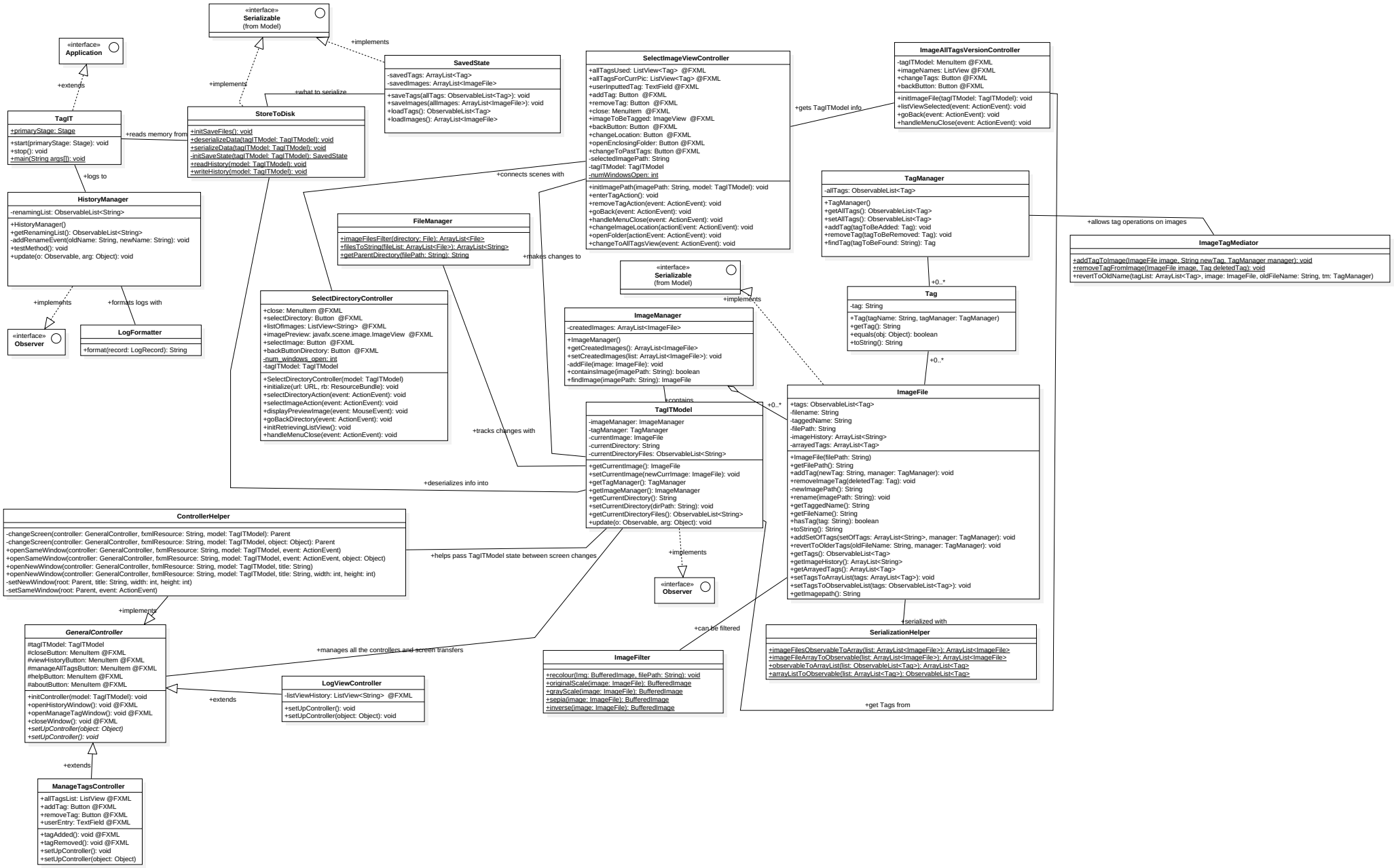
Model::Main


«interface»
Serializable
(from Model)
«interface»
Application
+implements
+extends
+implements
+what to serialize
SavedState
-savedTags: ArrayList<Tag>
-savedImages: ArrayList<ImageFile>
+saveTags(allTags: ObservableList<Tag>): void
+saveImages(allImages: ArrayList<ImageFile>): void
+loadTags(): ObservableList<Tag>
+loadImages(): ArrayList<ImageFile>
SelectImageViewController
+allTagsUsed: ListView<Tag> @FXML
+allTagsForCurrPic: ListView<Tag> @FXML
+userInputtedTag: TextField @FXML
+addTag: Button @FXML
+removeTag: Button @FXML
+close: MenuItem @FXML
+imageToBeTagged: ImageView @FXML
+backButton: Button @FXML
+changeLocation: Button @FXML
+openEnclosingFolder: Button @FXML
+changeToPastTags: Button @FXML
-selectedImagePath: String
-tagITModel: TagITModel
-numWindowsOpen: int
+initImagePath(imagePath: String, model: TagITModel): void
+enterTagAction(): void
+removeTagAction(event: ActionEvent): void
+goBack(event: ActionEvent): void
+handleMenuClose(event: ActionEvent): void
+changeToAllTagsView(event: ActionEvent): void
+gets TagITModel info
ImageAllTagsVersionController
-tagITModel: MenuItem @FXML
+imageNames: ListView @FXML
+changeTags: Button @FXML
+backButton: Button @FXML
+initImageFile(tagITModel: TagITModel): void
+listViewSelected(event: ActionEvent): void
+goBack(event: ActionEvent): void
+handleMenuClose(event: ActionEvent): void
TagIT
+primaryStage: Stage
+start(primaryStage: Stage): void
+stop(): void
+main(String args[]): void
+reads memory from
StoreToDisk
+initSaveFiles(): void
+deserializeData(tagITModel: TagITModel): void
+serializeData(tagITModel: TagITModel): void
-initSaveState(tagITModel: TagITModel): SavedState
+readHistory(model: TagITModel): void
+writeHistory(model: TagITModel): void
+logs to
+connects scenes with
HistoryManager
-renamingList: ObservableList<String>
+HistoryManager()
+getRenamingList(): ObservableList<String>
-addRenameEvent(oldName: String, newName: String): void
+testMethod(): void
+update(o: Observable, arg: Object): void
FileManager
+imageFilesFilter(directory: File): ArrayList<File>
+filesToString(fileList: ArrayList<File>): ArrayList<String>
+getParentDirectory(filePath: String): String
+makes changes to
TagManager
-allTags: ObservableList<Tag>
+TagManager()
+getAllTags(): ObservableList<Tag>
+setAllTags(): ObservableList<Tag>
+addTag(tagToBeAdded: Tag): void
+removeTag(tagToBeRemoved: Tag): void
+findTag(tagToBeFound: String): Tag
+allows tag operations on images
ImageTagMediator
+addTagToImage(ImageFile image, String newTag, TagManager manager): void
+removeTagFromImage(ImageFile image, Tag deletedTag): void
+revertToOldName(tagList: ArrayList<Tag>, image: ImageFile, oldFileName: String, tm: TagManager)
+implements
+formats logs with
«interface»
Observer
LogFormatter
+format(record: LogRecord): String
SelectDirectoryController
+close: MenuItem @FXML
+selectDirectory: Button @FXML
+listOfImages: ListView<String> @FXML
+imagePreview: javafx.scene.image.ImageView @FXML
+selectImage: Button @FXML
+backButtonDirectory: Button @FXML
-num_windows_open: int
-tagITModel: TagITModel
+SelectDirectoryController(model: TagITModel)
+initialize(url: URL, rb: ResourceBundle): void
+selectDirectoryAction(event: ActionEvent): void
+selectImageAction(event: ActionEvent): void
+displayPreviewImage(event: MouseEvent): void
+goBackDirectory(event: ActionEvent): void
+initRetrievingListView(): void
+handleMenuClose(event: ActionEvent): void
«interface»
Serializable
(from Model)
+implements
ImageManager
-createdImages: ArrayList<ImageFile>
+ImageManager()
+getCreatedImages(): ArrayList<ImageFile>
+setCreatedImages(list: ArrayList<ImageFile>): void
-addFile(image: ImageFile): void
+containsImage(imagePath: String): boolean
+findImage(imagePath: String): ImageFile
+contains
+0..*
Tag
-tag: String
+Tag(tagName: String, tagManager: TagManager)
+getTag(): String
+equals(obj: Object): boolean
+toString(): String
+0..*
+0..*
ImageFile
+tags: ObservableList<Tag>
-filename: String
-taggedName: String
-filePath: String
-imageHistory: ArrayList<String>
-arrayedTags: ArrayList<Tag>
+ImageFile(filePath: String)
+getFilePath(): String
+addTag(newTag: String, manager: TagManager): void
+removeImageTag(deletedTag: Tag): void
-newImagePath(): String
+rename(imagePath: String): void
+getTaggedName(): String
+getFileName(): String
+hasTag(tag: String): boolean
+toString(): String
+addSetOfTags(setOfTags: ArrayList<String>, manager: TagManager): void
+revertToOlderTags(oldFileName: String, manager: TagManager): void
+getTags(): ObservableList<Tag>
+getImageHistory(): ArrayList<String>
+getArrayedTags(): ArrayList<Tag>
+setTagsToArrayList(tags: ArrayList<Tag>): void
+setTagsToObservableList(tags: ObservableList<Tag>): void
+getImagepath(): String
+tracks changes with
TagITModel
-imageManager: ImageManager
-tagManager: TagManager
-currentImage: ImageFile
-currentDirectory: String
-currentDirectoryFiles: ObservableList<String>
+getCurrentImage(): ImageFile
+setCurrentImage(newCurrImage: ImageFile): void
+getTagManager(): TagManager
+getImageManager(): ImageManager
+getCurrentDirectory(): String
+setCurrentDirectory(dirPath: String): void
+getCurrentDirectoryFiles(): ObservableList<String>
+update(o: Observable, arg: Object): void
+deserializes info into
ControllerHelper
-changeScreen(controller: GeneralController, fxmlResource: String, model: TagITModel): Parent
-changeScreen(controller: GeneralController, fxmlResource: String, model: TagITModel, object: Object): Parent
+openSameWindow(controller: GeneralController, fxmlResource: String, model: TagITModel, event: ActionEvent)
+openSameWindow(controller: GeneralController, fxmlResource: String, model: TagITModel, event: ActionEvent, object: Object)
+openNewWindow(controller: GeneralController, fxmlResource: String, model: TagITModel, title: String)
+openNewWindow(controller: GeneralController, fxmlResource: String, model: TagITModel, title: String, width: int, height: int)
-setNewWindow(root: Parent, title: String, width: int, height: int)
-setSameWindow(root: Parent, event: ActionEvent)
+helps pass TagITModel state between screen changes
+implements
«interface»
Observer
+implements
GeneralController
#tagITModel: TagITModel
#closeButton: MenuItem @FXML
#viewHistoryButton: MenuItem @FXML
#manageAllTagsButton: MenuItem @FXML
#helpButton: MenuItem @FXML
#aboutButton: MenuItem @FXML
+initController(model: TagITModel): void
+openHistoryWindow(): void @FXML
+openManageTagWindow(): void @FXML
+closeWindow(): void @FXML
+setUpController(object: Object)
+setUpController(): void
+manages all the controllers and screen transfers
+can be filtered
ImageFilter
+recolour(Img: BufferedImage, filePath: String): void
+originalScale(image: ImageFile): BufferedImage
+grayScale(image: ImageFile): BufferedImage
+sepia(image: ImageFile): BufferedImage
+inverse(image: ImageFile): BufferedImage
+serialized with
SerializationHelper
+imageFilesObservableToArray(list: ArrayList<ImageFile>): ArrayList<ImageFile>
+imageFileArrayToObservable(list: ArrayList<ImageFile>): ArrayList<ImageFile>
+observableToArrayList(list: ObservableList<Tag>): ArrayList<Tag>
+arrayListToObservable(list: ArrayList<Tag>): ObservableList<Tag>
+extends
LogViewController
-listViewHistory: ListView<String> @FXML
+setUpController(): void
+setUpController(object: Object): void
+get Tags from
+extends
ManageTagsController
+allTagsList: ListView @FXML
+addTag: Button @FXML
+removeTag: Button @FXML
+userEntry: TextField @FXML
+tagAdded(): void @FXML
+tagRemoved(): void @FXML
+setUpController(): void
+setUpController(object: Object)