Rochester Institute of Technology
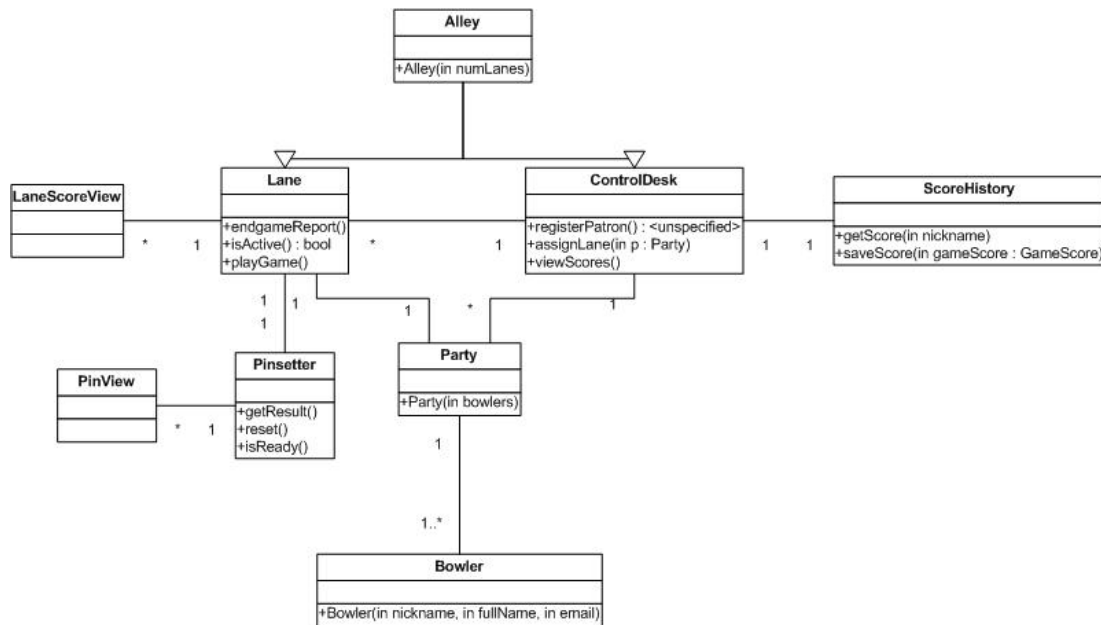Software Engineering Department

Team Name/Project Name	xxx
Document	Product Design Document

Note: this is a "living document", meaning its content and format will grow with the implementation of the project. Use it to capture key project concepts and to document important design decisions.
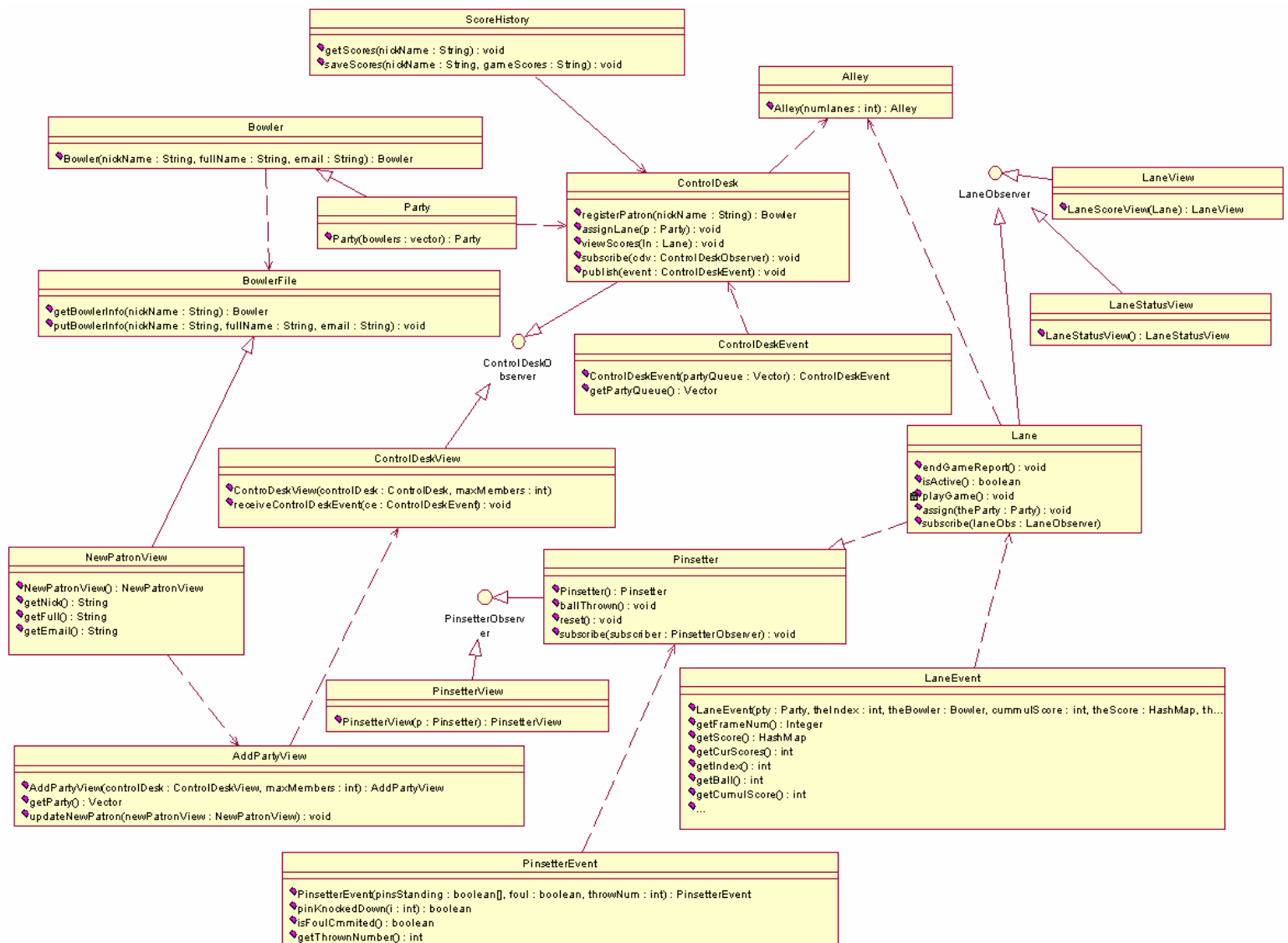
Conceptual Design

Components and Services

Patron Management	- Register Patron - Patron Checks in with System - Patron Checks out - Patron wishes to e-mail scores
Lane Management	- Party is assigned to a lane - It's a Bowler's turn to Bowl - The Pinsetter Cycles - The Pinsetter Resets - The Pinsetter Sends an Event - ControlDesk Operator wishes to view lane scores - Bowler encounters mechanical problem - Lane receives an update event from the pinsetter - The party leaves the lane - The party wishes to bowl another game
Game Management	- A Party stars a new Game - A Party finishes a Game
Score Management	- The Pinsetter returns the results of a user's throw - The Score display is updated
User Interface	- A LaneView receives a LaneEvent - A PinsetterView receives a PinsetterEvent - User is prompted if their party wants to play another game

Class Diagram

ScoreHistory

- getScores(nickName : String) : void
- saveScores(nickName : String, gameScores : String) : void

Alley

- Alley(numlanes : int) : Alley

Bowler

- Bowler(nickName : String, fullName : String, email : String) : Bowler

Party

- Party(bowlers : vector) : Party

ControlDesk

- registerPatron(nickName : String) : Bowler
- assignLane(p : Party) : void
- viewScores(ln : Lane) : void
- subscribe(cdv : ControlDeskObserver) : void
- publish(event : ControlDeskEvent) : void

LaneView

- LaneScoreView(Lane) : LaneView

LaneObserver

LaneStatusView

- LaneStatusView() : LaneStatusView

BowlerFile

- getBowlerInfo(nickName : String) : Bowler
- putBowlerInfo(nickName : String, fullName : String, email : String) : void

ControlDeskObserver

ControlDeskEvent

- ControlDeskEvent(partyQueue : Vector) : ControlDeskEvent
- getPartyQueue() : Vector

ControlDeskView

- ControlDeskView(controlDesk : ControlDesk, maxMembers : int)
- receiveControlDeskEvent(ce : ControlDeskEvent) : void

Lane

- endGameReport() : void
- isActive() : boolean
- playGame() : void
- assign(theParty : Party) : void
- subscribe(laneObs : LaneObserver)

NewPatronView

- NewPatronView() : NewPatronView
- getNick() : String
- getFull() : String
- getEmail() : String

Pinsetter

- Pinsetter() : Pinsetter
- ballThrown() : void
- reset() : void
- subscribe(subscriber : PinsetterObserver) : void

PinsetterObserver

LaneEvent

- LaneEvent(pty : Party, theIndex : int, theBowler : Bowler, cummulScore : int, theScore : HashMap, th...
- getFrameNum() : Integer
- getScore() : HashMap
- getCurScores() : int
- getIndex() : int
- getBall() : int
- getCumulScore() : int
- ...

PinsetterView

- PinsetterView(p : Pinsetter) : PinsetterView

AddPartyView

- AddPartyView(controlDesk : ControlDeskView, maxMembers : int) : AddPartyView
- getParty() : Vector
- updateNewPatron(newPatronView : NewPatronView) : void

PinsetterEvent

- PinsetterEvent(pinsStanding : boolean[], foul : boolean, throwNum : int) : PinsetterEvent
- pinKnockedDown(i : int) : boolean
- isFoulCmmited() : boolean
- getThrownNumber() : int

Design Notes

- *Decided to use Model View Controller for display of user interface*

- *Adopted Observer Pattern for passing events between Objects*

- *Multithreading was implemented for each lane and the control desk to manage its tasks.*