**Criterion B: Design**

The solution will incorporate three databases:

* One for the tutors registered for the program
* One for the tutees registered for the program
* One for the matches made (for the “active” tutors/tutees)

For the offline portion that will take care of managing of the database and attendance:

The offline portion will have 4 classes:

* Tutee
* Tutor
* Match
* TutoringDBMS (database management system)

Classes Tutee and Tutor have:

* Class fields (<type> < identifier>):
* int MATH/CHEMISTRY/etc… (to store the integer identifiers of each subject)
* used by classes Match and Student
* Instance fields (<type> < identifier>):
* String name
* int studentNumber
* char gender
* String email
* String homeroom teacher
* String subject
* This will be stored as “1 2 3” etc, each integer identifier separated by spaces
* boolean monday/tuesday/etc…
* this is for the days on which the tutor/tutee is free
* int attendance
* Methods:
* Associated methods for **accessing** and **mutating** the above instance fields
* **Write** to Student database

class Match (renamed from Pair to accommodate for more than one tutor/tutee) has:

* Class fields (<type> < identifier>):
* int MATH/CHEMISTRY/etc… (to store the integer identifiers of each subject)
* used by classes Match and Student
* Instance fields (<type> < identifier>):
* Student[] tutor
* Student[] tutee
* int numberOfTutors
* int numberOfTutees
* boolean monday/tuesday/etc…
* this is for the days on which the pair (or group) will meet
* String subject
* The subjects that the match will be tutoring/learning
* Methods:
* Associated methods for **accessing** and **mutating** the above instance fields
* Method(s) to **email** the tutor(s) and tutee(s) upon a match being made or for an attendance alert
* **Sort** by day of meeting
* **Write** to Match database

class TutoringDBMS has:

* Methods:
* main (to be run at the beginning of every session)
* prompt for day of week
* here the coordinator enters Monday, Tuesday, etc.
* read from Match database the pairs that are to meet on the given day
* prompt for a string (loop)

a) if a name, find the student and increment attendance by 1

b) if a predetermined code known only by coordinators, present a coordinator menu

* coordinator menu (allows coordinators to make changes)
* add/delete match
* sort