Project 1 Report (Bonus)

Task 1e: Transition System

User Management

The Action Space U

$U = \{CreateUser, AuthenticateUser, UpdateUser, DeleteUser, LogOut\}$

The Output Space Y

$Y = \{UserNotCreated, UserCreated, UserNotAuthenticated, UserAuthenticated\}$

The State Space X

$X = \{NotCreated, Created, NotAuthenticated, Authenticated\}$

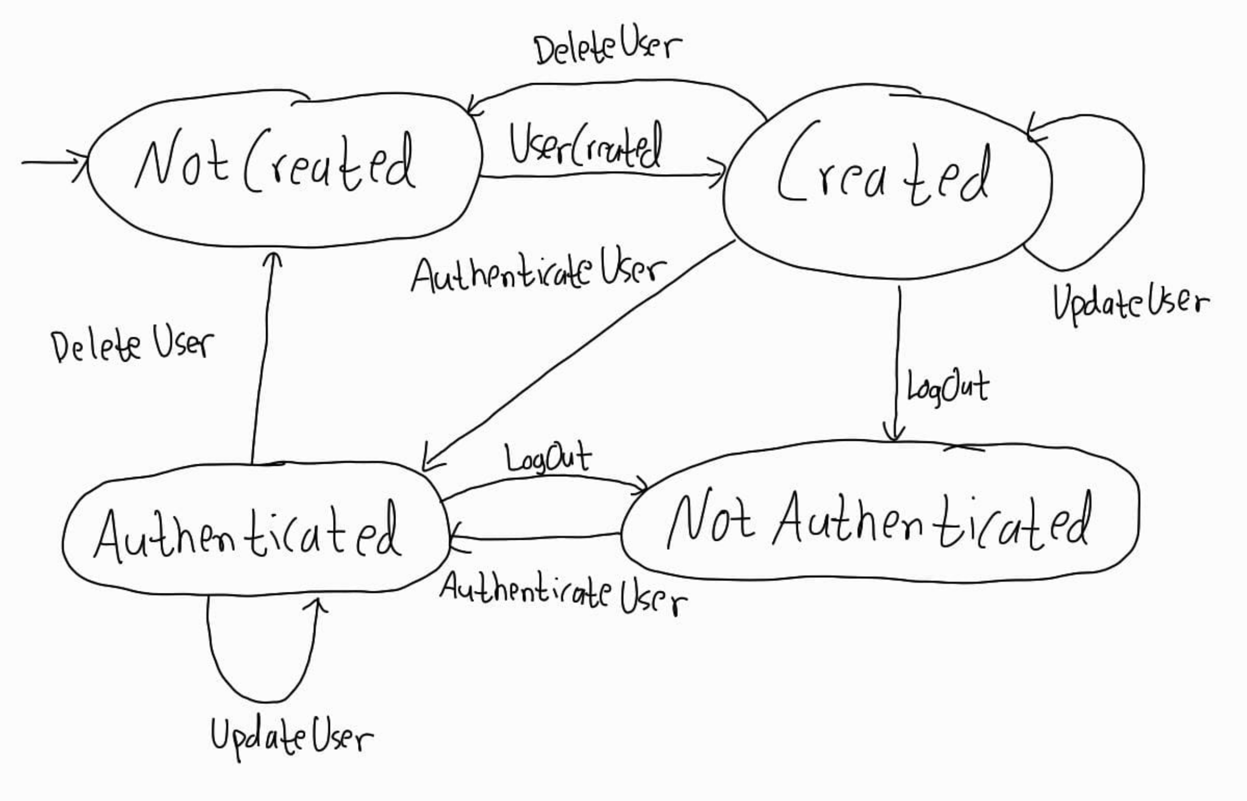
The Initial State X_0

$X_0 = \{UserNotCreated\}$

Transition Table

	CreateUser	AuthenticateUser	UpdateUser	LogOut	DeleteUser
UserNotCreated	UserCreated	x	x	x	x
UserCreated	x	UserAuthenticated	UserCreated	UserNotAuthenticated	UserNotCreated
UserNotAuthenticated	x	UserAuthenticated	x	x	x
UserAuthenticated	x	x	UserAuthenticated	UserNotAuthenticated	UserNotCreated

Diagram



Library Management

The Action Space U

$U = \{Create, Update, Delete, Play, Like\}$

The Output Space Y

$$Y = \{ArtistNotCreated, ArtistCreated, TrackNotCreated, TrackCreated, AlbumNotCreated, AlbumCreated, PlaylistNotCreated, PlaylistCreated, TranscoderNotCreated, TranscoderCreated, PlayNotStarted, PlayStarted, PlayCompleted, TrackNotLiked, TrackLiked\}$$

The State Space X

$X = \{NotCreated, Created, NotLiked, Liked, Played, NotPlayed\}$

The Initial State X_0

$$X_0 = \{ArtistNotCreated, TrackNotCreated, AlbumNotCreated, PlaylistNotCreated, TranscoderNotCreated, PlayNotStarted, TrackNotLiked\}$$

Transition Table

	Create	Update	Delete	Play	Like
ArtistNotCreated	ArtistCreated	x	x	x	x

	Create	Update	Delete	Play	Like
ArtistCreated	x	ArtistCreated	ArtistNotCreated	x	x
TrackNotCreated	TrackCreated	x	x	x	x
TrackCreated	x	TrackCreated	TrackNotCreated	x	x
AlbumNotCreated	AlbumCreated	x	x	x	x
AlbumCreated	x	AlbumCreated	AlbumNotCreated	x	x
PlaylistNotCreated	PlaylistCreated	x	x	x	x
PlaylistCreated	x	PlaylistCreated	PlaylistNotCreated	x	x
TranscoderNotCreated	TranscoderCreated	x	x	x	x
TranscoderCreated	x	TranscoderCreated	TranscoderNotCreated	x	x
PlayNotStarted	x	x	x	PlayStarted	x
PlayStarted	x	x	x	x	x
PlayCompleted	x	x	x	PlayStarted	x
TrackNotLiked	x	x	x	x	TrackLiked
TrackLiked	x	x	x	x	TrackNotLiked

Last.fm Integration

Action Space U

$U = \{Create, Update, Delete, Pause, Love\}$

The Output Space Y

$$Y = \{UnlovedTrack, LovedTrack, UserSessionNotCreated,\\ UserSessionCreated, TrackPaused, TrackUnpaused\}$$

The Space X

$X = \{Unloved, Loved, Created, NotCreated, Paused, Unpaused\}$

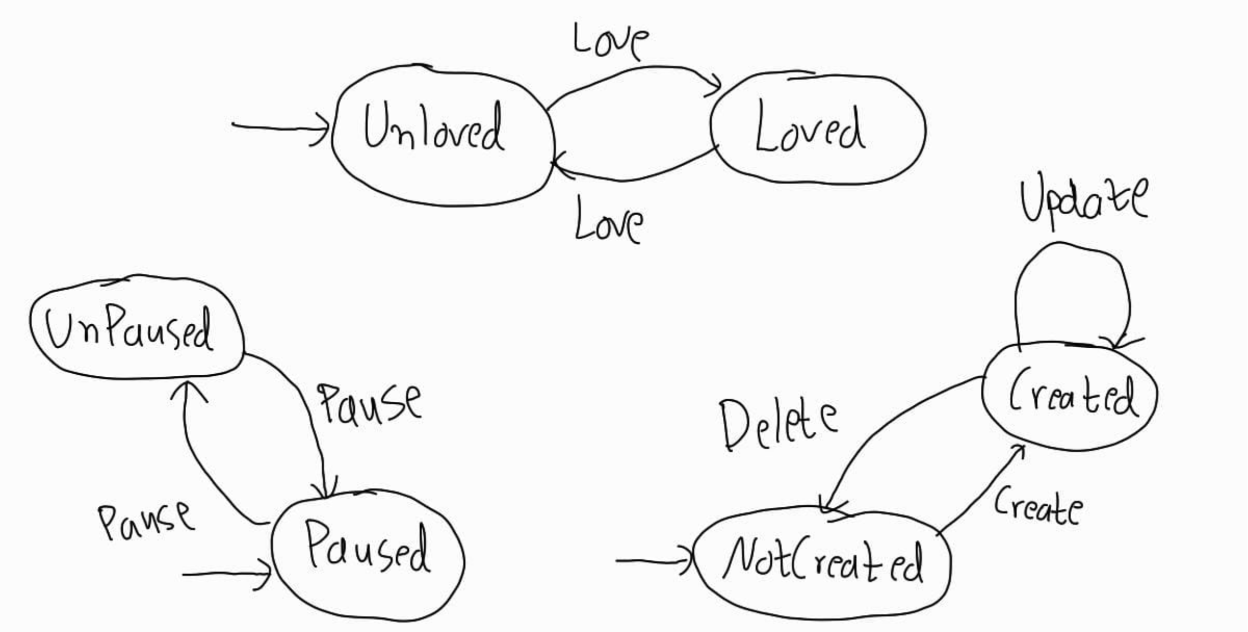
The Initial State X_0

$$X_0 = \{UnlovedTrack, UserSessionNotCreated, TrackPaused\}$$

Transition Table

	Create	Update	Delete	Pause	Love
UnlovedTrack	x	x	x	x	LovedTrack
LovedTrack	x	x	x	x	UnlovedTrack
UserSessionNotCreated	UserSessionCreated	x	x	x	x
UserSessionCreated	x	UserSessionCreated	UserSessionNotCreated	x	x
TrackPaused	x	x	x	TrackUnpaused	x
TrackUnpaused	x	x	x	TrackPaused	x

Diagram



Administrative Features

The Action Space U

$U = \{Create, Update, Delete, AssignRole\}$

The Output Space Y

$Y = \{RoleNotCreated, RoleCreated, RoleAssigned\}$

The State Space X

$X = \{NotCreated, Created, Assigned\}$

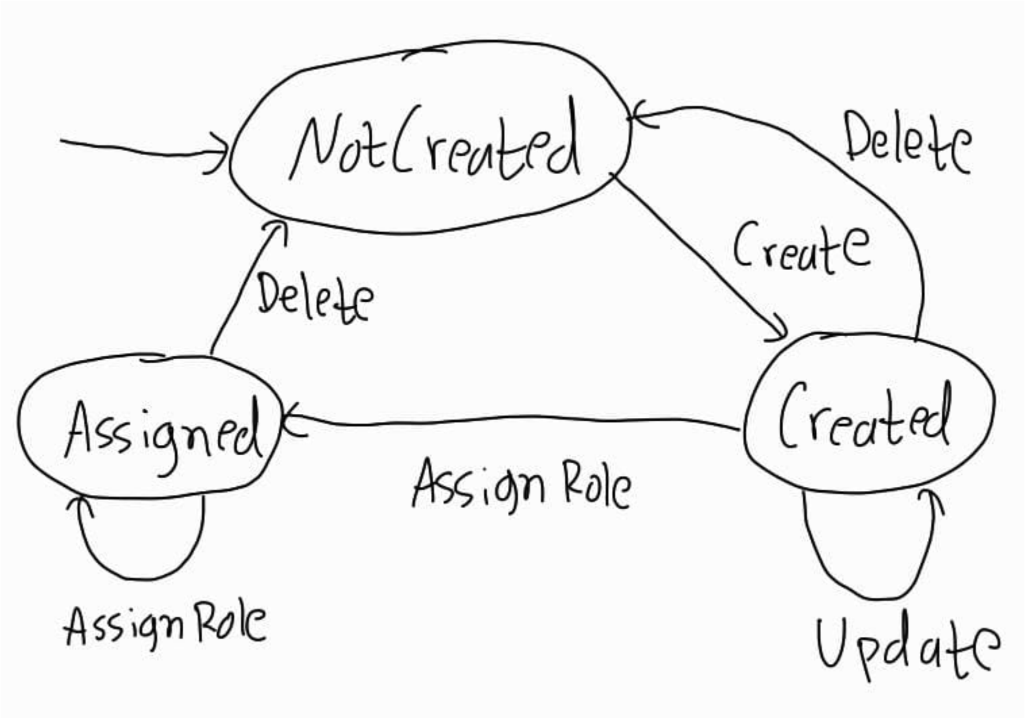
The Initial State X_0

$X_0 = \{RoleNotCreated\}$

Transition Table

	Create	Update	Delete	AssignRole
RoleNotCreated	RoleCreated	x	x	x
RoleCreated	x	RoleCreated	RoleNotCreated	RoleAssigned
RoleAssigned	x	x	RoleNotCreated	RoleAssigned

Diagram



Task 3e: Automatic Refactoring

The paper "Automated refactorings in Java using IntelliJ IDEA to extract and propagate constants" describes a technique for improving the maintainability and readability of Java code through automated refactoring. The authors focus on extracting constants from code and propagating them throughout the codebase.

The paper first explains the motivation for extracting constants, which includes reducing duplication and improving code maintainability. The authors then describe their approach to extracting and propagating constants using the IntelliJ IDEA IDE. They demonstrate using several automated refactoring tools within the IDE, including "Extract Constant" and "Inline Constant".

The authors evaluate their approach using a set of open-source Java projects and report on the improvements achieved through automated constant extraction and propagation. They find that their approach reduces code duplication, improves code readability, and reduces the number of bugs caused by changes to constant values.

Overall, the paper provides a practical and effective approach for improving Java code quality through automated refactoring, specifically on constant extraction and propagation.

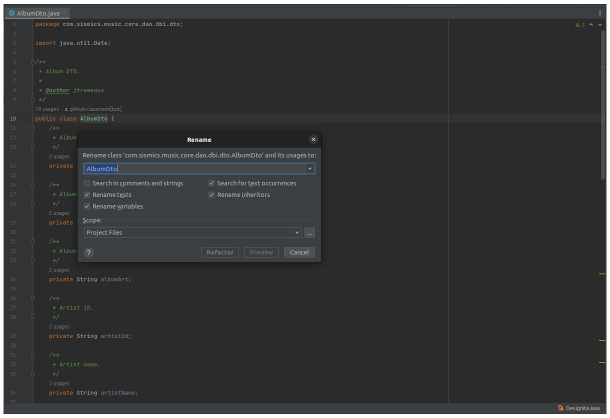
Functionalities Provided by IntelliJ IDEA

IntelliJ IDEA provides several functionalities to refactor code. A summary of these functionalities:

1. Renaming: Renaming involves changing the name of a class, method, variable, or other elements in the codebase. This can be done using IntelliJ IDEA's "Rename" refactoring tool, which automatically updates all references to the renamed element.
2. Extracting methods involve taking a block of code and turning it into a separate method, which can improve readability and maintainability. This can be done using IntelliJ IDEA's "Extract Method" refactoring tool.
3. Moving code involves relocating a class, method, or other elements to a different package or file. This can be done using IntelliJ IDEA's "Move" refactoring tool.
4. Changing signatures: Changing signatures involves modifying the parameters or returning the type of a method. This can be done using IntelliJ IDEA's "Change Signature" refactoring tool.
5. Inlining code: Inlining code involves taking a method call and replacing it with the contents of the method. This can be done using IntelliJ IDEA's "Inline" refactoring tool.

6. Introducing variables: Introducing variables involves replacing a complex expression with a named variable, which can improve readability and maintainability. This can be done using IntelliJ IDEA's "Introduce Variable" refactoring tool.
7. Extracting interfaces: Extracting interfaces involves creating an interface that defines a set of methods and then implementing that interface in multiple classes. This can be done using IntelliJ IDEA's "Extract Interface" refactoring tool.

Preview of the Refactoring Functionalities

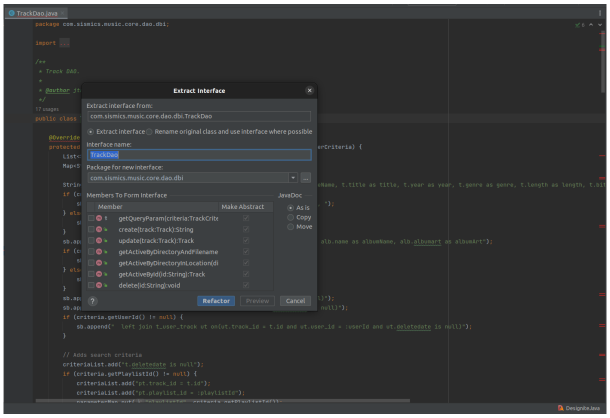


Rename class 'com.sismics.music.core.dao.dbi.dto.AlbumDto' and its usages to: AlbumDto

☐ Search in comments and strings ☑ Search for text occurrences
☑ Rename tests ☑ Rename inheritors
☑ Rename variables

Scope: Project Files

Renaming



Extract interface from: com.sismics.music.core.dao.dbi.TrackDao

● Extract interface ○ Rename original class and use interface where possible

Interface name: TrackDao

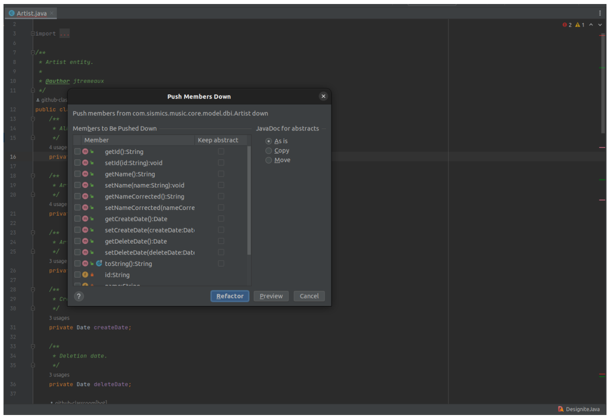
Package for new interface: com.sismics.music.core.dao.dbi

Members To Form Interface

Member	Make Abstract
☐ getQueryParam(criteria:TrackCrite	☑
☐ create(track:Track):String	☑
☐ update(track:Track):Track	☑
☐ getActiveByDirectoryAndFilename	☑
☐ getActiveByDirectoryInLocation(di	☑
☐ getActiveById(id:String):Track	☑
☐ delete(id:String):void	☑

JavaDoc: ● As is ○ Copy ○ Move

Extract



Push members from com.sismics.music.core.model.dbi.Artist down

Members to Be Pushed Down

Member	Keep abstract
☐ getId():String	☐
☐ setId(id:String):void	☐
☐ getName():String	☐
☐ setName(name:String):void	☐
☐ getNameCorrected():String	☐
☐ setNameCorrected(nameCorre	☐
☐ getCreateDate():Date	☐
☐ setCreateDate(createDate:Dat	☐
☐ getDeleteDate():Date	☐
☐ setDeleteDate(deleteDate:Dat	☐
☐ toString():String	☐
☐ id:String	

JavaDoc for abstracts: ● As is ○ Copy ○ Move

Push Down