CS112 Final Project Program Specification

Nodes:

1) Nodes is a class used to create a Node used in the generic data structure. It represents 2 numbers and the position of the next node in the chain so that can be called upon request. The number of nodes present at a time depends on the number and length of the data structures in the code.
2) Nodes has the member variables N, C and Next, they are of type E, E, and Node respectively, both E so that 2 objects can be stored here, Node to allow for the recursion of nodes and so that one node can be called from the rest. They are all private to stop their use anywhere else other than in this class (since the variables themselves can be accessed, there is no need to have them public.
3) 2 constructors, one that takes in a number and creates a node from it and the second which takes a node and creates a duplicate node to allow for copies to be made without the alteration of the original.
4) No methods in this class

MyGenericDS:

1) This class is the generic data structure class used to store several of certain variables in such a way that they can be called upon later, similar to an array. The number of instances of this class depends on whether there is any checking going on at which point an extra data structure is created as a variable for the checker. Otherwise there should be one instance to store the moves already used.
2) The member variables are the node node and the node end, both used as methods for storing objects. They are all private since other classes can look into them without giving full access by calling methods.
3) There are two constructors, one which simply creates a node called end, the other which creates a duplicate of the input for use when you want to create a duplicate.
4) Method append lets you add a node to the structure, public to allow access to all. Method peek lets you look at the last added node, public to allow all access. Method pop lets you look at the last added node and removes it from the pile, public to allow all access. Method remove removes a specific node from the chain, public to allow all access. Method toString converts the list into a readable object public to allow all access. Method length allows you to check the length of the list, public to allow all access. Method attach allows the addition of a same type object to that objects place. Method peekSecondary allows you to look at the second object in this place.

Faces:

1) This class is used to construct a singular face of the cube, no instances of the class are created when the code is run.
2) The member variables are 4D char array FaceArray, used to store the values of the face number and each move whether it has been played or not. Integer GameOver that states whether the game is over yet or not and whether there is a win for either side or if the game is a draw. Integer TurnCount that stores the value of the number of turns which have been played. Character Move which stores the character of the move that it is, "X" or "O". MyGenericDS Used that stores the values of all moves already taken and their positions on the playing board. Two ints IChange and JChange which stores the value of the chane in the I and J value when the face is rotated.
3) No constructor used.
4) Method FaceInitiation is used to initiate the face with it's initial values. Method UpdateBoard is used to update the states of the board once a move is played and to update the state of all used moves. Method CheckGameStatus lets you know what the position of the game is win loss draw or not over. Method PrintBoard prints a cmd line version of the board.

NaughtsAndCrossesInnerCube:

1) This class is used to create a cube of Naughts and Crosses to be played.

2) Member variables FaceNumber and BoardLocation used to store which FaceNumber the turn is being played on and which location on the face the player wants to move.
3) Constructor is called once and is used to initiate the cube.
4) Method InputFaceNumber, takes an input for the number face you wish to place the piece on, checks the number is valid also (for use without the graphics). Method InputPlaceNumber, allows an input for the location on the face where the piece will be played, checks for validity also (for use without the graphics). Method CheckUnused checks to ensure the place and face number combination aren't already in use.

Rubik:

1) This is the main class of our program and a subclass of JavaFX Application. Contrary to what we intended originally, this class now also contains the NaughtsAndCrossesInnerCube class – the class, that without graphical considerations, would have been the main class as it contains the core of the game engine. If we were to have one main class, it would've needed to extend both Applications and Faces class, however, Java does not support extending multiple classes. Hence, we decided to use this structure of a class within a class. NaughtsAndCrossesInnerCube class has been explained previously, so the focus of this part of the design specification will be on the graphical elements, which are created within the start() method of the class.
Within the start() method, nodes, a scene and a stage is created. Nodes make up the 3D objects (i.e. the individual cubes that make up the large rubik's cube), as well as the user interface controls (i.e. buttons to rotate the rubik's cube left-right or up-down) and the user events (i.e. selecting a rubik's cube to place a move). These nodes then form a scene; a scene is hence a collection of all the physical contents that appear in the program. The stage encompasses the scene and displays all the contents.
2) There are three protected static double variables, which represent the mid-x, -y, and –z coordinates relative to the scene. They must be protected and static because I will refer to them in other classes of the same package.
There is a private double variable called spinTime, which represents how long it takes to complete one single animation. It is private because I will only need it in this class. I want this to be a variable so I could easily try out different animation speeds.
3) There is no constructor in this class due to the unique nature of the JavaFX Application system.
4) In addition to NaughtsAndCrossesInnerCube class, there is only one method: start(). This is a built-in public method of JavaFX, that I then overwrite. In this method, 1) I call to createCubes to create the 3D cubes and their respective rotation objects; 2) create the graphical elements pertaining to the screen (i.e. the title of the game); 3) actions corresponding to user action (i.e. clicking on one cube places his/her move there.)

createCubes:

1) This class creates the graphics for the 3D Rubik's cube. One instance of it exists.
2) There is a protected static ArrayList of type Box called **cubes** that stores each cube of the 3D Rubik cube. It has to be protected because I would reference it in other classes. It is static because I would like to only move/modify the same set of parts every time.
There are two protected static ArrayList of type Rotate that stores the rotation object for each cube in the left-right and up-down direction respectively. They are called **cubelrt** and **cubeurt**. It has to be protected because I would refer to them in other classes. They are static because I would like to only move/modify the same set of parts every time.
There are three static private double lists that contain the center x, y and z coordinates of each cube. They are all set to private because they are only needed within this class.
3) There is one constructor. It is where the cubes are created and stored to cubes.
4) There are four methods.
The first method, called lrt, creates the left-right rotation objects for each cube. The second method, called urt, creates the up-down rotation objects for each cube.
The third and fourth method, called Cross and Circle respectively, creates the method that places a cross or a circle on a cube, when a player places his move.
All four methods are static and protected because I would refer to them in other classes and would not like their values to change.

createPopup:

1) This class creates a popup window that congratulates a player when he wins (or declares a game to be a draw, when a draw is reached). It mainly uses methods and objects from JavaFx.
2) Within the method display(), there is one Stage object, one Scene object, and one VBox object (which defines the layout of the window). These are necessary structures to create a window using JavaFX. There are two nodes: one of type Label which displays the message to the user, and one of type Button which is the close button a user can press on to close the popup window. They have the same visibility (protected) as the method they are in.
3) There is no constructor due to the unique nature of JavaFX.
4) There is only one method called display(). Once called, it displays the popup window.

rubik is the main class, NaughtsAndCrossesInnerCube is created inside of this class (since rubik cannot be a subclass of NaughtsAndCrossesInnerCube because it already extends Application.) NaughtsAndCrossesInnerCube is a sub class of faces however. createCube is an independent class which is called when the cubes are to be called. MyGenericDS is also a standalone class used to store the capability of the user defined data structure and this implements the interface GenericOrderedCollection.