Assessment Item 2:

Designing Object-Oriented Program

Bryce Sandilands: 11649401

ITC206-201860

Task One: Logic Explanation

The ProcessMarks class contains all the processing methods to process the array of marks. A separate class TestProcessMarks method is use to test all of the methods in ProcessMarks. All of the methods in TestProcessMarks are designed to help display the results and separate the logic out of the main method

- The `min` method assigns the first element in array to a variable and uses a loop to compare the rest of the elements against the variable to determine and return smallest value.
- The `max` method uses similar logic to the `min` method, instead compares for max values.
- The `range` method calls both min and max methods to determine range. Calling these methods within the `range` method eliminates the need to check the methods are called in a particular order.
- The `mean` method loops through the array and accumulates values to a variable that is then divided by the length of the array to return the mean value.
- The `median` method creates a new array by calling the `selectionSort` method on the array passed in and assigns it to a variable. An if statement determines whether the length of the array is even or odd and returns the corresponding median value
- The `mode` method again uses the `selectionSort` method to create a new sorted array from the one passed in. Two `modeNum` variables are created to keep track of current and last mode numbers. Two accumulators are created to keep track of numbers that matched the previous number in the last iteration of the loop. If it matches the, `occurrences` accumulates by one, else `occurrences` and `latestModeNum` get reset. If `occurrences` is greater than `maxOccurences,` both `maxOccurences` and `currentModeNum` get updated. Once all iterations have finished the `currentModeNum` is returned.
- The `grades` method creates a new `char[]` with a length matching that of the array accepted in the argument. A loop iterates through the marks array passed in and compares the `marks[i]` against the given `boundaries[i]` and assigns a grade and returns the array

Task One: Logic Explanation

- The `gradeDistn` method creates a new `int[]` and iterates over the `char[]` accepted in the arguments with a switch statement to determine what the current grade char and accumulates the `gradeCount` array based off the following.
  `A = 0, B = 1, C = 2, D = 3, E = 4, F = 5`
- The `selectionSort` method is a basic sorting algorithm used for processing via the `mode` and `median` methods.

Task One: Sample Output

```
Marks:
78 80 66 85 45 65 66 56 73 28 85 62 54 67 91 62 29 57 43 61 77 41 45 49 47 47 40 62 44 58
84 32 91 53 60 83 50 50 64 72 82 24 78 80 64 60 67 66 88 65 44 70 73 75 78 48 52 48 56 85
93 50 62 48 54 51 85 65 26 48 76 37 32 38 38 62 53 72 76 53 59 78 60 53 61 63 64 57 67 53
96 71 76 58 68 76 30 77 57 77 77 67 73 58 60 81 52 77 56 80 57 89 97 67 49 61 71 47 42 78
72 46 85 61 35

Minimum Mark: 24
Maximum Mark: 97
Range of Marks: 73
Mean: 62.104
Median: 62.0
Mode: 53.0

Grades:
B B C B E C C D C F B C D C A C F D F C B F E E E E F C F D
B F A D C B D D C C B F B B C C C C B C F C C B B E D E D B
A D C E D D B C F E B F F F F C D C B D D B C D C C C D C D
A C B D C B F B D B B C C D C B D B D B D B A C E C C E F B
C E B C F

Distributions:
A: 5
B: 29
C: 38
D: 24
E: 12
F: 17
```

Run Succeeded | Time 501 ms | M main

Task Two: Logic Explanation

The Point3D class contains all 4 properties `x`, `y`, `z` of the `double` type and `color` of `String` type. The class has accessor methods for the three values and two constructors, one accepting a point and the other accepting the 4 values noted above. The class also contains two `distance` methods, one accepting a point as an argument and the other accepting the 3D coordinates, both of which return the distance from the passed in values to the point created with the constructors.

- The `TestPoint3D` class is where most of the logic lies in this program. The main method is kept clean and simple only containing a new array of points and three methods to process and `display` the contents of the array in full and the `max` and `min d` distance from two farthest and closest points
- Starting with the `getPointArray,` the method accepts an int to represent the numbers of points in the array. Then assigns the first point to index 0 as specified and fills the rest of the array by iterating through and creating new point with random values received from `getRandomNumber` and `getColor`
- Both the `max` and `min` methods work in the same way in that they create two points and assign them two index 0 and 1. They both iterate through the array with a nested loop, the outside loop creates a distance value comparing the distance from the inside loops point to the outside loops point, if it matches the according condition (`max` or `min`) the `maximumDistance` / `minimumDistance` values are updated before being displayed via the `println` method.
- The `display` method takes an array of `Point3D` items, iterates through the items and prints the `color` and `x`, `y`, `z` values.
- The `getRandomNumber` is a standard random number generating function to produce a specified number of random integers which is used to fill the `pointArray` instead of hard coding the values.
- The `getColor` method returns a color from an array given a specified index. This is called within the loop to fill the array of points and uses the iteration number (minus 1 for bounds of the array)

Task Two: Sample Outputs

Sample Output One

```
Red Point: (0.0, 0.0, 0.0)
Aqua Point: (9.35982974138591, 6.316443489440845, 3.3156689011715095)
Cyan Point: (3.4648411400622883, 7.601864408495809, 8.458597652558847)
Emerald Point: (2.3886841104747427, 8.304083755818334, 1.5077907285408534)
Magenta Point: (2.4970044023521654, 6.281348347951792, 8.352969324015994)
Turquoise Point: (1.4769402428725145, 8.812226964316402, 9.588890050137465)
Orange Point: (2.2976183415901743, 6.641093804477198, 4.58586209326074)
Olive Point: (8.205011994301374, 6.268804876262461, 2.3622467012804638)
Yellow Point: (4.719595942140764, 2.978584664155134, 8.306451988726597)
Onyx Point: (2.4912389944281443, 10.4625885085808, 6.070201881617059)

Maximum distance occurs between the Red point and the Turquoise point
The distance between the points Red & Turquoise is: 13.106620805732751

Minimum distance occurs between the Aqua point and the Olive point
The distance between the points Aqua & Olive is: 1.4982948166806305
```

Run Succeeded | Time 580 ms

Sample Output Two

```
Red Point: (0.0, 0.0, 0.0)
Aqua Point: (8.656238901809758, 9.737525150172681, 9.755869668631364)
Cyan Point: (6.026960094045299, 2.3465559281053983, 1.796061295893612)
Emerald Point: (1.285181491901757, 1.0967869335518823, 9.085066701021168)
Magenta Point: (8.675474135517266, 8.888793287652511, 2.299224368929127)
Turquoise Point: (7.628879975302786, 6.640435178267876, 10.07718554328183)
Orange Point: (1.5559105918384577, 7.894597757900069, 7.066334680167891)
Olive Point: (3.5769075567637554, 2.6077265902888183, 2.6730308854939127)
Yellow Point: (3.0267480587464624, 6.806442216534591, 1.0705165099729443)
Onyx Point: (7.994586622890044, 9.929298780111209, 2.32516517446747)

Maximum distance occurs between the Red point and the Aqua point
The distance between the points Red & Aqua is: 16.27657399352737

Minimum distance occurs between the Magenta point and the Onyx point
The distance between the points Magenta & Onyx is: 1.2437573758898395
```

Run Succeeded | Time 596 ms

Task Two: UML Diagram