**Short Answer Questions Guidelines**

* Download this assessment to your local computer
* Upload your answers to your repository at the end of each period (Today & Tomorrow)
* Answer the questions using MS Word
* For each question clearly identify each of the points you are answering
* Provide complete sentences for each point with clear details and justification
* Clearly format included Java code samples as required for some questions
* Answer any 8 out of the 9 questions from the list below
* Only the first 8 questions will be marked
* Each question is worth 5 marks
* The total for this summative is 40 marks

**Short Answer Questions**

1. Describe a situation in class where one-dimensional and two-dimensional arrays were used to store and manage data. Structure your answer as follows.
   * Summarize the work or activity you did that links to the topic.
   * Explain specifically how the work or activity is related to the topic
   * Provide or explain specific examples of your work. Include sample Java code.
   * For additional marks, provide sample Java code to add, change, and delete elements of the array.

A one dimensional array is commonly used for storing single row elements. A situation where I used a one-dimensional array was when I had to store the String based titles/headers within a data table in which was when we were working on the Student DataBase assignment and my part was the interface. More specifically, the program needed to store student information as a record and various functionalities such as adding, deleting and modifying any existing records in relation to generating unique IDs for new student records.

**The array was stored as a private constant as shown below:**

//Stores the row titles as strings

private static String[] titles = {"Last Name:", "First Name:", "Grade:", "Student ID:", " ", " " };

When I was working on the 3D TicTacToe assignment I had to create a two dimensional array that would represent the 36 possibilities of the Gameboard. In this case, when looking at the sample code below in which is stored as a private variable created for the purpose of only my personal class that is the gameboard. This represents the possibilities of 1 to 8, 9 to 17, 18 to 36 on the dimensions of the GameBoard

**This is the line of code:** private String[] gameBoard = new String[36];

3. Describe a situation in class where code was developed to implement classes and objects. Structure your answer as follows.

* + Summarize the work or activity you did that links to the topic.
  + Explain how classes and objects are related but are also different
  + Provide an example of a class that includes a constructor and at least one method. Include sample Java code.
  + Explain, using your example class, how an object can be created and used. Include sample Java code.

I discovered the basics of class and object oriented programs when we worked on the Student Database project. Where the design of the program was to store student records and have the ability to add, delete, and modify existing records while generating unique IDs. My part being to create the SWT based interface so that way all the student data can displayed and used.

Classes are basically a place to store specific data. The code below shows a class called StudentInfoClientInterface. The constructor called StudentDatabase studentDB stores the user’s first name, last name and student ID. In this case the constructor uses 3 parameters. The Other methods are created for the purpose of the many button functionalities and overall table structure needed for the program’s interface.

public class StudentInfoClientInterface {

//Stores the row titles as strings

private static String[] titles = {"Last Name:", "First Name:", "Grade:", "Student ID:", " ", " " };

public static void main(String[] args) {

StudentDatabase studentDB = new StudentDatabase();

StudentRecord newStudent = new StudentRecord("Daniel", "Gopal", "1234");

studentDB.addStudent(newStudent);

newStudent = new StudentRecord("Kiran", "Hart", "5678");

studentDB.addStudent(newStudent);

newStudent = new StudentRecord("Antonio", "N-J", "2468");

studentDB.addStudent(newStudent);

newStudent = new StudentRecord("Robin", "Saran", "1357");

studentDB.addStudent(newStudent);

newStudent = new StudentRecord("Rahul", "Tailor", "7899");

studentDB.addStudent(newStudent);

newStudent = new StudentRecord("Calvin", "Ye", "1111");

studentDB.addStudent(newStudent);

Public static void AddButton () { // for the Add Button function

This would be a method to call code functionality for the Add Button by Robin

Public static void DeleteButton () { // for the Delete Button function

This would be a method to call code functionality for the Delete Button by Robin

Public static void EditButton () { // for the Edit Button function

This would be a method to call the code functionality for the Edit Button by Calvin

Public static void Columns () { // Creates the table structure by outputting the number of columns

Public static void TableBoxes () { // Holds and lays out the student info from the database (Currently filled as a blank String)

4. Describe a situation in class where code was developed to implement private and public constants, variables and methods in a Java class. Structure your answer as follows.

* + Summarize the work or activity you did that links to the topic.
  + Explain the difference between making a variable “public” or “private”.
  + Provide sample Java code for public and private constants, variables and methods.

When me and Antonio had the part to create the GameBoard for the 3D TicTacToe Assignment. We were assigned to create Public Constants & Variables called playerFree, playerX, and playerO.

I created them as public constants:

public static String playerFree = " ";

public static String playerX = "X";

public static String playerO = "O";

I made these following methods as private because they have no need of being accessed from the group members working on the game’s interface and AI.

private int mapFromLvlRowCol(int lvl, int row, int col) {

int map = (lvl - 1) \* 9; // 1:0 to 8, 2: 9 to 17, 3: 18 to 36

map = map + (row - 1) \* 3;

map = map + (col - 1);

return map;

}

private int levelFromMap(int map) {

int level = java.lang.Math.round(map / 9);

return level + 1;

}

private int rowFromMap(int map) {

int subMap = map % 9;

int row = java.lang.Math.round(subMap / 3);

return row + 1;

}

private int colFromMap(int map) {

int subMap = map % 9;

subMap = subMap % 3;

int col = java.lang.Math.round(subMap / 3);

return col + 1;

}

I made these following methods as public because they need to be accessed from Rahul and Kiran who are working on the game’s interface and require these methods to implement and manage gameboard moves.

//Returns Free, X, or O depending how the board space is occupied

public String checkMove(int lvl, int row, int col) {

//Checks if there are any valid or invalid board positions

int map = mapFromLvlRowCol(lvl, row, col);

return gameBoard[map];

}

//Returns true/false depending if move is valid

public boolean makeMove(String playerXO,int lvl, int row, int col) {

//Checks to see if the user input is valid when they select a letter

if (playerX != "X" || playerO != "O") //If not equal to correct String

{

return false; //Return as invalid

}

//Checks if Board position is occupied/taken

if (checkMove(lvl, row, col) != playerFree) {

return false; //Returns false (invalid) if taken

}

int map = mapFromLvlRowCol(lvl, row, col);

gameBoard[map] = playerXO;

return true; //Returns as true (valid) if not

}

}

When comparing a public to a private variable, public allows for that variable to be accessed from any secondary (other) java classes that are called/extended from that program. When a variable is made private this keeps the variable’s access only within the primary java program it is defined in. Simply said, private is only accessible within the same class it is declared in. While as, public can be accessed by any other class.

5. Describe a situation in class where code was developed to implement a standard mathematical algorithm or to implement a specification provided by your teacher.

* + Summarize the work or activity you did that links to the topic.
  + Explain specifically how the work or activity is related to the topic
  + Provide or explain specific examples of your work. Include sample Java code.

In Module B.3, where we did the SWT Math Project that was a program that will get an integer number from the user using a Spinner widget. The user will then select a common math function to calculate using the list widget. The program implemented many different math functions in blocks of code and as a result was calculated based on the selected Spinner and List values. Simply, our programs contained mathematical formulas represented in Java code that would function to give an answer/output.

We ended up implementing many different math functions such as Square Root, Exponents, Factorial, Logs, Trigonometry, EXP, Percentage and additionally the Clear/Off button. Which was very similar to the functions found on a scientific calculator.

I had the part come up with a few of these functions and how they worked.

**Exponents:** Within this program, there is an integer variable called number which represents the base number that is stored when a user inputs a number and an integer variable called exponent when the user types a number that should represent the power of an exponential equation.

import java.util.Scanner;

public class exponents {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("please type in your number");

int number= sc.nextInt();

System.out.println("please type in your exponent");

int exponent = sc.nextInt();

int E=2;

//variable for answer

int A=0;

E++;

A= number\*number;

while(exponent >= E){

A= A\*number;

E++;

}

System.out.println(A);

}

}

**Off Button:** This program demonstrates the function of turning off (ending) a java program. In this case, when the Off button is selected it will end the java program by closing the SWT window that was originally displayed.

import org.eclipse.swt.SWT;

import org.eclipse.swt.events.SelectionAdapter;

import org.eclipse.swt.events.SelectionEvent;

import org.eclipse.swt.events.SelectionListener;

import org.eclipse.swt.layout.RowLayout;

import org.eclipse.swt.widgets.Button;

import org.eclipse.swt.widgets.Composite;

import org.eclipse.swt.widgets.Display;

import org.eclipse.swt.widgets.Label;

import org.eclipse.swt.widgets.Shell;

public class SWTOffButton {

public static void main(String[] args) {

Display display = new Display();

Shell shell = new Shell(display);

shell.setText("SWT Toggle Button");

RowLayout rowLayout = new RowLayout(SWT.VERTICAL);

rowLayout.marginLeft = 10;

rowLayout.marginTop = 10;

rowLayout.spacing = 15;

shell.setLayout(rowLayout);

//Composite

Composite composite = new Composite(shell, SWT.NONE);

composite.setLayout(new RowLayout(SWT.HORIZONTAL));

Label label = new Label(composite, SWT.NONE);

label.setText("Select: ");

//OFF Button

Button toggleOFF = new Button(composite, SWT.TOGGLE);

toggleOFF.setText("OFF");

Label labelAnswer = new Label(shell, SWT.NONE);

labelAnswer.setForeground(display.getSystemColor(SWT.COLOR\_BLUE));

SelectionListener selectionListener = new SelectionAdapter() {

@Override

public void widgetSelected(SelectionEvent e) {

Button source = (Button) e.getSource();

if (source.getSelection())

{

System.exit(0); //Ends/Closes the Java program

}

}

};

toggleOFF.addSelectionListener(selectionListener);

shell.setSize(400, 250);

shell.open();

while (!shell.isDisposed())

{

if (!display.readAndDispatch())

display.sleep();

}

display.dispose();

}

}

**Clear Button:** This program demonstrates the function of setting already inputted text to a blank string variable. As when the button is clicked this will fill the box as a blank string removing/resetting what was originally typed.

import org.eclipse.swt.SWT;

import org.eclipse.swt.events.SelectionAdapter;

import org.eclipse.swt.events.SelectionEvent;

import org.eclipse.swt.events.SelectionListener;

import org.eclipse.swt.layout.GridLayout;

import org.eclipse.swt.layout.RowLayout;

import org.eclipse.swt.widgets.Button;

import org.eclipse.swt.widgets.Composite;

import org.eclipse.swt.widgets.Display;

import org.eclipse.swt.widgets.Label;

import org.eclipse.swt.widgets.Shell;

import org.eclipse.swt.widgets.Text;

public class SWTClearButton {

public static void main(String[] args)

{

Display display = new Display();

Shell shell = new Shell(display);

shell.setLayout(new GridLayout(1, false));

shell.setText("SWT Clear Button");

//Example text/numbers

Text group = new Text(shell, SWT.NONE);

group.setText("123456");

RowLayout rowLayout = new RowLayout(SWT.VERTICAL);

rowLayout.marginLeft = 10;

rowLayout.marginTop = 10;

rowLayout.spacing = 15;

shell.setLayout(rowLayout);

//Composite

Composite composite = new Composite(shell, SWT.NONE);

composite.setLayout(new RowLayout(SWT.HORIZONTAL));

//Clear Button

Button CLEAR = new Button(composite, SWT.TOGGLE);

CLEAR.setText("Clear");

Label labelAnswer = new Label(shell, SWT.NONE);

labelAnswer.setForeground(display.getSystemColor(SWT.COLOR\_BLUE));

SelectionListener selectionListener = new SelectionAdapter() {

@Override

public void widgetSelected(SelectionEvent e) {

Button source = (Button) e.getSource();

if (source.getSelection())

{

group.setText(" "); //Clears the numbers/text

}

}

};

CLEAR.addSelectionListener(selectionListener);

shell.setSize(400, 250);

shell.open();

while (!shell.isDisposed())

{

if (!display.readAndDispatch())

display.sleep();

}

display.dispose();

}

}

**Trigonometry:**

**//Program for the function of Sine**

package swt\_projects.java;

import java.util.Scanner;

public class Sine {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

double a = 180, b = 40500, c = 4;

System.out.println("Enter your number");

double x = sc.nextInt();

double d = x;

a = a-x;

c = c\*x;

c = c\*a;

d = d\*a;

b = b-d;

c = c/b;

if(180 < x) {

c = c\*-1.0;

if(x < 360) {

c = c\*-1.0;

}

}

System.out.println("x = " + c);

}

}

6. Describe a situation in class where code was developed to implement a graphical user interfaces (GUI). Structure your answer as follows.

* + Summarize the work or activity you did that links to the topic.
  + Explain specifically how the work or activity is related to the topic
  + Provide or explain specific examples of the widgets used to implement the GUI. Include sample Java code.
  + For additional marks, provide sample Java code to add, change, and delete elements of the widgets.

When working on the Student Database the plan was to assign a different part to each person in the group. I was assigned to create the UI of the SWT based interface. It contained the on-screen functionalities such as the buttons and the layout of the table.

I was able to create this output by importing SWT widgets such as:

**import org.eclipse.swt.widgets.Button;**

Which allowed me to create the various perspectives of the graphical buttons and give them a name creating many methods like this one listed below

public static void EditButton(Composite composite) {

//Edit Button (used to modify records)

Button EDIT = new Button(composite, SWT.TOGGLE);

EDIT.setText("Edit");

**import org.eclipse.swt.widgets.Table;**

Allowed me to create the general cell structure of each table box.

//Creates the Table Structure

Table table = new Table(shell, SWT.MULTI | SWT.BORDER | SWT.FULL\_SELECTION);

table.setLinesVisible(true);

table.setHeaderVisible(true);

**import org.eclipse.swt.widgets.TableColumn;**

Allowed me to create and generate the table layout of the rows and columns.

//Counts the number of row headers/titles and outputs a table in relation to the number of rows.

for (int i = 0; i < titles.length; i++)

{

TableColumn column = new TableColumn(table, SWT.NONE);

column.setText(titles[i]);

}

for (int i = 0; i < titles.length; i++)

{

table.getColumn(i).pack();//Generates the cell for each column

}

**import org.eclipse.swt.widgets.TableItem;**

This allowed me to fill the table with any text based content. In this case I filled them all as blank so that way when we generate the student data it will fill in those spots.

//Loop that generates blank boxes and the Add/Delete Button into the table

for (int i=0; i<8; i++) { // Creates 8 columns within 5 rows

TableItem item = new TableItem (table, SWT.NONE);

item.setText (0," " ); //Blank string/box

item.setText (1," " );

item.setText (2," " );

item.setText (3," " );

item.setText (4," " );

item.setText (5," " );

7. Explain the importance of designing reusable and partitioned code in computer programs. Structure your answer as follows.

* + Explain the benefits of separating code into well-defined classes and objects
  + Explain the importance of having well defined interfaces (e.g. public methods)
  + Describe a situation in class where you implemented code based on a specification that was provided.
  + Describe a situation in class where you documented the interface and specification for code you developed.

Coding a program that is structured and organized with defined methods and objects allows for better overall organization of the code itself as creating a method allows for all those sections/blocks of code to be based under the method’s name and not just hard coded into chunks of lines that make it more complex and harder to identify the meaning of each line of code. It makes the program’s code easier to read and understand as they are all split up into separate parts of the program that are in different classes which also makes it much easier to debug and find errors because every block of code is specific to its class.

This makes the program not only easier for the original programmer to understand but in the work we do that is group oriented, others who use the same program can get an understanding of how various methods work and what to call in order to implement it into their own programs.

Splitting up a program into multiple classes can also become quite handy as sometimes you may need to make another program with similar functionalities or other programs that need similar code and you can simply call your already created class instead of having to hard code or reprogram everything.

For example, when I was working on the interface of the Student Database project, I originally had all of my code in the main method of my class.

public class StudentInfoClientInterface {

private static String[] titles = {"Last Name:", "First Name:", "Grade:", "Student ID:",}; //Stores the row titles

public static void main(String[] args) {

// TODO Auto-generated method stub

Display display = new Display();

Shell shell = new Shell(display);

shell.setLayout(new GridLayout(1, false));

shell.setText("StudentInfoClient");

//Creates a Table

final Table table = new Table(shell, SWT.MULTI | SWT.BORDER

| SWT.FULL\_SELECTION);

table.setLinesVisible(true);

table.setHeaderVisible(true);

//Counts the number of row titles and outputs a table in relation to the number of rows

for (int i = 0; i < titles.length; i++)

{

TableColumn column = new TableColumn(table, SWT.NONE);

column.setText(titles[i]);

}

TableItem item = new TableItem(table, SWT.NONE);

//Sets each box as String/text type within the table

item.setText(0, "Antonio"); item.setText(1, "N-J"); item.setText(2, "12"); item.setText(3, "2468");

item = new TableItem(table, SWT.NONE);

item.setText(0, "Kiran"); item.setText(1, "Hart"); item.setText(2, "12"); item.setText(3, "5678");

item = new TableItem(table, SWT.NONE);

item.setText(0, "Robin"); item.setText(1, "Saran"); item.setText(2, "12"); item.setText(3, "1357");

item = new TableItem(table, SWT.NONE);

item.setText(0, "Rahul"); item.setText(1, "Tailor"); item.setText(2, "12"); item.setText(3, "7899");

item = new TableItem(table, SWT.NONE);

item.setText(0, "Calvin"); item.setText(1, "Ye"); item.setText(2, "12"); item.setText(3, "1111");

item = new TableItem(table, SWT.NONE);

item.setText(0, "Daniel"); item.setText(1, "Gopal"); item.setText(2, "12"); item.setText(3, "6143");

for (int i = 0; i < titles.length; i++)

{

table.getColumn(i).pack();//Generates the number of columns

}

//Composite class is a container which is capable of containing other widgets like shell.

//It draws a border around itself and allows you to set a header for the grouped widgets.

Composite composite = new Composite(shell, SWT.NONE);

composite.setLayout(new RowLayout(SWT.HORIZONTAL));

//Edit Button (used to modify records)

Button EDIT = new Button(composite, SWT.TOGGLE);

EDIT.setText("Edit");

//Add Button

Button ADD = new Button(composite, SWT.TOGGLE);

ADD.setText("Add");

//Delete Button

Button DELETE = new Button(composite, SWT.TOGGLE);

DELETE.setText("Delete");

//Exit Button

Button EXIT = new Button(composite, SWT.TOGGLE);

EXIT.setText("Exit");

SelectionListener selectionListener = new SelectionAdapter() {

@Override

public void widgetSelected(SelectionEvent e) {

Button source = (Button) e.getSource();

if (source.getSelection())

{

System.exit(0); //Ends/Closes the Java program

}

}

};

EXIT.addSelectionListener(selectionListener); //Enables the listener when selected

shell.setSize(500, 500);

shell.open();

while (!shell.isDisposed())

{

if (!display.readAndDispatch())

display.sleep();

}

display.dispose();

}

}

This would only make the code less usable for the project as this is supposed to be group oriented programming by which somehow one user’s program must retrieve data or call the other’s methods.

So I ended up splitting this program by creating multiple methods:

public class StudentInfoClientInterface {

//Stores the row titles as strings

private static String[] titles = {"Last Name:", "First Name:", "Grade:", "Student ID:", " ", " " };

public static void main(String[] args) {

// TODO Auto-generated method stub

Display display = new Display();

Shell shell = new Shell(display);

shell.setLayout(new GridLayout(1, false));

shell.setText("StudentInfoClient");

//Creates the Table Structure

Table table = new Table(shell, SWT.MULTI | SWT.BORDER | SWT.FULL\_SELECTION);

table.setLinesVisible(true);

table.setHeaderVisible(true);

//Composite class is a container which is capable of containing other widgets like shell.

//It draws a border around itself and allows you to set a header for the grouped widgets.

Composite composite = new Composite(shell, SWT.NONE);

composite.setLayout(new RowLayout(SWT.HORIZONTAL));

//Initialize data for created methods

Columns(table);//Method for table columns

TableBoxes(table);//Method for generating table boxes/cells

EditButton(composite);//Method for the Edit Button

ExitButton(composite);//Method for the Exit Button

shell.setSize(500, 500);

shell.open();

while (!shell.isDisposed())

{

if (!display.readAndDispatch())

display.sleep();

}

display.dispose();

}

public static void Columns(Table table) {

//Counts the number of row titles and outputs a table in relation to the number of rows

for (int i = 0; i < titles.length; i++)

{

TableColumn column = new TableColumn(table, SWT.NONE);

column.setText(titles[i]);

}

for (int i = 0; i < titles.length; i++)

{

table.getColumn(i).pack();//Generates the cell for each column

}

}

public static void TableBoxes(Table table) {

Button ADD = null;

Button DELETE = null;

//Loop that generates blank boxes and the Add/Delete Button into the table

for (int i=0; i<8; i++) { // Creates 8 columns within 5 rows

TableItem item = new TableItem (table, SWT.NONE);

item.setText (0," " ); //Blank string/box

item.setText (1," " );

item.setText (2," " );

item.setText (3," " );

item.setText (4," " );

item.setText (5," " );

//Add Button

ADD = new Button(table, SWT.TOGGLE);

ADD.setText("Add");

ADD.setEnabled(true);

//Allows for other widgets like buttons to be used with the Table item (cell) functions

TableEditor editor = new TableEditor(table);

editor.grabHorizontal = editor.grabVertical = true;

editor.setEditor(ADD, item, 4);

//Delete Button

DELETE = new Button(table, SWT.TOGGLE);

DELETE.setText("Delete");

editor = new TableEditor(table);

editor.grabHorizontal = editor.grabVertical = true;

editor.setEditor(DELETE, item, 5);

}

//Methods for the Add and Delete Button

AddButton(ADD);

DeleteButton(DELETE);

}

public static void AddButton(Button ADD){

}

public static void DeleteButton(Button DELETE){

}

public static void EditButton(Composite composite) {

//Edit Button (used to modify records)

Button EDIT = new Button(composite, SWT.TOGGLE);

EDIT.setText("Edit");

}

public static void ExitButton(Composite composite) {

//Exit Button

Button EXIT = new Button(composite, SWT.TOGGLE);

EXIT.setText("Exit");

SelectionListener selectionListener = new SelectionAdapter() {

@Override

public void widgetSelected(SelectionEvent e) {

Button source = (Button) e.getSource();

if (source.getSelection()) //checks if button is selected

{

System.exit(0); //Ends/Closes the Java program

}

}

};

EXIT.addSelectionListener(selectionListener); //Enables the listener when the button is selected

}

}

More simply, I have created methods that are supposed to do the following:

**Public static void AddButton () { // for the Add Button function**

This would be a method to call code functionality for the Add Button by Robin

**Public static void DeleteButton () { // for the Delete Button function**

This would be a method to call code functionality for the Delete Button by Robin

**Public static void EditButton () { // for the Edit Button function**

This would be a method to call the code functionality for the Edit Button by Calvin

**Public static void TableBoxes () { // Holds and lays out the student info from**

**the database (Currently filled as a blank String)**

This would be called from the Student Database to access the table and fill it with

student data.

**Public static void Columns () {**

Creates the table structure by outputting the number of columns

8. Describe a situation in class where you participated in a multi-student project involving Java code. Structure your answer as follows.

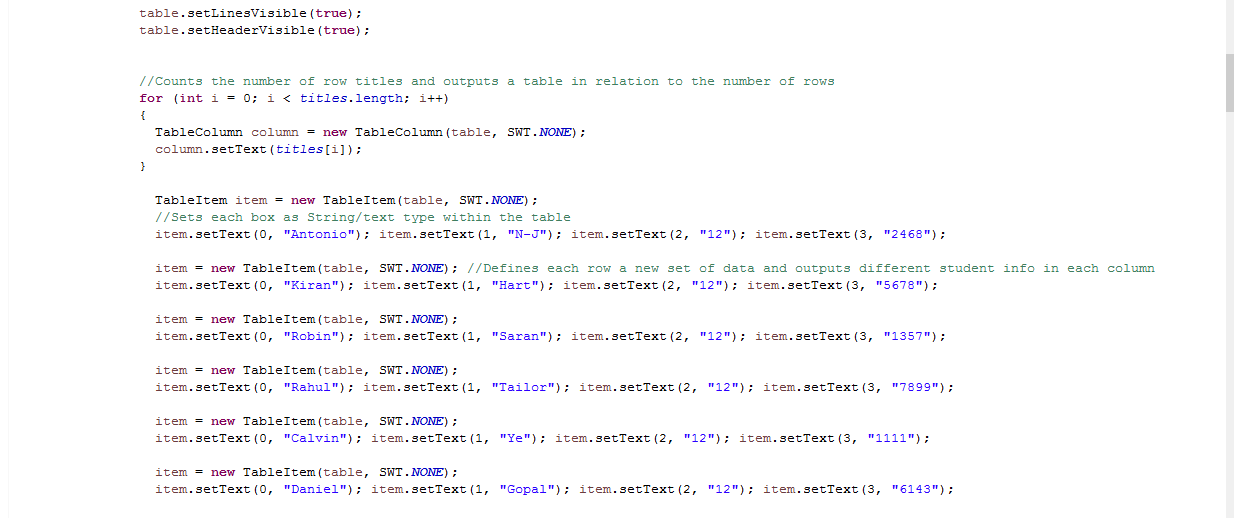
* + Summarize the work or activity you did that links to the topic.
  + Explain the software development plan that was created for the project
  + Explain how students communicated with each other regarding the status of their individual assigned tasks.
  + Explain how the code developed by different students was merged into one project
  + Explain how industry-standard programming tools (e.g. Eclipse, GitHub) are used to support multi-student software projects.

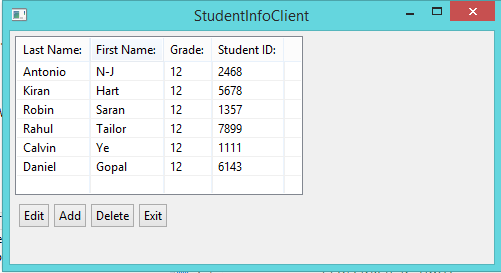
A multi-student project we did was the Student Database. When working on the Student Database the overall plan was to assign a different part to each person in the group. In this case, I was assigned to create the UI of the SWT based interface. It contained the on-screen functionalities such as the buttons and the layout of the table.

When looking at everything we tried to develop with this project, overall we needed to store student records and have the capabilities of adding, deleting, and modifying existing records, and generate unique IDs for the student information.

When it came to communication, we all created small presentations to give a visual and informational look at what each of our individual parts purpose is and how they can all be somehow integrated with each other. While coding our programs we communicated by helping each other like when I discovered some code for the functionality of the Edit (modify) button, I let the person who was responsible for that part in this case Calvin and provided him with some insight on what it actually does so that way it could help him improve his own program.

Me and Antonio parts where situated in the same class as he had to display the Student Info in the SWT window (interface). So we both worked together on parts of the program that were hard coded to display some student info as we tried to see if our program could run and display info without calling the Student Database.





With the program being divided between everyone in the group, each person made separate classes that were based on their assigned parts. We all as a team used a Github repository to upload our programs to a source folder when updated with changes on a daily to weekly basis in which made a good way to understand the structure of how our java projects within Eclipse should similarly be. While also providing a good resource for if we ever need to use another group members program.

The database was supposed to have the main method in their part their program

so that way all methods can somehow be called and executed.

I had created methods to call other members assigned parts:

**Public static void AddButton () { // for the Add Button function**

This would be a method to call code functionality for the Add Button by Robin.

**Public static void DeleteButton () { // for the Delete Button function**

This would be a method to call code functionality for the Delete Button by Robin.

**Public static void EditButton () { // for the Edit Button function**

This would be a method to call the code functionality for the Edit Button by Calvin.

**Public static void TableBoxes () { // Holds and lays out the student info from**

**the database (Currently filled as a blank String)**

This would be called from the Student Database to access the table and fill it with

student data.

9. Describe a situation in class where you worked independently to develop Java code. Structure your answer as follows.

* + Explain how you used help functions and reference documentation to resolve syntax issues (coding issues) while programming. Provide specific examples.
  + Explain how you used reference documentation to find sample code that you could use and modify implement parts of your program. Provide specific examples.
  + Explain how you used the Eclipse environment to debug your program

After we were introduced to the SWT widgets of Eclipse, I worked on my part which is the Student Databases interface. At first, I was getting syntax errors with the errors being indicated on the upper lines of my program that was a few of my imports such as

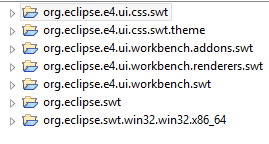
import org.eclipse.swt.widgets.Table;

import org.eclipse.swt.layout.GridLayout;

import org.eclipse.swt.layout.RowLayout;

import org.eclipse.swt.widgets.Button;

This caused every created variable that I have used with the imported types to also have errors because it was not defined correctly. I figured out through an online forum called stackoverflow.com that to define other SWT widget imports such as the ones listed above you need to have a few of these following below enabled within the properties of your Java Project’s Build Path.



As I originally had org.eclipse.swt installed and that was only able to support the SWT widget basics such as Text, Display and Shell.

More importantly, I learned how to originally work with SWT paths and builds by following the guided help instructions listed in Eclipse. Like for example, when we were assigned to setup the Hello World SWT program that we did in class.