Cuestionario Ajedrez.

1.- ¿Cuál fue el proceso de programación del ajedrez?

La clase más primitiva de todo el programa es la clase Posición, por lo que fue la primera que se diseñó. Ésta incluía como atributos dos enteros x, y para representar en coordenadas cartesianas la posición de una pieza en el tablero. A partir de la clase Posición se pudo definir a la clase abstracta Pieza, sus atributos incluían una posición, un color y un tipo. El color y el tipo se definieron a través de dos Enum distintos; uno que definía color, y otro que definía tipo de pieza. El constructor de esta clase abstracta recibe y asigna los valores para el color de la pieza y su posición.

Posteriormente se definió la clase Board, que es una matriz cuadrada de objetos de la clase Piece. La clase Board tiene la particularidad de que su constructor es un método privado. Se pretende que sólo se pueda crear un tablero por programa. Por ello, se crea el método getInstance(), que es público y estático. Este método detecta si existe un tablero, si no existe, lo crea usando el constructor privado, y si existe, lo regresa.

Posteriormente se diseñaron las clases para representar a todas las piezas de ajedrez. La clase abstracta Piece tenía como método abstracto getLegalMoves(), que en las clases hijas calcularía los movimientos que pueden realizar de acuerdo con su tipo.

Una vez hecho las clases para todas las piezas, se procede a ordenarlas en el tablero. Más concretamente en el constructor de la clase Board. Esto se hace asignando en this.matrix una nueva pieza.

Hecho esto, se creará la clase GUI, que importa los métodos de PApplet, es decir, Processing, y además el paquete chess, que contiene todo lo que se ha estado desarrollando hasta el momento. En GUI, vamos a importar las imágenes que representarán a las piezas, también dibujaremos la interfaz gráfica y en general todo lo que sea visual y tenga que ver con la interacción usuario-máquina.

Una vez implementada la GUI, empezó el proceso de depuración, donde se buscaron los errores existentes en el ajedrez. El primero que se encontró permitía seleccionar una pieza de ajedrez cuando no era su turno si ya se había seleccionado a una que si fuera su turno. El segundo error hacía que, si se cambiaba de pieza tres veces, se acabara el turno, y el ultimo error no dejaba que se refrescara la lista de movimientos de una pieza una vez calculada. Por eso mismo, se tuvo que reescribir el método donde el programa reacciona al dar clic en su totalidad. Después se hicieron algunos ajustes a las piezas, y aspi concluyó el desarrollo del programa Ajedrez.

2.- ¿Cuál es la complejidad del programa completo?(Qué tanto poder de cómputo requiere).

La complejidad seria de n^3

3.-¿Cuál es el algoritmo o la función más compleja de ejecutar?

El void draw() de processing, porque combina prácticamente todos los métodos y funciones de el programa.

4.-¿Qué conceptos aplicados en clase viste y dónde?

El más notorio de todos estos conceptos es la herencia. Todas las piezas que se colocan en el tablero heredan de una clase abstracta llamada Piece, que contiene todos los métodos comunes a las piezas del ajedrez.

Los arrays también fueron un concepto crucial para el manejo de las piezas en el tablero.

Y finalmente, los bucles como el for fueron una parte vital en poder implementar prácticamente todos los métodos y funciones del programa.

5.-¿Es un proyecto difícil?

No, es más tedioso que difícil por lo repetitivo que se vuelve en ocasiones programar las listas de movimientos.

6.- Después de haberlo hecho entre todos ¿Crees que ahora podrías implementarlo tu solo?

Si. Varios conceptos e idea me han quedado más claras gracias a ésta práctica, como por ejemplo el uso de return; en un método que regresa void.

7.- Describe como implementarías la regla Peón al paso.

Crearía un método único para el peón. Donde si un peon hace su primer movimiento a dos casillas adelante, busca a sus lados si hay otro peón de color opuesto. Si lo encuentra, entonces añade a la lista de movimientos de ese otro peón, el movimiento hacia adelante y el movimiento diagonal, después, el peón se desplaza solo en la matriz a un lugar hacia atrás. Así, el peón opuesto en el siguiente turno puede comérselo, y si no se lo come, se debe de volver a poner la posición de este peón en donde estaba antes.

8.- Describe como implementarías detectar que hay un jaque.

Usaría primera 4 ciclos for que revisen todas las diagonales con respecto al rey en búsqueda de alguna pieza. Si en la primera diagonal detectan un peón del color opuesto, se declara jaque, si no, prosigue. Si en alguna o varias de las diagonales se detecta un alfil o reina de color opuesto, se declara jaque. Se repite esto hacia los lados, pero buscando solamente una reina o alfil de color opuesto, si no, prosigue.

Después, se aplica la regla de los caballos, que es buscar en las posiciones que estén a ±2, ±1 o ±1, ±2 unidades desplazadas con respecto a la posición del rey. Si detecta un caballo de color opuesto en cualquiera de estas posiciones, declara jaque. En caso contrario, no hay jaque.

9.-Describe como implementarías el enroque.

Primero añadiría al rey y a la torre un nuevo atributo, un boolean llamado moved, que una vez se hace un movimiento se vuelve falso. Y al calcular las listas de movimientos, si no existe ninguna pieza entre el rey y la torre y además nunca se han movido las piezas, se añade a la lista de movimientos legales que ambas piezas puedan moverse a las posiciones del enroque.

Además, se modifica el método MoveTo en éstas dos clases para que, si se hace el movimiento a estos lugares especiales, se mueve la otra pieza en consecuencia. Pues el método actualmente sólo mueve una pieza.