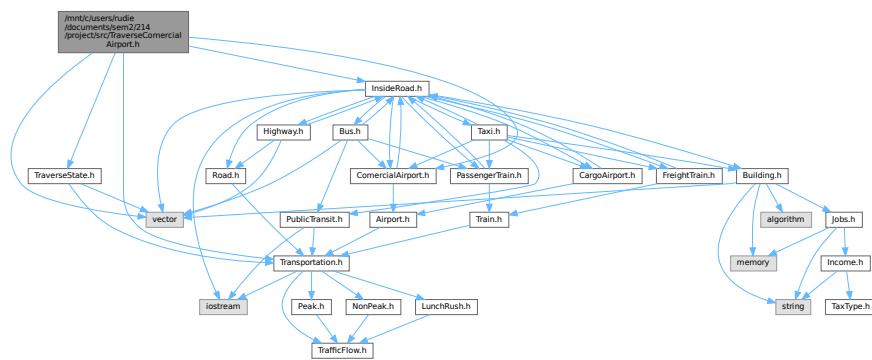
Date

2024-11-04

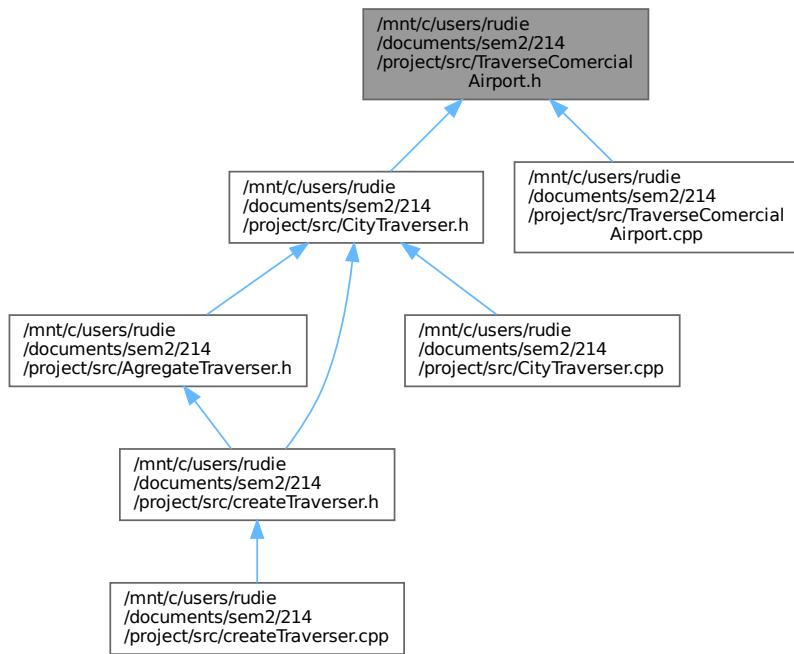
5.211 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAirport.h File Reference

Header file for the [TraverseComercialAirport](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "ComercialAirport.h"
Include dependency graph for TraverseComercialAirport.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseComercialAirport](#)
A class to traverse through a commercial airport.

5.211.1 Detailed Description

Header file for the [TraverseComercialAirport](#) class.

This file contains the definition of the [TraverseComercialAirport](#) class, which is a subclass of [TraverseState](#). It provides functionality to traverse through a commercial airport using different lists.

Version

1.0

Date

2024-11-04

5.212 TraverseComercialAirport.h

[Go to the documentation of this file.](#)

```

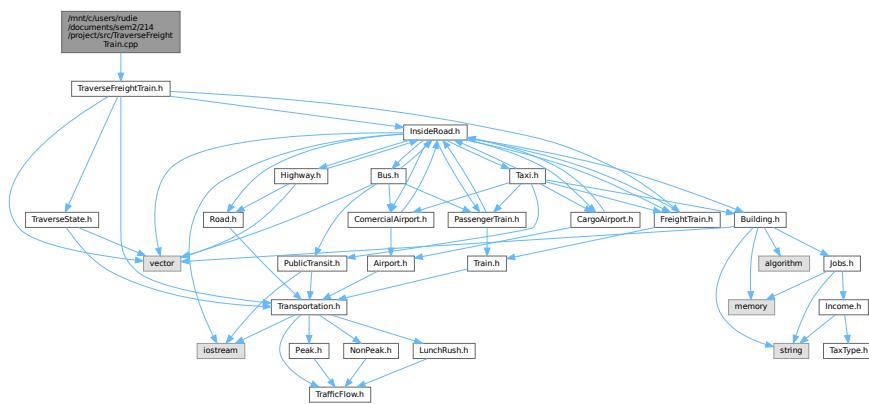
00001
00013 #ifndef TRAVERSECOMERCIALAIRPORT_H
00014 #define TRAVERSECOMERCIALAIRPORT_H
00015
00016 #include <vector>
00017
00018 #include "TraverseState.h"
00019 #include "Transportation.h"
00020 #include "InsideRoad.h"
00021 #include "ComercialAirport.h"
00022
00023 class Transportation;
00024 class InsideRoad;
00025 class ComercialAirport;
00026
00035 class TraverseComercialAirport: public TraverseState{
00036     private:
00037         int currentList = 0;
00038         int upperBound = 1;
00039
00040     public:
00048     TraverseComercialAirport(Transportation *element);
00049
00057     bool nextList();
00058
00066     bool prevList();
00067
00076     Transportation* getPos(size_t x);
00077 };
00078
00079 #endif // TRAVERSECOMERCIALAIRPORT_H

```

5.213 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.cpp File Reference

Implementation of the [TraverseFreightTrain](#) class.

```
#include "TraverseFreightTrain.h"
Include dependency graph for TraverseFreightTrain.cpp:
```



5.213.1 Detailed Description

Implementation of the [TraverseFreightTrain](#) class.

This file contains the implementation of the [TraverseFreightTrain](#) class, which provides methods to traverse through different lists of transportation elements related to freight trains.

Version

1.0

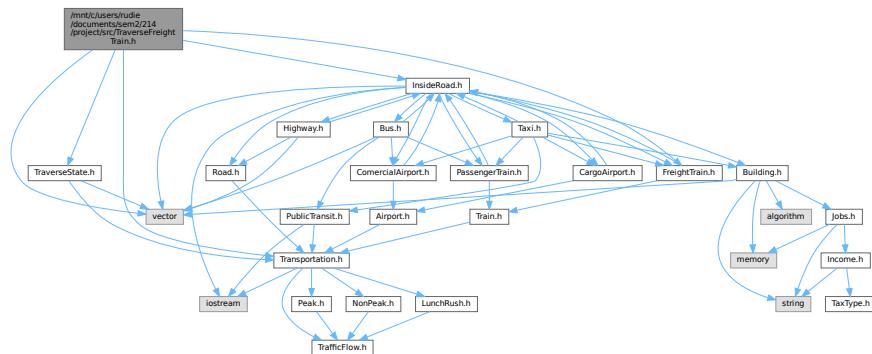
Date

2024-11-04

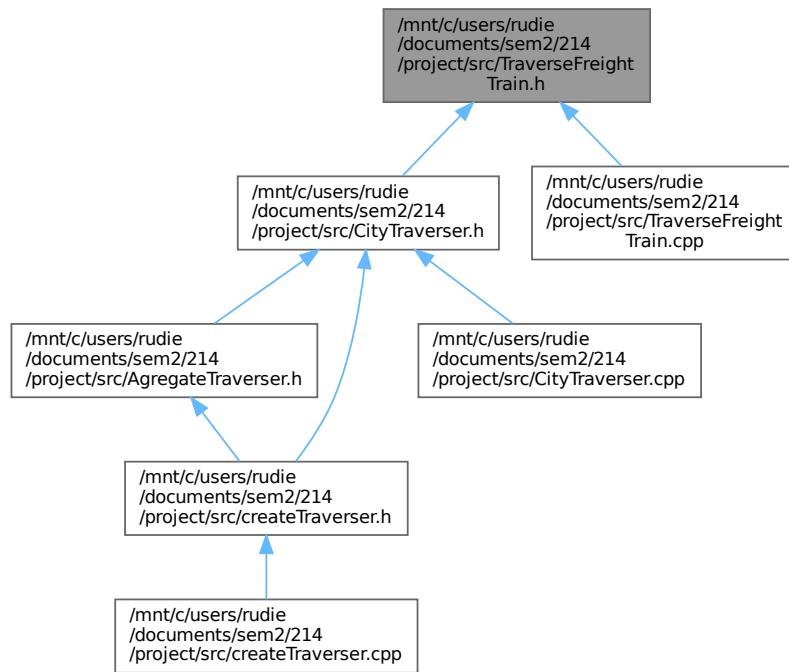
5.214 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.h File Reference

Header file for the [TraverseFreightTrain](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "FreightTrain.h"
Include dependency graph for TraverseFreightTrain.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseFreightTrain](#)
A class to traverse through FreightTrain objects.

5.214.1 Detailed Description

Header file for the [TraverseFreightTrain](#) class.

This file contains the definition of the [TraverseFreightTrain](#) class, which is a subclass of [TraverseState](#). It provides functionality to traverse through a list of [FreightTrain](#) objects within a [Transportation](#) system.

Version

1.0

Date

2024-11-04

5.215 TraverseFreightTrain.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef TRAVERSEFREIGHTTRAIN_H
00013 #define TRAVERSEFREIGHTTRAIN_H
00014
00015 #include <vector>
00016
00017 #include "TraverseState.h"
00018 #include "Transportation.h"
00019 #include "InsideRoad.h"
00020 #include "FreightTrain.h"
00021
00022 class Transportation;
00023 class InsideRoad;
00024 class FreightTrain;
00025
00026 class TraverseFreightTrain: public TraverseState{
00027     private:
00028         int currentList = 0;
00029         int upperBound = 1;
00030
00031     public:
00032         TraverseFreightTrain(Transportation *element);
00033
00034         bool nextList();
00035
00036         bool prevList();
00037
00038         Transportation* getPos(size_t x);
00039
00040     };
00041
00042 #endif // TRAVERSEFREIGHTTRAIN_H

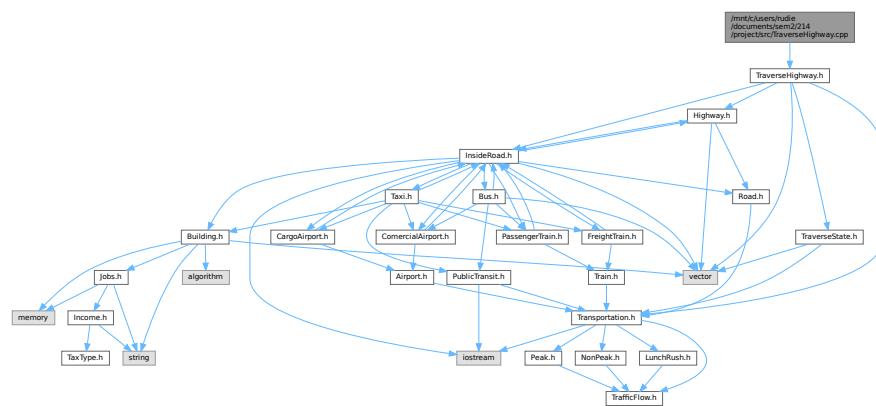
```

5.216 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.cpp File Reference

Implementation of the [TraverseHighway](#) class.

```
#include "TraverseHighway.h"
```

Include dependency graph for TraverseHighway.cpp:



5.216.1 Detailed Description

Implementation of the [TraverseHighway](#) class.

This file contains the implementation of the [TraverseHighway](#) class, which provides methods to traverse through different lists of transportation elements related to highways.

Version

1.0

Date

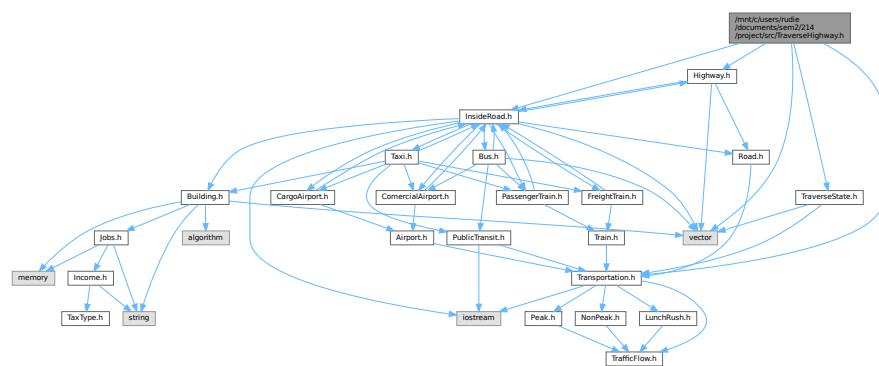
2024-11-04

5.217 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.h File Reference

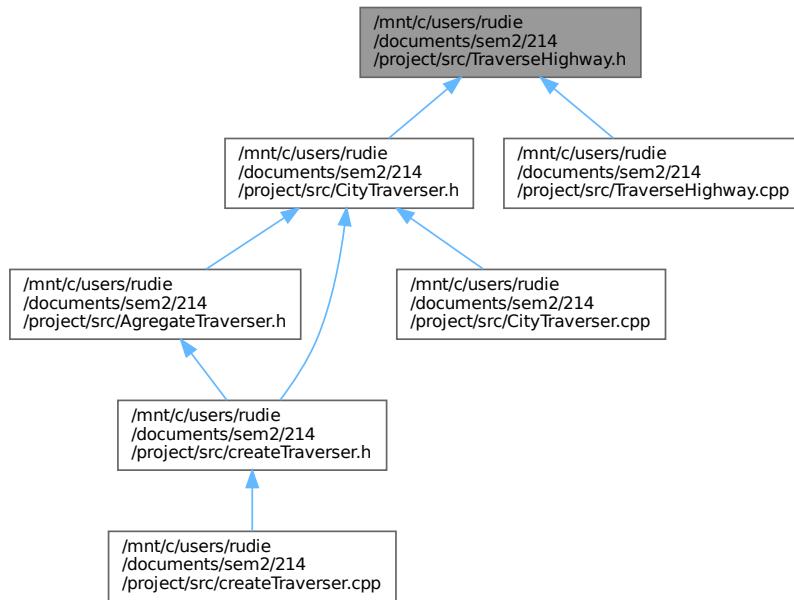
Header file for the [TraverseHighway](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "Highway.h"
#include "InsideRoad.h"
```

Include dependency graph for TraverseHighway.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseHighway](#)
A class to traverse through highways in a transportation system.

5.217.1 Detailed Description

Header file for the [TraverseHighway](#) class.

This file contains the definition of the [TraverseHighway](#) class, which is a subclass of [TraverseState](#). It provides functionality to traverse through highways in a transportation system.

Version

1.0

Date

2024-11-04

5.218 TraverseHighway.h

[Go to the documentation of this file.](#)

```

0001
00013 #ifndef TRAVERSEHIGHWAY_H
00014 #define TRAVERSEHIGHWAY_H
00015
00016 #include <vector>
00017
00018 #include "TraverseState.h"
00019 #include "Transportation.h"
00020 #include "Highway.h"
00021 #include "InsideRoad.h"
00022
00031 class TraverseHighway: public TraverseState{
00032     private:
00033         int currentList = 0;
00034         int upperBound = 1;
00035
00036     public:
00041         TraverseHighway(Transportation *element);
00042
00047         bool nextList();
00048
00053         bool prevList();
00054
00060         Transportation* getPos(size_t x);
00061     };
00062
00063 #endif // TRAVERSEHIGHWAY_H

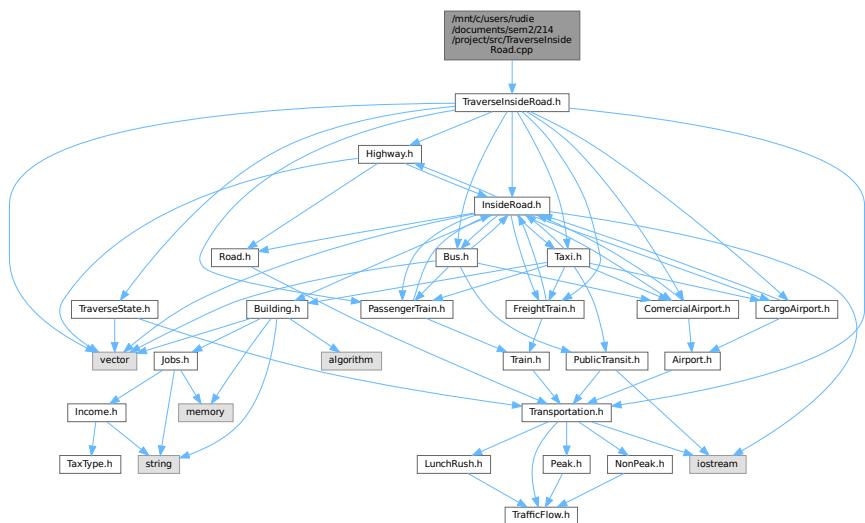
```

5.219 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.cpp File Reference

Implementation of the [TraverseInsideRoad](#) class.

```
#include "TraverseInsideRoad.h"
```

Include dependency graph for TraverseInsideRoad.cpp:



5.219.1 Detailed Description

Implementation of the [TraverseInsideRoad](#) class.

This file contains the implementation of the [TraverseInsideRoad](#) class, which provides methods to traverse through different lists of transportation elements related to inside roads.

Version

1.0

Date

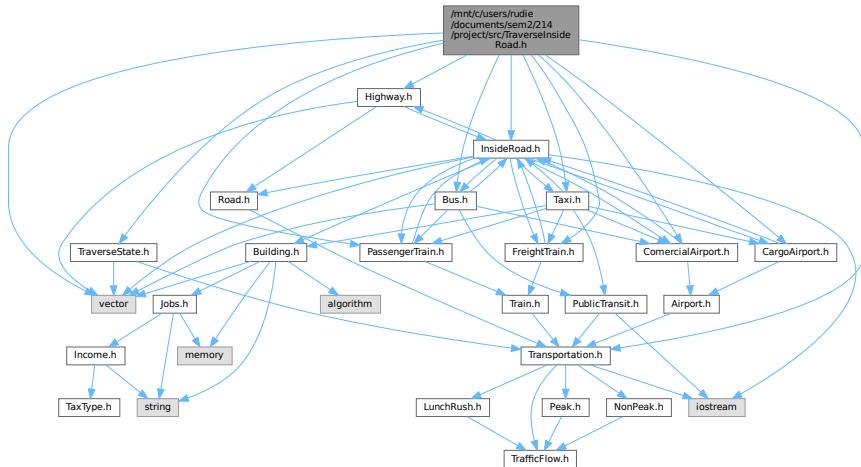
2024-11-04

5.220 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.h File Reference

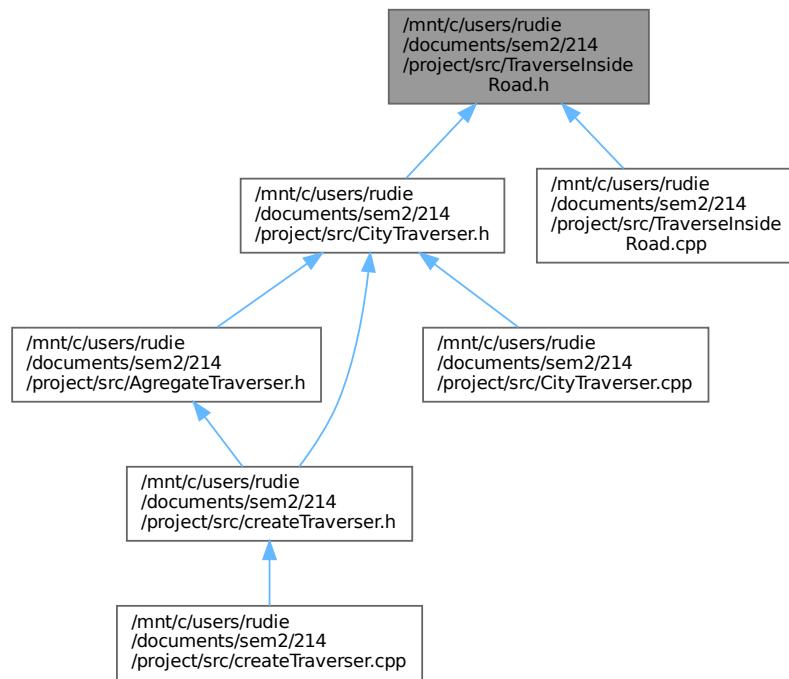
Header file for the [TraverseInsideRoad](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "Highway.h"
#include "InsideRoad.h"
#include "Bus.h"
#include "Taxi.h"
#include "ComercialAirport.h"
#include "CargoAirport.h"
#include "PassengerTrain.h"
#include "FreightTrain.h"
```

Include dependency graph for TraverseInsideRoad.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseInsideRoad](#)
A class to traverse inside roads in a transportation system.

5.220.1 Detailed Description

Header file for the [TraverseInsideRoad](#) class.

This file contains the definition of the [TraverseInsideRoad](#) class, which is a state class used to traverse inside roads in a transportation system.

Version

1.0

Date

2024-11-04

5.221 TraverseInsideRoad.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef TRAVERSEINSIDEROAD_H
00013 #define TRAVERSEINSIDEROAD_H
00014
00015 #include <vector>
00016
00017 #include "TraverseState.h"
00018 #include "Transportation.h"
00019 #include "Highway.h"
00020 #include "InsideRoad.h"
00021 #include "Bus.h"
00022 #include "Taxi.h"
00023 #include "ComercialAirport.h"
00024 #include "CargoAirport.h"
00025 #include "PassengerTrain.h"
00026 #include "FreightTrain.h"
00027
00028 class TraverseInsideRoad: public TraverseState{
00029     private:
00030         int currentList = 0;
00031         int upperBound = 7;
00032
00033     public:
00034         TraverseInsideRoad(Transportation *element);
00035
00036         bool nextList();
00037
00038         bool prevList();
00039
00040         Transportation* getPos(size_t x);
00041
00042     };
00043
00044 #endif // TRAVERSEINSIDEROAD_H

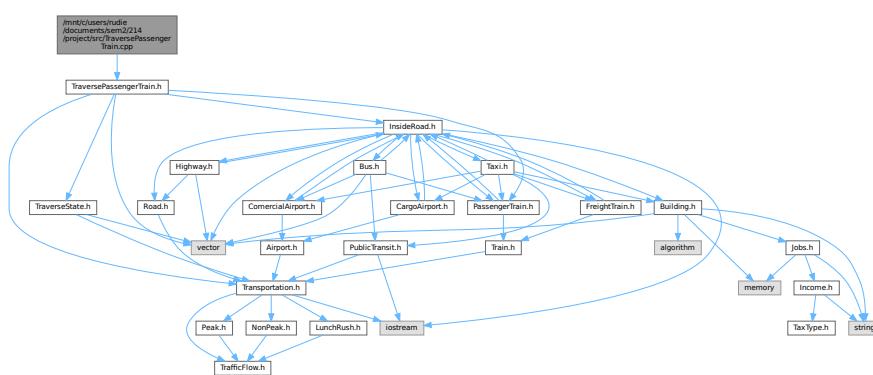
```

5.222 /mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.cpp File Reference

Implementation of the [TraversePassengerTrain](#) class.

```
#include "TraversePassengerTrain.h"
```

Include dependency graph for TraversePassengerTrain.cpp:



5.222.1 Detailed Description

Implementation of the [TraversePassengerTrain](#) class.

This file contains the implementation of the [TraversePassengerTrain](#) class, which provides methods to traverse through different lists of transportation elements related to passenger trains.

Version

1.0

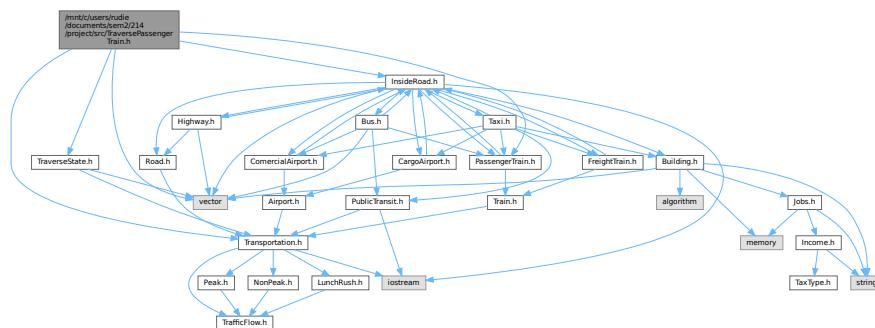
Date

2024-11-04

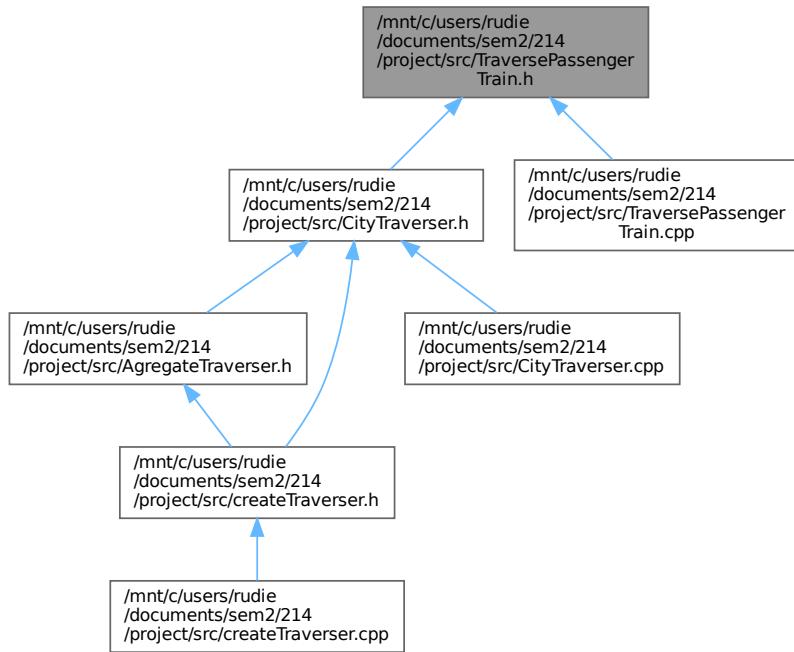
5.223 /mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.h File Reference

Header file for the [TraversePassengerTrain](#) class.

```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "PassengerTrain.h"
Include dependency graph for TraversePassengerTrain.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraversePassengerTrain](#)
A class to traverse through a passenger train.

5.223.1 Detailed Description

Header file for the [TraversePassengerTrain](#) class.

This file contains the definition of the [TraversePassengerTrain](#) class, which is used to traverse through a passenger train in a transportation system.

Version

1.0

Date

2024-11-04

5.224 TraversePassengerTrain.h

[Go to the documentation of this file.](#)

```

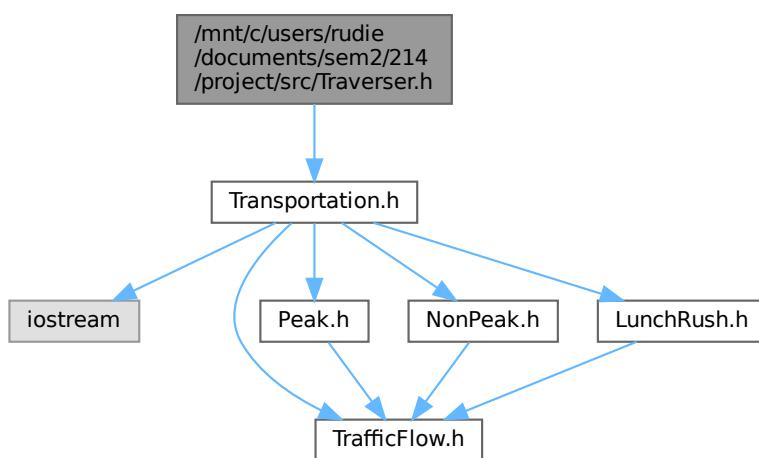
00001
00012 #ifndef TRAVERSEPASSENGERTRAIN_H
00013 #define TRAVERSEPASSENGERTRAIN_H
00014
00015 #include <vector>
00016
00017 #include "TraverseState.h"
00018 #include "Transportation.h"
00019 #include "InsideRoad.h"
00020 #include "PassengerTrain.h"
00021
00022 class Transportation;
00023 class InsideRoad;
00024 class PassengerTrain;
00025
00026 class TraversePassengerTrain: public TraverseState{
00027     private:
00028         int currentList = 0;
00029         int upperBound = 1;
00030
00031     public:
00032         TraversePassengerTrain(Transportation *element);
00033
00034         bool nextList();
00035
00036         bool prevList();
00037
00038         Transportation* getPos(size_t x);
00039
00040     };
00041
00042 #endif // TRAVERSEPASSENGERTRAIN_H

```

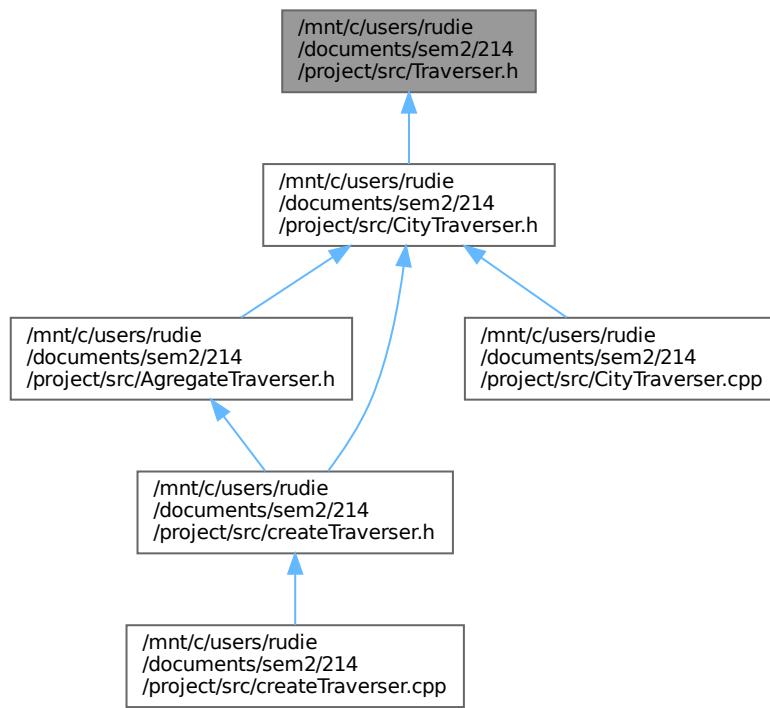
5.225 /mnt/c/users/rudie/documents/sem2/214/project/src/Traverser.h File Reference

Defines the [Traverser](#) interface for iterating over [Transportation](#) objects.

```
#include "Transportation.h"
Include dependency graph for Traverser.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Traverser](#)

Interface for iterating over [Transportation](#) objects.

5.225.1 Detailed Description

Defines the [Traverser](#) interface for iterating over [Transportation](#) objects.

This file contains the declaration of the [Traverser](#) class, which provides an interface for incrementing, decrementing, and dereferencing iterators over [Transportation](#) objects.

Version

1.0

Date

2024-11-04

5.226 Traverser.h

[Go to the documentation of this file.](#)

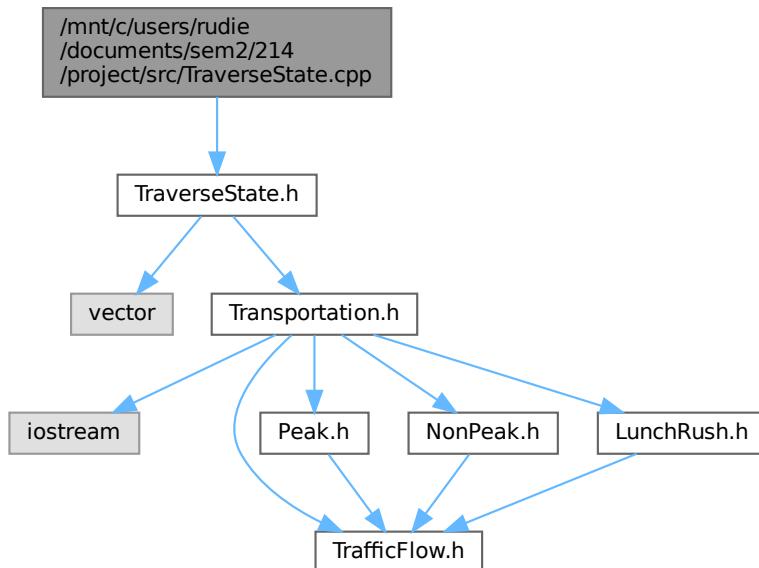
```
00001
00012 #ifndef TRAVERSER_H
00013 #define TRAVERSER_H
00014
00015 #include "Transportation.h"
00016
00025 class Traverser {
00026     public:
00034         virtual bool operator++() = 0;
00035
00043         virtual bool operator--() = 0;
00044
00052         virtual Transportation* operator*() = 0;
00053 };
00054
00055 #endif // TRAVERSER_H
```

5.227 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.cpp File Reference

Implementation of the [TraverseState](#) class.

```
#include "TraverseState.h"
```

Include dependency graph for TraverseState.cpp:



5.227.1 Detailed Description

Implementation of the [TraverseState](#) class.

This file contains the implementation of the [TraverseState](#) class, which is responsible for managing the state of transportation layers.

Version

1.0

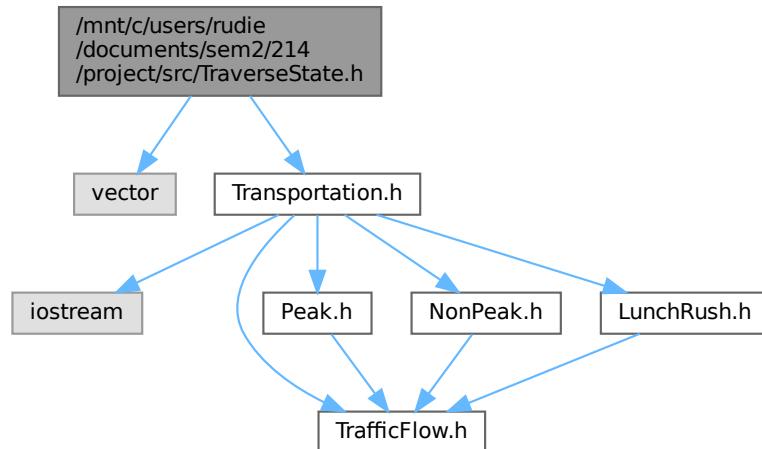
Date

2024-11-04

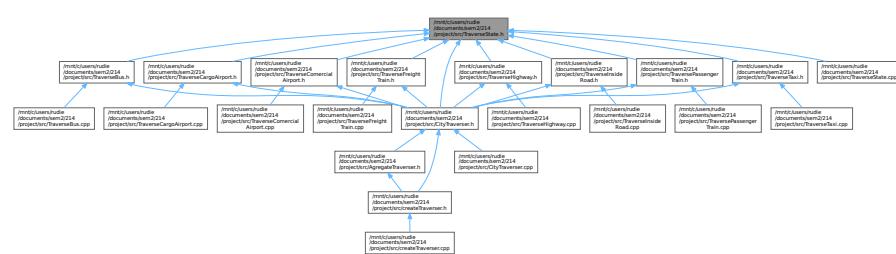
5.228 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.h File Reference

Defines the [TraverseState](#) class and its interface for traversing through [Transportation](#) elements.

```
#include <vector>
#include "Transportation.h"
Include dependency graph for TraverseState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseState](#)

Abstract class that provides an interface for traversing through a list of [Transportation](#) elements.

5.228.1 Detailed Description

Defines the [TraverseState](#) class and its interface for traversing through [Transportation](#) elements.

This file contains the definition of the [TraverseState](#) class, which provides an abstract interface for traversing through a list of [Transportation](#) elements.

Version

1.0

Date

2024-11-04

5.229 TraverseState.h

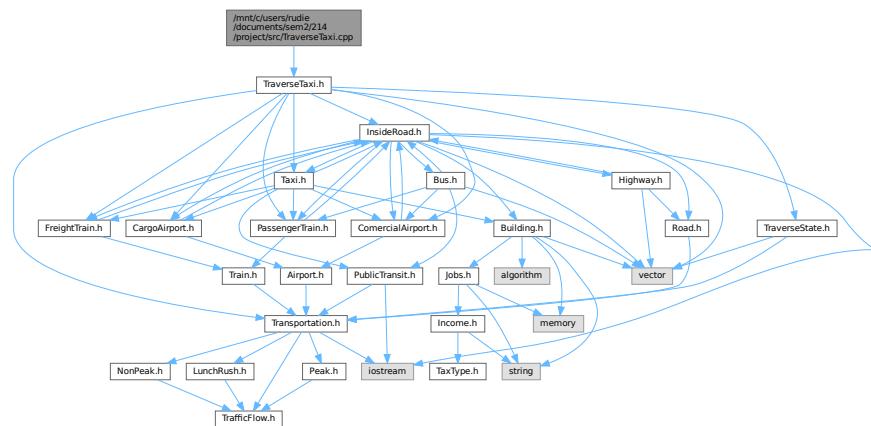
[Go to the documentation of this file.](#)

```
00001
00012 #ifndef TRAVERSESTATE_H
00013 #define TRAVERSESTATE_H
00014
00015 #include <vector>
00016 #include "Transportation.h"
00017
00022 class TraverseState {
00023     private:
00024         Transportation *layer;
00025
00026     public:
00031         TraverseState(Transportation *element);
00032
00037         virtual bool nextList() = 0;
00038
00043         virtual bool prevList() = 0;
00044
00050         virtual Transportation* getPos(size_t x) = 0;
00051
00056         Transportation* getLayer();
00057         virtual ~TraverseState() = default;
00058 };
00059
00060 #endif // TRAVERSESTATE_H
```

5.230 /mnt/c/users/rudie/documents/sem2/214/project/src/Traverse← Taxi.cpp File Reference

Implementation of the [TraverseTaxi](#) class.

```
#include "TraverseTaxi.h"
Include dependency graph for TraverseTaxi.cpp:
```



5.230.1 Detailed Description

Implementation of the `TraverseTaxi` class.

This file contains the implementation of the `TraverseTaxi` class, which provides specific traversal logic for `Taxi` transportation elements.

Version

1.0

Date

2024-11-04

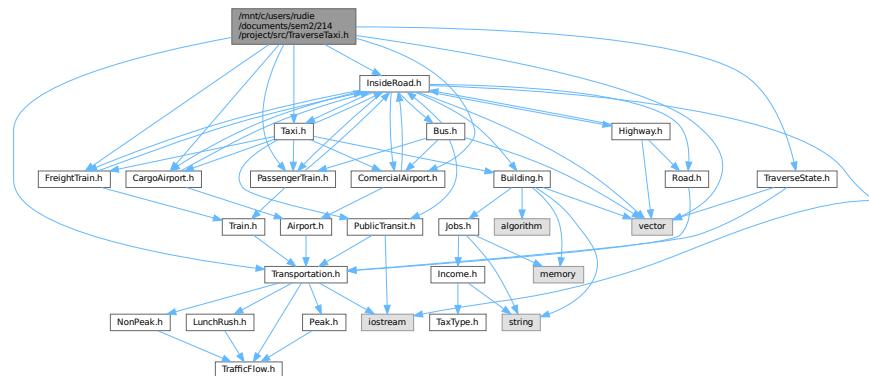
5.231 /mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.h File Reference

Header file for the `TraverseTaxi` class.

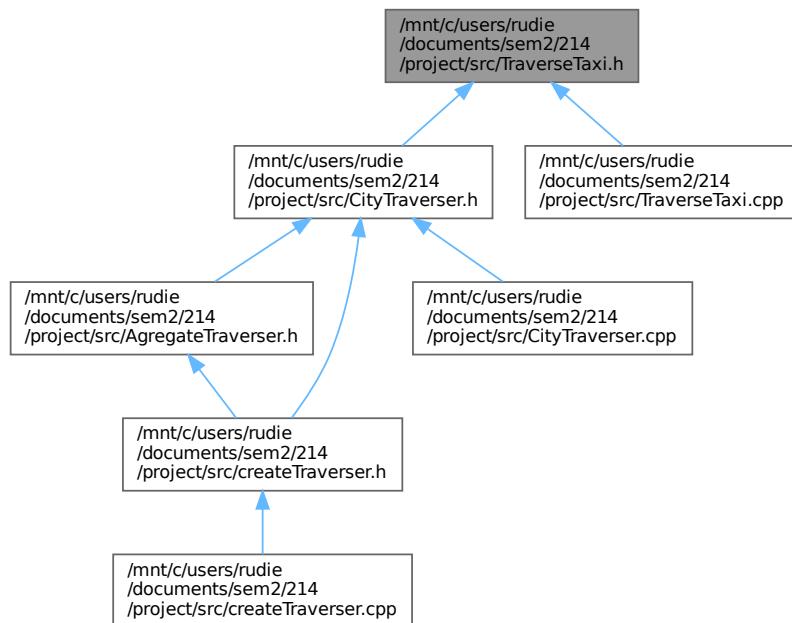
```
#include <vector>
#include "TraverseState.h"
#include "Transportation.h"
#include "InsideRoad.h"
#include "ComercialAirport.h"
#include "CargoAirport.h"
#include "PassengerTrain.h"
#include "FreightTrain.h"
```

```
#include "Taxi.h"
```

Include dependency graph for TraverseTaxi.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TraverseTaxi](#)

Manages the traversal state of a [Taxi](#) object.

5.231.1 Detailed Description

Header file for the [TraverseTaxi](#) class.

This file contains the definition of the [TraverseTaxi](#) class, which is a subclass of [TraverseState](#). The [TraverseTaxi](#) class is responsible for managing the traversal state of a [Taxi](#) object within a transportation system.

Version

1.0

Date

2024-11-04

5.232 TraverseTaxi.h

[Go to the documentation of this file.](#)

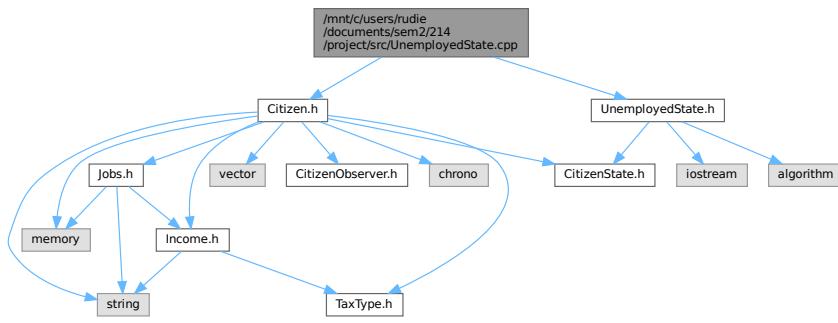
```
00001
00013 #ifndef TRAVERSETAXI_H
00014 #define TRAVERSETAXI_H
00015
00016 #include <vector>
00017
00018 #include "TraverseState.h"
00019 #include "Transportation.h"
00020 #include "InsideRoad.h"
00021 #include "ComercialAirport.h"
00022 #include "CargoAirport.h"
00023 #include "PassengerTrain.h"
00024 #include "FreightTrain.h"
00025 #include "Taxi.h"
00026
00027 class Transportation;
00028 class InsideRoad;
00029 class ComercialAirport;
00030 class CargoAirport;
00031 class PassengerTrain;
00032 class FreightTrain;
00033 class Taxi;
00034
00043 class TraverseTaxi: public TraverseState{
00044     private:
00045         int currentList = 0;
00046         int upperBound = 4;
00047
00048     public:
00049         TraverseTaxi(Transportation *element);
00054
00059         bool nextList();
00060
00065         bool prevList();
00066
00072         Transportation* getPos(size_t x);
00073 };
00074
00075 #endif // TRAVERSETAXI_H
```

5.233 /mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.cpp File Reference

Implementation of the [UnemployedState](#) class.

```
#include "UnemployedState.h"
#include "Citizen.h"
```

Include dependency graph for UnemployedState.cpp:



5.233.1 Detailed Description

Implementation of the `UnemployedState` class.

This file contains the implementation of the `UnemployedState` class, which handles the behavior of a citizen in an unemployed state.

Version

1.0

Date

2024-11-04

Note

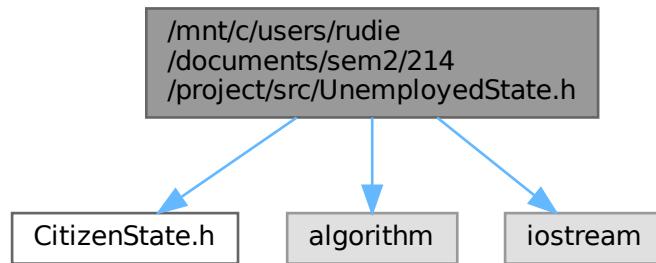
This file is part of a larger project that simulates the behavior of citizens in various states.

5.234 /mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.h File Reference

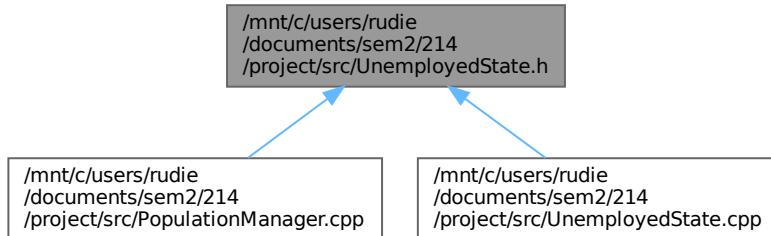
Declaration of the `UnemployedState` class.

```
#include "CitizenState.h"
#include <algorithm>
```

```
#include <iostream>
Include dependency graph for UnemployedState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [UnemployedState](#)
A class that represents the unemployed state of a citizen.

5.234.1 Detailed Description

Declaration of the [UnemployedState](#) class.

This file contains the declaration of the [UnemployedState](#) class, which handles the behavior of a citizen in an unemployed state.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens in various states.

5.235 UnemployedState.h

[Go to the documentation of this file.](#)

```

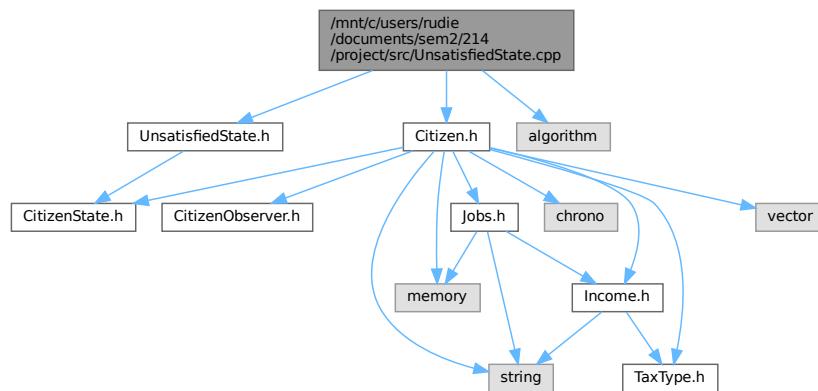
00001
00013 #ifndef UNEMPLOYEDSTATE_H
00014 #define UNEMPLOYEDSTATE_H
00015
00016 #include "CitizenState.h"
00017 #include <algorithm>
00018 #include <iostream>
00019
00026 class UnemployedState : public CitizenState {
00027 public:
00036     void handleState(Citizen& citizen) const override;
00037 };
00038
00039 #endif // UNEMPLOYEDSTATE_H

```

5.236 /mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.cpp File Reference

Implementation of the [UnsatisfiedState](#) class.

```
#include "UnsatisfiedState.h"
#include "Citizen.h"
#include <algorithm>
Include dependency graph for UnsatisfiedState.cpp:
```



5.236.1 Detailed Description

Implementation of the [UnsatisfiedState](#) class.

This file contains the implementation of the [UnsatisfiedState](#) class, which handles the behavior of a citizen in an unsatisfied state.

Version

1.0

Date

2024-11-04

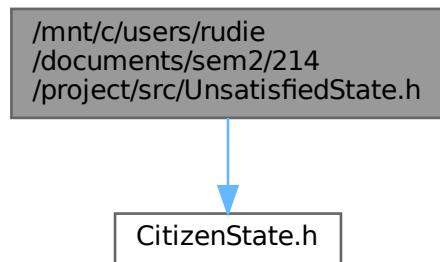
Note

This file is part of a larger project that simulates the behavior of citizens in various states.

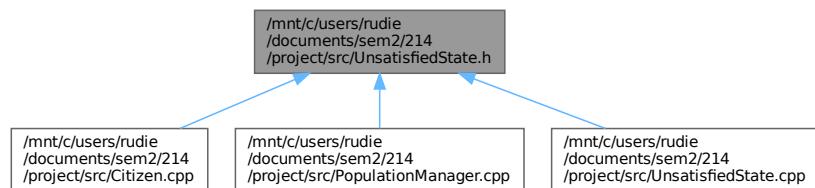
5.237 /mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.h File Reference

Declaration of the [UnsatisfiedState](#) class.

```
#include "CitizenState.h"  
Include dependency graph for UnsatisfiedState.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [UnsatisfiedState](#)
A class that represents the unsatisfied state of a citizen.

5.237.1 Detailed Description

Declaration of the [UnsatisfiedState](#) class.

This file contains the declaration of the [UnsatisfiedState](#) class, which handles the behavior of a citizen in an unsatisfied state.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens in various states.

5.238 UnsatisfiedState.h

[Go to the documentation of this file.](#)

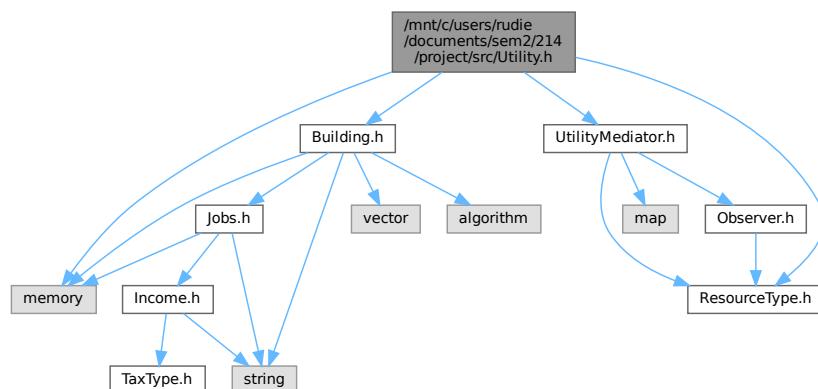
```
00001
00013 #ifndef UNSATISFIEDSTATE_H
00014 #define UNSATISFIEDSTATE_H
00015
00016 #include "CitizenState.h"
00017
00024 class UnsatisfiedState : public CitizenState {
00025 public:
00034     void handleState(Citizen& citizen) const override;
00035 };
00036
00037 #endif // UNSATISFIEDSTATE_H
```

5.239 /mnt/c/users/rudie/documents/sem2/214/project/src/Utility.h File Reference

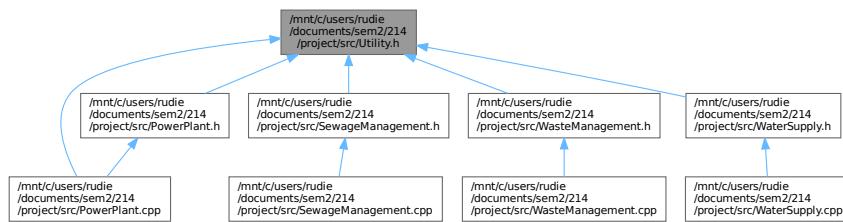
Declaration of the [Utility](#) class.

```
#include "Building.h"
#include "ResourceType.h"
#include "UtilityMediator.h"
#include <memory>
```

Include dependency graph for Utility.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Utility](#)

A class that represents a utility service in the city.

5.239.1 Detailed Description

Declaration of the [Utility](#) class.

This file contains the declaration of the [Utility](#) class, which handles the management of resources for buildings.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.240 Utility.h

[Go to the documentation of this file.](#)

```

00001
00013 #ifndef UTILITY_H
00014 #define UTILITY_H
00015
00016 #include "Building.h"
00017 #include "ResourceType.h"
00018 #include "UtilityMediator.h"
00019 #include <memory>
00020
00027 class Utility {
00028 protected:
00029     UtilityMediator* mediator;
00030
00031 public:
00037     Utility(UtilityMediator* mediator) : mediator(mediator) {}
00038
00044     virtual void registerBuilding(Building* building) = 0;
00045
00051     virtual void supplyResources(Building* building) = 0;
00052
00058     virtual void adjustForPopulation(int newPopulation) = 0;
00059
00063     virtual ~Utility() = default;
00064 };
00065
00066 #endif // UTILITY_H
  
```

5.241 UtilityManager.h

```

00001 #ifndef UTILITYMANAGER_H
00002 #define UTILITYMANAGER_H
00003
00004 // #include "CityGrowthObserver.h"
00005 // #include "Citizen.h"
00007
00008 // class UtilityManager : CityGrowthObserver {
00009
00010
00011 // public:
00012 // void update(int newPopulation);
00013 // };
00014
00015 #endif

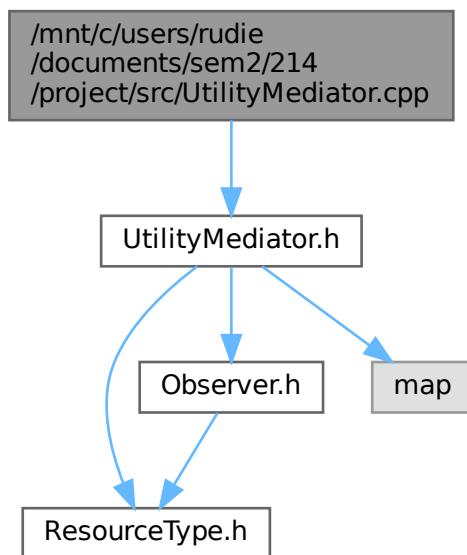
```

5.242 /mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.cpp File Reference

Implementation of the [UtilityMediator](#) class.

```
#include "UtilityMediator.h"
```

Include dependency graph for UtilityMediator.cpp:



5.242.1 Detailed Description

Implementation of the [UtilityMediator](#) class.

This file contains the implementation of the [UtilityMediator](#) class, which manages resource distribution for utilities.

Version

1.0

Date

2024-11-04

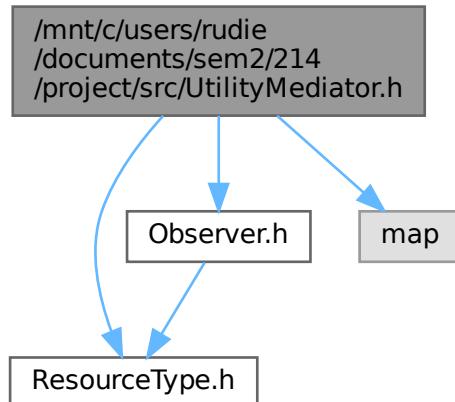
Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

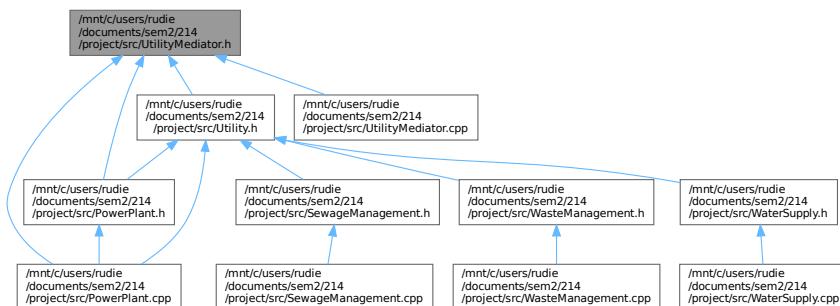
5.243 /mnt/c/users/rudie/documents/sem2/214/project/src/Utility Mediator.h File Reference

Declaration of the [UtilityMediator](#) class.

```
#include "ResourceType.h"
#include "Observer.h"
#include <map>
Include dependency graph for UtilityMediator.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [UtilityMediator](#)

A class that manages resource distribution for utilities.

5.243.1 Detailed Description

Declaration of the [UtilityMediator](#) class.

This file contains the declaration of the [UtilityMediator](#) class, which manages resource distribution for utilities.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.244 UtilityMediator.h

[Go to the documentation of this file.](#)

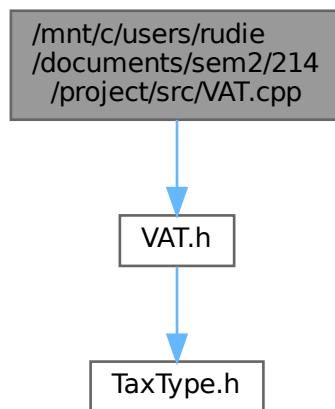
```
00001
00013 #ifndef UTILITYMEDIATOR_H
00014 #define UTILITYMEDIATOR_H
00015
00016 #include "ResourceType.h"
00017 #include "Observer.h"
00018 #include <map>
00019
00026 class UtilityMediator : public Observer {
00027 private:
00028     std::map<ResourceType, int> resourceInventory;
00029
00030 public:
00034     UtilityMediator() = default;
00035
00045     bool requestResources(ResourceType type, int amount);
00046
00055     void releaseResources(ResourceType type, int amount);
00056
00065     void produceResource(ResourceType type, int amount);
00066
00075     int getAvailableResource(ResourceType type) const;
00076
00085     void update(ResourceType type, int quantity) override;
00086 };
00087
00088 #endif // UTILITYMEDIATOR_H
```

5.245 /mnt/c/users/rudie/documents/sem2/214/project/src/VAT.cpp File Reference

Implementation of the [VAT](#) class.

```
#include "VAT.h"
```

Include dependency graph for VAT.cpp:



5.245.1 Detailed Description

Implementation of the [VAT](#) class.

This file contains the implementation of the [VAT](#) class, which handles the calculation and setting of Value Added Tax ([VAT](#)).

Version

1.0

Date

2024-11-04

Note

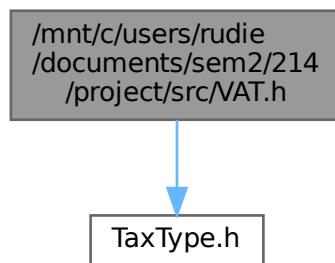
This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.246 /mnt/c/users/rudie/documents/sem2/214/project/src/VAT.h File Reference

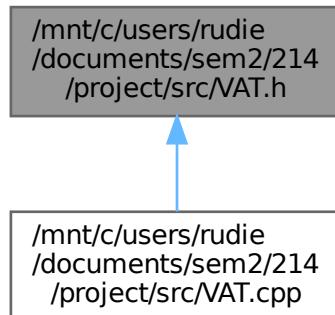
Declaration of the [VAT](#) class.

```
#include "TaxType.h"
```

Include dependency graph for VAT.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [VAT](#)

A class that represents Value Added Tax ([VAT](#)).

5.246.1 Detailed Description

Declaration of the [VAT](#) class.

This file contains the declaration of the [VAT](#) class, which handles the calculation and setting of Value Added Tax ([VAT](#)).

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.247 VAT.h

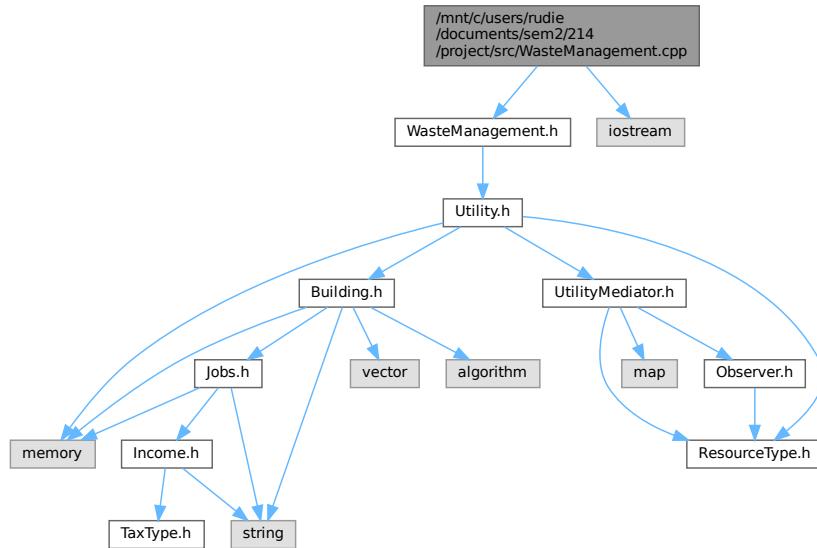
[Go to the documentation of this file.](#)

```
00001  
00013 #ifndef VAT_H  
00014 #define VAT_H  
00015  
00016 #include "TaxType.h"  
00017  
00024 class VAT : public TaxType {  
00025 private:  
00026     double vat;  
00027     char cType;  
00028  
00029 public:  
00037     void setTax(double rate);  
00038  
00044     void calculateTax();  
00045 };  
00046  
00047 #endif // VAT_H
```

5.248 /mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.cpp File Reference

Implementation of the [WasteManagement](#) class.

```
#include "WasteManagement.h"
#include <iostream>
Include dependency graph for WasteManagement.cpp:
```



5.248.1 Detailed Description

Implementation of the [WasteManagement](#) class.

This file contains the implementation of the [WasteManagement](#) class, which handles waste management services for buildings.

Version

1.0

Date

2024-11-04

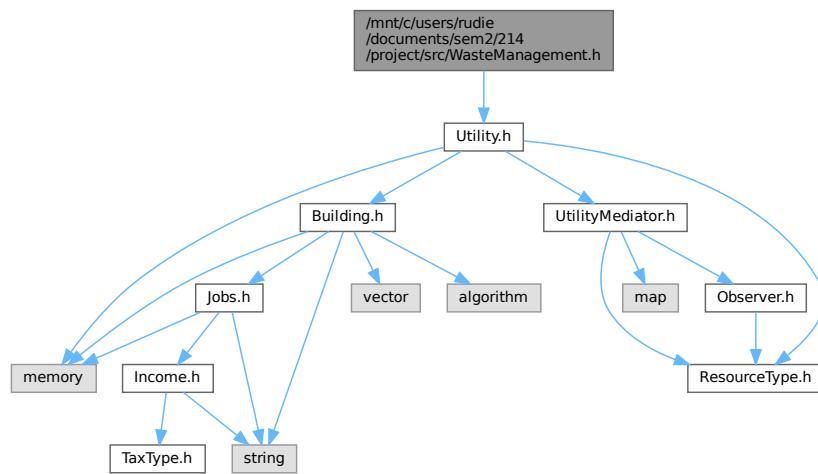
Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

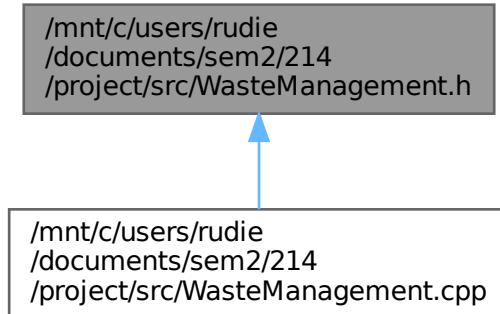
5.249 /mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.h File Reference

Declaration of the [WasteManagement](#) class.

```
#include "Utility.h"
Include dependency graph for WasteManagement.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [WasteManagement](#)
A class that represents waste management services in the city.

5.249.1 Detailed Description

Declaration of the [WasteManagement](#) class.

This file contains the declaration of the [WasteManagement](#) class, which handles waste management services for buildings.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.250 WasteManagement.h

[Go to the documentation of this file.](#)

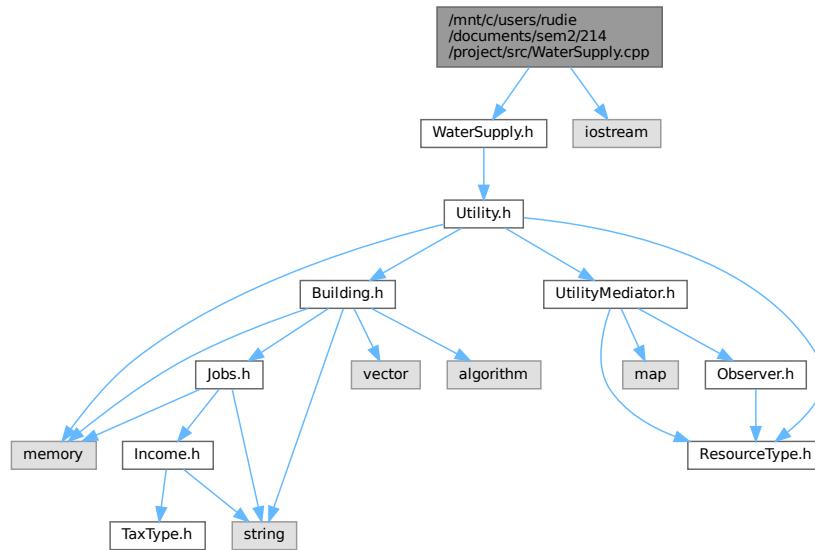
```
00001
00013 #ifndef WASTEMANAGEMENT_H
00014 #define WASTEMANAGEMENT_H
00015
00016 #include "Utility.h"
00017
00024 class WasteManagement : public Utility {
00025
00026 public:
00032     WasteManagement(UtilityMediator* mediator);
00033
00039     void registerBuilding(Building* building) override;
00040
00046     void supplyResources(Building* building) override;
00047
00053     void adjustForPopulation(int newPopulation) override;
00054
00055     // /**
00056     // * @brief Adjusts waste management services based on a citizen's requirements.
00057     // *
00058     // * @param citizen A pointer to the citizen whose requirements are being considered.
00059     // */
00060     // void adjustForCitizen(Citizen* citizen) override;
00061 };
00062
00063 #endif // WASTEMANAGEMENT_H
```

5.251 /mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.cpp File Reference

Implementation of the [WaterSupply](#) class.

```
#include "WaterSupply.h"
#include <iostream>
```

Include dependency graph for WaterSupply.cpp:



5.251.1 Detailed Description

Implementation of the [WaterSupply](#) class.

This file contains the implementation of the [WaterSupply](#) class, which handles water supply services for buildings.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

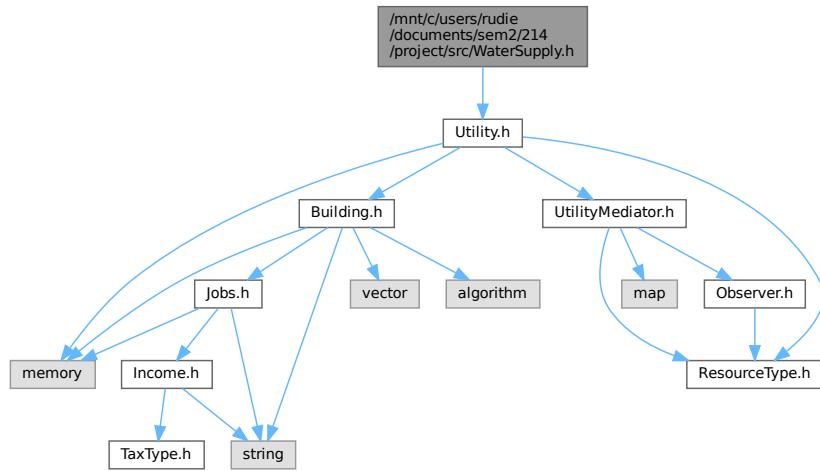
5.252 /mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.h

File Reference

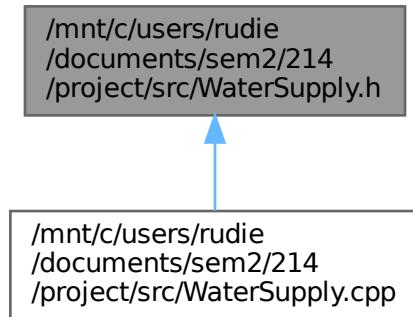
Declaration of the [WaterSupply](#) class.

```
#include "Utility.h"
```

Include dependency graph for WaterSupply.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [WaterSupply](#)
A class that represents water supply services in the city.

5.252.1 Detailed Description

Declaration of the [WaterSupply](#) class.

This file contains the declaration of the [WaterSupply](#) class, which handles water supply services for buildings.

Version

1.0

Date

2024-11-04

Note

This file is part of a larger project that simulates the behavior of citizens and buildings in a city.

5.253 WaterSupply.h

[Go to the documentation of this file.](#)

```
00001
00013 #ifndef WATERSUPPLY_H
00014 #define WATERSUPPLY_H
00015
00016 #include "Utility.h"
00017
00024 class WaterSupply : public Utility {
00025
00026 public:
00032     WaterSupply(UtilityMediator* mediator);
00033
00039     void registerBuilding(Building* building) override;
00040
00046     void supplyResources(Building* building) override;
00047
00053     void adjustForPopulation(int newPopulation) override;
00054
00055     // /**
00056     // * @brief Adjusts water supply services based on a citizen's requirements.
00057     // *
00058     // * @param citizen A pointer to the citizen whose requirements are being considered.
00059     // */
00060     // void adjustForCitizen(Citizen* citizen) override;
00061 };
00062
00063 #endif // WATERSUPPLY_H
```

Index

/mnt/c/users/rudie/documents/sem2/214/project/src/AggregateTaxesCommand.cpp, 415, 417

/mnt/c/users/rudie/documents/sem2/214/project/src/Airport.cpp, 417

/mnt/c/users/rudie/documents/sem2/214/project/src/Airport.h, 418, 419

/mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.cpp, 420

/mnt/c/users/rudie/documents/sem2/214/project/src/AllocateBudgetCommand.h, 421, 422

/mnt/c/users/rudie/documents/sem2/214/project/src/BudgetAllocationHandler.cpp, 423, 424

/mnt/c/users/rudie/documents/sem2/214/project/src/Builder.cpp, 424

/mnt/c/users/rudie/documents/sem2/214/project/src/Builder.h, 426, 427

/mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.cpp, 428, 429

/mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.h, 430

/mnt/c/users/rudie/documents/sem2/214/project/src/BuildingManager.cpp, 431, 432

/mnt/c/users/rudie/documents/sem2/214/project/src/Bus.cpp, 433

/mnt/c/users/rudie/documents/sem2/214/project/src/Bus.h, 434, 435

/mnt/c/users/rudie/documents/sem2/214/project/src/BusinessCommand.cpp, 436

/mnt/c/users/rudie/documents/sem2/214/project/src/BusinessCommand.h, 437, 438

/mnt/c/users/rudie/documents/sem2/214/project/src/CargoAirport.cpp, 439, 440

/mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.cpp, 440

/mnt/c/users/rudie/documents/sem2/214/project/src/Citizen.h, 442, 443

/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenObserver.h, 445, 446

/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenStateObserver.h, 446, 447

/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenStateObserver.cpp, 447

/mnt/c/users/rudie/documents/sem2/214/project/src/CitizenStateObserver.h, 448, 449

/mnt/c/users/rudie/documents/sem2/214/project/src/CityServiceCommand.cpp, 449

/mnt/c/users/rudie/documents/sem2/214/project/src/CityServiceCommand.h, 450, 451

TaxesCommand.cpp, 451

TaxesCommand.h, 452, 454

CollectTaxesCommand.cpp, 454

CollectTaxesCommand.h, 455, 456

ComercialAirport.cpp, 457

ComercialAirport.h, 458, 459

CommercialBuilding.cpp, 459

CommercialBuilding.h, 460, 461

CommercialBuildingBuilder.cpp, 462

CommercialBuildingBuilder.h, 463, 464

Director.cpp, 467

Director.h, 468

EmployedState.cpp, 468

EmployedState.h, 469, 471

EnforcePolicyCommand.cpp, 471

EnforcePolicyCommand.h, 472, 473

FemaleCitizen.h, 474, 475

FreightTrain.cpp, 475

FreightTrain.h, 476, 477

GovCommand.cpp, 478

GovCommand.h, 478, 480

GovObserver.cpp, 483

GovObserver.h, 483, 484

Government.cpp, 480

/mnt/c/users/rudie/documents/sem2/214/project/src/Government.cpp, 481, 482
/mnt/c/users/rudie/documents/sem2/214/project/src/Highway.cpp, 484
/mnt/c/users/rudie/documents/sem2/214/project/src/Highway.h, 485, 487
/mnt/c/users/rudie/documents/sem2/214/project/src/Housing.cpp, 487
/mnt/c/users/rudie/documents/sem2/214/project/src/Housing.h, 488, 490
/mnt/c/users/rudie/documents/sem2/214/project/src/Income.cpp, 491, 492
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.cpp, 492
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuilding.h, 493, 495
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingManager.cpp, 495
/mnt/c/users/rudie/documents/sem2/214/project/src/IndustrialBuildingManager.h, 496, 498
/mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.cpp, 498
/mnt/c/users/rudie/documents/sem2/214/project/src/InsideRoad.h, 499, 501
/mnt/c/users/rudie/documents/sem2/214/project/src/JobSaturationStrategy.cpp, 504
/mnt/c/users/rudie/documents/sem2/214/project/src/JobSaturationStrategy.h, 505, 507
/mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.cpp, 502
/mnt/c/users/rudie/documents/sem2/214/project/src/Jobs.h, 503, 504
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.cpp, 507
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilder.h, 508, 510
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilderManager.cpp, 510
/mnt/c/users/rudie/documents/sem2/214/project/src/LandmarkBuildingBuilderManager.h, 511, 513
/mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityStrategy.cpp, 513
/mnt/c/users/rudie/documents/sem2/214/project/src/LeavingCityStrategy.h, 514, 515
/mnt/c/users/rudie/documents/sem2/214/project/src/LunchPoster.cpp, 516
/mnt/c/users/rudie/documents/sem2/214/project/src/LunchPoster.h, 516, 518
/mnt/c/users/rudie/documents/sem2/214/project/src/MaleCitizen.cpp, 518, 519
/mnt/c/users/rudie/documents/sem2/214/project/src/NonPeakHour.cpp, 520
/mnt/c/users/rudie/documents/sem2/214/project/src/NonPeakHour.h, 520, 522
/mnt/c/users/rudie/documents/sem2/214/project/src/Observation.cpp, 522, 523
/mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.cpp, 523
/mnt/c/users/rudie/documents/sem2/214/project/src/PassengerTrain.h, 524, 525
/mnt/c/users/rudie/documents/sem2/214/project/src/Peak.cpp, 526
/mnt/c/users/rudie/documents/sem2/214/project/src/Peak.h, 526, 528
/mnt/c/users/rudie/documents/sem2/214/project/src/Policy.cpp, 528
/mnt/c/users/rudie/documents/sem2/214/project/src/Policy.h, 529, 530
/mnt/c/users/rudie/documents/sem2/214/project/src/PolicyCheckerHandler.cpp, 530, 531
/mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.cpp, 531
/mnt/c/users/rudie/documents/sem2/214/project/src/PopulationManager.h, 532, 534
/mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.cpp, 534
/mnt/c/users/rudie/documents/sem2/214/project/src/PowerPlant.h, 535, 537
/mnt/c/users/rudie/documents/sem2/214/project/src/Property.cpp, 537
/mnt/c/users/rudie/documents/sem2/214/project/src/Property.h, 538, 539
/mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.cpp, 539
/mnt/c/users/rudie/documents/sem2/214/project/src/PublicTransit.h, 540, 542
/mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.cpp, 542
/mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuilding.h, 543, 544
/mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.cpp, 545
/mnt/c/users/rudie/documents/sem2/214/project/src/ResidentialBuildingBuilder.h, 545, 547
/mnt/c/users/rudie/documents/sem2/214/project/src/Resource.h, 547, 549
/mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAllocationStrategy.cpp, 549, 551
/mnt/c/users/rudie/documents/sem2/214/project/src/ResourceApprovalHandler.cpp, 551, 552
/mnt/c/users/rudie/documents/sem2/214/project/src/ResourceAvailabilityHandler.cpp, 553, 554
/mnt/c/users/rudie/documents/sem2/214/project/src/ResourceManager.cpp, 554, 555
/mnt/c/users/rudie/documents/sem2/214/project/src/ResourceManager.h, 556, 558
/mnt/c/users/rudie/documents/sem2/214/project/src/Road.cpp, 558
/mnt/c/users/rudie/documents/sem2/214/project/src/Road.h, 559, 560
/mnt/c/users/rudie/documents/sem2/214/project/src/Sales.cpp, 560
/mnt/c/users/rudie/documents/sem2/214/project/src/Sales.h, 561, 563

/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfactionStrategy.h
563, 564

/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.cpp
565

/mnt/c/users/rudie/documents/sem2/214/project/src/SatisfiedState.h
565, 567

/mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxContoller.h
567

/mnt/c/users/rudie/documents/sem2/214/project/src/SetTaxContoller.cpp
568, 569

/mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.h
569

/mnt/c/users/rudie/documents/sem2/214/project/src/SewageManagement.cpp
570, 571

/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.h
574

/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSatisfactionStrategy.cpp
575

/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.cpp
576

/mnt/c/users/rudie/documents/sem2/214/project/src/TaxSystem.h
577

/mnt/c/users/rudie/documents/sem2/214/project/src/TaxType.cpp
578

/mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.cpp
579

/mnt/c/users/rudie/documents/sem2/214/project/src/Taxi.h
580, 582

/mnt/c/users/rudie/documents/sem2/214/project/src/TrafficFront.h
580, 590

/mnt/c/users/rudie/documents/sem2/214/project/src/Train.cpp
583, 584

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.h
584

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportManager.cpp
585, 587

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.h
585, 587

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.cpp
588, 590

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.h
591, 593

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.cpp
593

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.h
594, 596

/mnt/c/users/rudie/documents/sem2/214/project/src/TransportFactory.cpp
596

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseComercialAir.h
597, 599

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.h
599

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseFreightTrain.h
600, 602

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.h
602

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseHighway.h
603, 605

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.h
605

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseInsideRoad.h
606, 608

/mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.h
608

/mnt/c/users/rudie/documents/sem2/214/project/src/TraversePassengerTrain.h
609, 611

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.cpp
613

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseState.h
614, 615

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.cpp
615

/mnt/c/users/rudie/documents/sem2/214/project/src/TraverseTaxi.h
616, 618

/mnt/c/users/rudie/documents/sem2/214/project/src/Traverser.h
611, 613

/mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.cpp
618

/mnt/c/users/rudie/documents/sem2/214/project/src/UnemployedState.h
619, 621

/mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.cpp
621

/mnt/c/users/rudie/documents/sem2/214/project/src/UnsatisfiedState.h
622, 623

/mnt/c/users/rudie/documents/sem2/214/project/src/Utility.h
623, 624

/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityManager.h
625

/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.cpp
625

/mnt/c/users/rudie/documents/sem2/214/project/src/UtilityMediator.h
626, 627

/mnt/c/users/rudie/documents/sem2/214/project/src/VAT.cpp
628

/mnt/c/users/rudie/documents/sem2/214/project/src/VAT.h
629, 630

/mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.cpp
630

/mnt/c/users/rudie/documents/sem2/214/project/src/WasteManagement.h
631, 633

/mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.cpp
633

/mnt/c/users/rudie/documents/sem2/214/project/src/WaterSupply.h
634, 636

/mnt/c/users/rudie/documents/sem2/214/project/src/createTraverser.cpp
465

/mnt/c/users/rudie/documents/sem2/214/project/src/createTraversee
 ~GovCommand
 ~GovObserver
 ~TransportManager
 ~Transportation
 addBonus
 addBuilding
 addBus
 addCargoAirport
 addCitizen
 addComercialAirport
 addFreightTrain
 addGovernment
 addHighway
 addIncomeTaxBuilding
 addInsideRoad
 addJob
 addObserver
 addPassengerTrain
 addPolicy
 addPropertyTaxBuilding
 addResidents
 addResource
 addRoad
 addSalesTaxBuilding
 addSatisfactionStrategy
 addService
 addTaxesToBudget
 addTaxi
 addTaxRate
 addVATTaxPayer
 adjustForPopulation
 aggregateTraverser
 Airport
 allocate
 allocateAdditionalBudget
 allocateBudget
 AllocateBudgetCommand
 canExecute
 execute
 executeWithValidation
 getAmount
 getDescription
 getName
 getPrevAllocation
 returnVal
 setAmount
 setPrevAllocation
 undo

PassengerTrain, 244
 Taxi, 318
 Business, 67
 TaxSystem, 329
 ResidentialBuilding, 274
 ResourceManager, 290
 ComercialAirport, 118
 TaxSystem, 329
 Citizen, 84
 Business, 68
 Government, 163
 InsideRoad, 205
 TaxSystem, 329
 Utility, 398
 WasteManagement, 408
 WaterSupply, 412
 createCityTraverser, 18, 19
 Airport, 22
 calculateCommute, 22
 getAirportName, 22
 EqualDistributionStrategy, 146
 PriorityDistributionStrategy, 260
 Resource, 282
 ResourceAllocationStrategy, 284
 CityService, 98
 Government, 163
 AllocateBudgetCommand, 23
 AllocateBudgetCommand, 25
 execute
 executeWithValidation, 26
 getAmount, 26
 getDescription, 26
 getName, 27
 getPrevAllocation, 27
 returnVal, 27
 setAmount, 27
 setPrevAllocation, 28
 undo, 28

validateAllocation, 28
allocateResources
 ResourceManager, 290
applyDeductions
 Income, 182
assignJobToCitizen
 BuildingManager, 52

Budget
 ResourceType.h, 557
BudgetApprovalHandler, 29
 BudgetApprovalHandler, 31
 handleRequest, 31

build
 Builder, 34
 CommercialBuildingBuilder, 131
 IndustrialBuildingBuilder, 195
 LandmarkBuildingBuilder, 227
 ResidentialBuildingBuilder, 280

Builder, 32
 build, 34
 setArea, 34
 setCapacity, 35
 setCitizenSatisfaction, 35
 setEconomicGrowth, 35
 setFloors, 36
 setName, 36
 setResourceConsumption, 36

Building, 37
 addJob, 43
 Building, 41, 42
 construct, 43
 displayJobInfo, 43, 44
 getAvailableJob, 44
 getEconomicGrowth, 44
 getJobs, 45
 getName, 45
 getResourceConsumption, 45, 46
 getSatisfaction, 46
 getType, 46
 hireEmployee, 47
 payTaxes, 47, 48
 releaseEmployee, 48
 setName, 49
 undoCollectTaxes, 49
 updateImpacts, 49

BuildingManager, 50
 addBuilding, 51
 addCitizen, 51
 assignJobToCitizen, 52
 BuildingManager, 51
 findAvailableJob, 52
 getBuildings, 52
 listAllJobsInBuilding, 52
 releaseCitizenFromJob, 54

Bus, 54
 addBus, 58
 addComercialAirport, 59
 addInsideRoad, 59

addPassengerTrain, 60
Bus, 58
getBus, 60
getBusNumber, 61
getCapacity, 61
getComercialAirport, 61
getInsideRoad, 62
getPassengerTrain, 62
getRouteName, 63

Business, 63
 addPolicy, 67
 addService, 68
 Business, 67
 calculateTax, 68
 payTax, 68, 70
 payTaxes, 70
 removePolicy, 70, 71
 removeService, 71
 setTaxCooldownPeriod, 71, 72
 updatePolicy, 72
 updateServices, 72, 73
 updateTaxRate, 73

calculateCommute
 Airport, 22
 PublicTransit, 267
 Road, 295
 Train, 339

calculateEconomicImpact
 CommercialBuilding, 126
 IndustrialBuilding, 189
 LandmarkBuilding, 222
 ResidentialBuilding, 274

calculateMonthlyIncome
 Income, 182

calculatePropertyTax
 ResidentialBuilding, 274

calculateResourceConsumption
 CommercialBuilding, 126
 IndustrialBuilding, 190
 LandmarkBuilding, 222
 ResidentialBuilding, 275

calculateSatisfaction
 HousingSatisfactionStrategy, 177
 JobSatisfactionStrategy, 215
 SatisfactionStrategy, 301
 TaxSatisfactionStrategy, 325

calculateSatisfactionImpact
 CommercialBuilding, 126
 IndustrialBuilding, 190
 LandmarkBuilding, 222
 ResidentialBuilding, 275

calculateTax
 Business, 68
 Income, 182
 Property, 263
 Sales, 298
 TaxType, 334
 VAT, 405

canExecute
 AllocateBudgetCommand, 26
 CollectTaxesCommand, 112
 EnforcePolicyCommand, 142
 GovCommand, 159
 SetTaxCommand, 307
 CargoAirport, 74
 addCargoAirport, 78
 addInsideRoad, 78
 CargoAirport, 77
 getCargoAirport, 78
 getInsideRoad, 79
 getName, 79
 checkAvailability
 ResourceAvailability, 288
 checkImpact
 TaxSystem, 330
 Citizen, 79
 addObserver, 84
 addSatisfactionStrategy, 84
 Citizen, 83
 clone, 84
 getAge, 84
 getBankBalance, 84
 getJob, 85
 getjobobj, 85
 getMarriageDuration, 85
 getName, 85
 getRelationshipStatus, 85
 getSatisfactionLevel, 86
 getTaxCooldown, 86
 getTaxRate, 86
 increaseBankBalance, 86
 isEmployed, 87
 isLeaving, 87
 isOnCooldown, 87
 payTaxes, 87
 removeObserver, 88
 searchAndApplyForJob, 88
 setBankBalance, 88
 setEmployed, 88
 setIncome, 89
 setJob, 89
 setJobTitle, 89
 setRelationshipStatus, 89
 setSatisfactionLevel, 90
 setState, 90
 setTaxCooldown, 90
 setTaxRate, 90
 subtractBankBalance, 91
 updateSatisfaction, 91
 Citizen.cpp
 minMax, 441
 CitizenObserver, 91
 update, 93
 CitizenSatisfactionObserver, 93
 update, 94
 CitizenState, 95
 handleState, 96
 CityService, 96
 allocateAdditionalBudget, 98
 CityService, 98
 getBudgetAllocated, 98
 getServiceName, 99
 isWithinBudget, 99
 provideService, 99
 reduceBudget, 99
 setServiceName, 100
 updateBudget, 100
 CityTraverser, 100
 CityTraverser, 103, 104
 getCurrentLayer, 104
 operator+, 104
 operator++, 105
 operator-, 105
 operator--, 105
 operator=, 106
 operator*, 104
 setState, 107
 stepIn, 107
 stepOut, 107
 clamp
 EmployedState.cpp, 469
 clone
 Citizen, 84
 FemaleCitizen, 151
 MaleCitizen, 237
 collectTaxes
 Government, 163
 TaxSystem, 330
 CollectTaxesCommand, 108
 canExecute, 112
 CollectTaxesCommand, 111
 execute, 112
 executeWithValidation, 112
 getCollectedTaxes, 112
 getDescription, 112
 getName, 113
 getTaxesCollected, 113
 returnVal, 113
 setCollectedTaxes, 113
 undo, 114
 validateCollection, 114
 ComercialAirport, 114
 addComercialAirport, 118
 addRoad, 118
 ComercialAirport, 117
 getComercialAirport, 119
 getName, 119
 getRoad, 120
 CommercialBuilding, 120
 calculateEconomicImpact, 126
 calculateResourceConsumption, 126
 calculateSatisfactionImpact, 126
 CommercialBuilding, 125
 construct, 126

getType, 126
 payTaxes, 126
 setBusiness, 127
 undoCollectTaxes, 127
 updateCustomer, 127
 updateImpacts, 127
CommercialBuildingBuilder, 128
 build, 131
 getBusinessUnits, 131
 getCustomerTraffic, 131
 setBusinessUnits, 131
 setCustomerTraffic, 132
construct
 Building, 43
 CommercialBuilding, 126
 IndustrialBuilding, 190
 LandmarkBuilding, 222
 ResidentialBuilding, 275
constructLargeBuilding
 Director, 137
constructMediumBuilding
 Director, 137
constructSmallBuilding
 Director, 137
createBus
 TransportationFactory, 344
 TransportManager, 350
createCargoAirport
 TransportationFactory, 344
 TransportManager, 351
createCityTraverser
 AggregateTraverser, 18, 19
 CreateTraverser, 134, 135
createComercialAirport
 TransportationFactory, 345
 TransportManager, 351
createFreightTrain
 TransportationFactory, 345
 TransportManager, 352
createHighway
 TransportationFactory, 346
 TransportManager, 353
createInsideRoad
 TransportationFactory, 347
 TransportManager, 353
createPassengerTrain
 TransportationFactory, 347
 TransportManager, 354
createTaxi
 TransportationFactory, 348
 TransportManager, 354
CreateTraverser, 132
 createCityTraverser, 134, 135
Director, 136
 constructLargeBuilding, 137
 constructMediumBuilding, 137
 constructSmallBuilding, 137
 setBuilder, 137
displayJobInfo
 Building, 43, 44
EmployedState, 138
 handleState, 139
EmployedState.cpp
 clamp, 469
enforcePolicy
 Government, 163
EnforcePolicyCommand, 140
 canExecute, 142
 EnforcePolicyCommand, 142
 execute, 143
 executeWithValidation, 143
 getDescription, 143
 getName, 143
 isPolicyEnforced, 143
 returnVal, 144
 setPolicyEnforced, 144
 undo, 144
 validateEnforcement, 144
EqualDistributionStrategy, 145
 allocate, 146
execute
 AllocateBudgetCommand, 26
 CollectTaxesCommand, 112
 EnforcePolicyCommand, 143
 GovCommand, 159
 SetTaxCommand, 307
executeWithValidation
 AllocateBudgetCommand, 26
 CollectTaxesCommand, 112
 EnforcePolicyCommand, 143
FemaleCitizen, 146
 clone, 151
 FemaleCitizen, 150
findAvailableJob
 BuildingManager, 52
findJobsForUnemployedCitizens
 PopulationManager, 255
FreightTrain, 151
 addFreightTrain, 155
 addInsideRoad, 155
 FreightTrain, 154
 getFreightTrain, 156
 getInsideRoad, 156
 getLength, 157
 getTrainLine, 157
 getWeight, 157
getAge
 Citizen, 84
getAirportName
 Airport, 22
getAmount
 AllocateBudgetCommand, 26
getAvailableJob
 Building, 44

getAvailableResource
 UtilityMediator, 401
getAvgStopTime
 InsideRoad, 205
getBankBalance
 Citizen, 84
getBaseSalary
 Income, 183
getBonus
 Income, 183
getBudget
 Government, 163
 ResourceManager, 291
getBudgetAllocated
 CityService, 98
getBuilding
 InsideRoad, 205
 Taxi, 318
getBuildings
 BuildingManager, 52
getBus
 Bus, 60
 InsideRoad, 206
getBusinessUnits
 CommercialBuildingBuilder, 131
getBusNumber
 Bus, 61
getCapacity
 Bus, 61
getCargoAirport
 CargoAirport, 78
 InsideRoad, 207
 Taxi, 319
getCitizens
 PopulationManager, 255
getCollectedTaxes
 CollectTaxesCommand, 112
getComercialAirport
 Bus, 61
 ComercialAirport, 119
 InsideRoad, 207
 Taxi, 319
getComfort
 ResidentialBuildingBuilder, 280
getCulturalValue
 LandmarkBuildingBuilder, 227
getCurrentLayer
 CityTraverser, 104
getCustomerTraffic
 CommercialBuildingBuilder, 131
getDeductions
 Income, 183
getDescription
 AllocateBudgetCommand, 26
 CollectTaxesCommand, 112
 EnforcePolicyCommand, 143
 GovCommand, 160
 SetTaxCommand, 307
getEconomicGrowth
 Building, 44
getFreightTrain
 FreightTrain, 156
 InsideRoad, 208
 Taxi, 320
getHighway
 Highway, 173
 InsideRoad, 208
getHighwaysList
 Highway, 173
getImpactLevel
 Policy, 249
getIncome
 Jobs, 212
getInsideRoad
 Bus, 62
 CargoAirport, 79
 FreightTrain, 156
 Highway, 173
 InsideRoad, 209
 PassengerTrain, 245
 Taxi, 320
getInsideRoadsList
 Highway, 174
getIsHistoric
 LandmarkBuildingBuilder, 227
getJob
 Citizen, 85
getjobobj
 Citizen, 85
getJobs
 Building, 45
getLayer
 TraverseState, 388
getLength
 FreightTrain, 157
getLine
 Train, 339
getMarriageDuration
 Citizen, 85
getName
 AllocateBudgetCommand, 27
 Building, 45
 CargoAirport, 79
 Citizen, 85
 CollectTaxesCommand, 113
 ComercialAirport, 119
 EnforcePolicyCommand, 143
 GovCommand, 160
 Road, 295
 SetTaxCommand, 307
getPassengerTrain
 Bus, 62
 InsideRoad, 209
 PassengerTrain, 245
 Taxi, 321
getPolicyName

Policy, 249
getPollutionLevel
 IndustrialBuildingBuilder, 195
getPopulation
 PopulationManager, 256
getPos
 TraverseBus, 358
 TraverseCargoAirport, 362
 TraverseComercialAirport, 366
 TraverseFreightTrain, 370
 TraverseHighway, 374
 TraverseInsideRoad, 378
 TraversePassengerTrain, 382
 TraverseState, 388
 TraverseTaxi, 391
getPrevAllocation
 AllocateBudgetCommand, 27
getProductionCapacity
 IndustrialBuildingBuilder, 195
getQuantity
 Resource, 283
getResourceStatus
 Citizen, 85
getResidentialUnit
 ResidentialBuildingBuilder, 280
getResidentialUnits
 ResidentialBuilding, 275
getResource
 ResourceManager, 291
getResourceConsumption
 Building, 45, 46
getRoad
 ComercialAirport, 120
getRoadName
 Highway, 174
 InsideRoad, 210
getRoute
 PublicTransit, 267
getRouteName
 Bus, 63
 Taxi, 321
getSatisfaction
 Building, 46
getSatisfactionLevel
 Citizen, 86
getServiceName
 CityService, 99
getSpeedLimit
 Highway, 174
getState
 LunchRush, 232
 NonPeak, 238
 Peak, 247
 TrafficFlow, 336
getTaxCooldown
 Citizen, 86
getTaxesCollected
 CollectTaxesCommand, 113
getTaxi
 InsideRoad, 210
getTaxiCompany
 Taxi, 322
getTaxiNumber
 Taxi, 322
getTaxRate
 Citizen, 86
 Government, 164
 TaxType, 334
getTaxType
 TaxType, 334
getTitle
 Jobs, 212
getTrafficFlow
 LunchRush, 232
 NonPeak, 238
 Peak, 247
 TrafficFlow, 336
 Transportation, 341
getTrainLine
 FreightTrain, 157
 PassengerTrain, 246
getTransportation
 TransportManager, 355
getType
 Building, 46
 CommercialBuilding, 126
 IndustrialBuilding, 190
 LandmarkBuilding, 222
 ResidentialBuilding, 275
 Resource, 283
 Transportation, 341
getVisitorCapacity
 LandmarkBuildingBuilder, 227
getWeight
 FreightTrain, 157
GovCommand, 158
 ~GovCommand, 159
 canExecute, 159
 execute, 159
 getDescription, 160
 getName, 160
 returnVal, 160
 undo, 160
Government, 161
 addTaxesToBudget, 163
 allocateBudget, 163
 collectTaxes, 163
 enforcePolicy, 163
 getBudget, 163
 getTaxRate, 164
 Government, 162
 refundTaxes, 164
 registerObserver, 164
 revertBudgetAllocation, 164
 setTax, 165
 unregisterObserver, 165

update, 165
GovObserver, 166
 ~GovObserver, 167
 updatePolicy, 167
 updateServices, 168
 updateTaxRate, 168

handleRequest
 BudgetApprovalHandler, 31
 PolicyCheckerHandler, 253
 ResourceApprovalHandler, 286
handleState
 CitizenState, 96
 EmployedState, 139
 LeavingCityState, 230
 SatisfiedState, 303
 UnemployedState, 393
 UnsatisfiedState, 395
Highway, 169
 addHighway, 172
 addInsideRoad, 172
 getHighway, 173
 getHighwaysList, 173
 getInsideRoad, 173
 getInsideRoadsList, 174
 getRoadName, 174
 getSpeedLimit, 174
 Highway, 171
hireEmployee
 Building, 47
hostEvent
 LandmarkBuilding, 222
HousingSatisfactionStrategy, 175
 calculateSatisfaction, 177
 updateForHousingChange, 177
 updateForJobChange, 177
 updateForTaxChange, 177

implement
 Policy, 250
Income, 178
 addBonus, 182
 applyDeductions, 182
 calculateMonthlyIncome, 182
 calculateTax, 182
 getBaseSalary, 183
 getBonus, 183
 getDeductions, 183
 Income, 181
 payTaxes, 183
 setTax, 184
increaseBankBalance
 Citizen, 86
IndustrialBuilding, 184
 calculateEconomicImpact, 189
 calculateResourceConsumption, 190
 calculateSatisfactionImpact, 190
 construct, 190
 getType, 190

 IndustrialBuilding, 189
 payTaxes, 190
 undoCollectTaxes, 191
 updateImpacts, 191
 upgradeTech, 191
IndustrialBuildingBuilder, 191
 build, 195
 getPollutionLevel, 195
 getProductionCapacity, 195
 setPollutionLevel, 195
 setProductionCapacity, 196
InsideRoad, 196
 addBuilding, 200
 addBus, 201
 addCargoAirport, 201
 addComercialAirport, 202
 addFreightTrain, 203
 addHighway, 203
 addInsideRoad, 204
 addPassengerTrain, 204
 addTaxi, 205
 getAvgStopTime, 205
 getBuilding, 205
 getBus, 206
 getCargoAirport, 207
 getComercialAirport, 207
 getFreightTrain, 208
 getHighway, 208
 getInsideRoad, 209
 getPassengerTrain, 209
 getRoadName, 210
 getTaxi, 210
 InsideRoad, 200
isEmployed
 Citizen, 87
isLeaving
 Citizen, 87
isOccupied
 Jobs, 213
isOnCooldown
 Citizen, 87
isPolicyEnforced
 EnforcePolicyCommand, 143
 PolicyCheckerHandler, 253
isWithinBudget
 CityService, 99

Jobs, 211
 getIncome, 212
 getTitle, 212
 isOccupied, 213
 Jobs, 212
JobSatisfactionStrategy, 213
 calculateSatisfaction, 215
 updateForHousingChange, 215
 updateForJobChange, 215
 updateForTaxChange, 215

 LandmarkBuilding, 216

calculateEconomicImpact, 222
calculateResourceConsumption, 222
calculateSatisfactionImpact, 222
construct, 222
getType, 222
hostEvent, 222
LandmarkBuilding, 221
payTaxes, 223
undoCollectTaxes, 223
updateImpacts, 223
LandmarkBuildingBuilder, 224
build, 227
getCulturalValue, 227
getIsHistoric, 227
getVisitorCapacity, 227
setCulturalValue, 227
setIsHistoric, 228
setVisitorCapacity, 228
LeavingCityState, 228
handleState, 230
listAllJobsInBuilding
BuildingManager, 52
LunchRush, 230
getState, 232
getTrafficFlow, 232

MaleCitizen, 233
clone, 237
MaleCitizen, 236
Materials
 ResourceType.h, 557
minMax
 Citizen.cpp, 441

nextList
 TraverseBus, 358
 TraverseCargoAirport, 362
 TraverseComercialAirport, 366
 TraverseFreightTrain, 370
 TraverseHighway, 374
 TraverseInsideRoad, 378
 TraversePassengerTrain, 382
 TraverseState, 388
 TraverseTaxi, 391
NonPeak, 237
 getState, 238
 getTrafficFlow, 238

Observer, 239
 update, 240
operator+
 CityTraverser, 104
operator++
 CityTraverser, 105
 Traverser, 385
operator-
 CityTraverser, 105
operator--
 CityTraverser, 105

Traverser, 386
operator=
 CityTraverser, 106
operator*
 CityTraverser, 104
 Traverser, 385

PassengerTrain, 241
 addInsideRoad, 244
 addPassengerTrain, 244
 getInsideRoad, 245
 getPassengerTrain, 245
 getTrainLine, 246
 PassengerTrain, 243
payTax
 Business, 68, 70
payTaxes
 Building, 47, 48
 Business, 70
 Citizen, 87
 CommercialBuilding, 126
 Income, 183
 IndustrialBuilding, 190
 LandmarkBuilding, 223
 ResidentialBuilding, 275
Peak, 246
 getState, 247
 getTrafficFlow, 247
Policy, 248
 getImpactLevel, 249
 getPolicyName, 249
 implement, 250
 Policy, 249
 revoke, 250
PolicyCheckerHandler, 250
 handleRequest, 253
 isPolicyEnforced, 253
 PolicyCheckerHandler, 252
 setPolicyEnforced, 253
PopulationManager, 254
 addCitizen, 255
 findJobsForUnemployedCitizens, 255
 getCitizens, 255
 getPopulation, 256
 PopulationManager, 255
Power
 ResourceType.h, 557
PowerPlant, 256
 PowerPlant, 258
 registerBuilding, 258
 supplyResources, 259
prevList
 TraverseBus, 359
 TraverseCargoAirport, 363
 TraverseComercialAirport, 367
 TraverseFreightTrain, 371
 TraverseHighway, 375
 TraverseInsideRoad, 379
 TraversePassengerTrain, 383

TraverseState, 388
 TraverseTaxi, 392
 PriorityDistributionStrategy, 259
 allocate, 260
 produceResource
 UtilityMediator, 401
 Property, 261
 calculateTax, 263
 Property, 263
 setAdditionalFees, 263
 setMunicipalLevy, 264
 setTax, 264
 provideService
 CityService, 99
 PublicTransit, 264
 calculateCommute, 267
 getRoute, 267
 PublicTransit, 267
 Recycling
 ResourceType.h, 557
 reduceBudget
 CityService, 99
 refundTaxes
 Government, 164
 registerBuilding
 PowerPlant, 258
 SewageManagement, 310
 Utility, 398
 WasteManagement, 409
 WaterSupply, 413
 registerObserver
 Government, 164
 release
 Resource, 283
 releaseCitizenFromJob
 BuildingManager, 54
 releaseEmployee
 Building, 48
 releaseResources
 ResourceManager, 291
 UtilityMediator, 402
 removeIncomeTaxBuilding
 TaxSystem, 330
 removeObserver
 Citizen, 88
 removePolicy
 Business, 70, 71
 removePropertyTaxBuilding
 TaxSystem, 330
 removeSalesTaxBuilding
 TaxSystem, 331
 removeService
 Business, 71
 removeTaxRate
 TaxSystem, 331
 removeVATTaxPayer
 TaxSystem, 331
 requestResources
 UtilityMediator, 402
 ResidentialBuilding, 268
 addResidents, 274
 calculateEconomicImpact, 274
 calculatePropertyTax, 274
 calculateResourceConsumption, 275
 calculateSatisfactionImpact, 275
 construct, 275
 getResidentialUnits, 275
 getType, 275
 payTaxes, 275
 ResidentialBuilding, 273
 undoCollectTaxes, 276
 updateImpacts, 276
 upgradeComfort, 276
 ResidentialBuildingBuilder, 277
 build, 280
 getComfort, 280
 getResidentialUnit, 280
 setComfort, 280
 setResidentialUnit, 281
 Resource, 281
 allocate, 282
 getQuantity, 283
 getType, 283
 release, 283
 Resource, 282
 ResourceAllocationStrategy, 283
 allocate, 284
 ResourceApprovalHandler, 285
 handleRequest, 286
 setNextHandler, 287
 ResourceAvailability, 287
 checkAvailability, 288
 ResourceAvailability, 288
 ResourceManager, 289
 addObserver, 290
 addResource, 290
 allocateResources, 290
 getBudget, 291
 getResource, 291
 releaseResources, 291
 ResourceManager, 290
 setAllocationStrategy, 292
 ResourceType
 ResourceType.h, 557
 ResourceType.h
 Budget, 557
 Materials, 557
 Power, 557
 Recycling, 557
 ResourceType, 557
 Sewage, 557
 Waste, 557
 Water, 557
 returnVal
 AllocateBudgetCommand, 27
 CollectTaxesCommand, 113

EnforcePolicyCommand, 144
GovCommand, 160
SetTaxCommand, 307
revertBudgetAllocation
 Government, 164
revoke
 Policy, 250
Road, 292
 calculateCommute, 295
 getName, 295
 Road, 295
Sales, 296
 calculateTax, 298
 Sales, 298
 setEnvironmentalLevy, 299
 setServiceFee, 299
 setTax, 299
SatisfactionStrategy, 300
 calculateSatisfaction, 301
 updateForHousingChange, 301
 updateForJobChange, 301
 updateForTaxChange, 302
SatisfiedState, 302
 handleState, 303
searchAndApplyForJob
 Citizen, 88
setAdditionalFees
 Property, 263
setAllocationStrategy
 ResourceManager, 292
setAmount
 AllocateBudgetCommand, 27
setArea
 Builder, 34
setBankBalance
 Citizen, 88
setBuilder
 Director, 137
setBusiness
 CommercialBuilding, 127
setBusinessUnits
 CommercialBuildingBuilder, 131
setCapacity
 Builder, 35
setCitizenSatisfaction
 Builder, 35
setCollectedTaxes
 CollectTaxesCommand, 113
setComfort
 ResidentialBuildingBuilder, 280
setCulturalValue
 LandmarkBuildingBuilder, 227
setCustomerTraffic
 CommercialBuildingBuilder, 132
setEconomicGrowth
 Builder, 35
setEmployed
 Citizen, 88
setEnvironmentalLevy
 Sales, 299
setFloors
 Builder, 36
setIncome
 Citizen, 89
setIsHistoric
 LandmarkBuildingBuilder, 228
setJob
 Citizen, 89
setJobTitle
 Citizen, 89
setMunicipalLevy
 Property, 264
setName
 Builder, 36
 Building, 49
setNextHandler
 ResourceApprovalHandler, 287
setPolicyEnforced
 EnforcePolicyCommand, 144
 PolicyCheckerHandler, 253
setPollutionLevel
 IndustrialBuildingBuilder, 195
setPrevAllocation
 AllocateBudgetCommand, 28
setProductionCapacity
 IndustrialBuildingBuilder, 196
setRelationshipStatus
 Citizen, 89
setResidentialUnit
 ResidentialBuildingBuilder, 281
setResourceConsumption
 Builder, 36
setSatisfactionLevel
 Citizen, 90
setServiceFee
 Sales, 299
setServiceName
 CityService, 100
setState
 Citizen, 90
 CityTraverser, 107
 Transportation, 342
setTax
 Government, 165
 Income, 184
 Property, 264
 Sales, 299
 TaxSystem, 331
 TaxType, 335
 VAT, 405
SetTaxCommand, 304
 canExecute, 307
 execute, 307
 getDescription, 307
 getName, 307
 returnVal, 307

SetTaxCommand, 306
 undo, 308
 setTaxCooldown
 Citizen, 90
 setTaxCooldownPeriod
 Business, 71, 72
 setTaxRate
 Citizen, 90
 setVisitorCapacity
 LandmarkBuildingBuilder, 228
 Sewage
 ResourceType.h, 557
 SewageManagement, 308
 registerBuilding, 310
 SewageManagement, 310
 supplyResources, 311
 stepIn
 CityTraverser, 107
 stepOut
 CityTraverser, 107
 subtractBankBalance
 Citizen, 91
 supplyResources
 PowerPlant, 259
 SewageManagement, 311
 Utility, 399
 WasteManagement, 409
 WaterSupply, 413

 Taxi, 311
 addBuilding, 315
 addCargoAirport, 316
 addComercialAirport, 316
 addFreightTrain, 317
 addInsideRoad, 317
 addPassengerTrain, 318
 getBuilding, 318
 getCargoAirport, 319
 getComercialAirport, 319
 getFreightTrain, 320
 getInsideRoad, 320
 getPassengerTrain, 321
 getRouteName, 321
 getTaxiCompany, 322
 getTaxiNumber, 322
 Taxi, 315
 TaxSatisfactionStrategy, 322
 calculateSatisfaction, 325
 updateForHousingChange, 325
 updateForJobChange, 326
 updateForTaxChange, 326
 TaxSystem, 326
 addGovernment, 328
 addIncomeTaxBuilding, 328
 addPropertyTaxBuilding, 329
 addSalesTaxBuilding, 329
 addTaxRate, 329
 addVATTaxPayer, 329
 checkImpact, 330

 collectTaxes, 330
 removeIncomeTaxBuilding, 330
 removePropertyTaxBuilding, 330
 removeSalesTaxBuilding, 331
 removeTaxRate, 331
 removeVATTaxPayer, 331
 setTax, 331
 updateTaxRate, 331

 TaxType, 332
 calculateTax, 334
 getTaxRate, 334
 getTaxType, 334
 setTax, 335
 TaxType, 334

 TrafficFlow, 335
 getState, 336
 getTrafficFlow, 336

 Train, 337
 calculateCommute, 339
 getLine, 339
 Train, 339

 Transportation, 340
 ~Transportation, 341
 getTrafficFlow, 341
 getType, 341
 setState, 342
 Transportation, 341

 TransportationFactory, 342
 createBus, 344
 createCargoAirport, 344
 createComercialAirport, 345
 createFreightTrain, 345
 createHighway, 346
 createInsideRoad, 347
 createPassengerTrain, 347
 createTaxi, 348

 TransportManager, 349
 ~TransportManager, 350
 createBus, 350
 createCargoAirport, 351
 createComercialAirport, 351
 createFreightTrain, 352
 createHighway, 353
 createInsideRoad, 353
 createPassengerTrain, 354
 createTaxi, 354
 getTransportation, 355
 TransportManager, 350

 TraverseBus, 356
 getPos, 358
 nextList, 358
 prevList, 359
 TraverseBus, 358

 TraverseCargoAirport, 360
 getPos, 362
 nextList, 362
 prevList, 363
 TraverseCargoAirport, 362

TraverseComercialAirport, 363
 getPos, 366
 nextList, 366
 prevList, 367
 TraverseComercialAirport, 366

TraverseFreightTrain, 368
 getPos, 370
 nextList, 370
 prevList, 371
 TraverseFreightTrain, 370

TraverseHighway, 372
 getPos, 374
 nextList, 374
 prevList, 375
 TraverseHighway, 374

TraverseInsideRoad, 375
 getPos, 378
 nextList, 378
 prevList, 379
 TraverseInsideRoad, 378

TraversePassengerTrain, 379
 getPos, 382
 nextList, 382
 prevList, 383
 TraversePassengerTrain, 382

Traverser, 384
 operator++, 385
 operator--, 386
 operator*, 385

TraverseState, 386
 getLayer, 388
 getPos, 388
 nextList, 388
 prevList, 388
 TraverseState, 387

TraverseTaxi, 389
 getPos, 391
 nextList, 391
 prevList, 392
 TraverseTaxi, 391

undo
 AllocateBudgetCommand, 28
 CollectTaxesCommand, 114
 EnforcePolicyCommand, 144
 GovCommand, 160
 SetTaxCommand, 308

undoCollectTaxes
 Building, 49
 CommercialBuilding, 127
 IndustrialBuilding, 191
 LandmarkBuilding, 223
 ResidentialBuilding, 276

UnemployedState, 392
 handleState, 393

unregisterObserver
 Government, 165

UnsatisfiedState, 394
 handleState, 395

update
 CitizenObserver, 93
 CitizenSatisfactionObserver, 94
 Government, 165
 Observer, 240
 UtilityMediator, 402

updateBudget
 CityService, 100

updateCustomer
 CommercialBuilding, 127

updateForHousingChange
 HousingSatisfactionStrategy, 177
 JobSatisfactionStrategy, 215
 SatisfactionStrategy, 301
 TaxSatisfactionStrategy, 325

updateForJobChange
 HousingSatisfactionStrategy, 177
 JobSatisfactionStrategy, 215
 SatisfactionStrategy, 301
 TaxSatisfactionStrategy, 326

updateForTaxChange
 HousingSatisfactionStrategy, 177
 JobSatisfactionStrategy, 215
 SatisfactionStrategy, 302
 TaxSatisfactionStrategy, 326

updateImpacts
 Building, 49
 CommercialBuilding, 127
 IndustrialBuilding, 191
 LandmarkBuilding, 223
 ResidentialBuilding, 276

updatePolicy
 Business, 72
 GovObserver, 167

updateSatisfaction
 Citizen, 91

updateServices
 Business, 72, 73
 GovObserver, 168

updateTaxRate
 Business, 73
 GovObserver, 168
 TaxSystem, 331

upgradeComfort
 ResidentialBuilding, 276

upgradeTech
 IndustrialBuilding, 191

Utility, 396
 adjustForPopulation, 398
 registerBuilding, 398
 supplyResources, 399
 Utility, 398

UtilityMediator, 399
 getAvailableResource, 401
 produceResource, 401
 releaseResources, 402
 requestResources, 402
 update, 402

validateAllocation
 AllocateBudgetCommand, 28
validateCollection
 CollectTaxesCommand, 114
validateEnforcement
 EnforcePolicyCommand, 144
VAT, 403
 calculateTax, 405
 setTax, 405

Waste
 ResourceType.h, 557

WasteManagement, 405
 adjustForPopulation, 408
 registerBuilding, 409
 supplyResources, 409
 WasteManagement, 408

Water
 ResourceType.h, 557

WaterSupply, 409
 adjustForPopulation, 412
 registerBuilding, 413
 supplyResources, 413
 WaterSupply, 412