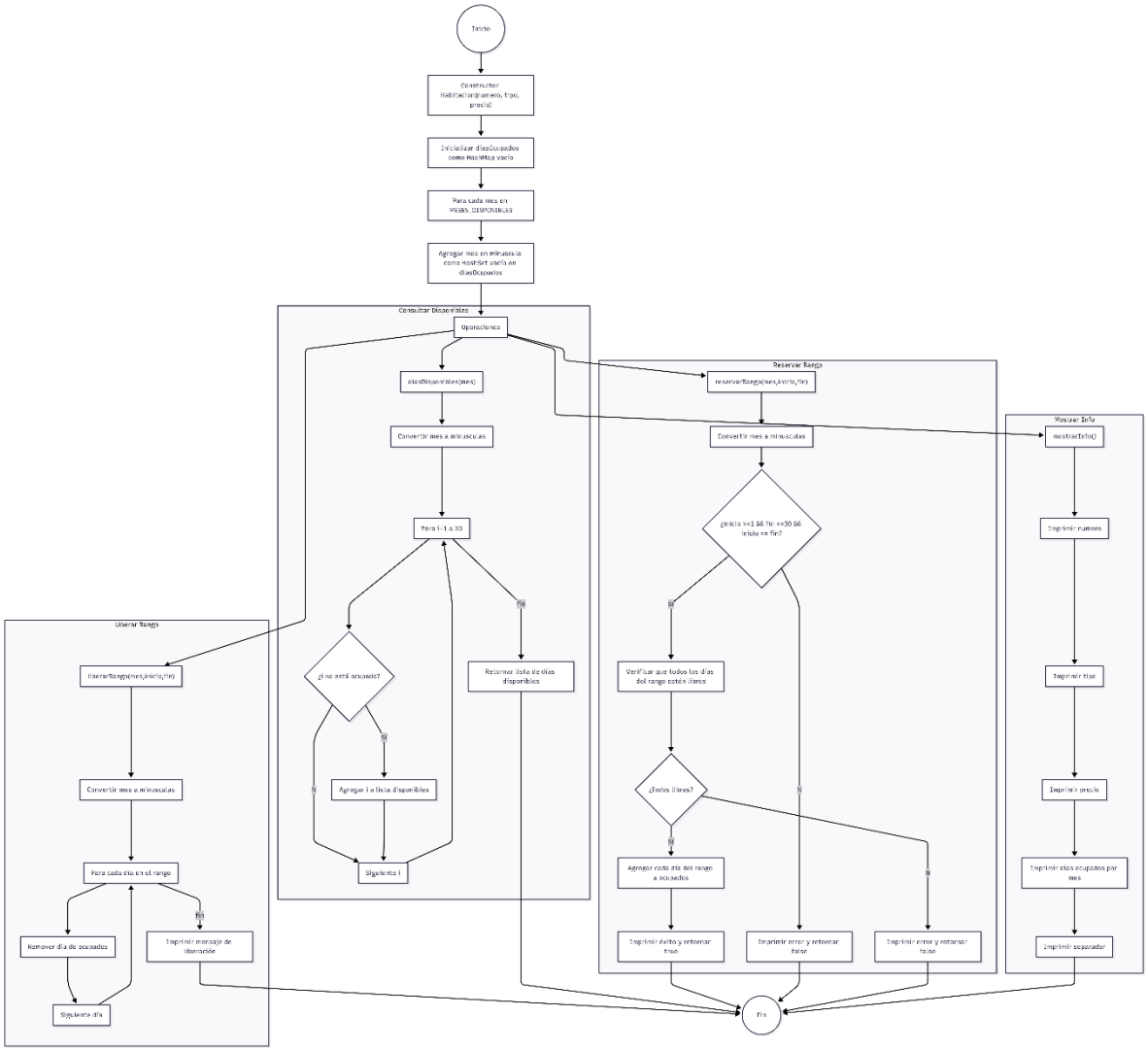
Pseudocodigos y diagramas

1. Clase Habitacion

```
Constructor Habitacion(numero, tipo, precio):
    Asignar numero = numero
    Asignar tipo = tipo
    Asignar precio = precio
    Inicializar diasOcupados como un mapa vacío (HashMap)

    Para cada mes en la lista MESES_DISPONIBLES:
        Convertir mes a minúsculas
        Agregar a diasOcupados: clave = mes, valor = conjunto vacío de días (HashSet)

diasDisponibles(mes):
    Convertir mes a minúsculas
    Crear lista vacía llamada disponibles
    Para i desde 1 hasta 30:
        Si i no está en diasOcupados[mes]:
            Agregar i a disponibles
    Retornar disponibles

reservarRango(mes, inicio, fin):
    Convertir mes a minúsculas
    Si inicio < 1 o fin > 30 o inicio > fin:
        Imprimir "Rango de días inválido."
        Retornar falso

    Para cada dia desde inicio hasta fin:
        Si diasOcupados[mes] contiene dia:
```

```
            Imprimir "El día [dia] de [mes] no está disponible. Reserva fallida."
            Retornar falso

        Para cada dia desde inicio hasta fin:
            Agregar dia a diasOcupados[mes]

        Imprimir "Habitación [numero] reservada del día [inicio] al [fin] de [mes]."
        Retornar verdadero

liberarRango(mes, inicio, fin):
    Convertir mes a minúsculas
    Para cada dia desde inicio hasta fin:
        Remover dia de diasOcupados[mes]

    Imprimir "Días del [inicio] al [fin] de [mes] liberados en habitación [numero]."

mostrarInfo():
    Imprimir "Habitación #[numero]"
    Imprimir "Tipo: [tipo]"
    Imprimir "Precio: $[precio]"
    Imprimir "Días ocupados por mes:"
    Para cada mes en diasOcupados:
        Imprimir "- [mes]: [diasOcupados[mes]]"
    Imprimir "----------------------"

Getters (getNumero, getTipo, getPrecio):
    Retornar el valor correspondiente (numero, tipo, precio)
```



Clase Hotel

Constructor Hotel(nombre):

Asignar nombre = nombre

Inicializar usuarios como lista vacía (ArrayList)

Inicializar habitaciones como lista vacía (ArrayList)

agregarUsuario(usuario):

Agregar usuario a la lista usuarios

Imprimir "Usuario agregado al hotel."

```
buscarUsuarioPorEmail(email):
    Para cada usuario en usuarios:
        Si usuario.getEmail() == email:
            Retornar usuario
    Retornar null

agregarHabitacion(habitacion):
    Agregar habitacion a la lista habitaciones

buscarHabitacionPorTipo(tipo):
    Para cada habitacion en habitaciones:
        Si habitacion.getTipo() coincide con tipo (ignorando mayúsculas):
            Retornar habitacion
    Retornar null

mostrarHabitaciones():
    Imprimir "=== Habitaciones de [nombre] ==="
    Para cada habitacion en habitaciones:
        Llamar habitacion.mostrarInfo()

hacerReservaInteractiva(usuario):
    Si usuario no está logueado:
        Imprimir "Debes iniciar sesión para hacer una reserva."
        Retornar

    Imprimir "=== Hacer Reserva ==="
    Inicializar habitacion = null
    Mientras habitacion == null:
        Imprimir "Tipo de habitación (1.simple, 2.doble, 3.suite): "
        Leer tipo del usuario
```

```
    Asignar habitacion = buscarHabitacionPorTipo(tipo)
    Si habitacion == null:
        Imprimir "Tipo inválido o no disponible. Intenta de nuevo."

Inicializar mesElegido = ""
Mientras mesElegido no es válido:
    Imprimir "Meses disponibles para reservar:"
    Para cada mes en MESES_DISPONIBLES:
        Imprimir "- [mes]"
    Imprimir "Elige un mes: "
    Leer mesElegido del usuario, convertir a minúsculas
    Para cada m en MESES_DISPONIBLES:
        Si m == mesElegido:
            Marcar como válido
    Si no válido:
        Imprimir "Mes no válido. Intenta de nuevo."

Asignar diasLibres = habitacion.diasDisponibles(mesElegido)
Si diasLibres está vacío:
    Imprimir "No hay días disponibles en [mesElegido] para esta habitación."
    Retornar

Imprimir "Días disponibles en [mesElegido]: [diasLibres]"

Inicializar diaInicio = 0, diaFin = 0
Mientras rango no válido:
    Imprimir "Ingresa día de inicio: "
    Leer diaInicio (validar si es número)
    Imprimir "Ingresa día de fin: "
    Leer diaFin (validar si es número)
```

```
        Si diaInicio < 1 o > 30 o diaFin < 1 o > 30 o diaInicio > diaFin:
            Imprimir "Rango de días inválido. Intenta de nuevo."
        Sino:
            Marcar rango como válido

    Calcular total = habitacion.getPrecio() * (diaFin - diaInicio + 1)
    Imprimir "El precio total para esta reserva será: Cop_[total]"

    Mientras numeroTarjeta no válido:
        Imprimir "Ingresa el número de tarjeta (solo dígitos): "
        Leer numeroTarjeta
        Si no tiene 13-19 dígitos numéricos:
            Imprimir "Número de tarjeta inválido."

    Mientras cvc no válido:
        Imprimir "Ingresa el CVC (3 o 4 dígitos): "
        Leer cvc
        Si no tiene 3-4 dígitos numéricos:
            Imprimir "CVC inválido."

    Imprimir "Datos de pago validados"

    Si habitacion.reservarRango(mesElegido, diaInicio, diaFin):
        Crear nueva Reserva con usuario, habitacion, mesElegido, diaInicio, diaFin
        Agregar reserva a usuario
        Imprimir "Reserva realizada con éxito del día [diaInicio] al [diaFin] de [mesElegido]."
    Sino:
        Imprimir "No se pudo realizar la reserva."

mostrarTodasLasReservas():
```

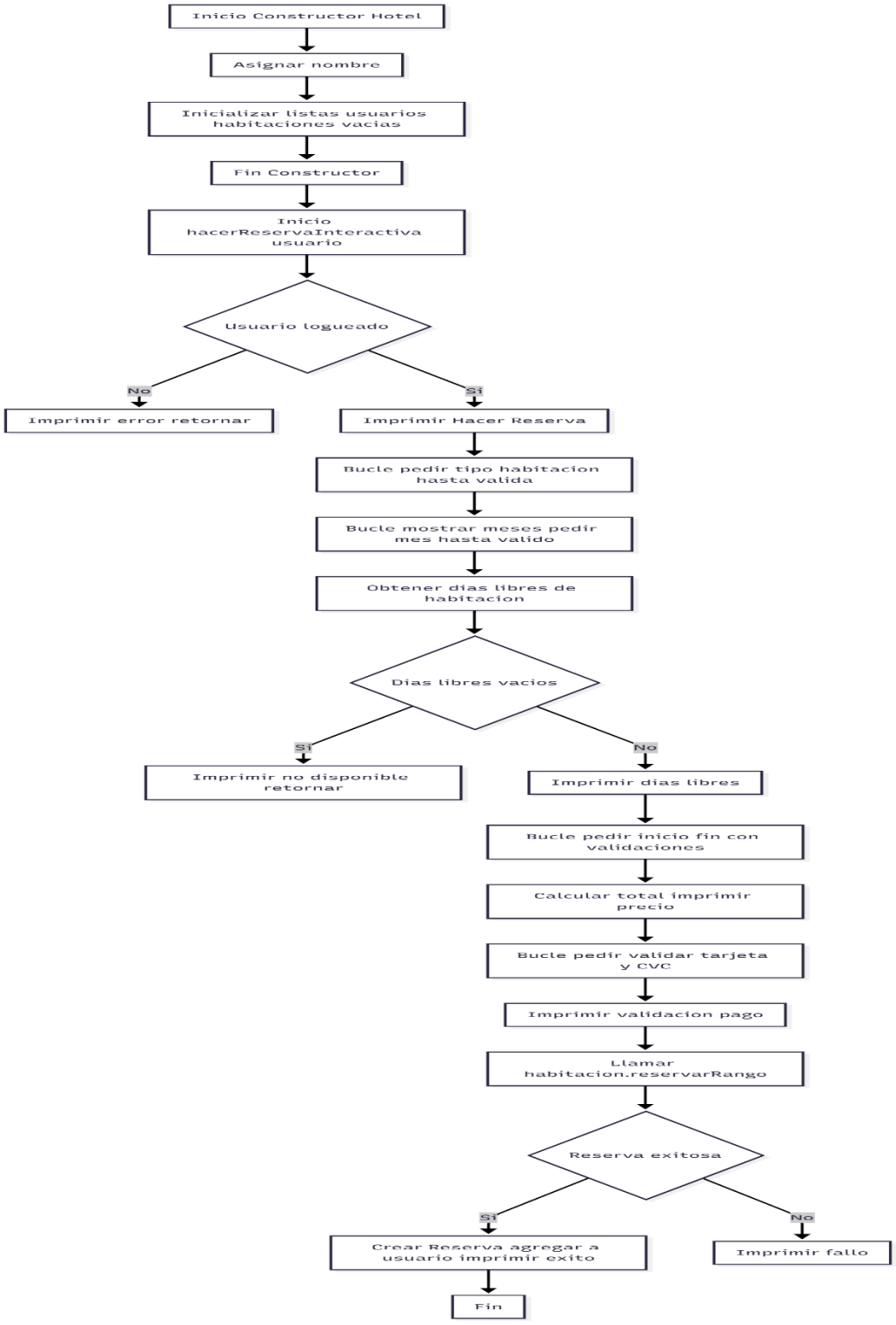
Imprimir "=== Todas las reservas de [nombre] ==="

Para cada usuario en usuarios:

Llamar usuario.consultarReservas()

getNombre():

Retornar nombre



3. Clase Main

main(args):

    Crear Scanner sc

    Crear Hotel hotel con nombre "Silvery Deluxe"

    Agregar habitaciones de ejemplo: Habitacion(101, "simple", 50000), etc.

    Imprimir "==================================="

    Imprimir " Bienvenido al Hotel Silvery Deluxe "

    Imprimir "===================================\n"

    Inicializar aceptarDatos = falso, respuestaValida = falso

    Mientras no respuestaValida:

        Imprimir "¿Acepta el tratamiento de sus datos personales y el uso de cookies?"

        Imprimir "1. Sí, acepto y deseo continuar"

        Imprimir "2. No, salir del sistema"

        Imprimir "Seleccione una opción (1-2): "

        Leer opcion (validar si es número)

        Si opcion == 1:

            aceptarDatos = verdadero, respuestaValida = verdadero

        Sino si opcion == 2:

            Imprimir "\nGracias por usar nuestros servicios. ¡Hasta pronto!"

            Cerrar sc, terminar programa

        Sino:

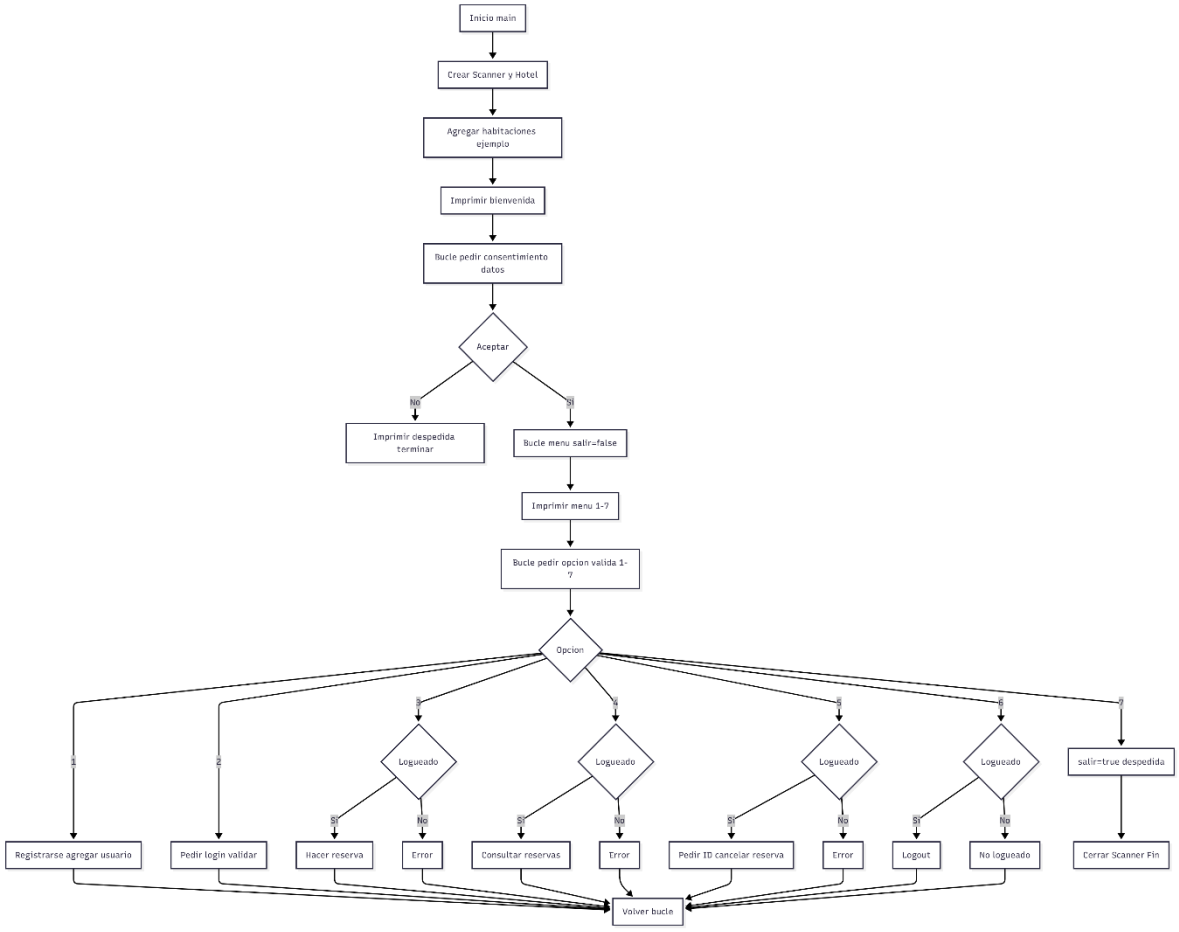
            Imprimir "Opción inválida. Debe ser 1 o 2.\n"

    Inicializar salir = falso, usuarioActual = null

    Mientras no salir:

```
Imprimir "\n=== Menú Principal del Hotel Silvery Deluxe ==="
Imprimir opciones 1-7 (Registrarse, Loguearse, etc.)

Inicializar opcionValida = falso
Mientras no opcionValida:
    Imprimir "Elige una opción (1-7): "
    Leer opcion (validar si es número y en 1-7)
    Si válido:
        opcionValida = verdadero
    Sino:
        imprimir error

Según opcion:
    Caso 1:
        usuarioActual = Usuario.registrarse(), hotel.agregarUsuario(usuarioActual)

    Caso 2:
        Leer email y pass, buscar usuario en hotel,
        si encontrado y loguearse exitoso: usuarioActual = usuario

    Caso 3:
        Si usuarioActual != null:
            hotel.hacerReservaInteractiva(usuarioActual)
        Sino:
            imprimir error

    Caso 4:
        Si usuarioActual != null:
            usuarioActual.consultarReservas()
        Sino:
```

```
            imprimir error

        Caso 5:
            Si usuarioActual != null:
                Leer idReserva (validar número), usuarioActual.cancelarReserva(idReserva)
            Sino:
                imprimir error

        Caso 6:
            Si usuarioActual != null:
                usuarioActual.desloguearse(), usuarioActual = null
            Sino:
                imprimir "No hay usuario logueado."

        Caso 7:
            salir = verdadero, imprimir "\nGracias por usar nuestros servicios. ¡Hasta pronto!"

Cerrar sc
```



4. Clase Reserva

Constructor Reserva(usuario, habitacion, mes, diaInicio, diaFin):

Convertir mes a minúsculas

Si no validarMes(mes):

Lanzar IllegalArgumentException "Mes inválido para la reserva: [mes]"

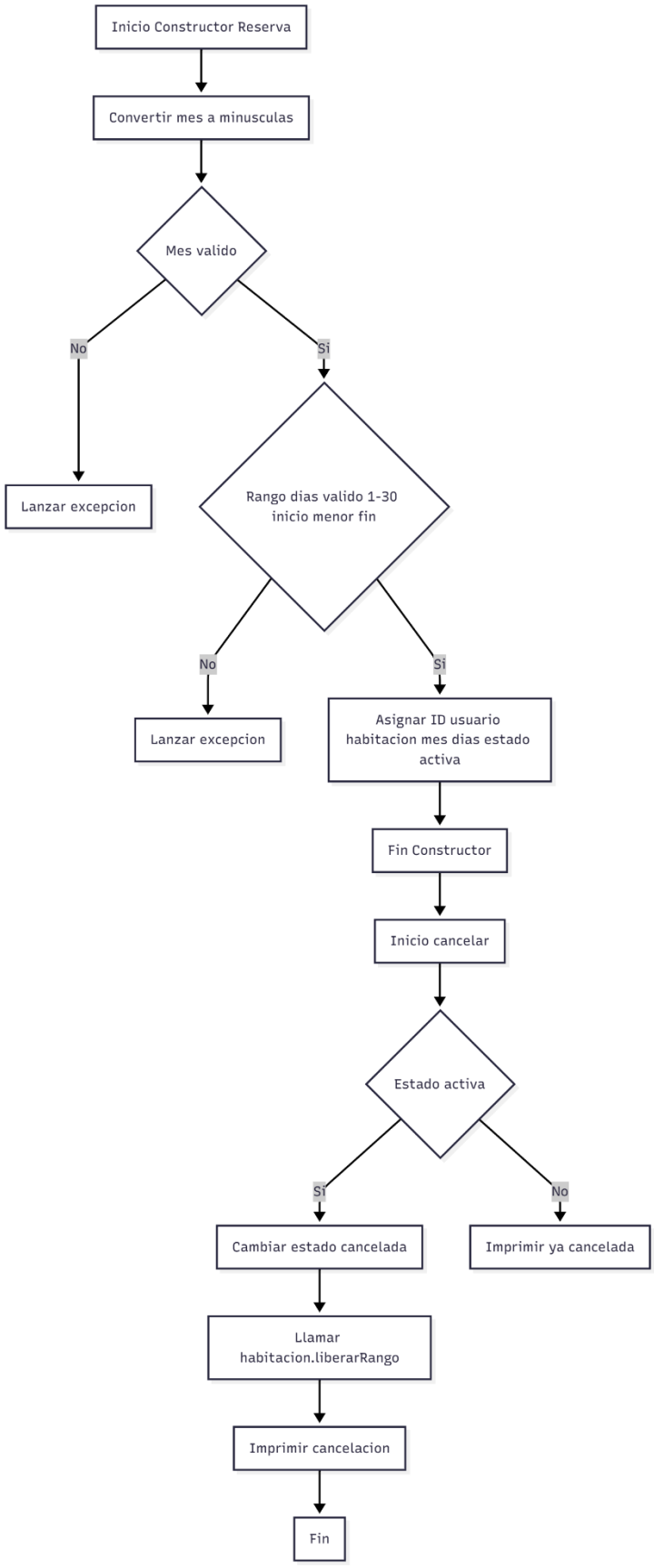
Si diaInicio < 1 o > 30 o diaFin < 1 o > 30 o diaInicio > diaFin:

Lanzar IllegalArgumentException "Rango de días inválido: [diaInicio] - [diaFin]"

Asignar idReserva = contador.getAndIncrement() (ID único)

```
    Asignar usuario = usuario
    Asignar habitacion = habitacion
    Asignar mes = mes
    Asignar diaInicio = diaInicio
    Asignar diaFin = diaFin
    Asignar estado = "activa"

validarMes(mes) (privado):
    Para cada m en MESES_DISPONIBLES:
        Si m coincide con mes (ignorando mayúsculas):
            Retornar verdadero
    Retornar falso

calcularTotal():
    Asignar numDias = diaFin - diaInicio + 1
    Retornar numDias * habitacion.getPrecio()

cancelar():
    Si estado == "activa":
        Asignar estado = "cancelada"
        Llamar habitacion.liberarRango(mes, diaInicio, diaFin)
        Imprimir "Reserva #[idReserva] cancelada."
    Sino:
        Imprimir "La reserva #[idReserva] ya está cancelada."

mostrarInfo():
    Imprimir "ID Reserva: [idReserva]"
    Imprimir "Usuario: [usuario.getNombre()] [usuario.getApellido()]"
    Imprimir "Habitación #[habitacion.getNumero()] ([habitacion.getTipo()])"
    Imprimir "Días: [diaInicio] al [diaFin]"
```

```
    Imprimir "Precio total: Cop_[calcularTotal()]"
    Imprimir "Mes: [mes]"
    Imprimir "Estado: [estado]"
    Imprimir "----------------------"

Getters (getIdReserva, getUsuario, etc.):
    Retornar el valor correspondiente
```


Inicio Constructor Reserva
Convertir mes a minusculas
Mes valido
No
Si
Lanzar excepcion
Rango dias valido 1-30
inicio menor fin
No
Si
Lanzar excepcion
Asignar ID usuario
habitacion mes dias estado
activa
Fin Constructor
Inicio cancelar
Estado activa
Si
No
Cambiar estado cancelada
Imprimir ya cancelada
Llamar
habitacion.liberarRango
Imprimir cancelacion
Fin

5. Clase Usuario

Constructor Usuario(nombre, apellido, email, contraseña):

Asignar nombre = nombre

Asignar apellido = apellido

Asignar email = email

Asignar contraseña = contraseña

Asignar logueado = falso

Inicializar reservas como lista vacía (ArrayList)

registrarse() (estático):

Crear Scanner local sc

Mientras nombre no válido:

Imprimir "Nombre: ", leer nombre

Si no coincide con patrón de letras/espacios:

imprimir error

Mientras apellido no válido:

Imprimir "Apellido: ", leer apellido

Si no coincide con patrón de letras/espacios:

imprimir error

Mientras email no válido:

Imprimir "Email: ", leer email

Si no coincide con regex de email:

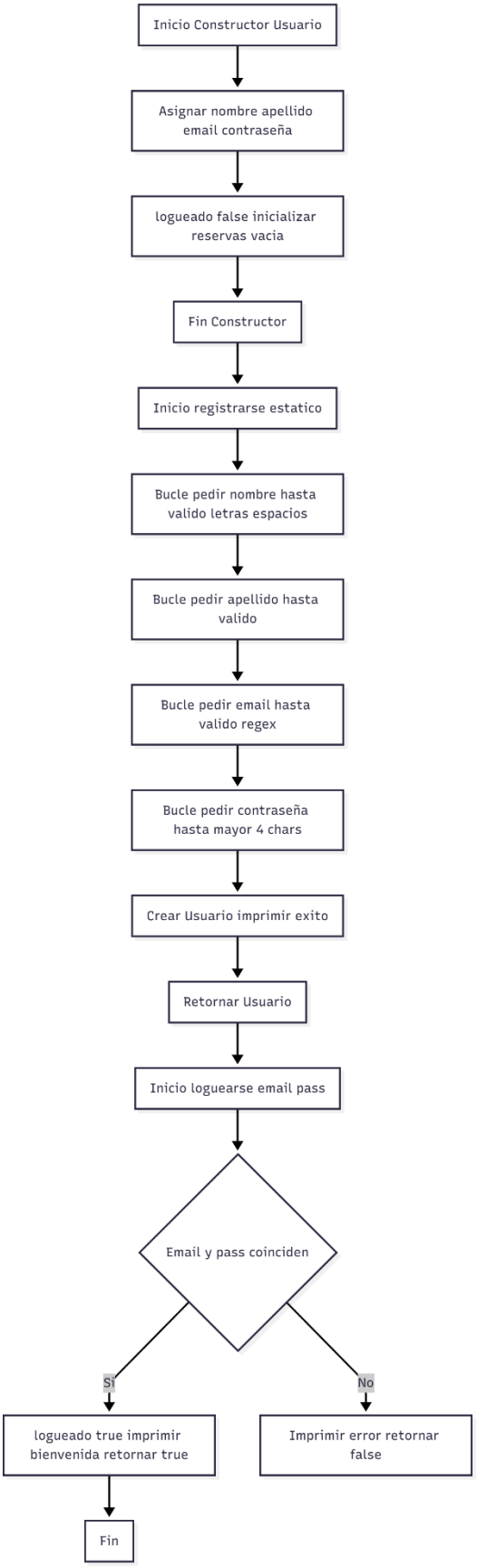
imprimir error

Mientras contraseña no válida:

Imprimir "Contraseña (mínimo 4 caracteres): ", leer contraseña

```
        Si longitud < 4:
            imprimir error

    Crear nuevoUsuario = new Usuario(nombre, apellido, email, contraseña)
    Imprimir "Usuario registrado exitosamente.\n"
    Retornar nuevoUsuario

loguearse(email, contraseña):
    Si this.email == email y this.contraseña == contraseña:
        Asignar logueado = verdadero
        Imprimir "Login exitoso. Bienvenido [nombre] [apellido]!"
        Retornar verdadero
    Sino:
        Imprimir "Email o contraseña incorrectos."
        Retornar falso

desloguearse():
    Si logueado:
        Asignar logueado = falso
        Imprimir "Usuario [nombre] [apellido] ha cerrado sesión."
    Sino:
        Imprimir "No hay usuario logueado."

agregarReserva(reserva):
    Agregar reserva a la lista reservas

consultarReservas():
    Si reservas está vacío:
        Imprimir "No tienes reservas activas."
        Retornar
```

```
    Imprimir "=== Reservas de [nombre] [apellido] ==="
    Para cada reserva en reservas:
        Llamar reserva.mostrarInfo()

cancelarReserva(idReserva):
    Para cada reserva en reservas:
        Si reserva.getIdReserva() == idReserva:
            Llamar reserva.cancelar()
            Retornar

    Imprimir "No se encontró una reserva con ID [idReserva]"

Getters (getNombre, getApellido, getEmail, isLogueado):
    Retornar el valor correspondiente
```


Inicio Constructor Usuario
Asignar nombre apellido email contraseña
logueado false inicializar reservas vacia
Fin Constructor
Inicio registrarse estatico
Bucle pedir nombre hasta valido letras espacios
Bucle pedir apellido hasta valido
Bucle pedir email hasta valido regex
Bucle pedir contraseña hasta mayor 4 chars
Crear Usuario imprimir exito
Retornar Usuario
Inicio loguearse email pass
Email y pass coinciden
Si
No
logueado true imprimir bienvenida retornar true
Imprimir error retornar false
Fin