**Análisis del ejercicio de la tortugaAnálisis Conceptual:**

El análisis conceptual del ejercicio propuesto implica comprender los elementos clave y la lógica subyacente requerida para resolver el problema. Aquí están los aspectos conceptuales fundamentales:

**Entorno de simulación**: Se describe un entorno de simulación que consiste en un "cuarto" donde una tortuga mecánica se mueve. Este cuarto está representado por un arreglo bidimensional (piso) de 20x20, donde cada celda puede contener información sobre si la tortuga ha dejado un rastro en esa posición o no.

**Tortuga mecánica**: La tortuga mecánica es el agente principal en la simulación. Puede moverse en cuatro direcciones (arriba, abajo, izquierda, derecha) en el plano del piso y puede tener su bolígrafo en dos estados (arriba o abajo). La posición de la tortuga se rastrea mediante coordenadas (x, y) en el arreglo piso.

**Bolígrafo**: El bolígrafo puede estar en dos estados: arriba o abajo. Cuando está abajo, la tortuga deja un rastro en el piso mientras se mueve; cuando está arriba, la tortuga se mueve sin dejar rastro.

**Comandos de control**: Se deben proporcionar comandos que controlen el movimiento de la tortuga y el estado del bolígrafo. Estos comandos pueden incluir avanzar, girar a la izquierda, girar a la derecha, bajar el bolígrafo y subir el bolígrafo.

**Interacción con el entorno**: La tortuga interactúa con el entorno (el piso) mediante la modificación de las celdas del arreglo piso. Cuando el bolígrafo está abajo, la tortuga marca las celdas por donde pasa; cuando está arriba, la tortuga simplemente se mueve sin modificar el piso.

**Registro y seguimiento de estado**: Es necesario llevar un registro del estado actual de la tortuga, incluyendo su posición (coordenadas x, y) y el estado del bolígrafo (arriba o abajo). Esto permite garantizar que se apliquen los comandos correctamente y se actualice adecuadamente el estado del piso.

**Gestión de errores**: Se deben considerar posibles errores durante la ejecución, como movimientos fuera de los límites del piso o comandos incorrectos. La implementación debe manejar estos casos de manera adecuada para evitar resultados inesperados o errores en la simulación.

**Análisis desde el área matemática:**

Aplicaciones prácticas: Aunque el ejercicio en sí no tiene aplicaciones directas en la teoría de números, la capacidad de representar gráficamente conceptos matemáticos es fundamental en la educación y comprensión de las matemáticas. Además, el ejercicio puede ser útil para enseñar conceptos de programación y lógica.

Clases y operaciones relacionadas: En el contexto de la programación de computadoras, este ejercicio puede implementarse utilizando clases para representar la tortuga y el "piso" virtual. Las operaciones relacionadas podrían incluir métodos para controlar el movimiento de la tortuga y dibujar figuras en el "piso".

**Análisis desde el área de la lógica:**

Desde el área lógica, el ejercicio presenta un conjunto de reglas y operaciones que gobiernan el comportamiento de una tortuga mecánica en un entorno simulado. Aquí hay un análisis desde esta perspectiva:

**Reglas de movimiento**: La tortuga puede moverse en cuatro direcciones: arriba, abajo, izquierda y derecha. Cada movimiento implica un cambio en las coordenadas x e y de la posición de la tortuga en el piso. Estos movimientos están definidos de manera lógica y deben aplicarse de acuerdo con la dirección actual de la tortuga.

**Estado del bolígrafo**: El bolígrafo puede estar en dos estados: arriba o abajo. Este estado determina si la tortuga dejará un rastro mientras se mueve. La lógica detrás de esta funcionalidad dicta que cuando el bolígrafo está abajo, la tortuga debe dejar un rastro marcando las posiciones en el piso por donde pasa.

**Interacción con el piso**: El piso se representa mediante un arreglo bidimensional. La interacción lógica entre la tortuga y el piso implica marcar ciertas posiciones del arreglo cuando el bolígrafo está abajo y la tortuga se mueve. Esto implica la modificación de los valores en el arreglo para indicar qué áreas han sido "dibujadas" por la tortuga.

**Secuencia de comandos**: Los comandos proporcionados (avanzar, girar a la izquierda, girar a la derecha, bajar el bolígrafo y subir el bolígrafo) deben ser ejecutados en secuencia. La lógica de ejecución de estos comandos implica realizar las operaciones adecuadas en función del estado actual de la tortuga y del bolígrafo.

**Validación de movimientos**: Debe aplicarse lógica para validar los movimientos de la tortuga, asegurándose de que no salga del área definida por el piso. Esto implica verificar si los movimientos están dentro de los límites del arreglo y manejar de manera adecuada cualquier intento de movimiento fuera de estos límites.

**Actualización del estado**: Después de cada comando, es necesario actualizar el estado actual de la tortuga y del piso. Esto implica registrar la nueva posición de la tortuga y, si el bolígrafo está abajo, marcar las celdas correspondientes en el arreglo para reflejar el dibujo de la tortuga.

**Gestión de errores**: La lógica debe incluir la detección y el manejo de posibles errores, como comandos incorrectos o movimientos fuera de los límites del piso. Esto garantiza que la simulación se ejecute de manera coherente y produce resultados esperados.

**Análisis desde el área de la programación:**

Desde el área de la programación, el ejercicio propuesto implica la implementación de un sistema que simula el movimiento de una tortuga mecánica en un entorno bidimensional. Aquí está el análisis desde esta perspectiva:

Estructuras de datos: Es necesario definir estructuras de datos para representar el entorno de simulación y el estado de la tortuga. Esto podría incluir un arreglo bidimensional para representar el piso y variables para almacenar la posición de la tortuga, la dirección en la que mira y el estado del bolígrafo.

Control de flujo: Se deben implementar estructuras de control de flujo para interpretar y ejecutar los comandos proporcionados en el arreglo. Esto podría hacerse mediante un bucle que itere sobre cada comando y ejecute la acción correspondiente.

Operaciones de movimiento: Se deben implementar funciones para manejar el movimiento de la tortuga en todas las direcciones posibles, así como funciones para girar a la derecha o izquierda. Estas funciones actualizarán la posición de la tortuga según las reglas definidas.

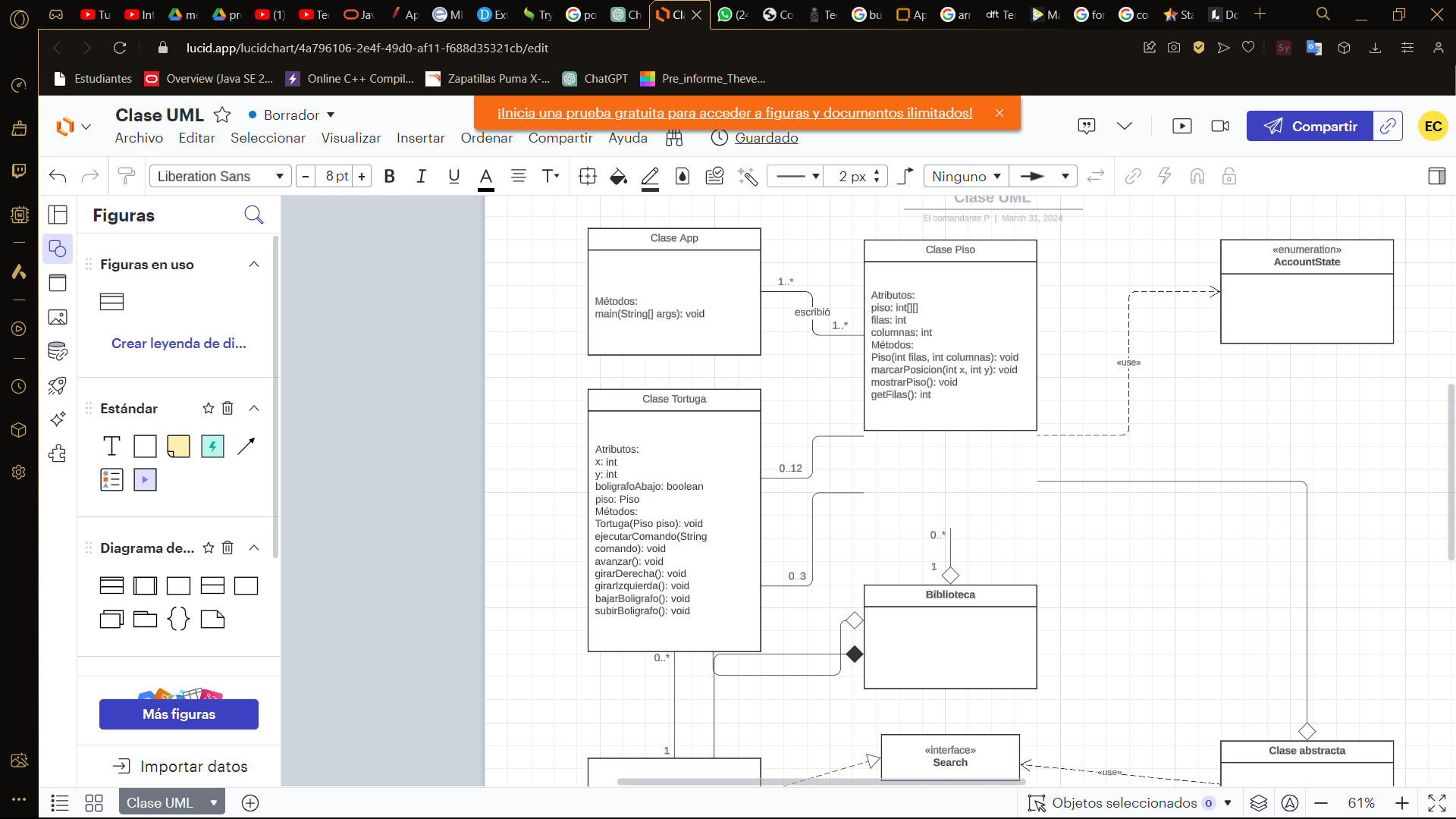
Dibujar en el piso: Cuando el bolígrafo está abajo, se debe modificar el arreglo del piso para marcar las posiciones que la tortuga ha recorrido. Esto implica cambiar los valores en el arreglo en función de la posición actual de la tortuga y la dirección en la que se mueve.

Manejo de estado: Se deben implementar funciones para controlar el estado de la tortuga, incluyendo si el bolígrafo está arriba o abajo, y para actualizar este estado según los comandos proporcionados.

Validación de movimientos: Se deben agregar comprobaciones para garantizar que la tortuga no salga del límite del piso durante sus movimientos. Esto implica verificar si la nueva posición está dentro de los límites del arreglo antes de actualizarla.

Gestión de errores: Se deben implementar mecanismos para manejar errores como comandos inválidos o movimientos fuera de los límites del piso. Esto podría incluir la generación de mensajes de error y la interrupción de la ejecución de comandos incorrectos.

Salida del resultado: Finalmente, se debe proporcionar una forma de visualizar el resultado de la simulación, que podría ser mostrando el arreglo del piso con las marcas dejadas por la tortuga.

**Diseño:**

Diagrama

Descripción generada automáticamente con confianza baja

Diagrama

Descripción generada automáticamente con confianza baja

**Código del programa:**

**Clase App**

|  |
| --- |
| import java.util.ArrayList;  public class App {      public static void main(String[] args) {          // Se inicializa el piso          Piso piso = new Piso(20, 20);          // Se crean los comandos para la tortuga          ArrayList<String> comandos = new ArrayList<>();          comandos.add("ABAJO");          comandos.add("AVANZA");          comandos.add("AVANZA");          comandos.add("DERECHA");          comandos.add("AVANZA");          comandos.add("ARRIBA");          comandos.add("AVANZA");          comandos.add("ABAJO");          comandos.add("AVANZA");          // Se crea una instancia de la tortuga con el piso          Tortuga tortuga = new Tortuga(piso);          // Se ejecutan los comandos de la tortuga          for (String comando : comandos) {              tortuga.ejecutarComando(comando);          }          // Se imprime el piso con las figuras trazadas por la tortuga          piso.mostrarPiso();      }  } |

**Clase Piso**

|  |
| --- |
| class Piso {      private int[][] piso;      private int filas;      private int columnas;      public Piso(int filas, int columnas) {          this.filas = filas;          this.columnas = columnas;          piso = new int[filas][columnas];      }      public void marcarPosicion(int x, int y) {          if (x >= 0 && x < filas && y >= 0 && y < columnas) {              piso[x][y] = 1;          }      }      public void mostrarPiso() {          for (int[] fila : piso) {              for (int valor : fila) {                  System.out.print(valor == 1 ? "\*" : " ");              }              System.out.println();          }      }      public int getFilas() {          return filas;      }  } |

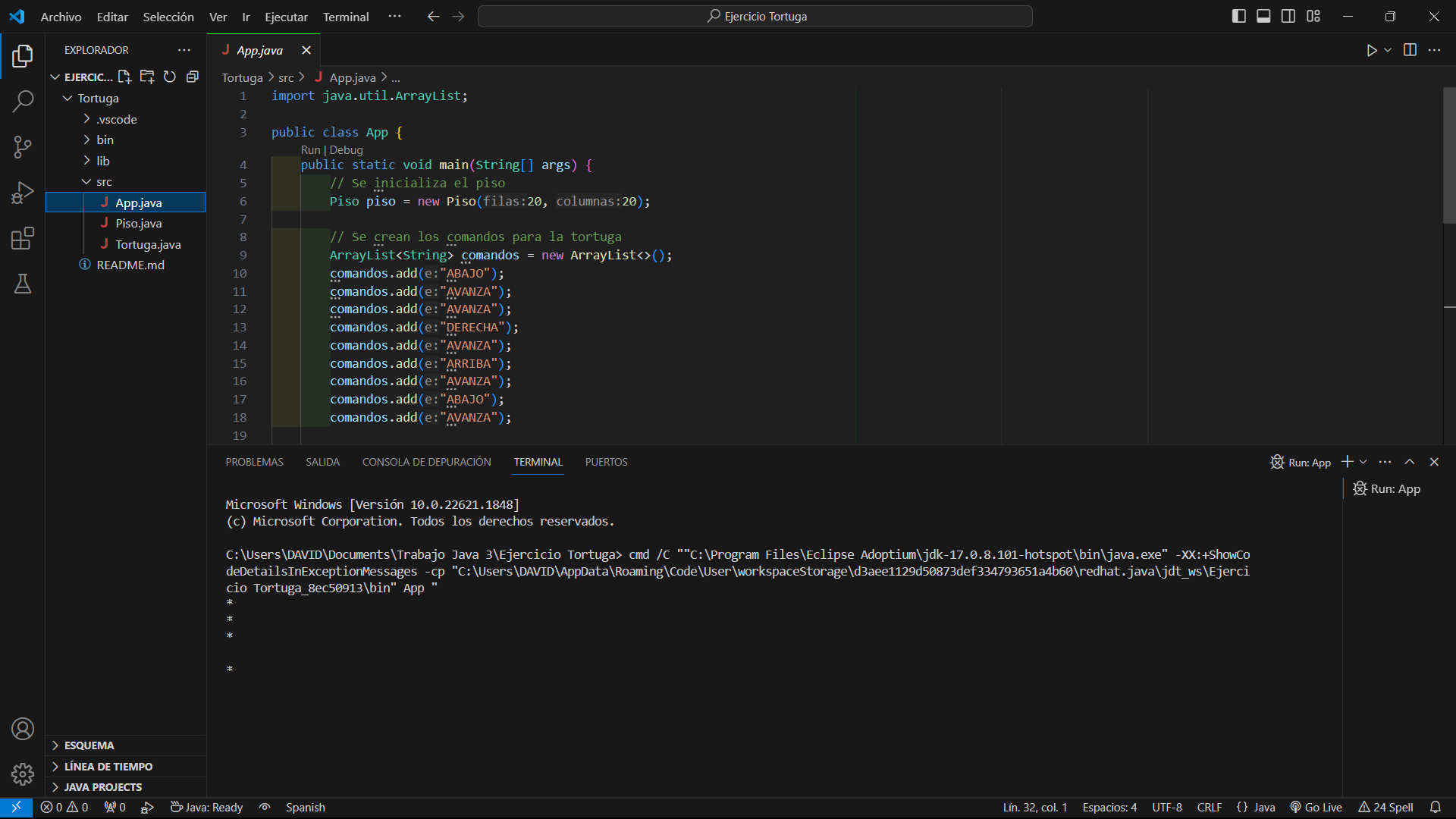
**Clase Tortuga**

|  |
| --- |
| class Tortuga {      private int x;      private int y;      private boolean boligrafoAbajo;      private Piso piso;      public Tortuga(Piso piso) {          this.x = 0;          this.y = 0;          this.boligrafoAbajo = false;          this.piso = piso;      }      public void ejecutarComando(String comando) {          switch (comando) {              case "AVANZA":                  avanzar();                  break;              case "DERECHA":                  girarDerecha();                  break;              case "IZQUIERDA":                  girarIzquierda();                  break;              case "ABAJO":                  bajarBoligrafo();                  break;              case "ARRIBA":                  subirBoligrafo();                  break;          }      }      private void avanzar() {          if (boligrafoAbajo) {              piso.marcarPosicion(x, y);          }          // Lógica para avanzar (actualiza las coordenadas x, y)          x++;          if (x >= piso.getFilas()) {              x = piso.getFilas() - 1;          }      }      private void girarDerecha() {          // Lógica para girar a la derecha      }      private void girarIzquierda() {          // Lógica para girar a la izquierda      }      private void bajarBoligrafo() {          boligrafoAbajo = true;      }      private void subirBoligrafo() {          boligrafoAbajo = false;      }  } |

**.TXT**

|  |
| --- |
| public static void guardarEstadoTortuga(Tortuga tortuga, String nombreArchivo) {          try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File(nombreArchivo)))) {              // Escribir información sobre la posición de la tortuga              int[] posicion = tortuga.getPosicion();              writer.write("Posición de la tortuga:\n");              writer.write("X: " + posicion[0] + "\n");              writer.write("Y: " + posicion[1] + "\n");              // Escribir información sobre el piso de dibujo              int[][] piso = tortuga.getPiso();              writer.write("\nPiso de dibujo:\n");              for (int i = 0; i < piso.length; i++) {                  for (int j = 0; j < piso[i].length; j++) {                      writer.write(piso[i][j] + " ");                  }                  writer.write("\n");              }          } catch (IOException e) {              e.printStackTrace();          }      }  } |

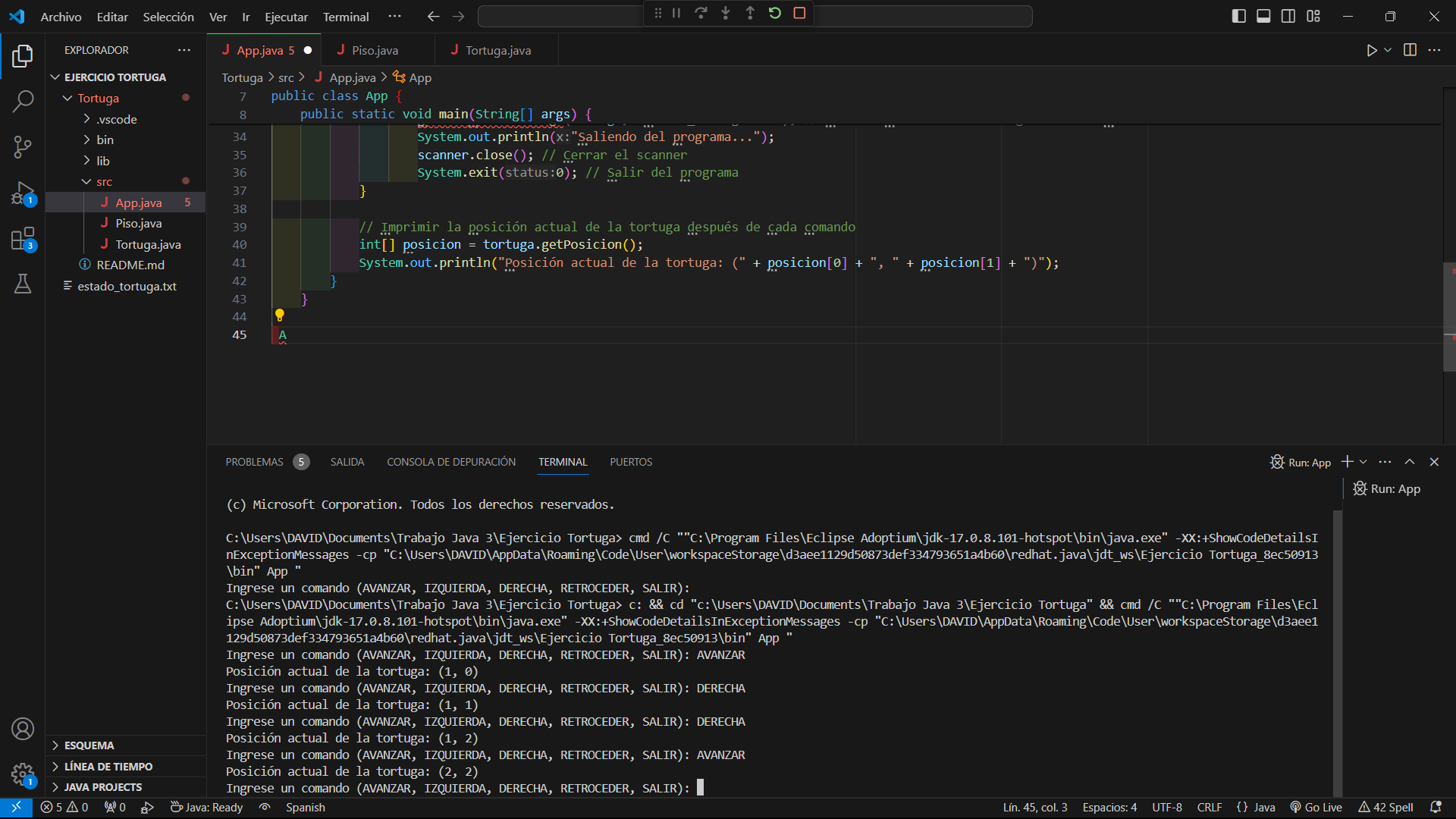
**Testeo App normal:**



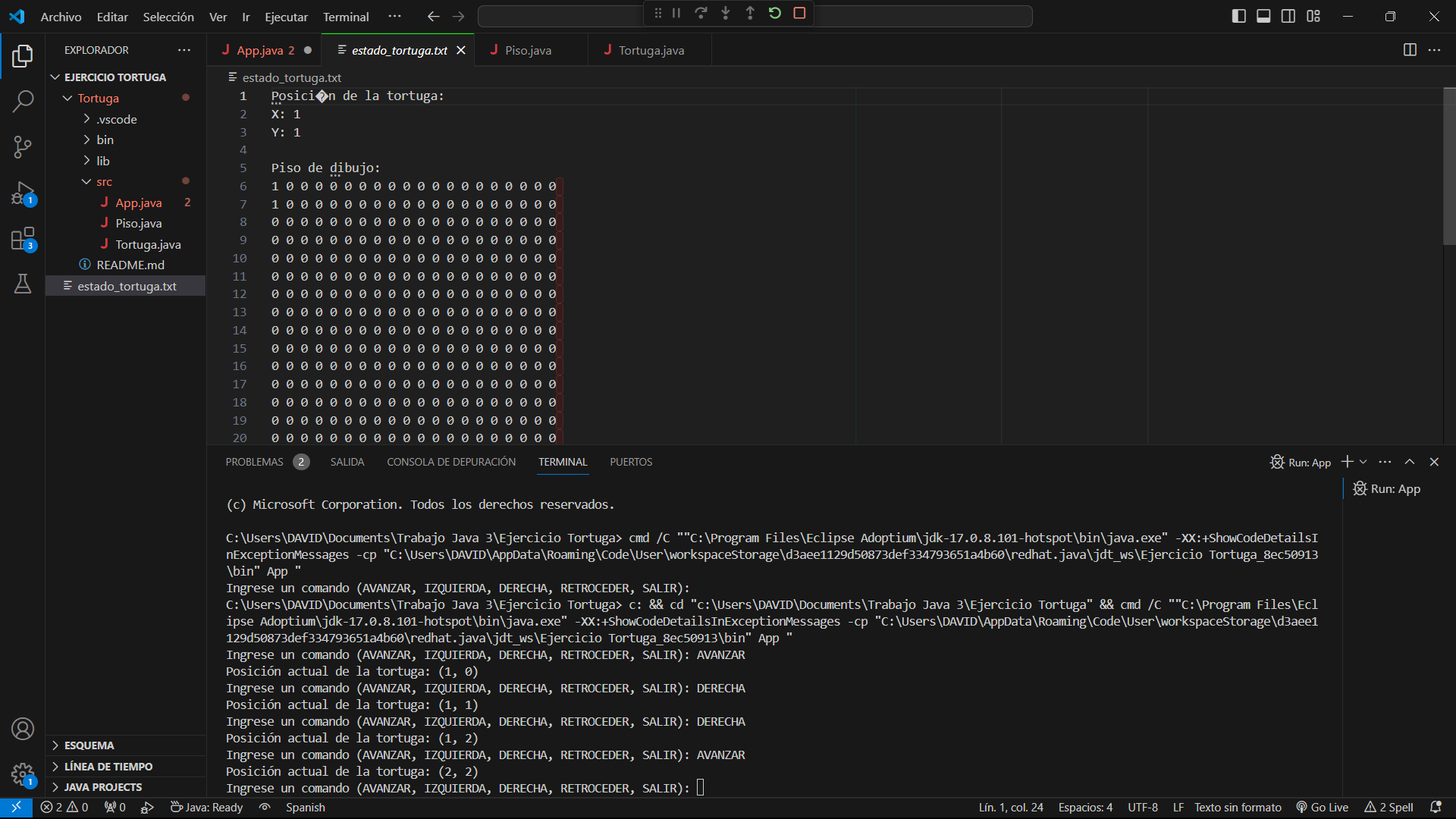
La prueba del programa muestra los pasos de la tortuga mediante las instrucciones fijas que ya se le dan para poder próximamente dar las instrucciones desde teclado y así la tortuga siga las instrucciones dadas por el usuario final.

Se muestran los pasos de la tortuga y donde está con asterisco es por donde pasó la tortuga y los espacios en blanco es donde la tortuga no paso.

**Testeo App .Txt:**



En el testeo el programa pide la dirección en la coordenada X y Y en la cuál se va a mover la tortuga y el mismo programa da las coordenadas del movimiento de la tortuga en 2 dimensiones.



Al ejecutar la clase App a comparación del testeo inicial esta clase genera un archivo plano que se guarda en visual studio code como archivo.txt en el cual se ejecuta el código y muestra la posición de la tortuga en 2 dimensiones y muestra la posición de la tortuga de manera gráfica en el arreglo 20\*20.