

# **Proyecto – Entrega 1**

**Santiago Castro Zuluaga**

**Daniela Martínez Quiroga**

**María Isabella Rodríguez Arévalo**

**Lina María Salamanca Tovar**

**Pontificia Universidad Javeriana**

**Ingeniería de Sistemas**

**Estructura de Datos**



Pontificia Universidad  
**JAVERIANA**  
Colombia

## Tabla de Contenido

1.	Diseño .....	3
I.	Especificación de requisitos .....	3
II.	Diseño de TADs.....	4
III.	Diagramas .....	8
2.	Plan de pruebas .....	14
I.	Presentación .....	<b>¡Error! Marcador no definido.</b>

# 1. Diseño

## I. Especificación de requisitos

Especificación del comando inicializar	
Descripción	Realiza las operaciones necesarias para inicializar el juego. Permite determinar la cantidad de jugadores, elegir a cada uno su identificador y sus territorios por turnos.
Entradas	No tiene entradas
Salidas	No tiene salidas
Condiciones	<ol style="list-style-type: none"><li>Validación del comando<ul style="list-style-type: none"><li>Condición 1: El comando debe estar correctamente escrito y en minúscula.</li><li>Acción 1: Mostrar mensaje de error: Comando inválido.</li></ul></li><li>Condición 2: El juego debe estar sin inicializar<ul style="list-style-type: none"><li>Acción 2: Mostrar mensaje de error: Juego en curso</li></ul></li></ol>

Especificación del comando salir	
Descripción	Termina la ejecución de la aplicación
Entradas	No tiene entradas
Salidas	No tiene salida por pantalla
Condiciones	<ol style="list-style-type: none"><li>Validación del comando<ul style="list-style-type: none"><li>Condición: El comando debe estar correctamente escrito y en minúscula.</li><li>Acción: Mostrar mensaje de error: Comando inválido.</li></ul></li></ol>

Especificación del comando turno	
Descripción	Permite a un jugador obtener nuevas unidades, atacar y fortificar.
Entradas	<ol style="list-style-type: none"><li>&lt;id_jugador&gt;<ul style="list-style-type: none"><li>Tipo: Número entero</li><li>Restricciones: El número debe ser entero sin signo</li></ul></li></ol>
Salidas	
Condiciones	<ol style="list-style-type: none"><li>Validación del comando<ul style="list-style-type: none"><li>Condición: El comando debe estar correctamente escrito y en minúscula.</li><li>Acción: Mostrar mensaje de error: Comando inválido</li></ul></li><li>Validación de la entrada<ul style="list-style-type: none"><li>Condición 1: Se debe ingresar en la consola &lt;id_jugador&gt; junto al comando.</li><li>Acción 1: Mensaje de error: Comando incompleto, ingrese el parámetro.</li><li>Condición 2: &lt;id_jugador&gt; debe cumplir con las especificaciones de entrada, y debe ser existente en la partida.</li><li>Acción 2: Mensaje de error: Jugador no válido.</li></ul></li></ol>

	<ul style="list-style-type: none"> <li>- Condición 3: El &lt;id_jugador&gt; ingresado debe respetar los turnos</li> <li>- Acción 3: Mensaje de error: Jugador fuera de turno.</li> <li>- Condición 4: Debe haber una partida previamente inicializada.</li> <li>- Acción 4: Mensaje de error: Juego no inicializado.</li> <li>- Condición 5: La partida debe estar activa, sin un ganador.</li> <li>- Acción 5: Juego terminado.</li> </ul>
--	---

## II. Diseño de Tipos de Datos Abstractos

Diseño hecho de acuerdo con la plantilla de clase:

### TAD Carta

Conjunto mínimo de datos

- id, número entero, identificador de la carta.
- Figura, cadena de caracteres, nombre de la figura.
- País, cadena de caracteres, nombre del país.
- Continente, cadena de caracteres, nombre del continente.

Operaciones

- Carta(id, figura, continente, pais), inicializa los valores del objeto carta
- getID(), retorna el número entero que representa el identificador de la carta
- getFigura(), retorna una cadena de caracteres correspondiente a la figura en la carta
- getContinente(), retorna una cadena de caracteres del Continente el cual tiene la carta
- getPais(), retorna una cadena de caracteres que contiene el País representado en la carta

### TAD Jugador

Conjunto mínimo de datos

- Id, numero entero, identificador único del jugador
- Color, cadena de caracteres, color representativo del jugador
- Alias, cadena de caracteres, nombre del jugador
- Unidades, numero entero, cantidad de unidades que posee el jugador
- Cartas, lista de Carta, cartas que tiene
- Puntaje , numero entero, indica la cantidad de países que tiene el jugador

Operaciones

- Jugador(), genera un nuevo Jugador
- Jugador(id, color, alias), genera un nuevo Jugador con estas variables inicializadas.
- setId(idN), fija el nuevo identificador del Jugador
- setColor(colorN), fija el nuevo color del Jugador
- setAlias(aliasN), fija el nuevo alias del Jugador
- setUnidades(unidades), fija la cantidad de unidades del Jugador
- setCartas(cartasN), fija la nueva lista de cartas del Jugador
- setPuntaje (puntaje), fija el puntaje del jugador
- getId(), retorna el número entero correspondiente al identificador

- getColor(), retorna la cadena de caracteres del color del Jugador
- getAlias(), retorna la cadena de caracteres del alias del Jugador
- getUnidades(), retorna el número entero de la cantidad de unidades del Jugador
- getCartas(), retorna la lista de cartas del Jugador
- getActivo(): Retorna un booleano que indica si el Jugador se encuentra activo o no
- getPuntaje(), retorna la cantidad de países que tiene el Jugador
- getActivo(), retorna el booleano de si el Jugador se encuentra activo o no
- agregarCarta(carta), permite agregar una Carta a la lista de Cartas del Jugador
- quitarCarta(idP), permite remover una carta que coincida con el identificador enviado
- tresCartasCumplen(\*ganaIguales,\*ganaTodas), permite verificar si el Jugador cumplió con los requisitos del juego (relacionadas a las cartas) para obtener recompensas

## TAD País

### Conjunto mínimo de datos

- id, número entero, identifica al país con un número único
- Nombre, cadena de caracteres, nombre del país
- Continente, cadena de caracteres, nombre del continente
- Unidades, numero entero, unidades que hay en el territorio del jugador
- ID jugador, numero entero, identificador del jugador que tiene el territorio
- Conexiones, lista de números enteros, lista que contiene a los países vecinos.

### Operaciones

- País(id, nombre, continente), genera un país con estas variables inicializadas
- Set\_unidades(unidades), fija la cantidad de unidades que tiene el país
- Set\_id\_jugador(id\_jugador), fija el identificador del Jugador al cual le pertenece el país
- Get\_id(), retorna el valor del identificador del País
- Get\_nombre(), retorna una cadena de caracteres que es el nombre del País
- Get\_Continente(), retorna la cadena de caracteres del nombre del continente al cual pertenece el país
- Get\_unidades(), retorna el número entero de las unidades ubicadas en el País
- Get\_conexiones(), retorna la lista de números enteros que contiene los identificadores de los vecinos del país
- Get\_id\_jugador(), retorna el id del jugador al cual pertenece
- Agg\_conexion(conexión), recibe un número entero para agregarlo a la lista de conexiones

## TAD Continente

### Conjunto mínimo de datos

- Nombre, cadena de caracteres, nombre del continente,
- Paises, lista de País, lista que contiene todos los países que pertenecen al continente

### Operaciones

- Continente(nombre), genera un continente con el nombre inicializado
- Get\_nombre(), retorna una cadena de caracteres del nombre del Continente
- Get\_paises(), retorna una lista de Países que pertenecen al Continente

- setNombre(nombre), fija el nombre del continente
- aggPais(país), recibe un país para agregarlo al continente
- aggConexion(país, vecino) recibe los identificadores del país principal yb de su vecino para agregarlo a la lista conexiones de País
- lleno(), retorna una verdadero o falso de acuerdo a si el continente ya ha sido ocupado completamente
- ocuparPais(idJugador, idPais), recibe el identificador del jugador y del país para asignarle al jugador el país que seleccionó
- paisExiste(idP), retorna un booleano que indica si el País que estás buscando existe o no
- jugadorOcupaPais(idP, idJ), retorna un booleano que indica si el Jugador enviado está ocupando el País consultado
- paisVecino(origen, destino), retorna un booleano que indica si los dos países enviados son vecinos
- quitarUnidad(idP, &encontrado), retorna un booleano que indica si el Jugador pierde el País de acuerdo a la cantidad de unidades que haya
- intercambioPorPaises(idJ), retorna un booleano que indica si el Jugador enviado coincide con el Jugador registrado en algún País de la lista
- moverUnidad(idP, unidades), retorna un booleano indicando si hubo movimientos de las unidades
- fortificar(idP, idJ, unidades), retorna un booleano indicando que se pudo hacer el proceso de agregar más unidades a un País

## TAD Partida

### Conjunto mínimo de datos

- id, número entero, identificador de la partida
- jugadores, vector de Jugador, jugadores de la partida
- tablero, lista de Continente, tablero de juego
- cartas, lista de Carta, cartas en la partida

### Operaciones

- Partida(id), genera una partida con el identificador inicializado
- get\_jugadores(), retorna el vector de jugadores
- get\_cartas() retorna la lista de cartas
- get\_tablero(), retorna la lista de continentes
- set\_id(id), fija el identificador
- countLines( archivo\_cartas), contar líneas del archivo de cartas
- cargarCartas(archivo\_cartas), cargar cartas leyendo archivo
- buscarColorRepetido (color), retornar si un color ya está elegido por jugador
- asignarUnidades(), asignar unidades a jugadores
- inicializarJugadores(), inicializar información de jugadores
- repetido(nombre), retornar si un continente ya está repetido
- inicializarTablero(), cargar continentes
- llenarContinentes(), llenar continentes los países
- aggConexion(pais, vecino), agregar vecino a un país
- cargarConexiones(archivo), leer países vecinos del archivo de conexiones
- tableroLleno(), retornar si el tablero está lleno
- paisLleno(id), saber si país está ocupado

- ocuparPais(id, idPais, unidades), ocupar país por jugador
- obtenerCarta(idPais), reclamar carta por ocupar país
- ubicarUnidades(inicializado, numUnidades), ubicar unidades de cada jugador
- mostrarInicializacion(), imprimir resumen
- paisExiste(idP), retornar si país existe
- jugadorOcupaPais(idJ, idP), retornar si jugador ocupa país
- paisVecino(paisOrigen, paisDestino), retornar si país es vecino de otro
- paisAtacable(idJ, idP), retornar si país es atacable
- puedeAtacar(posJ), retornar si jugador puede atacar
- origenAptoParaAtaque(posJ, idP), retornar si país origen permite atacar
- elegirUbicacionAtaque(posJug, paisOrigen, paisDestino), seleccionar origen y destino de ataque
- buscarAtacado(idP), buscar jugador atacado
- lanzarDados(numDados), lanzar dados
- quitarUnidad(idP), quitar unidad a jugador perdedor del ataque
- atacar(posAtacante, origen, destino), ejecutar ataque
- puedeUbicar(idJ), Jugador tiene unidades y territorio para ubicar fichas
- ubicarNuevasUnidades(posJ, gana, propias), ubicar unidades ganadas por jugador
- calcularPaises(idJ), contar países que pertenecen a un jugador
- intercambioNormal(posJ), dar unidades ganadas según países poseídos y ubicarlas
- intercambioPorPaises(posJ), dar unidades ganadas según continentes conquistados y ubicarlas
- intercambioPorCartasCondicionales(posJ), retornar si jugador posee cartas con las condiciones de entrega de unidades
- elegirCartasIntercambio(posJ, figura, mismas), se realiza el intercambio de cartas de acuerdo a los parámetros ingresados por el jugador
- ubicarUnidadesDeCartas(figura, posJ, gana, mismas), permite hacer el intercambio de las recompensas al tener 3 cartas válidas
- intercambiarCartas(posJ, gana), permite hacer intercambios de cartas de acuerdo al tipo de unidad
- unidadesSuficientes(posJ, idP, unidades), retornar si jugador tiene unidades suficientes
- paisFortificable(idJ, idP), retornar si país es fortificable
- puedeFortificar(posJ), retornar si jugador puede fortificar
- aptoParaFortificar(idP, posJ), retornar si país de origen permite fortificar
- moverUnidades(posJ, origen, destino, unidadesM), mover unidades en fortificación
- fortificarTerritorio(jugadorIndex), recibir ubicación para fortificar
- jugadorVigente(posJ), retornar si jugador aún tiene cartas para intercambiar, unidades en el tablero o unidades propias
- finalizado(ganador), retornar si un jugador ya gano, junto con su id

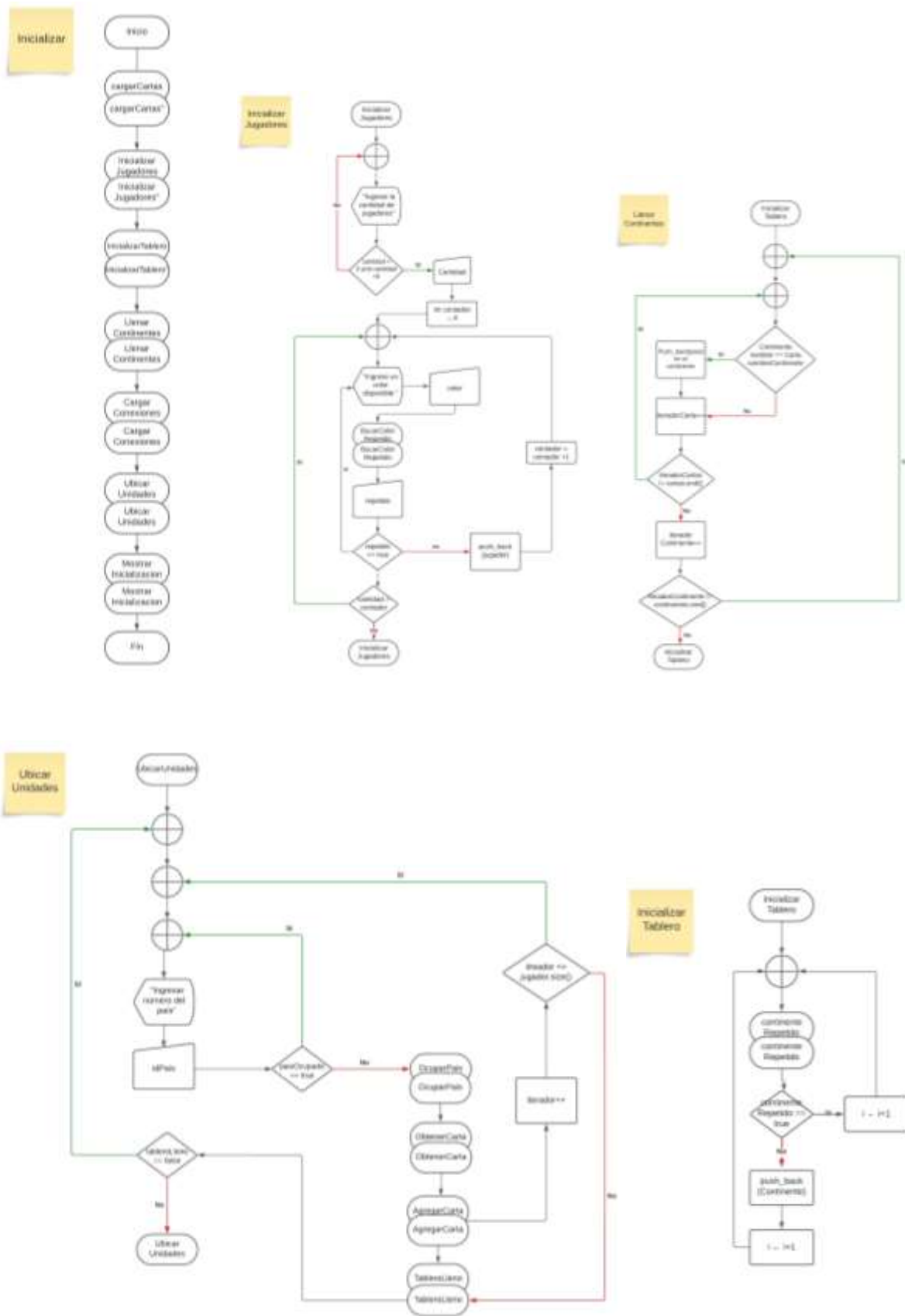
Diagrama de TADs:



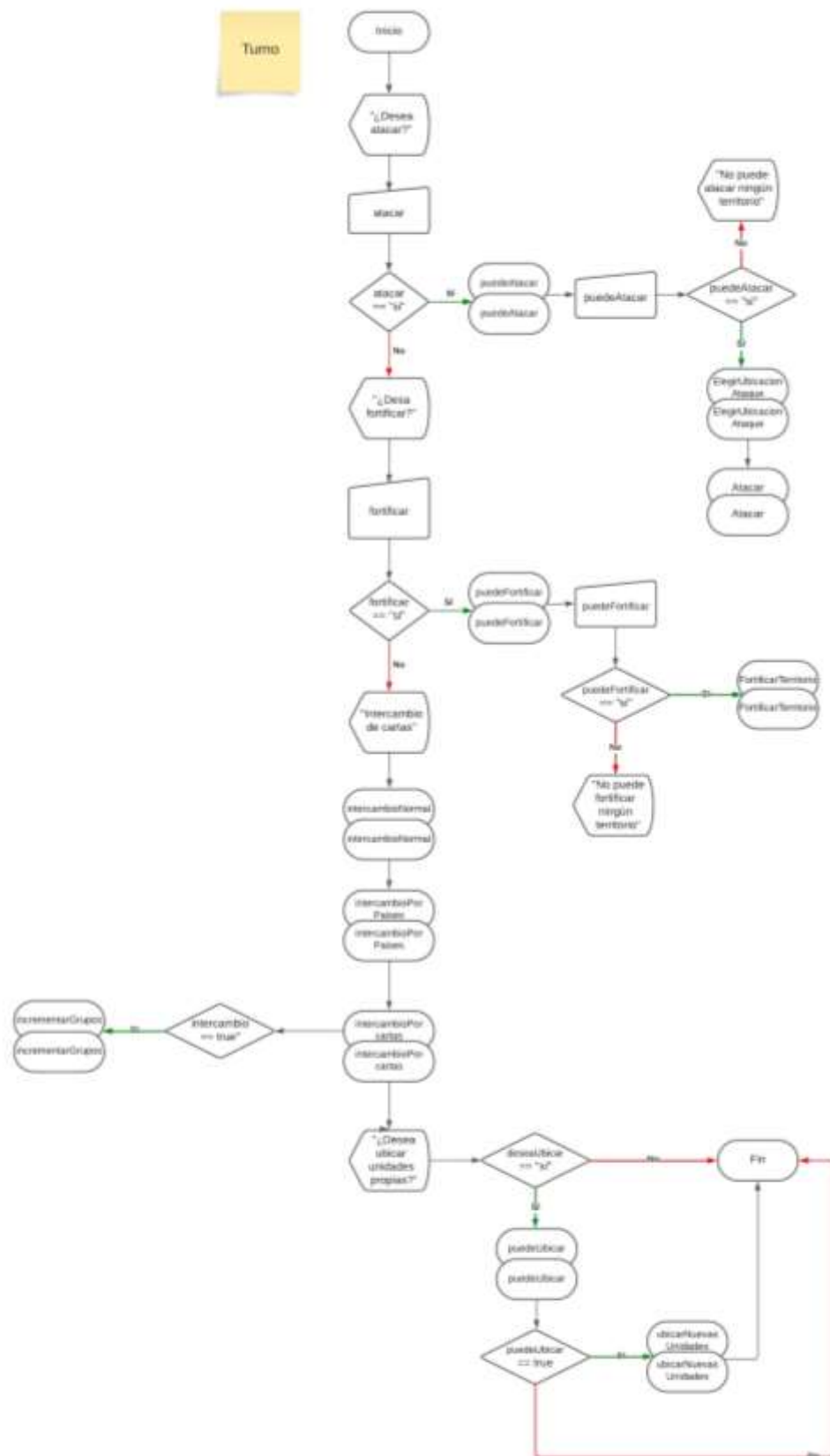


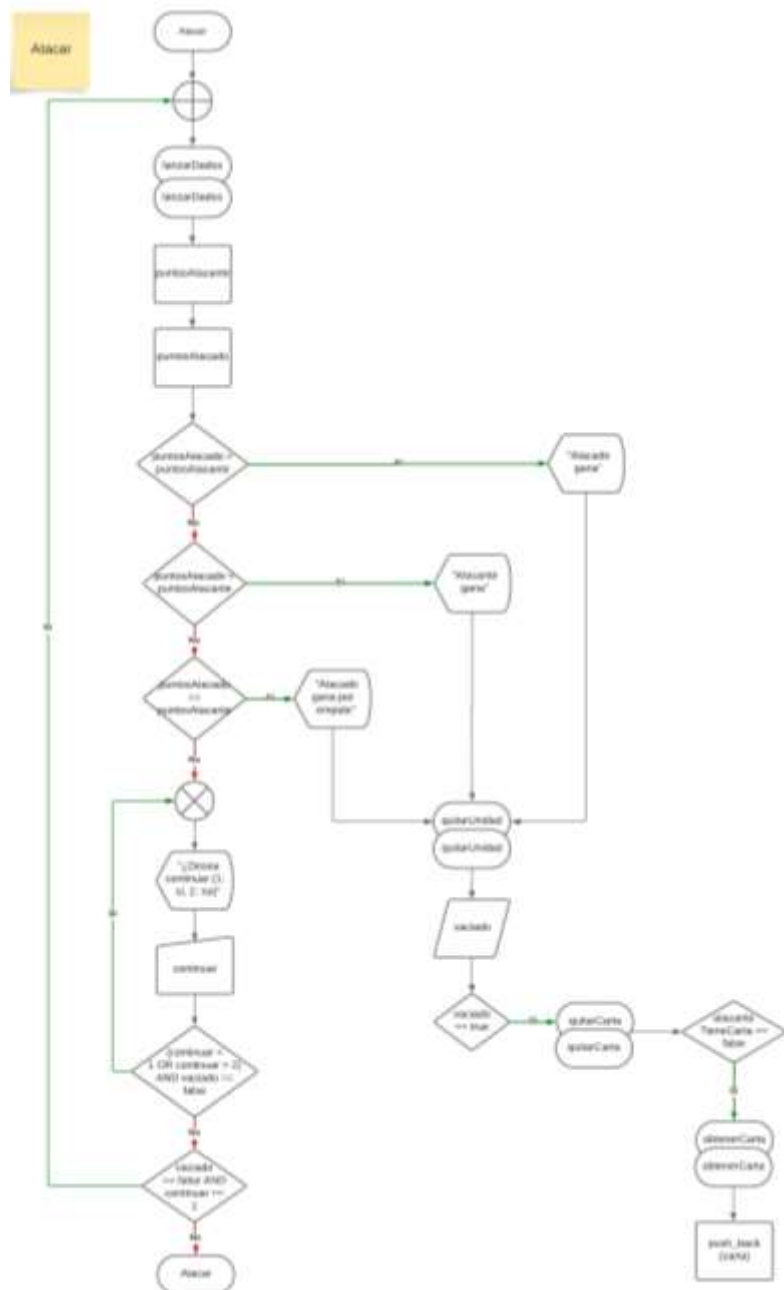
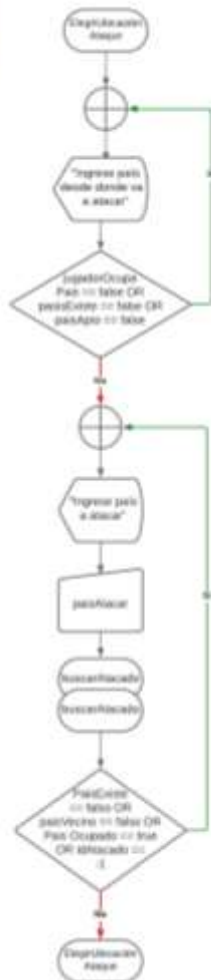


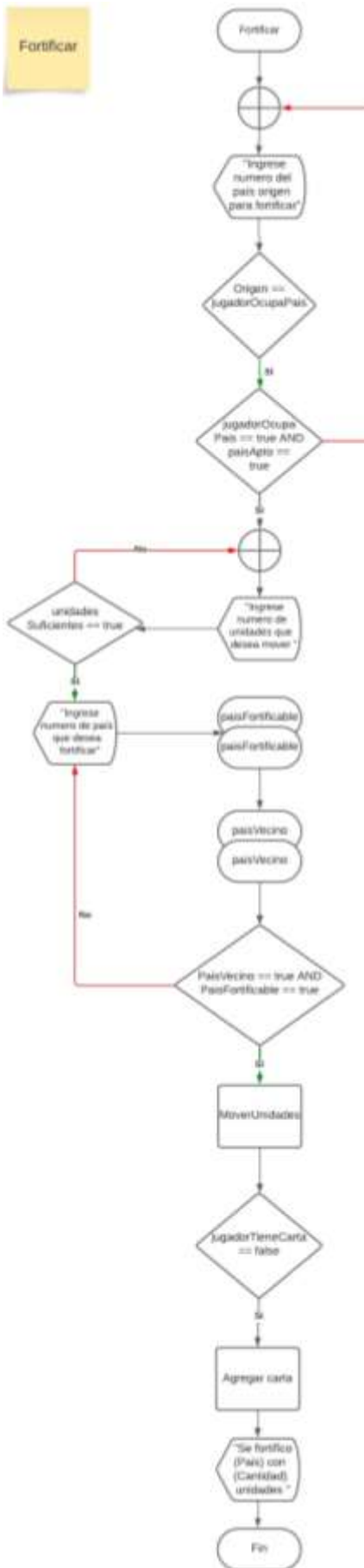
b. Diagramas relacionados a la inicialización



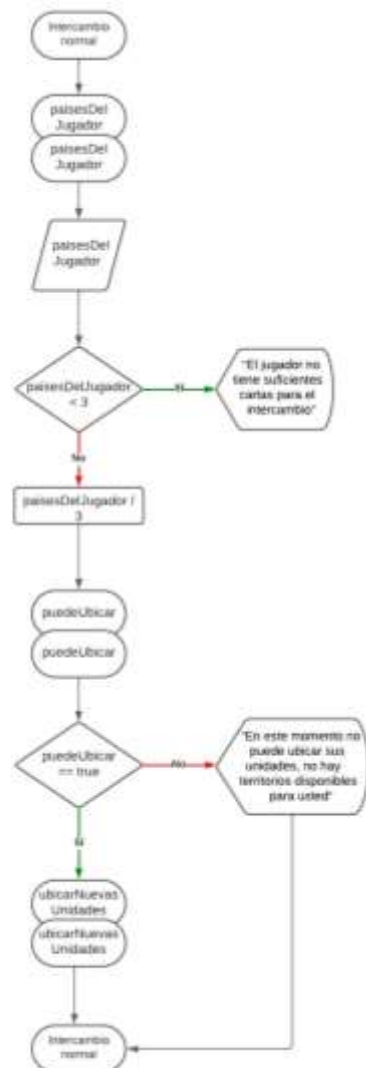
c. Diagramas relacionados a turno







Intercambio normal

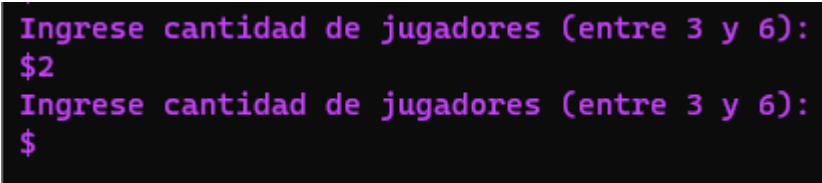
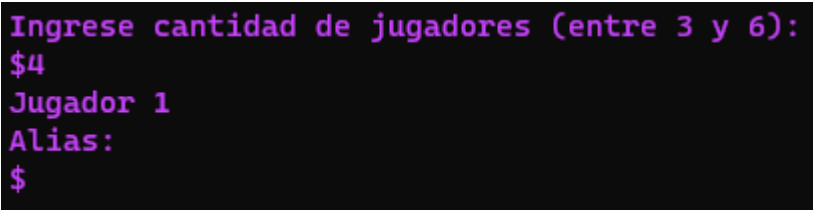




## 2. Plan de pruebas

PLAN DE PRUEBAS DEL COMANDO INICIALIZAR			
Descripción del caso	Valores de entrada	Valor esperado	Valor obtenido
1. Inicializar el juego con cantidad de jugadores fuera del rango permitido (3 a 6).	$a = 2$ Nota: $a$ número de jugadores	“Ingrese cantidad de jugadores (entre 3 y 6)”	“Ingrese cantidad de jugadores (entre 3 y 6)”
2. Inicializar el juego con cantidad de jugadores dentro del rango permitido (3 a 6).	$a = 4$ Nota: $a$ número de jugadores	Ingreso de información de jugadores: “Jugador 1, Alias: ”	Ingreso de información de jugadores: “Jugador 1, Alias: ”
3. Inicializar los jugadores seleccionando colores repetidos o ya elegidos previamente por otro jugador.	$a = 3$ $b1 = “J1”, c1 = 1$ $b2 = “J2”, c2 = 1$ Nota: $b(i)$ alias del jugador y $c(i)$ color del jugador.	“Color Repetido”	“Color Repetido”
4. Inicializar los jugadores seleccionando color distinto para cada uno.	$a = 3$ $b1 = “J1”, c1 = 1$ $b2 = “J2”, c2 = 2$ $b3 = “J3”, c3 = 3$ Nota: $b(i)$ alias del jugador y $c(i)$ color del jugador.	Asignación correcta de los colores, se procede a asignar los territorios	Asignación correcta de los colores, se procede a asignar los territorios

5. Ubicar unidades en países no disponibles (ya elegidos previamente) o no existentes.	$a = 3$ $b1 = \text{"J1"}, c1 = 1$ $b2 = \text{"J2"}, c2 = 2$ $b3 = \text{"J3"}, c3 = 3$ $d1 = 1$ $d2 = 1$ Nota: $b(i)$ alias del jugador, $c(i)$ color del jugador y $d(i)$ id del país seleccionado por el jugador $b(i)$	"País ocupado o no válido"	"País ocupado o no válido"
6. Ubicar unidades correctamente, cada país ocupado por un único jugador.	$a = 3$ $b1 = \text{"J1"}, c1 = 1$ $b2 = \text{"J2"}, c2 = 2$ $b3 = \text{"J3"}, c3 = 3$ $d1 = 1$ $d2 = 2$ $d3 = 3$ $\dots$ $d42 = 42$ Nota: $b(i)$ alias del jugador, $c(i)$ color del jugador y $d(i)$ id del país seleccionado por cada jugador $b(i)$ en su turno.	Se obtiene un resumen de los países que tiene cada jugador y su color	Se obtiene un resumen de los países que tiene cada jugador y su color

EVIDENCIAS DE EJECUCIÓN	
Descripción del caso	Resultado
1. Inicializar el juego con cantidad de jugadores fuera del rango permitido (3 a 6).	 <pre> Ingrese cantidad de jugadores (entre 3 y 6): \$2 Ingrese cantidad de jugadores (entre 3 y 6): \$           </pre>
2. Inicializar el juego con cantidad de jugadores dentro del rango permitido (3 a 6).	 <pre> Ingrese cantidad de jugadores (entre 3 y 6): \$4 Jugador 1 Alias: \$           </pre>

<p>3. Inicializar los jugadores seleccionando colores repetidos o ya elegidos previamente por otro jugador.</p>	<div> <div> Alias: \$J1 Ingrese numero para elegir color: 1. verde 2. azul 3. rojo 4. amarillo 5. rosado 6. morado \$1 </div> <div> Alias: \$J2 Ingrese numero para elegir color: 1. verde 2. azul 3. rojo 4. amarillo 5. rosado 6. morado \$1 Color repetido </div> </div>
<p>4. Inicializar los jugadores seleccionando color distinto para cada uno.</p>	<div> <div> Alias: \$J1 Ingrese numero para elegir color: 1. verde 2. azul 3. rojo 4. amarillo 5. rosado 6. morado \$1 Alias: \$J3 Ingrese numero para elegir color: 1. verde 2. azul 3. rojo 4. amarillo 5. rosado 6. morado \$3 </div> <div> Alias: \$J2 Ingrese numero para elegir color: 1. verde 2. azul 3. rojo 4. amarillo 5. rosado 6. morado \$2  Jugador:J1 Ingrese el numero del pais: \$ </div> </div>
<p>5. Ubicar unidades en países no disponibles (ya elegidos previamente) o no existentes.</p>	<div> Jugador:J1 Ingrese el numero del pais: \$1 Jugador:J2 Ingrese el numero del pais: \$1 Pais ocupado o no valido Jugador:J2 Ingrese el numero del pais: \$ </div>
<p>6. Ubicar unidades correctamente, cada país ocupado por un único jugador.</p>	<div> Muestra parte del resumen:  Jugador 1:J1  color: verde tiene 14  1:Alaska  4:Estados Unidos Orientales  7:Ontario  11:Brasil  13:Colombia  16:Europa del Norte  19:Ucrania  22:Africa Oriental  25:Africa del Norte  28:China  31:Japon  34:Mongolia  37:Ural  40:Indonesia </div>



Presentación: [https://www.canva.com/design/DAFs-B0D9C8/IHBYdRtNkj\\_Y1Z\\_jQiHuYA/view?utm\\_content=DAFs-B0D9C8&utm\\_campaign=designshare&utm\\_medium=link&utm\\_source=publishsharelink](https://www.canva.com/design/DAFs-B0D9C8/IHBYdRtNkj_Y1Z_jQiHuYA/view?utm_content=DAFs-B0D9C8&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink)