Assignment Report for Assignment 04

Course and Section	CSC.215
Assignment Name	Assignment 04
Due Date and Time	October 11th @ 11:55 PM
First Name and Last Name	AJ Arce
SFSU Email Account	Aarce3@sfsu.edu
First Name and Last Name of Teammate	[First Name] [Last Name]
SFSU Email Account of Teammate	[SFSU Email Account]

PART A

Question Description and Analysis:

This part of the assignment asks that I create 1 method that can display the alphabet stored in 2 arrays. One array is a 2D array and wants us to use the shorthand notation to store the alphabet. The other array is a 2D ragged/jagged irregular array and wants us to use the multi-step approach to store the English alphabet.

Answer:

1. This problem asks us to create two different types of 2D arrays to store and display the alphabet. The first way to store the alphabet is through a standard array and the second way is through a jagged/irregular array. The standard array uses shorthand notation while the jagged array uses the multistep method for manual initialization. The client expects the program to display both arrays in a neatly formatted way, where the irregular array aligns the same way as the standard one. In order to solve this, I planned to define the arrays and then create a separate method that prints both of them with the proper spacing. I calculated the number of spaces before printing the standard array so that the alignment

with the regular array was perfect. The goal is to produce output that matches a specific format, so paying extra attention to the spacing and structure is key.

3. The program I created was able to compile and run fine, displaying both the arrays in the correct format and alignment. I was able to align the jagged array by calculating the difference in array lengths and adding however much spaces was needed based on that difference. This made sure that the irregular array matched the format of the regular array. The program meets the client's requirements and works as its supposed to, but there's always room for improvement in every program. In the future, I want to make the method a little more flexible, allowing it to handle different types of arrays or just to format it in a more complex way. I could also add user input to make the program more dynamic.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished…

PART B

Question Description and Analysis:

This part of the assignment asks that I create a program that takes a 2D array with four different rows. Row 1 is 3 Integers, row 2 is 3 characters, row 3 is 3 strings, and row 4 is 1 int, 1 char, and 1 string. Then the program will print out the 2D array in different ways. The first view is the Data Type view and that's when the program automatically detects the data type of the stored data in the array for each type. Then we print the Data Value view which shows what's inside of

the stored data and prints it. Then I'm asked to answer some questions about the problem analysis and solving, and the result analysis and future development.

Answer:

1. The problem asks to create a program that uses a 2D array to store multiple data types using the Object[][] array structure. The key objective is to promp the user to input values of different data types such as integers, characters, and strings. Then we're supposed to store those values into an array. After the data is entered, the program needs to display it in two different ways: "Data Type view" and "Data Value View". Data Type View identifies and displays each data type of the each piece of data stored in the array. The Data Value View retrieves and displays the actual data themselves. I solved this by breaking the task into two different parts. The first, create an input process to prompt the user to enter the data needed for the 2D array. The second, designing a method that will accurately detect the data types and display them in proper format.

3. The program is able to compile and run successfully. It also meets the client's requirements: it stores the different data types in the 2D array and accurately displays them in hoth the Data Type View and Data Value View. What worked well for me is the automatic detection of the data types and the output of the content of the array. There is definitely room for improvement, especially when it comes to making the input process. Adding validation to the input process could prevent incorrect data types from being entered. In future developments, I would also consider allowing the program to handle more complex data structures, such as custom objects and more dynamic input.

PART C

Question Description and Analysis:

This part of the assignment asks that I create a java program to track the growth of a plant. We have set arrays for the average temperature and average rainfall. It was us to take user input for the minimum temperature, maximum temperature, and minimum rainfall for the plant. With these values, we're asked to calculate the plant growth and height and display it within a table that's well formatted and easy to use. Then we're asked about problem analysis and result analysis.

Answer:

1. The problem asks us to create a program that helps keep track of the growth of a plant throughout the year, given the average temp and rainfall data for each month. The goal is to prompt the user for the plant's growing conditions. This includes the min and max temperature as well as the minimum rainfall. The program then has to calculate whether the plant grows or dies back each month, depending on whether the average temperature falls within the range the user inputs. If the temperature is outside of the range, the plant's growth is set to -1. The growth is calculated as the difference between the average rainfall and the plants minimum rainfall. To solve the problem, I will use loops to iterate over the data, applying conditional logic to determine if the plant grows or dies each

month. The program will calculate the plant height as well. By dividing the tasks into different methods, the code will remain modular and easy to manage.

3. The program works as expected, calculating plant growth and height for each month and preventing the height from dropping below 0. The welcomeAndInputs() method organizes all user inputs, while the plantGrowthCalc() method performs the core logic of determining the plants monthly gross. plantHeight() method ensures that the plant height is accurately tracker, and the printTable() method design improves code clarity and makes it easier to modify or extend in the future. Each method serves a distinct role and contributes to the functionality and readability of the program.

PART D

Question Description and Analysis:

This part of the assignment asks that I download and use the java class inside of the zip file. It then wants us to implement that java class into a new class by adding more code. It asks that I do a problem analysis and result analysis for the program as well.

Answer:

1. The problem required me to create a client program that interacts with the given Student class that was given to us. Iw as then asked to add functionality to create students, display their information, update their information, and show the updated results as well. The goal was to keep the solution easy to maintain without modifying the Student class. To solve the problem, I broke it down into smaller tasks. I used an array to store the student data, and created methods for each of functionalities. I had methods for adding the

students, displaying the information, and updating details on the students. Taking this approach kept the code organizded and made it a bit easier to understand and debug. I also made sure the user prompts were easy to go through, and that the output matched the format that was required. This way each method had a specific purpose and made the overall program easy to manage.

3. The program was able to meet all the requirements. I was able to add, display, and update student information easily. Each method played their role well and made sure the code was easy to test and modify. Here are the 5 meaningful methods I used:

1. **addstudents()**: This method handled the user input for creating new students. It ensured that each student is properly initialized and added to the list. Its role is important because it sets up the core data the program will change/update.
2. **displayStudentInfo()**: The method displays the details of a single student. It keeps the code modular by separating the display logic for individual students, which can be reused whenever needed.
3. **displayAllStudents():** This method displays all student's information by calling the previous method mentioned for each student in the list. This separation makes it easy to manage the display logic and provides a display thati s clear and shows all student data.
4. **updateStudentInfo():** This method allows the user to update an existing student's details based on their name and GPA. It is meaningful because it provides interactivity, allowing the program to modify values after the initial input.
5. **main():** The main method coordinates all the functionality, it acts at the control center for user interaction. It calls other methods in a logical sequence and ensures that the user is able to create, view, and update student information.

I noticed that adding input validation could make the program more user-friendly, as it would prevent incorrect inouts. Future imporvements could also include adding functionality to remove students. Swtiching from arrays to an ArrayList could also improve scalability. The program meets the requirements, but there are definitely more opportunities to make it even better and easier to use.

PART E

Question Description and Analysis:

This part of the assignment asks that I create two programs that take in the 4 arrays provided. With the 4 arrays we create program that is able to use them to create one 3D string and the other program we create a Container OOP with a 1D array.

Answer:

1c. In my approach, I made the four 1D arrays that represent a semester's courses. These arrays are then combined into a 3D array degreePlanner. The structure of the 3D array is [4][3][2]. I used nested loops to assign the courses from 1D arrays to the 3D array. By organizing the data like this, it was easy to retrieve and manipulate each course based on its semester and position within the pair.

1d. To print the data from the 3D array, I used a nested loop structure. The outer loop goes over each semester, the middle loop goes through the course pairs in each semester, and the innermost loop goes over each course within a pair. During the printing, I made sure that the output was formatted properly and neatly by printing the courses with commas between them, beside the end

of he list. This made sure that the output matches the specified format properly while displaying all the 24 items in the 3D array.

1e. The two meaningful methods I created are turnInto3DArray and main.

- turnInto3DArray: This method organizes the courses from four 1D arrays into one 3D array. It simplifies the data loading process and makes the program more modular and maintainable. This method is meaningful as it abstracts the data organization logic and allows the main method to focus on the output.
- main: The main method serves as the primary entry point of the program, it calls turnInto3DArray to organize the data in the way it wants and then prints the results. This is meaningful because by separating data organization from data input it enhances readability and maintainability.

2c. In this approach, the data for each semester is stored in a Semester class, which holds an array of six courses. The method getSemesterArray creates an array of four Semester objects, each of these objects represents a semester and assigns a semester with a set of six courses by invoking the method getSemester for each, The courses are generated dynamically using a base number to match the semester courses numbers. This structure allows each Semester object to manage its data independently. The key benefit to this approach is that it separarates the data management from the logic of organizing and retrieving them.

2d. To print the information, getAllSemesters method iterates over the Semester[] array, calling getSemester on each Semester object. Inside the getSemester, the getUserCourses method is called, and that returns a string of all courses for that semester. This design allows each semester to manage and print its own data. The benefit of this method is that the logic for printing the data is abstracted away into the getSemester method and this ensures that the main method remains

clean. This structure also allows you to easily adjust how semesters are printed without needing to modify how they are stored.

2e.

- getSemesterArray: This method is responsible for creating an array of Semester objects. This is meaningful because it abstracts the logic of initializing each semester which makes the code easy to maintain.
- getSemester: This method prints the courses for a given semester in a readable format. This is meaningful because it makes sure that the presentation log is separate from the logic used to retrieve data.
- getCourses: This method generates the course codes for each semester based on the semester number. This is meaningful because it makes the code flexible and eliminates the need to add the courses manually.
- getUserCourses: This method inside the Semester class formats the courses into a readable string. This is meaningful because it handles the internal representation of the course data and makes sure that the formatting and presentation is within the Semester object.
- getAllSemesters: This method loops through the array of Semester objects and prints each of them. This is meaningful because it serves as the main control for how the data is displayed and ensures that all semesters are printed in the right format.

Screenshots of Outputs and Explanation:

These screenshots show what I accomplished…