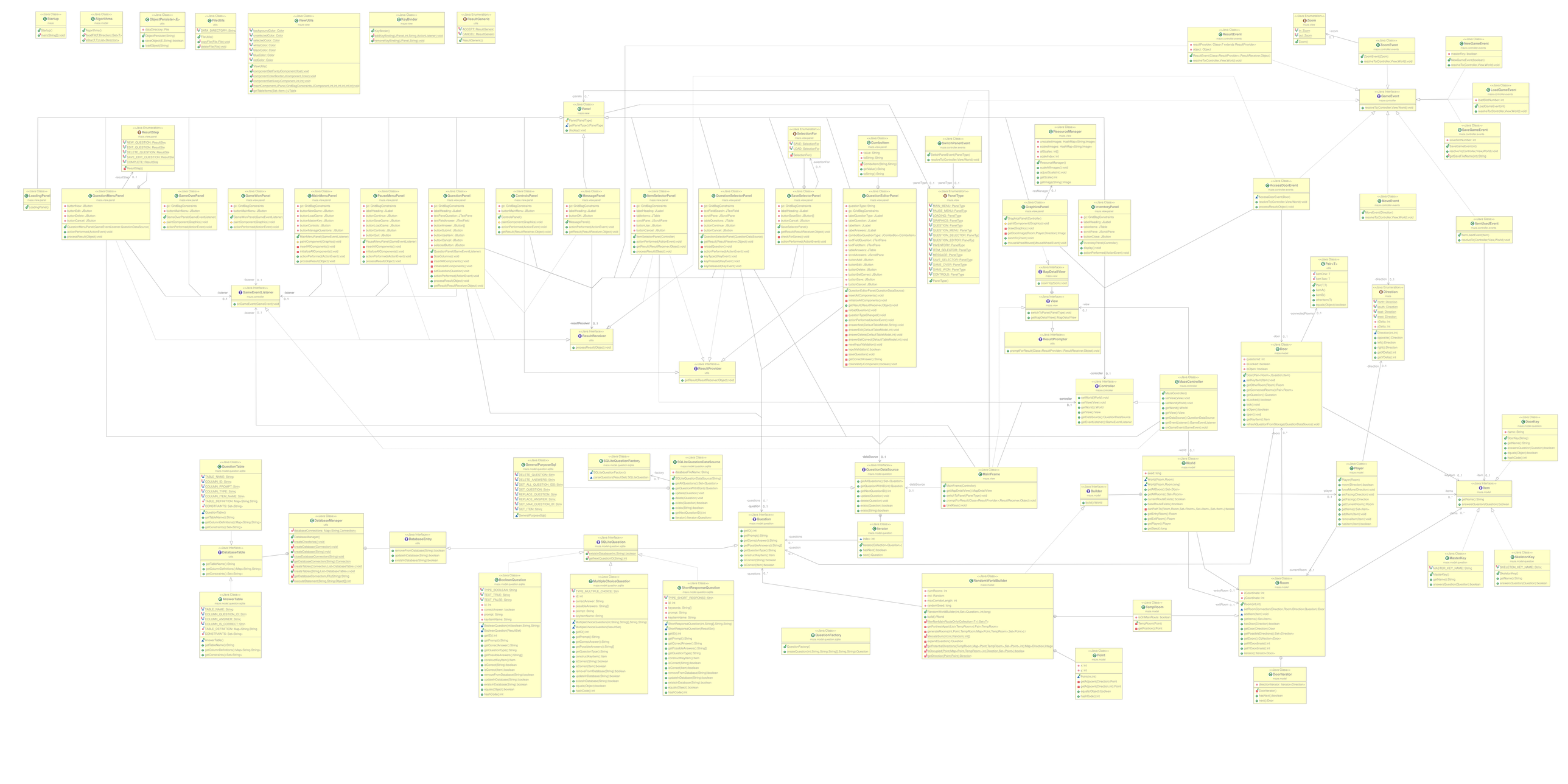
«Java Class»
Startup
view
Startup()
main(String[]): void
«Java Class»
Algorithms
maze.model
Algorithms()
floodFill(Direction): Set<Tuple>
dfSort(T,Map<T,List<Direction>>): List<Direction>
«Java Class»
ObjectPersister<E>
utils
dataDirectory: File
ObjectPersister(String)
saveObject(E,String): boolean
loadObject(String): E
«Java Class»
FileUtils
utils
DATA_DIRECTORY: String
FileUtils()
writeTextFile(File,File): void
deleteFileIfExist(File): void
«Java Class»
ViewUtils
maze.view
backgroundColor: Color
unselectedColor: Color
selectedColor: Color
whiteColor: Color
blackColor: Color
blueColor: Color
redColor: Color
ViewUtils()
addComponentBeforeFooter(JComponent,Footer): void
addComponentToCenter(JComponent,Center): void
addComponentBeforeSlot(JComponent,int,int): void
insertComponent(JPanel,GridBagConstraints,JComponent,int,int,int,int,int): void
getTableItemsSize(Item,int): Table
«Java Class»
KeyBinder
maze.view
KeyBinder()
addKeyBinding(JPanel,int,String,ActionListener): void
removeKeyBinding(JPanel,String): void
«Java Enumeration»
ResultGeneric
utils
ACCEPT: ResultGeneric
CANCEL: ResultGeneric
values(): ResultGeneric[]
«Java Class»
Panel
maze.view
panelType: PanelType
getPanelType(): PanelType
display(): void
«Java Enumeration»
ResultStep
maze.view.panel
NEW_QUESTION: ResultStep
EDIT_QUESTION: ResultStep
DELETE_QUESTION: ResultStep
SAVE_EDIT_QUESTION: ResultStep
COMPLETE: ResultStep
ResultStep()
«Java Class»
LoadingPanel
maze.view.panel
LoadingPanel()
«Java Class»
QuestionMenuPanel
maze.view.panel
buttonNew: JButton
buttonEdit: JButton
buttonDelete: JButton
buttonCancel: JButton
QuestionMenuPanel(GameEventListener,QuestionDataSource)
actionPerformed(ActionEvent): void
processResult(Object): void
«Java Class»
GameOverPanel
maze.view.panel
gc: GridBagConstraints
buttonMainMenu: JButton
GameOverPanel(GameEventListener)
initAllComponents(Graphics): void
actionPerformed(ActionEvent): void
«Java Class»
GameWonPanel
maze.view.panel
gc: GridBagConstraints
buttonMainMenu: JButton
GameWonPanel(GameEventListener)
initAllComponents(Graphics): void
actionPerformed(ActionEvent): void
«Java Class»
MainMenuPanel
maze.view.panel
gc: GridBagConstraints
buttonNewGame: JButton
buttonLoadGame: JButton
buttonMasterKey: JButton
buttonControls: JButton
buttonManageQuestions: JButton
MainMenuPanel(GameEventListener)
paint(Component,Graphics): void
insertAllComponents(): void
initializeAllComponents(): void
actionPerformed(ActionEvent): void
processResult(Object): void
«Java Class»
PauseMenuPanel
maze.view.panel
gc: GridBagConstraints
buttonContinue: JButton
buttonSaveGame: JButton
buttonLoadGame: JButton
buttonControls: JButton
buttonQuit: JButton
PauseMenuPanel(GameEventListener)
insertAllComponents(): void
initializeAllComponents(): void
actionPerformed(ActionEvent): void
processResult(Object): void
«Java Class»
QuestionPanel
maze.view.panel
gc: GridBagConstraints
labelHeading: JLabel
textFieldQuestion: JTextPane
textFieldAnswer: JTextField
buttonAnswer: JButton[]
buttonSubmit: JButton
buttonUseItem: JButton
buttonCancel: JButton
selectedButton: JButton
QuestionPanel(GameEventListener)
SizeColumns(): void
insertAllComponents(): void
initializeAllComponents(): void
setQuestion(Question): void
actionPerformed(ActionEvent): void
processResult(Object): void
getResult(ResultReceiver,Object): void
«Java Class»
ControlsPanel
maze.view.panel
gc: GridBagConstraints
buttonMainMenu: JButton
ControlsPanel()
paint(Component,Graphics): void
actionPerformed(ActionEvent): void
getResult(ResultReceiver,Object): void
«Java Class»
MessagePanel
maze.view.panel
gc: GridBagConstraints
labelHeading: JLabel
buttonOK: JButton
MessagePanel()
actionPerformed(ActionEvent): void
getResult(ResultReceiver,Object): void
«Java Class»
ItemSelectionPanel
maze.view.panel
gc: GridBagConstraints
labelHeading: JLabel
tableItems: JTable
scrollPane: JScrollPane
buttonUse: JButton
buttonCancel: JButton
ItemSelectionPanel(Controller)
actionPerformed(ActionEvent): void
getResult(ResultReceiver,Object): void
processResult(Object): void
«Java Class»
QuestionSelectorPanel
maze.view.panel
gc: GridBagConstraints
textFieldSearch: JTextField
scrollPane: JScrollPane
tableQuestions: JTable
buttonContinue: JButton
buttonCancel: JButton
QuestionSelectorPanel(QuestionDataSource)
getResult(ResultReceiver,Object): void
processQuestions(): void
actionPerformed(ActionEvent): void
keyTyped(KeyEvent): void
keyPressed(KeyEvent): void
keyReleased(KeyEvent): void
«Java Enumeration»
SelectionFor
maze.view.panel
SAVE: SelectionFor
LOAD: SelectionFor
SelectionFor()
«Java Class»
SaveSelectionPanel
maze.view.panel
labelHeading: JLabel
buttonSaveSlot: JButton[]
buttonCancel: JButton
SaveSelectionPanel()
getResult(ResultReceiver,Object): void
actionPerformed(ActionEvent): void
«Java Class»
Combobox
maze.view.panel
value: String
toString: String
Combobox(String,String)
getValue(): String
toString(): String
«Java Class»
QuestionEditorPanel
maze.view.panel
questionType: String
gc: GridBagConstraints
labelQuestionType: JLabel
labelQuestion: JLabel
labelItem: JLabel
labelAnswers: JLabel
comboboxQuestionType: JComboBox<Combobox>
textFieldQuestion: JTextPane
textFieldItem: JTextPane
tableAnswers: JTable
scrollAnswers: JScrollPane
buttonAdd: JButton
buttonEdit: JButton
buttonDelete: JButton
buttonSaveCorrect: JButton
buttonSave: JButton
buttonCancel: JButton
QuestionEditorPanel(QuestionDataSource)
insertAllComponents(): void
initializeAllComponents(): void
getResult(ResultReceiver,Object): void
processQuestion(): void
questionTypeChanged(): void
actionPerformed(ActionEvent): void
answerAddDefaultTableModel(String): void
answerEditDefaultTableModel(int): void
answerDeleteDefaultTableModel(int): void
answerSetCorrectDefaultTableModel(int): void
resetInputValidation(): void
inputValidation(): boolean
saveQuestion(): void
getCorrectAnswer(): String
isInvalid(JComponent,boolean): void
«Java Class»
SwitchPanelEvent
maze.controller.events
SwitchPanelEvent(PanelType)
resolveToController(View World): void
«Java Enumeration»
PanelType
maze.view
MAIN_MENU: PanelType
PAUSE_MENU: PanelType
LOADING: PanelType
GRAPHICS: PanelType
QUESTION: PanelType
QUESTION_MENU: PanelType
QUESTION_SELECTOR: PanelType
QUESTION_EDITOR: PanelType
INVENTORY: PanelType
ITEM_SELECTOR: PanelType
MESSAGE: PanelType
SAVE_SELECTOR: PanelType
GAME_OVER: PanelType
GAME_WON: PanelType
CONTROLS: PanelType
PanelType()
«Java Class»
ResourceManager
maze.view
cmscaleImages: HashMap<String,Image>
scaleImages: HashMap<String,Image>
allScales: int[]
scaleIndex: int
ResourceManager()
scaleAllImages(): void
adjustScale(int): void
getScale(): int
getImage(String): Image
«Java Class»
GraphicsPanel
maze.view
GraphicsPanel(Controller)
paint(Component,Graphics): void
drawGraphics(): void
getDoorImage(Room,Player,Direction): Image
zoomToZoom(): void
mouseWheelMoved(MouseWheelEvent): void
«Java Class»
InventoryPanel
maze.view.panel
gc: GridBagConstraints
labelHeading: JLabel
tableItems: JTable
scrollPane: JScrollPane
buttonClose: JButton
InventoryPanel(Controller)
display(): void
actionPerformed(ActionEvent): void
«Java Interface»
HouDetailView
maze.view
zoomToZoom(): void
«Java Interface»
View
maze.view
switchToPanel(PanelType): void
getMapDetailView(): MapDetailView
«Java Interface»
ResultPrompter
utils
promptForResult(Class<ResultProvider>,ResultReceiver,Object): void
«Java Interface»
ResultReceiver
utils
processResult(Object): void
«Java Interface»
ResultProvider
utils
getResult(ResultReceiver,Object): void
«Java Interface»
GameEventListener
maze.controller
onGameEvent(GameEvent): void
«Java Class»
ResultEvent
maze.controller.events
resultProvider: Class<?> extends ResultProvider>
object: Object
ResultEvent(Class<ResultProvider>,ResultReceiver,Object)
resolveToController(View World): void
«Java Enumeration»
Zoom
maze.view
IN: Zoom
OUT: Zoom
ZOOM(): Zoom
«Java Class»
ZoomEvent
maze.controller.events
ZoomEvent(Zoom)
resolveToController(View World): void
«Java Class»
NewGameEvent
maze.controller.events
masterKey: boolean
NewGameEvent(boolean)
resolveToController(View World): void
«Java Interface»
GameEvent
maze.controller
resolveToController(View World): void
«Java Class»
LoadGameEvent
maze.controller.events
slotSlotNumber: int
LoadGameEvent(int)
resolveToController(View World): void
«Java Class»
SaveGameEvent
maze.controller.events
saveSlotNumber: int
SaveGameEvent(int)
resolveToController(View World): void
getSaveFileName(int): String
«Java Class»
AccessDoorEvent
maze.controller.events
processDoorEvent(Door): void
resolveToController(View World): void
processResult(Object): void
«Java Class»
MouseEvent
maze.controller.events
MouseEvent(int,int)
resolveToController(View World): void
«Java Class»
ItemUseEvent
maze.controller.events
ItemUseEvent(Item)
resolveToController(View World): void
«Java Class»
Controller
maze.controller
availableWorld(World): void
getView(View): void
getWorld(World): void
getWorld(): World
getView(): View
getDataSource(): QuestionDataSource
getEventListener(): GameEventListener
«Java Class»
MazeController
maze.controller
MazeController()
getView(View): void
setWorld(World): void
getWorld(): World
getView(): View
getDataSource(): QuestionDataSource
getEventListener(): GameEventListener
onGameEvent(GameEvent): void
«Java Class»
Pair<T>
utils
itemOne: T
itemTwo: T
Pair(T,T)
hasA(): boolean
hasB(): boolean
isPartOf(T): boolean
equals(Object): boolean
«Java Enumeration»
Direction
maze
North: Direction
South: Direction
East: Direction
West: Direction
dCelsa: int
dDelta: int
Direction(int,int)
opposite(): Direction
left(): Direction
right(): Direction
getXDelta(): int
getYDelta(): int
«Java Class»
Door
maze.model
questionId: int
bLocked: boolean
bOpen: boolean
Door(Pair<Room>,Question,Item)
addKeyItem(Item): void
getOtherRoom(Room): Room
getConnectedRooms(): Pair<Room>
getQuestion(): Question
bLocked(): boolean
lock(): void
bOpen(): boolean
open(): void
getKeyItem(): Item
initializeQuestionFromStorage(QuestionDataSource): void
«Java Class»
DoorKey
maze.model.question
name: String
DoorKey(String)
getName(): String
answerQuestion(Question): boolean
equals(Object): boolean
hashCode(): int
«Java Class»
QuestionTable
maze.model.question.sqlite
TABLE_NAME: String
COLUMN_ID: String
COLUMN_PROMPT: String
COLUMN_TYPE: String
COLUMN_ITEM_NAME: String
TABLE_DEFINITION: Map<String,String>
CONSTRAINTS: Set<String>
QuestionTable()
getTableName(): String
getColumnDefinitions(): Map<String,String>
getConstraints(): Set<String>
«Java Class»
GeneralPurposeSql
maze.model.question.sqlite
DELETE_QUESTION: Stri
DELETE_ANSWERS: Stri
GET_ALL_QUESTION_IDS: Stri
GET_QUESTION: Strin
REPLACE_QUESTION: Strin
REPLACE_ANSWER: Strin
GET_MAX_QUESTION_ID: Stri
GET_ITEM: String
GeneralPurposeSql()
«Java Class»
SQLiteQuestionFactory
maze.model.question.sqlite
SQLiteQuestionFactory()
storeQuestion(ResultSet): SQLiteQuestion
«Java Class»
SQLiteQuestionDataSource
maze.model.question.sqlite
databaseFileName: String
SQLiteQuestionDataSource(String)
getAllQuestions(): Set<Question>
getQuestionWithId(int): Question
getNextQuestion(): void
deleteQuestion(): void
exists(Question): boolean
exists(String): boolean
getNextQuestionId(): int
iterator(): Iterator<Question>
«Java Interface»
QuestionDataSource
maze.model.question
getAllQuestions(): Set<Question>
getQuestionWithId(int): Question
getNextQuestionId(): int
updateQuestion(): void
deleteQuestion(): void
exists(Question): boolean
exists(String): boolean
«Java Class»
MainFrame
maze.view
MainFrame(Controller)
getMapDetailView(): MapDetailView
switchToPanel(PanelType): void
promptForResult(Class<ResultProvider>, ResultReceiver,Object): void
bindKeys(): void
«Java Interface»
Builder
maze.model
build(): World
«Java Class»
World
maze.model
seed: long
World(Room,Room)
World(Room,Room,long)
getAllDoors(): Set<Door>
getAllRooms(): Set<Room>
currentRouteExists(): boolean
baseRouteExists(): boolean
getPathToRoom,Room,Set<Room>,Set<Item>,Set<Item>): boolean
getEntryRoom(): Room
getExitRoom(): Room
getPlayer(): Player
getSeed(): long
«Java Class»
Player
maze.model
Player(Room)
move(Direction): boolean
forceMove(Direction): void
getFacingDirection(): Direction
getCurrentRoom(): Room
getItems(): Set<Item>
addItem(Item): void
removeItem(Item): void
hasItem(Item): boolean
«Java Class»
Item
maze.model
getName(): String
answerQuestion(Question): boolean
«Java Class»
MasterKey
maze.model.question
MASTER_KEY_NAME: String
MasterKey()
getName(): String
answerQuestion(Question): boolean
«Java Class»
SkeletonKey
maze.model.question
SKELETON_KEY_NAME: Strin
SkeletonKey()
getName(): String
answerQuestion(Question): boolean
«Java Class»
DatabaseManager
utils
databaseConnections: Map<String,Connection>
DatabaseManager()
createDirectories(): void
createDatabaseConnection(): void
createDatabase(String): void
closeDatabaseConnection(String): void
getDatabaseConnection(String): Connection
createTables(Connection,List<DatabaseTable>): void
dropTables(String,List<DatabaseTable>): void
getDatabaseConnectionURL(String): String
executeStatement(String,String,Object[]): int
«Java Interface»
DatabaseEntry
utils
removeFromDatabase(String): boolean
updateInDatabase(String): boolean
existsInDatabase(String): boolean
«Java Interface»
DatabaseTable
utils
getTableName(): String
getColumnDefinitions(): Map<String,String>
getConstraints(): Set<String>
«Java Class»
AnswerTable
maze.model.question.sqlite
TABLE_NAME: String
COLUMN_QUESTION_ID: Strin
COLUMN_ANSWER: String
COLUMN_IS_CORRECT: Strin
TABLE_DEFINITION: Map<String,String>
CONSTRAINTS: Set<String>
AnswerTable()
getTableName(): String
getColumnDefinitions(): Map<String,String>
getConstraints(): Set<String>
«Java Interface»
SQLiteQuestion
maze.model.question.sqlite
existsInDatabase(String): boolean
getNextQuestionId(String): int
«Java Class»
BooleanQuestion
maze.model.question.sqlite
TYPE_BOOLEAN: String
TEXT_TRUE: String
TEXT_FALSE: String
id: int
correctAnswer: boolean
prompt: String
keyItemName: String
BooleanQuestion(int,boolean,String,String)
BooleanQuestion(ResultSet)
getID(): int
getPrompt(): String
getCorrectAnswer(): String
getQuestionType(): String
getPossibleAnswers(): String[]
constructKeyItem(): Item
isCorrect(String): boolean
isCorrect(Item): boolean
removeFromDatabase(String): boolean
updateInDatabase(String): boolean
existsInDatabase(String): boolean
equals(Object): boolean
hashCode(): int
«Java Class»
MultipleChoiceQuestion
maze.model.question.sqlite
TYPE_MULTIPLE_CHOICE: Strin
id: int
correctAnswer: String
possibleAnswers: String[]
prompt: String
keyItemName: String
MultipleChoiceQuestion(int,String,String[],String,String)
MultipleChoiceQuestion(ResultSet)
getID(): int
getPrompt(): String
getCorrectAnswer(): String
getPossibleAnswers(): String[]
getQuestionType(): String
constructKeyItem(): Item
isCorrect(String): boolean
isCorrect(Item): boolean
removeFromDatabase(String): boolean
updateInDatabase(String): boolean
existsInDatabase(String): boolean
equals(Object): boolean
hashCode(): int
«Java Class»
ShortResponseQuestion
maze.model.question.sqlite
TYPE_SHORT_RESPONSE: Strin
id: int
keywords: String[]
prompt: String
keyItemName: String
ShortResponseQuestion(int,String[],String,String)
ShortResponseQuestion(ResultSet)
getID(): int
getPrompt(): String
getCorrectAnswer(): String
getPossibleAnswers(): String[]
getQuestionType(): String
constructKeyItem(): Item
isCorrect(String): boolean
isCorrect(Item): boolean
removeFromDatabase(String): boolean
updateInDatabase(String): boolean
existsInDatabase(String): boolean
equals(Object): boolean
hashCode(): int
«Java Interface»
Question
maze.model.question
getID(): int
getPrompt(): String
getCorrectAnswer(): String
getPossibleAnswers(): String[]
getQuestionType(): String
constructKeyItem(): Item
isCorrect(String): boolean
isCorrect(Item): boolean
«Java Class»
Iterator
maze.model.question
index: int
Iterator(Collection<Question>)
hasNext(): boolean
next(): Question
«Java Class»
QuestionFactory
maze.model.question.sqlite
QuestionFactory()
createQuestion(int,String,String,String[],String,String):Question
«Java Class»
RandomWorldBuilder
maze.model
numRooms: int
rnd: Random
maxCorridorLength: int
randomSeed: long
RandomWorldBuilder(int,Set<Question>,int,long)
build(): World
findNearestNonRouteOrig(Collection<?>):Set<?>
getFurthestApart(List<TempRoom>):Pair<TempRoom>
generateRooms(int,Point,TempRoom,Map<Point,TempRoom>,Set<Points>)
placeDoors(int,int,Random):void
expandQuestion():Question
getPotentialDirections(TempRoom,Map<Point,TempRoom>,Set<Point>,int):Map<Direction,Integer>
isCloseUpon(Point,Map<Point,TempRoom>,int,Direction,Set<Point>):boolean
getDirection(Point,Point):Direction
«Java Class»
TempRoom
maze.model
isOnMainRoute: boolean
TempRoom(Point)
getPosition():Point
«Java Class»
Room
maze.model
xCoordinate: int
yCoordinate: int
Room(int,int)
setRoomConnection(Direction,Room,Direction,Question):Door
addItem(Item):void
getItems():Set<Item>
hasDoor(Direction):boolean
getDoor(Direction):Door
getPossibleDirections():Set<Direction>
getDoors():Collection<Door>
getXCoordinate():int
getYCoordinate():int
iterator():Iterator<Door>
«Java Class»
Point
maze.model
x: int
y: int
Point(int,int)
getAdjacent(Direction):Point
getAdjacent(Direction,int):Point
equals(Object): boolean
hashCode(): int
«Java Class»
DoorIterator
maze.model
directionIterator: Iterator<Direction>
DoorIterator()
hasNext(): boolean
next(): Door