Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería
Introducción a la programación y computación 1 Sección “A”
Segundo Semestre 2024

ENRIQUE ALEXANDER TEBALAN HERNANDEZ
Carné: 202230026

MANUAL TECNICO PROYECTO #1 2024

{BIENVENIDO }

Manual Técnico

CREADO EN: NETBEANS IDE 22
LENGUAJE: JAVA
Utilizando Programación Estructural y Orientada a Objetos (POO)

Objetivos generales

- Familiarizar al estudiante con el lenguaje Java.
- Aplicar conceptos de programación orientada a objetos recibidos en clase magistral y laboratorio.
- Elaborar la lógica para la solución del problema planteado.

Objetivos específicos

- Construcción de algoritmos para los requerimientos de la actividad.
- Ampliar el conocimiento del lenguaje JAVA.
- Ampliar el conocimiento de Programación orientada a objetos en JAVA.
- Desarrollar diagramas de clase como parte del análisis del problema.
- Implementación de ciclos, sentencias de control y arreglos.
- Construcción de aplicaciones en consola.
- Implementación de clases, herencia, encapsulamiento, polimorfismo y reutilización de código.
- Desarrollo de manual técnico y de usuario

2. Descripción del Juego

Konquest es un juego de estrategia por turnos donde dos jugadores compiten para conquistar planetas en un mapa. Los jugadores envían flotas de naves llenas de guerreros para conquistar planetas neutrales y planetas enemigos. El juego se basa en la creación de mapas personalizados por los usuarios, y el jugador que conquiste todos los planetas del mapa será el ganador.

Características del Juego:

- **Mapas personalizados**: El usuario puede diseñar mapas con atributos definidos como número de filas, columnas y planetas neutrales.
- **Mecánicas de combate**: Los guerreros de diferentes planetas luchan usando factores de muerte y habilidades especiales.
- **Gestión de recursos**: Los jugadores deben gestionar recursos como dinero y constructores para producir naves y mejorar sus flotas.
- **Simulación de batallas**: Los jugadores pueden simular el resultado de enviar una flota antes de realizar el ataque real.

---

3. Componentes Principales

3.1 Clases del Juego

Clase Juego

La clase principal que controla el flujo del juego, gestiona los turnos y coordina las acciones de los jugadores.

- Atributos:
  - mapa: Contiene el mapa actual de juego.

    - jugadores: Arreglo que contiene los jugadores.
    - turnoActual: Indica el turno actual del juego.
- Métodos:
    - iniciarJuego(): Método principal que inicia el juego.
    - registrarAcciones(): Registra las acciones que realizan los jugadores en cada turno.
    - ejecutarAcciones(): Ejecuta las acciones después de que ambos jugadores han registrado sus movimientos.

Clase Mapa

Representa el tablero de juego donde se distribuyen los planetas.

- Atributos:
    - filas, columnas: Dimensiones del mapa.
    - planetas: Arreglo bidimensional que representa los planetas distribuidos en el mapa.
- Métodos:
    - diseñarMapa(): Permite al jugador diseñar un mapa personalizado.
    - colocarPlanetas(): Coloca los planetas en el mapa, ya sean iniciales o neutrales.

Clase Planeta

Cada instancia de planeta tiene sus atributos específicos y está asignada a un jugador o es neutral.

- Atributos:
    - nombre: Nombre del planeta.
    - dueño: Jugador que controla el planeta.
    - tipo: El tipo de planeta (Tierra, Agua, Fuego, Biotara, Radioactivo).
    - guerreros: La cantidad de guerreros presentes en el planeta.
    - dinero: Cantidad de dinero en el planeta.
- Métodos:
    - producirRecursos(): Produce dinero y guerreros al finalizar cada turno.
    - enviarFlota(): Permite enviar una flota de guerreros a otro planeta.
    - construirNave(): Crea una nueva nave en el planeta si hay constructores disponibles.

Clase Jugador

Representa a los dos jugadores del juego.

- Atributos:
    - nombre: Nombre del jugador.
    - planetas: Arreglo que contiene los planetas conquistados por el jugador.
- Métodos:
    - realizarAccion(): Método que permite al jugador realizar acciones como ver planetas, enviar flotas, o construir naves.

Clase Nave

Modela las naves que transportan guerreros entre planetas.

- Atributos:
    - tipo: Tipo de nave (Helios, Galaxia Prime, etc.).
    - capacidad: Capacidad máxima de guerreros que puede transportar.
    - tasaSupervivencia: Probabilidad de que la nave y sus guerreros lleguen a destino.
- Métodos:

- calcularTasaSupervivencia(): Calcula la tasa de supervivencia de la nave basada en la distancia y los recursos asignados al viaje.

Clase Guerrero

Representa a los guerreros que luchan en los planetas y son transportados por las naves.

- Atributos:
  - tipo: Tipo de guerrero (Terradiente, Aquaris, Ignis, etc.).
  - factorMuerte: Valor que indica la capacidad de un guerrero para vencer a otro en batalla.

5. Lógica del Juego

5.1 Fases del Juego

1. **Fase de Diseño del Mapa**: Los jugadores diseñan o seleccionan un mapa.
2. **Fase de Juego**: Los jugadores toman turnos para realizar acciones como enviar flotas, medir distancias y construir naves.
3. **Fase de Batalla**: Si una flota llega a un planeta enemigo o neutral, se inicia una batalla entre los guerreros de ambos bandos.
4. **Victoria**: El juego finaliza cuando uno de los jugadores ha conquistado todos los planetas.

5.2 Turnos

Cada jugador puede realizar múltiples acciones durante su turno. Las acciones incluyen:

- Ver información de planetas.
- Medir distancias entre planetas.
- Enviar flotas.
- Construir naves.

---

6. Requisitos Técnicos

6.1 Lenguaje y Herramientas Utilizadas

- **Lenguaje de Programación**: Java
- **Entorno de Desarrollo**: Cualquier IDE de Java (IntelliJ, Eclipse, NetBeans)
- **Modo de Ejecución**: Consola
- **Estructuras de Datos**: Arreglos para gestionar colecciones.

6.2 Consideraciones Importantes

- El código debe aplicar conceptos de **Programación Orientada a Objetos**: herencia, polimorfismo, encapsulamiento y abstracción.
- El uso de **ArrayList**, **LinkedList**, u otras estructuras de Java no está permitido. Solo se utilizan **arreglos**.

PSEUDOCODIGOS

INICIO PROGRAMA

```
 CREAR objeto Colores 'color'
CREAR objeto Herramientas 'h'
CREAR objeto Scanner 'sc'

LLAMAR a h.dobleLineaSeparadora()
LLAMAR a h.mensajeBienvenida()
LLAMAR a h.menuOpciones()

IMPRIMIR "Seleccione una opción: "
LEER 'opcion' DESDE teclado

SEGÚN (opcion)
   CASO 1:
      IMPRIMIR "Ingrese el número de filas para el mapa: "
      LEER 'filas' DESDE teclado

      IMPRIMIR "Ingrese el número de columnas para el mapa: "
      LEER 'columnas' DESDE teclado

      CREAR objeto AdminMapa 'mapa' CON filas y columnas

      LIMPIAR buffer de entrada

      IMPRIMIR "Ingrese el nombre del Jugador 1: "
      LEER 'nombre1' DESDE teclado

      IMPRIMIR "Ingrese el nombre del Jugador 2: "
      LEER 'nombre2' DESDE teclado

      'posicion1' ← LLAMAR seleccionarPosicion(sc, nombre1, filas, columnas)
      'planeta1' ← LLAMAR seleccionarTipoDePlaneta(sc, nombre1, posicion1)

      'posicion2' ← LLAMAR seleccionarPosicion(sc, nombre2, filas, columnas)
      'planeta2' ← LLAMAR seleccionarTipoDePlaneta(sc, nombre2, posicion2)

      CREAR objeto AdminJugador 'jugador1' CON nombre1 y planeta1
      CREAR objeto AdminJugador 'jugador2' CON nombre2 y planeta2

      LLAMAR mapa.colocarPlaneta(planeta1)
      LLAMAR mapa.colocarPlaneta(planeta2)

      IMPRIMIR "Ingrese el número de planetas neutrales a crear: "
      LEER 'numNeutrales' DESDE teclado

      LLAMAR generarPlanetasNeutrales(mapa, numNeutrales)
```

```
        CREAR objeto MotorDeJuego 'juego' CON jugador1, jugador2 y mapa
        LLAMAR h.separadorLinea()
        LLAMAR juego.primerJuego()

    CASO 2:
        IMPRIMIR "Aletioraridad SIN FUNCIONAR :("

    CASO 3:
        LLAMAR h.mensajeDespedida()

    DEFAULT:
        IMPRIMIR "INGRESE OPCIÓN VÁLIDA"

FIN PROGRAMA

---

FUNCIÓN seleccionarPosicion(sc, jugador, filas, columnas):
    IMPRIMIR jugador + ", ¿deseas ubicar tu planeta manualmente? (S/N): "
    LEER 'respuesta' DESDE teclado

    SI (respuesta ES 'S' o 's') ENTONCES
        IMPRIMIR "Ingrese la posición de su planeta (formato: x,y): "
        LEER 'coords' DESDE teclado
        'x' ← PARSEAR coords[0] A entero
        'y' ← PARSEAR coords[1] A entero
        RETORNAR nueva Posicion(x, y)
    SINO
        'x' ← GENERAR número aleatorio entre 0 y filas-1
        'y' ← GENERAR número aleatorio entre 0 y columnas-1
        IMPRIMIR "Posición aleatoria asignada: (" + x + "," + y + ")"
        RETORNAR nueva Posicion(x, y)

FIN FUNCIÓN

---

FUNCIÓN seleccionarTipoDePlaneta(sc, dueño, posicion):
    IMPRIMIR "Seleccione el tipo de planeta para " + dueño
    IMPRIMIR "1. Planeta Tierra"
    IMPRIMIR "2. Planeta Agua"
    IMPRIMIR "3. Planeta Fuego"
    IMPRIMIR "4. Planeta Biotara"
    IMPRIMIR "5. Planeta Radioactivo"
    LEER 'tipo' DESDE teclado

    SEGÚN (tipo)
        CASO 1: RETORNAR nuevo TierraPlaneta("Tierra", posicion, dueño)
        CASO 2: RETORNAR nuevo AguaPlaneta("Agua", posicion, dueño)
```

```
        CASO 3: RETORNAR nuevo FuegoPlaneta("Fuego", posicion, dueño)
        CASO 4: RETORNAR nuevo BiotaraPlaneta("Biotara", posicion, dueño)
        CASO 5: RETORNAR nuevo RadioactivoPlaneta("Radioactivo", posicion, dueño)
        DEFAULT:
            IMPRIMIR "Opción no válida. Seleccionando Planeta Tierra por defecto."
            RETORNAR nuevo TierraPlaneta("Tierra", posicion, dueño)
FIN FUNCIÓN

---

FUNCIÓN generarPlanetasNeutrales(mapa, numNeutrales):
    PARA i DESDE 0 HASTA numNeutrales-1 HACER
        HACER
            'fila' ← GENERAR número aleatorio entre 0 y mapa.getFilas()-1
            'columna' ← GENERAR número aleatorio entre 0 y mapa.getColumnas()-1
            'posicion' ← nueva Posicion(fila, columna)
        MIENTRAS (mapa.obtenerPlaneta(posicion) NO ES nulo)

        'tipo' ← GENERAR número aleatorio entre 1 y 5
        SEGÚN (tipo)
            CASO 1: 'planetaNeutral' ← nuevo TierraPlaneta("Neutral Tierra" + (i+1), posicion, "Neutral")
            CASO 2: 'planetaNeutral' ← nuevo AguaPlaneta("Neutral Agua" + (i+1), posicion, "Neutral")
            CASO 3: 'planetaNeutral' ← nuevo FuegoPlaneta("Neutral Fuego" + (i+1), posicion, "Neutral")
            CASO 4: 'planetaNeutral' ← nuevo BiotaraPlaneta("Neutral Biotara" + (i+1), posicion, "Neutral")
            CASO 5: 'planetaNeutral' ← nuevo RadioactivoPlaneta("Neutral Radioactivo" + (i+1), posicion, "Neutral")
            DEFAULT: 'planetaNeutral' ← nuevo TierraPlaneta("Neutral Tierra" + (i+1), posicion, "Neutral")

        LLAMAR mapa.colocarPlaneta(planetaNeutral)
    FIN PARA
FIN FUNCIÓN
```

PSEUDOCODIGO CLASE MOTORDE JUEGO

```
CLASE MotorDeJuego
    ATRIBUTOS
        AdminJugador jugador1
        AdminJugador jugador2
        AdminMapa mapa
        AdminTienda tienda
        entero turno
        AdminFlota[] flotasEnCamino
        Colores c
```

```
    Herramientas h
    Scanner sc

CONSTRUCTOR MotorDeJuego(jugador1, jugador2, mapa)
    ASIGNAR jugador1
    ASIGNAR jugador2
    ASIGNAR mapa
    ASIGNAR turno a 1
    INICIALIZAR tienda como AdminTienda
    INICIALIZAR sc como nuevo Scanner()

MÉTODO primerJuego()
    IMPRIMIR "KONQUEST - El objetivo del juego es conquistar todos los planetas."
    IMPRIMIR "¡El jugador que conquiste todos los planetas gana!"
    MAPA.mostrarMapa()

    MIENTRAS VERDADERO HACER
        h.separadorLinea()
        IMPRIMIR "Es el turno: " + turno

        SI turno % 2 != 0 ENTONCES
            jugarTurno(jugador1)
            SI verificarVictoria(jugador1) ENTONCES
                SALIR DEL BUCLE
            FIN SI
        SINO
            jugarTurno(jugador2)
            SI verificarVictoria(jugador2) ENTONCES
                SALIR DEL BUCLE
            FIN SI
        FIN SI

        actualizarFlotas()
        producirRecursos()
        mapa.mostrarMapa()
        turno++
    FIN MIENTRAS

    h.dobleLineaSeparadora()
    IMPRIMIR "¡Fin del juego!"

MÉTODO jugarTurno(jugador)
    booleano seguirTurno = VERDADERO

    MIENTRAS seguirTurno HACER
        verificarFlotas()
        h.separadorLinea()
        IMPRIMIR "Turno actual: " + jugador.getNombre()
        h.separadorLinea()
```

```
        IMPRIMIR "1. Consultar planeta"
        IMPRIMIR "2. Simular envío de flota"
        IMPRIMIR "3. Enviar flota"
        IMPRIMIR "4. Construir nave"
        IMPRIMIR "5. Ir a la tienda"
        IMPRIMIR "6. Ver recursos actuales"
        IMPRIMIR "7. Terminar turno"
        IMPRIMIR "8. Rendirse"
        IMPRIMIR "Seleccione una opción: "
        entero opcion = sc.nextInt()
        sc.nextLine()

        SEGÚN opcion HACER
          CASO 1:
            consultarPlaneta()
            ROMPER
          CASO 2:
            simularEnviarFlota(jugador)
            ROMPER
          CASO 3:
            enviarFlota(jugador)
            ROMPER
          CASO 4:
            construirNave(jugador)
            ROMPER
          CASO 5:
            tienda.mostrarMenu(jugador)
            ROMPER
          CASO 6:
            mostrarRecursosJugador(jugador)
            ROMPER
          CASO 7:
            IMPRIMIR "Turno terminado."
            seguirTurno = FALSO
            ROMPER
          CASO 8:
            IMPRIMIR jugador.getNombre() + " se ha rendido."
            h.mensajeDespedida()
            ROMPER
          DEFAULT:
            IMPRIMIR "Opción no válida."
        FIN SEGÚN
        h.separadorLinea()
    FIN MIENTRAS

MÉTODO verificarFlotas()
    h.separadorLinea()
    IMPRIMIR "Flotas en camino:"
    PARA i DESDE 0 HASTA flotasEnCamino.LONGITUD HACER
```

```
        SI flotasEnCamino[i] NO ES NULO Y flotasEnCamino[i].getTurnoLlegada() <= turno ENTONCES
            AdminPlanetas destino = flotasEnCamino[i].getDestino()
            AdminPlanetas origen = flotasEnCamino[i].getOrigen()
            AdminsGuerreros[] guerrerosAtacantes = flotasEnCamino[i].getGuerreros()

            IMPRIMIR "La flota ha llegado al planeta " + destino.getNombre()

            SI destino.getDueño() = "Neutral" O NO destino.getDueño() = flotasEnCamino[i].getJugador().getNombre()
ENTONCES
                booleano conquista = destino.batalla(guerrerosAtacantes, origen)
                SI conquista ENTONCES
                    IMPRIMIR "El planeta " + destino.getNombre() + " ha sido conquistado por " +
flotasEnCamino[i].getJugador().getNombre() + "!"
                    destino.setDueño(flotasEnCamino[i].getJugador().getNombre())
                SINO
                    IMPRIMIR "Los defensores han repelido el ataque en " + destino.getNombre() + "."
                FIN SI
            SINO
                destino.reforzarGuerreros(guerrerosAtacantes)
                IMPRIMIR "Los guerreros han sido enviados para reforzar el planeta " + destino.getNombre() + "."
            FIN SI

            flotasEnCamino[i] = NULO
        FIN SI
    FIN PARA
  FIN MÉTODO

  MÉTODO actualizarFlotas()
    PARA i DESDE 0 HASTA flotasEnCamino.LONGITUD HACER
        SI flotasEnCamino[i] NO ES NULO Y flotasEnCamino[i].getTurnoLlegada() <= turno ENTONCES
            AdminPlanetas destino = flotasEnCamino[i].getDestino()
            AdminPlanetas origen = flotasEnCamino[i].getOrigen()
            AdminsGuerreros[] guerrerosAtacantes = flotasEnCamino[i].getGuerreros()
            AdminJugador jugador = flotasEnCamino[i].getJugador()
            h.separadorLinea()
            IMPRIMIR "La flota ha llegado al planeta " + destino.getNombre()

            SI destino.getDueño() = "Neutral" O NO destino.getDueño() = jugador.getNombre() ENTONCES
                booleano conquista = destino.batalla(guerrerosAtacantes, origen)
                SI conquista ENTONCES
                    IMPRIMIR "El planeta " + destino.getNombre() + " ha sido conquistado por " + jugador.getNombre() +
"!"
                    destino.setDueño(jugador.getNombre())
                SINO
                    IMPRIMIR "Los defensores han repelido el ataque en " + destino.getNombre() + "."
                FIN SI
            SINO
                destino.reforzarGuerreros(guerrerosAtacantes)
                IMPRIMIR "Los guerreros han reforzado el planeta " + destino.getNombre() + "."
            FIN SI
```

```
            flotasEnCamino[i] = NULO
          FIN SI
        FIN PARA
    FIN MÉTODO

    MÉTODO mostrarRecursosJugador(jugador)
        h.separadorLinea()
        IMPRIMIR "---- RECURSOS DE " + jugador.getNombre() + " ----"
        IMPRIMIR "Monedas: " + jugador.getPlanetaInicial().getMonedas()
        IMPRIMIR "Planetas:"
        PARA i DESDE 0 HASTA jugador.getPlanetas().LONGITUD HACER
          SI jugador.getPlanetas()[i] NO ES NULO ENTONCES
            AdminPlanetas planeta = jugador.getPlanetas()[i]
            IMPRIMIR "- " + planeta.getNombre() + " (Coordenadas: " + planeta.getPosicion().getX() + "," +
planeta.getPosicion().getY() + ")"
          FIN SI
        FIN PARA
        h.separadorLinea()
    FIN MÉTODO

    MÉTODO producirRecursos()
        h.separadorLinea()
        IMPRIMIR "--- PRODUCCIÓN DE RECURSOS ---"
        PARA i DESDE 0 HASTA mapa.getNumPlanetas() HACER
          AdminPlanetas planeta = mapa.getPlaneta(i)
          SI planeta NO ES NULO ENTONCES
            IMPRIMIR "Produciendo recursos en el planeta " + planeta.getNombre()
            entero recursosPrevios = planeta.getMonedas()
            planeta.producirRecursos()
            IMPRIMIR "Recursos producidos: " + (planeta.getMonedas() - recursosPrevios) + " estelares."
          FIN SI
        FIN PARA
    FIN MÉTODO

    MÉTODO consultarPlaneta()
        h.separadorLinea()
        IMPRIMIR "---- CONSULTA DE PLANETA ----"
        IMPRIMIR "Ingrese las coordenadas del planeta (formato: x,y): "
        String[] coordenadas = sc.nextLine().split(",")
        entero x = convertirAEntero(coordenadas[0])
        entero y = convertirAEntero(coordenadas[1])

        AdminPlanetas planeta = mapa.obtenerPlaneta(new Posicion(x, y))

        SI planeta NO ES NULO ENTONCES
          planeta.mostrarInfo()
        SINO
          IMPRIMIR "El planeta no existe en esa posición."
```

```
    FIN SI
    h.separadorLinea()
FIN MÉTODO

MÉTODO construirNave(jugador)
    h.separadorLinea()
    AdminPlanetas planeta
    MIENTRAS VERDADERO HACER
        IMPRIMIR "Ingrese las coordenadas del planeta donde desea construir la nave (formato: x,y): "
        String[] coordenadas = sc.nextLine().split(",")
        entero x = convertirAEntero(coordenadas[0])
        entero y = convertirAEntero(coordenadas[1])

        planeta = mapa.obtenerPlaneta(new Posicion(x, y))
        SI planeta NO ES NULO Y planeta.getDueño() = jugador.getNombre() ENTONCES
            romper
        SINO
            IMPRIMIR "El planeta no existe o no es suyo. Intente nuevamente."
        FIN SI
    FIN MIENTRAS

    IMPRIMIR "¿Qué tipo de nave desea construir? (opciones: tipo1, tipo2, tipo3)"
    String tipoNave = sc.nextLine()
    jugador.construirNave(tipoNave, planeta)
FIN MÉTODO

MÉTODO simularEnviarFlota(jugador)
    h.separadorLinea()
    AdminPlanetas origen
    AdminPlanetas destino

    MIENTRAS VERDADERO HACER
        IMPRIMIR "Ingrese las coordenadas del planeta de origen (formato: x,y): "
        String[] coordenadasOrigen = sc.nextLine().split(",")
        entero xOrigen = convertirAEntero(coordenadasOrigen[0])
        entero yOrigen = convertirAEntero(coordenadasOrigen[1])

        origen = mapa.obtenerPlaneta(new Posicion(xOrigen, yOrigen))
        SI origen NO ES NULO Y origen.getDueño() = jugador.getNombre() ENTONCES
            romper
        SINO
            IMPRIMIR "El planeta de origen no existe o no es suyo. Intente nuevamente."
        FIN SI
    FIN MIENTRAS

    MIENTRAS VERDADERO HACER
        IMPRIMIR "Ingrese las coordenadas del planeta de destino (formato: x,y): "
        String[] coordenadasDestino = sc.nextLine().split(",")
        entero xDestino = convertirAEntero(coordenadasDestino[0])
```

```
        entero yDestino = convertirAEntero(coordenadasDestino[1])

        destino = mapa.obtenerPlaneta(new Posicion(xDestino, yDestino))
        SI destino NO ES NULO ENTONCES
          romper
        SINO
          IMPRIMIR "El planeta de destino no existe. Intente nuevamente."
        FIN SI
      FIN MIENTRAS

      h.separadorLinea()
      IMPRIMIR "¿Cuántos guerreros desea enviar?"
      entero cantidadGuerreros = sc.nextInt()
      sc.nextLine()
      AdminsGuerreros[] guerreros = jugador.enviarGuerreros(origen, destino, cantidadGuerreros)
      flotasEnCamino.add(new AdminFlota(origen, destino, guerreros, turno + 2, jugador))
      IMPRIMIR "Flota enviada desde " + origen.getNombre() + " hacia " + destino.getNombre() + "."
    FIN MÉTODO

    MÉTODO enviarFlota(jugador)
      // Lógica para enviar flotas de forma efectiva
      // Similar a simularEnviarFlota, pero efectivamente se envían los guerreros
    FIN MÉTODO

    MÉTODO verificarVictoria(jugador)
      // Lógica para verificar si un jugador ha ganado
    FIN MÉTODO

FIN CLASE
```

```
PSEUDOCODIGO CLASE ADMINMAPA

CLASE AdminMapa
  ATRIBUTOS
    Colores c
    Herramientas h
    AdminPlanetas[] planetas    // Arreglo de planetas en el mapa
    entero filas
    entero columnas
    entero numPlanetas          // Contador de planetas
```

```
    CONSTRUCTOR AdminMapa(filas, columnas)
        ASIGNAR this.filas a filas
        ASIGNAR this.columnas a columnas
        INICIALIZAR this.planetas como nuevo AdminPlanetas[60]   // Limitar a 60 planetas en el mapa
        ASIGNAR this.numPlanetas a 0

    // Método para colocar un planeta en el mapa
    MÉTODO colocarPlaneta(planeta)
        SI numPlanetas >= planetas.LONGITUD ENTONCES
            IMPRIMIR c.rojo("No se pueden colocar más planetas.")
            RETORNAR
        FIN SI
        Posicion pos = planeta.getPosicion()
        SI obtenerPlaneta(pos) es NULO ENTONCES
            planetas[numPlanetas] = planeta   // Añadir el planeta y aumentar el contador
            numPlanetas++
            h.separadorLinea()
            IMPRIMIR c.cian("Planeta " + planeta.getNombre() + " colocado en (" + pos.getX() + ", " + pos.getY() + ")")
        SINO
            IMPRIMIR c.amarillo("La posición (" + pos.getX() + ", " + pos.getY() + ") ya está ocupada.")
        FIN SI
    FIN MÉTODO

    // Obtener un planeta por su posición
    MÉTODO obtenerPlaneta(posicion)
        PARA i DESDE 0 HASTA numPlanetas HACER
            SI  planetas[i].getPosicion().getX()  =  posicion.getX()  Y  planetas[i].getPosicion().getY()  =  posicion.getY()
ENTONCES
                RETORNAR planetas[i]
            FIN SI
        FIN PARA
        RETORNAR NULO
    FIN MÉTODO

    // Método para obtener el número total de planetas en el mapa
    MÉTODO getNumPlanetas()
        RETORNAR numPlanetas
    FIN MÉTODO

    // Método para obtener un planeta por índice
    MÉTODO getPlaneta(index)
        SI index >= 0 Y index < numPlanetas ENTONCES
            RETORNAR planetas[index]
        FIN SI
        RETORNAR NULO
    FIN MÉTODO

    // Método para calcular la distancia entre dos planetas
```

```
MÉTODO calcularDistancia(p1, p2, nave)
    SI p1 es NULO O p2 es NULO ENTONCES
        IMPRIMIR c.amarillo("Uno de los planetas es nulo.")
        RETORNAR 0
    FIN SI
    Posicion pos1 = p1.getPosicion()
    Posicion pos2 = p2.getPosicion()

    // Calculamos la distancia en el plano 2D
    entero distancia = (int) sqrt((pos2.getX() - pos1.getX())^2 + (pos2.getY() - pos1.getY())^2)

    // Calculamos los turnos en base a la velocidad de la nave
    RETORNAR (int) ceil(distancia / nave.getVelocidad())
FIN MÉTODO

// Mostrar el mapa visualmente con más detalles
MÉTODO mostrarMapa()
    String[][] tablero = nuevo String[filas][columnas]

    // Inicializar el tablero vacío
    PARA i DESDE 0 HASTA filas HACER
        PARA j DESDE 0 HASTA columnas HACER
            tablero[i][j] = "|   |"  // Celda vacía
        FIN PARA
    FIN PARA

    // Colocar los planetas en el tablero
    PARA i DESDE 0 HASTA numPlanetas HACER
        AdminPlanetas planeta = planetas[i]
        entero x = planeta.getPosicion().getX()
        entero y = planeta.getPosicion().getY()

        String dueño = planeta.getDueño().esIgual("Neutral") ? "N" : planeta.getDueño().substring(0, 1)
        tablero[x][y] = VERDE + "|" + planeta.getNombre().charAt(0) + "-" + dueño + "|" + RESET
    FIN PARA

    // Mostrar el mapa con números de fila y columna
    h.dobleLineaSeparadora()
    IMPRIMIR c.morado("                  MAPA KONQUEST")

    // Mostrar los números de columna en la parte superior
    IMPRIMIR "    "  // Espacio para el índice de fila
    PARA j DESDE 0 HASTA columnas HACER
        IMPRIMIR "  " + VERDE + j + RESET + "  "
    FIN PARA
    IMPRIMIR ""

    // Mostrar el tablero con números de fila en el lateral
    PARA i DESDE 0 HASTA filas HACER
```

```
            IMPRIMIR VERDE + i + "  " + RESET  // Número de fila al inicio de cada línea
            PARA j DESDE 0 HASTA columnas HACER
                IMPRIMIR tablero[i][j] + " "
            FIN PARA
            IMPRIMIR ""
        FIN PARA
        h.dobleLineaSeparadora()
    FIN MÉTODO

    // Obtener el número de filas del mapa
    MÉTODO getFilas()
        RETORNAR filas
    FIN MÉTODO

    // Obtener el número de columnas del mapa
    MÉTODO getColumnas()
        RETORNAR columnas
    FIN MÉTODO
FIN CLASE
```