MUN Society System

Classes with Major Responsibilities

STUDENT
-create society
-join society

MEMBER
-all abilities of **STUDENT**
-leave society
-declare eligibility to be elected **PRESIDENT**
-vote for **PRESIDENT**
-create events
-invite members to society

SOCIETY (SYSTEM)
-promote society
-handle messaging, with input from President/Board Members
-remind students about various things, eg. society fees, upcoming events
-sanction society (20+ members), de-sanction society (19- members, no President elected within 4 weeks of sanctioning)
-delete societies (MUN's discretion, no election had within 6 weeks of creation)

BOARD MEMBER (PRIOR TO A3, WAS SIMPLY A STUDENT WITH A BOOLEAN TRUE IF A BOARD MEMBER)
-all abilities of **MEMBER**
-collect society fees
-call meetings
-accept applications to join society
-remove members from society

PRESIDENT (PRIOR TO A3, WAS SIMPLY A STUDENT WITH A BOOLEAN TRUE IF A PRESIDENT)
-all abilities of **BOARD MEMBER**
-call election
-appoint board members
-add students to ballot

-delete society at discretion of MUN, and if necessary, rest of society

ELECTION
-ability to create ballot
-ability to conduct vote electronically

<u>Classes with Features</u>

STUDENT
-includes name, student number, major

MEMBER
-all features of **STUDENT**
-boolean for eligibility to be elected

SOCIETY
-includes name, contact info, student number of current President, major, description, list of members, and boolean which is true if sanctioned

BOARD MEMBER (PRIOR TO A3, WAS SIMPLY A STUDENT WITH A BOOLEAN TRUE IF A BOARD MEMBER)
-all features of **STUDENT**

PRESIDENT (PRIOR TO A3, WAS SIMPLY A STUDENT WITH A BOOLEAN TRUE IF A PRESIDENT)
-all features of **STUDENT**

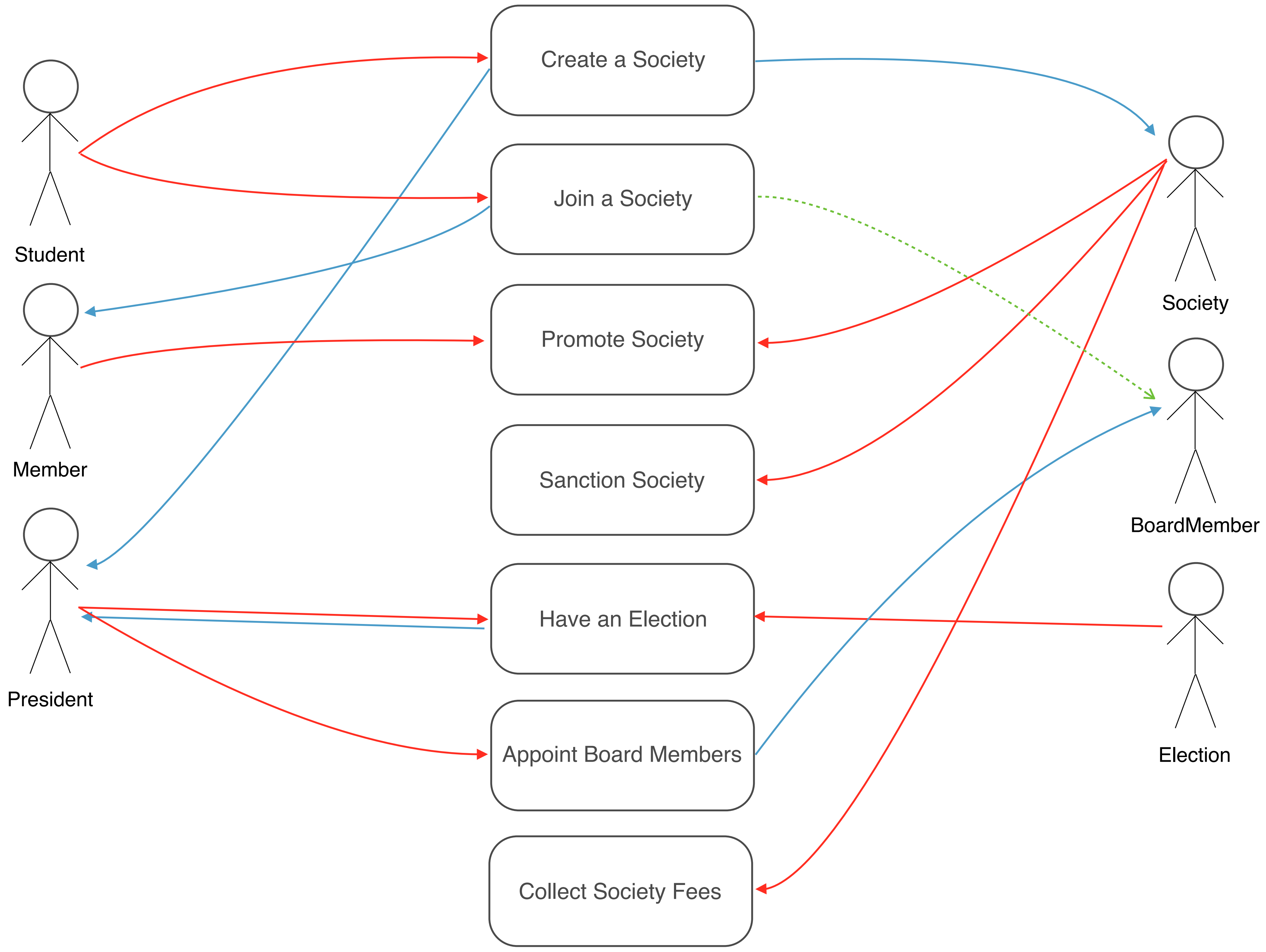
ELECTION
-includes list (ballot) of candidates

MEETING
-includes date, time, location, and purpose
-can be cancelled

EVENT
-all features of **MEETING**
-includes name

Use Cases

Main Path	Alternate Paths
1. Student creates society	
2. Other students join society	2.1 Steps 2 and 3 occur in reverse order
3. Society promotes itself to student body	
4. Society becomes sanctioned	4.1 Society fails to achieve 20 members (and therefore be sanctioned) within 6 weeks of creation and is deleted from system
5. Society has election	5.1 Society fails to have election within 4 weeks of being sanctioned, is no longer sanctioned. It has another 6 weeks to have an election or be deleted
6. President appoints Board Members	6.1 President does not appoint board members (step 6 is optional)
7. Society (Board Members) begins to collect Society fees	7.1 Society chooses not to collect society fees (step 7 is optional)
8. Members create events	Steps beyond this point have to do with the normal functioning of a society
9. Board Members call meetings	
10. Society continues to run so long as members exist (members can graduate, leave MUN, etc.)	10.1 All members of a society disband and society is deleted 10.2 Not enough members remain to remain sanctioned after some members leave, and after a period of 6 weeks society is deleted


Student
Member
President
Create a Society
Join a Society
Promote Society
Sanction Society
Have an Election
Appoint Board Members
Collect Society Fees
Society
BoardMember
Election

CRC Cards and Domain Model

Society
getName():String, setName(String)
getContact():String, setContact(String)
getPresidentID():double, setPresidentID(double)
getMajor():String, setMajor(String)
addMember(Student), removeMember(Member)
Sanction()
Promote()
haveElection()
haveMeeting()
haveEvent()
collectFees()
Disband():String

Election
createBallot(Society)
conductVote(Society)

Student
getName():String, setName(String)
getID():double, setID(double)
getMajor():String, setMajor(String), declareMajor(String)
createSociety():Society, joinSociety(Society)
isStudent():boolean

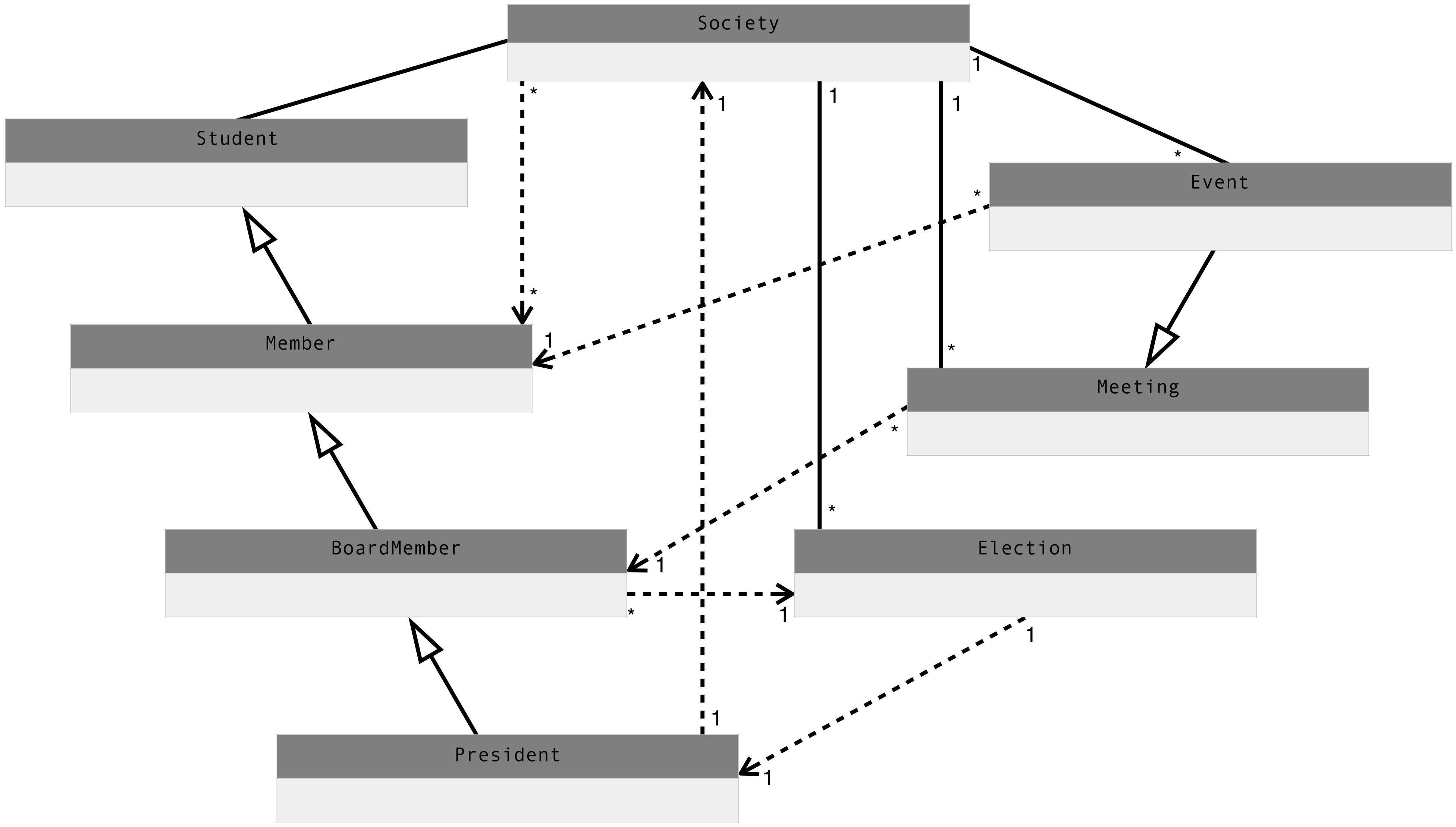
Member
isEligible():boolean, Declare(), Withdraw()
setPresident():President, setBoardMember():BoardMember
createEvent()
Promote(Society)
Leave():Student

BoardMember
reviewApplication(Student)
callMeeting()
removeMember(Society, Member)

President
appoint(Member)
callElection()

Meeting
getDate():String, setDate(String)
getTime():String, setTime(String)
getLocation():String, setLocation(String)
getPurpose():String, setPurpose(String)
cancelMeeting()

Event
getName():String, setName(String)
cancelEvent()


Society
Student
Event
Member
Meeting
BoardMember
Election
President
1
*
1
1
1
*
*
*
1
*
*
*
1
*
1
1
1
1

Modules Needed to Handle Functionality of Our System

- **Don’t Repeat Yourself** is necessary here; we are working with many classes which often make references to each other and therefore certain code could have been repeated but was not

- **Single Responsibility Principle** was applied; a Member cannot appoint itself to be a Board Member, but other things were changed to reflect this Principle

- **Liskov Substitution Principle** was applied; this was done before we knew of this principle. Any President has the same abilities and freedoms as any Member, or Student, and so on

- **Delegation** was used since the beginning; many functions called by Member, BoardMember, and President use each other but are all delegated through Society first

Traceability Matrix

Responsibility Class	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Society	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Election										✓	✓	✓
Student	✓	✓										
Member			✓	✓			✓					
Board Member		✓				✓						
President									✓	✓		
Meeting						✓						
Event							✓					

R1 - Student creating a Society	R6 - Having a Meeting	R11 - Creating a ballot
R2 - Student joining a Society	R7 - Having an Event	R12 - Conducting a vote
R3 - Member leaving a Society	R8 - Collecting Society Fees	
R4 - Promoting a Society	R9 - Disbanding a Society	
R5 - Sanctioning a Society	R10 - Having an Election	

Key Decisions Made During Assignment 3 Process and Why

- President and BoardMember classes are created because having Members shown to be the President or a Board Member using booleans is admittedly primitive

- Various changes were made to our code from Assignment 2 because it was skeleton code and was filled with basic errors

- **Delegation** was used from the beginning, but was used more in this assignment for simplicity

- Inheritance was kept for simplicity; we thought about making Member, BoardMember, and President separate but decided to keep it to fulfill **Don't Repeat Yourself**

Additional Notes

- Our code is not completely ready for testing, but the number of errors have been limited to one main error, where we attempt to search the ArrayList<Member> for BoardMembers and Presidents in 2 instances. Though BoardMember and President are Members, this error persists and we have yet to figure it out.

- Of the most **<u>ARCHITECTURAL SIGNIFICANCE</u>**:
  - Society (The pivot point of almost all interaction and the host of the important aspects of the system)
  - Member (The class that is the basis of all other types of member and is required for a Society to work)
  - Student (The class that is the basis of all other "people" classes)
  - Election (The class that changes which member is President, and is needed for a Society to be sanctioned)
  - President (The creator is made President by default, and a President is needed for some essential features)
  - Meeting/Event (Optional events)
  - BoardMember (Optional roles)