```
int[][][] grades = new int[allStudents.length][numberOfClasses][testsPerClass];
```

I chose a 3d array instead of a 2d array or a 1d array for ‘grades’. The reason for this is that ‘grades’ is to able to hold the data for each test in each class for each student, and a 3d array is the perfect type of array that can actually hold data for that amount of dimensions. If I had attempted to use a 1d array, I would have had to make an array for each student’s classes (which is thousands of arrays for a school with less than a hundred students). If I had attempted to use a 2d array, I would have had to create an array for the gradebooks of each of the students in the school (possibly hundreds of them). So, the best option was to create a singular 3d array that includes all of the gradebook data. In short, the pros of having ‘grades’ be a 2d or 1d array are close to zero. The only possible advantage a 2d array might have is that one could have a 2d array assigned to each student in the school (for classes and tests respectively), and that would make them possibly easier to access. However, the tradeoff of having the bulk of hundreds of arrays to forgo a couple of extra lines of code when writing a nested for loop to traverse the array is most definitely not worth it. Not only does your memory suffer because it had to keep the names of hundreds of arrays as well as their properties in memory, but also it’s a lot more difficult to traverse through multiple arrays instead of a single array. There’s a reason why one can create arrays of arrays in Javascript.

Also, the reason why I made ‘grades’ an integer array instead of a double array is because of the nature of its purpose-to hold grades on a scale from 0 to 100. Most schools don’t have a decimal point in the grades they give. Therefore, there’s no point to make the grades they give be ‘doubles’. All it would do is double the memory the data type would cost. In short, the advantages of having ‘grades’ be a double array are none and the disadvantages include but are not limited to: them being very inefficient because of the memory that they cost, and the fact that having them is completely pointless because non-whole-number grades aren’t even a thing in most schools.

```
Student[] allStudents = {new Student("Jake McMillan"), new Student("Jacob Jonis"), new
Student("Elana Munasinghe")......;
```

The reason why I created an object array for allStudents instead of a String array is because of the versatility of methods I can execute on the array. In the original ‘Student class’, I can write multiple methods to do things such as getting, and modifying. This would not be possible if allStudents was a String array. Also, this being an object array could pave the way for me making possible modifications to the constructor of the Student class, meaning I could hold more data such as studentID, age, school year, and more without having to write a new class or completely change the type of array I am using. In conclusion, the advantages of using an object array are for ease of method use and ease of increasing the capabilities of the array to be able to hold more data. The main disadvantage, however, is the memory and processing power that having an entirely new class that exists simply to hold the methods and data for one array would cost.