FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Algo-thief

[7507/9502] Algoritmos y Programación III
Segundo cuatrimestre de 2021
Primer Entrega

Grupo:	5
Alumno 1:	Alejandro Abraham Osco Cabrera
Padrón:	102256
Email:	aosco@fi.uba.ar
Alumno 2:	Agustin Rodriguez
Padrón:	101570
Email:	agrodriguez@fi.uba.ar
Alumno 3:	Ricardo Luizaga
Padrón:	87528
Email:	rluizaga@fi.uba.ar
Alumno 4:	Santigo Curetti
Padrón:	105133
Email:	scuretti@fi.uba.ar
Alumno 5:	Martin Anus
Padrón:	101033
Email:	manus@fi.uba.ar

Índice

1. Introducción

Desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua. Desarrollar la aplicación completa, incluyendo el modelo de clases e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño.

Algo-thief es un juego similar al Carmen Sandiego (wikipedia-Carmen Sandiego - Google Play-Carmen Sandiego). El objetivo del jugador es perseguir y atrapar al ladrón antes del término de una semana aproximadamente. Para ello se deberá emitir una orden de arresto con pistas recolectadas entre los testigos que identifique unívocamente al autor del ilícito.

2. Supuestos

Se toma como supuesto los siguientes actos:

- Al inicio del juego se asigna al jugador una misión aleatoria de las disponibles, inicializándose todas las pistas en los edificios que recorre el ladrón, ubicando al mismo en la ciudad y edificio donde será atrapado.
- El jugador duerme desde las 0 a.m. hasta las 8 a.m.
- El tiempo establecido para atrapar al ladrón es 154 horas. Desde el Lunes 7 a.m. hasta el Domingo 5 p.m.
- El jugador comienza con rango Novato.
- Dependiendo el rango del policía, se le asignaran casos más fáciles o difíciles. Esto incluye que las pistas sean más directas y que el ladrón recorra más ciudades.
- El jugador deberá emitir una orden de arresto con la descripción del sospechoso previo a encontrarse con el ladrón para poderlo arrestar. Caso contrario pierde la misión.
- Al visitar un edificio, el policía puede encontrar una pista, un herida (de cuchillo o arma de fuego) o al ladrón.
-

3. Modelo de dominio

El presente trabajo tiene como fin realizar un juego similar al Carmen Sandiego. Nuestro objetivo es perseguir y atrapar al ladrón antes de un tiempo límite. Para lograrlo se necesita recolectar pistas visitando edificios en distintas ciudades del mundo. Las mismas pueden ser del ladrón así también como de las ciudades que éste visitó. Cuando se tiene suficiente información del sospechoso se emite una orden de arresto para poder identificarlo y finalmente atraparlo en la ciudad donde se esconde.

4. Diagramas de clase

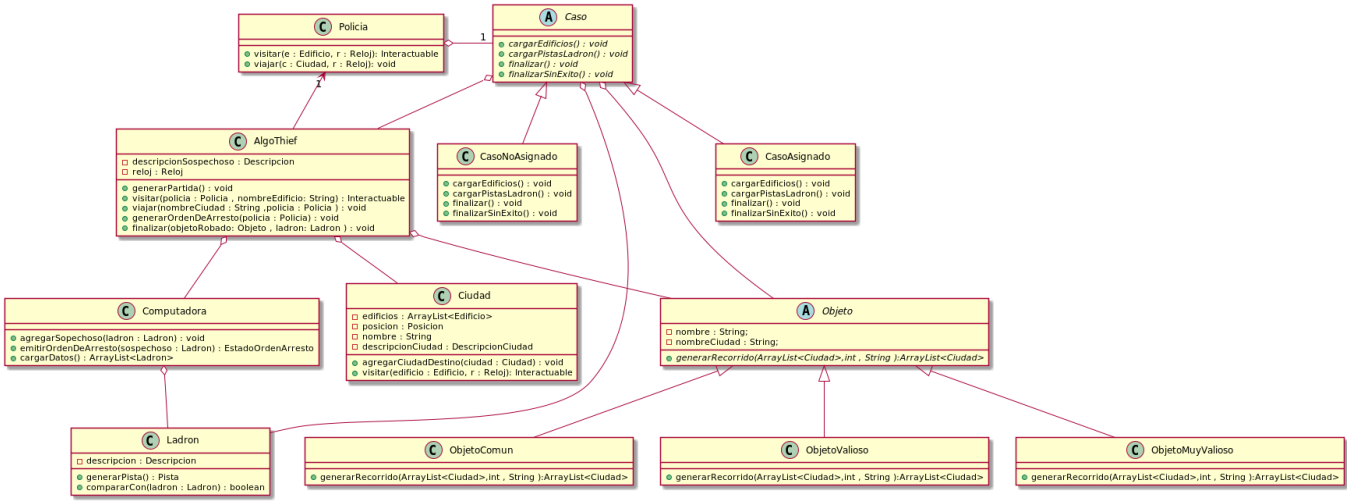

Figura 1: Diagrama de Clases General

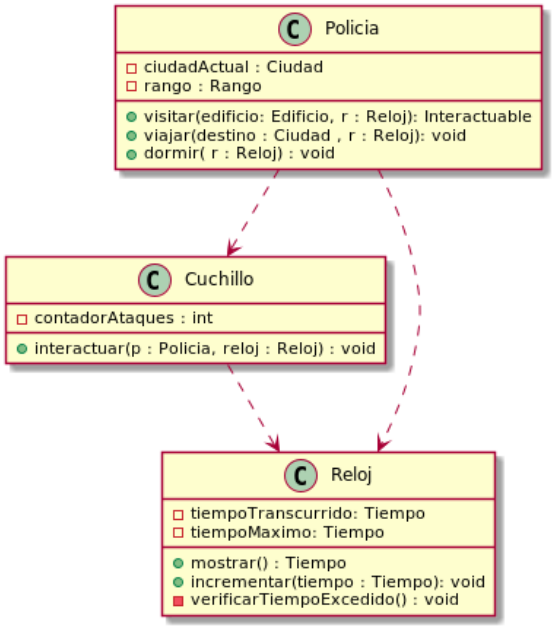

Figura 2: Diagrama de Clases que incluye cuchillo y reloj

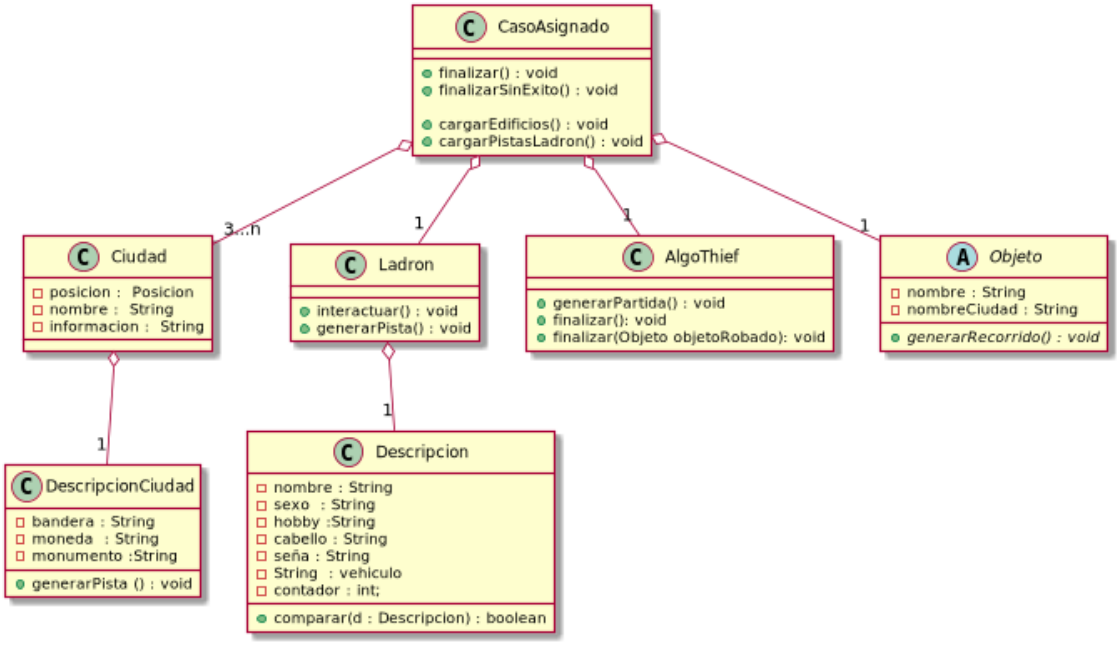

Figura 3: Diagrama de Clases de CasoAsignado

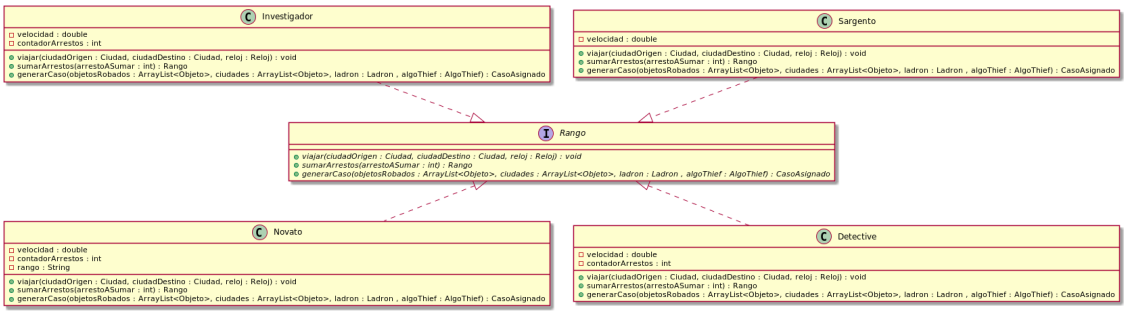

Figura 4: Diagrama de Clases del rango del policía

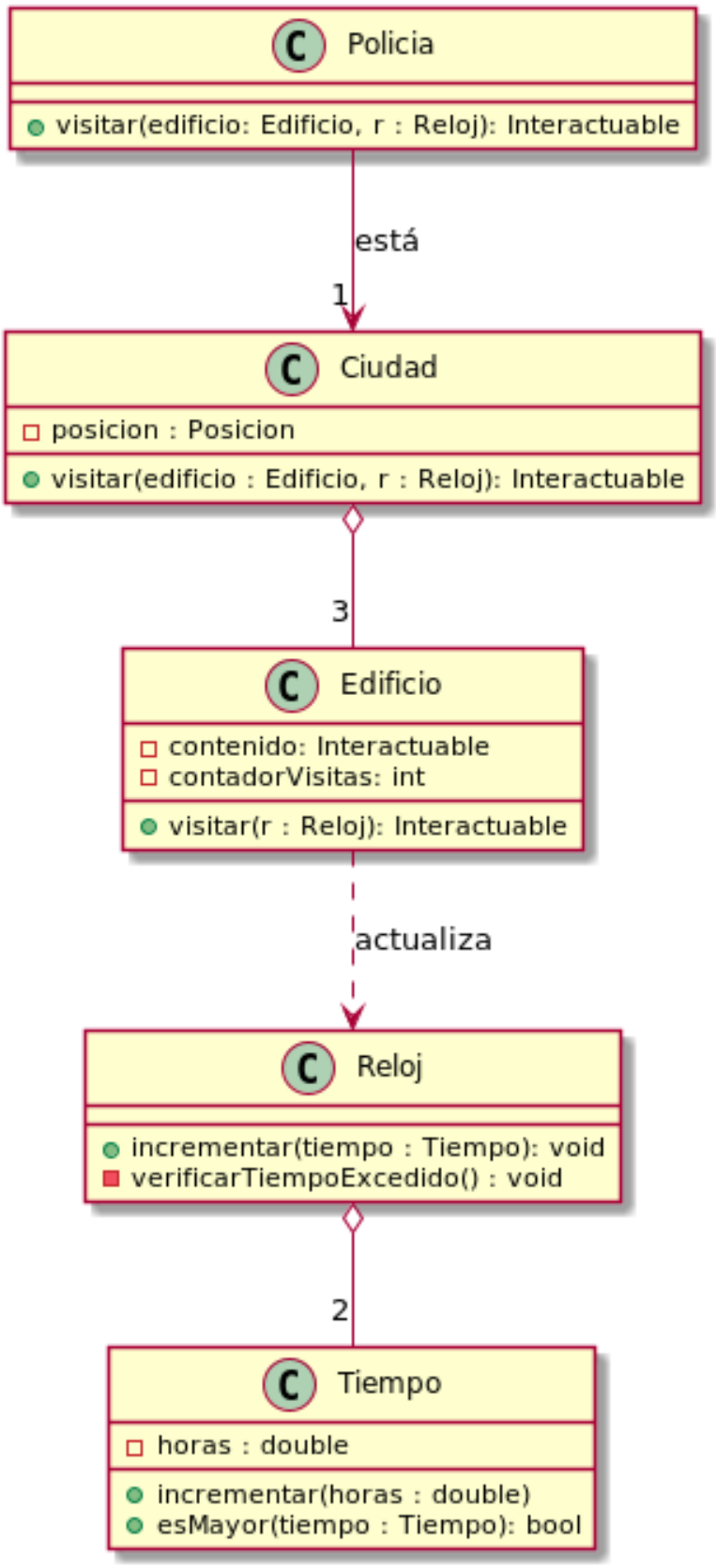

Figura 5: Diagrama de Clases que incluye reloj y tiempo

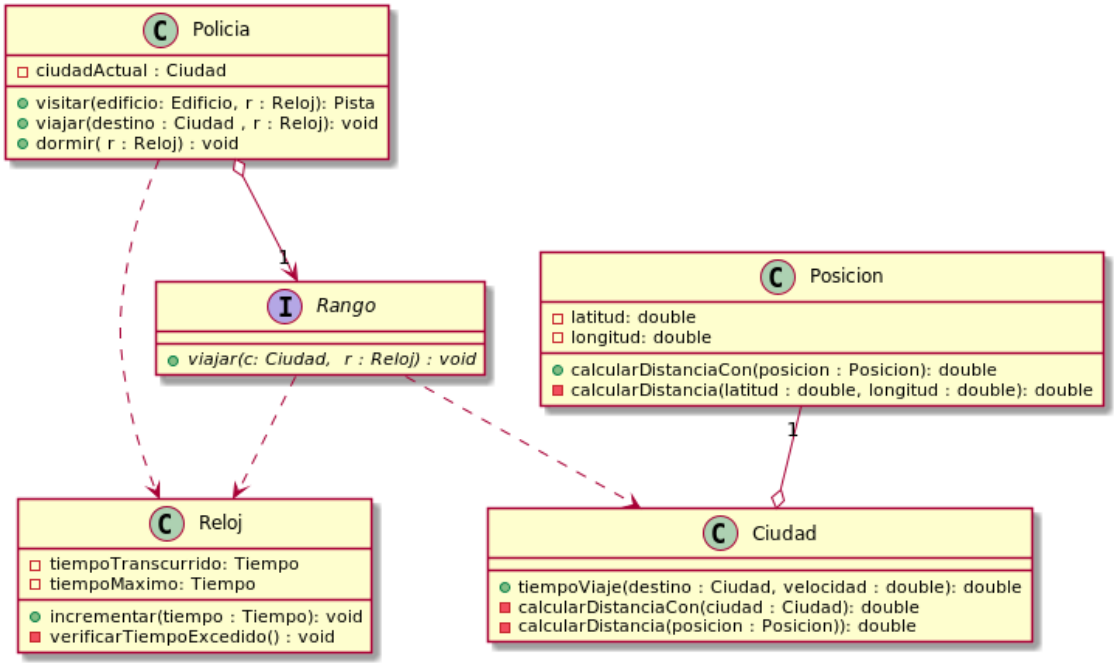

Figura 6: Diagrama de Clases que incluye posición de la ciudad

5. Detalles de implementación

5.1. Clase Algothief

Algothief es la clase que representa el juego, la clase que se comunica con el jugador y la que comienza con la ejecución del juego. Se encarga de delegar la apertura de archivos que contienen a las ciudades, las pistas y las descripciones de los ladrones para generar el caso. La clase AlgoThief delega metodos como viajar,visitar,etc a otras clases.

5.2. Clase Abstracta Caso

En la clase Caso implementamos el patrón "State"para poder definir el comportamiento cuando el policía no tiene un caso asignado o como cuando lo tiene. La clase CasoAsignado representa la entidad "partida.[en] el cual el jugador tiene que poder atrapar al ladrón.

5.3. Clase Policia

Policia es la clase que representa a la entidad jugador dentro del juego. Esta tiene la responsabilidad de visitar a un edificio o viajar a una ciudad. Al visitar un edificio para obtener una pista, delegará en la entidad Ciudad, (su ubicación actual). Para viajar a otra ciudad delegará en la entidad Rango, la cual determina su velocidad de viaje.

5.4. Clase Posición

Cada instancia de Ciudad posee una instancia de la clase Posición, la cual tiene como atributos latitud y longitud. Cada vez que el Jugador viaja desde una Ciudad a otra se calcula la distancia entre sus respectivas posiciones. Para ello se emplea la fórmula de Haversine.

5.5. Interfaz Interactuable

Implementamos esta interfaz con el objetivo de adaptar el comportamiento del contenido del edificio, el cual puede ser una pista, un cuchillo, un arma de fuego o un ladrón. Además se está cumpliendo con el principio Solid: Open-Closed porque en un futuro podemos seguir agregando nuevos objetos dentro del los edificios, y solamente se tendrían que crear nuevas clases que implementen los métodos de la interfaz Interactuable.

5.6. Clase Reloj

Es la entidad que se encarga de controlar y registrar el transcurso del tiempo durante la misión.

5.7. Clase Tiempo

Al tiempo se lo modela como una cantidad de horas discretas. Es decir, el siguiente valor de 1 hora es 2 horas, no hay 1,5 horas. Las acciones que incrementan el tiempo son visitar un edificio, viajar a una ciudad y emitir la orden de arresto. Dormir incrementa el tiempo transcurrido en 8 horas.

5.8. Clase Pista

La clase pista es la entidad que modelamos para poder representar a la información que nos ayudara a capturar al ladrón. En el modelo actual su contenido es un String y es lo que se devuelve al jugador cuando terminó de visitar un edifico. Sin embargo en un futuro ésta puede ser una imagen, un código QR, etc.

5.9. Clase Rango

La clase Rango es una interfaz implementada por las clases Novato, Detective, Investigador y Sargento. Dado que la estrategia de escape del Ladrón y la dificultad de las pistas depende del rango del policía, estas clases son las encargadas de seleccionar al objeto robado al momento de generar el caso, en función de la rareza del mismo. Se utiliza el Patrón State dado que cada rango es un estado con distinto comportamiento. Además, al completar una cierta cantidad de casos, el estado del rango evoluciona hacia uno nuevo, reflejando el hecho de que algunos estados conocen a otros y pueden cambiar por estos.

6. Diagramas de Secuencia

6.1. Diagrama para visitar un edificio

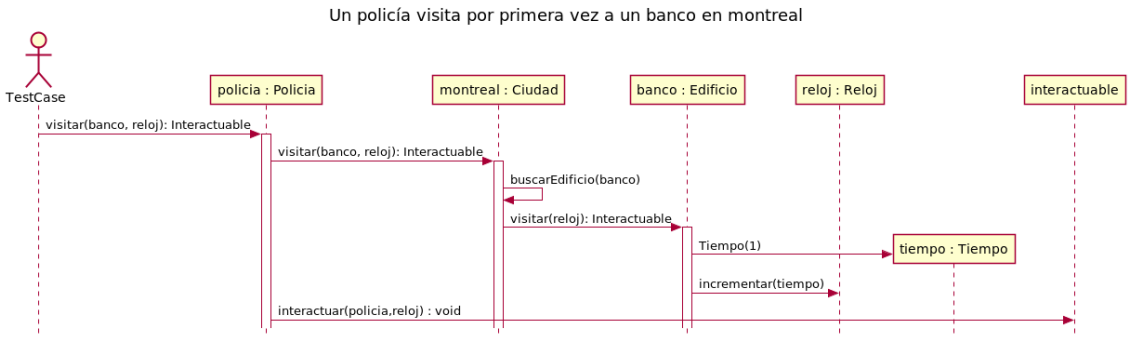

Figura 7: Un policía visita por primera vez un Banco.

6.1.1. Caso en el que el edificio contiene una pista

En este caso la pista recibe el mensaje interactuar de policía pero este no hace nada y se devuelve la pista

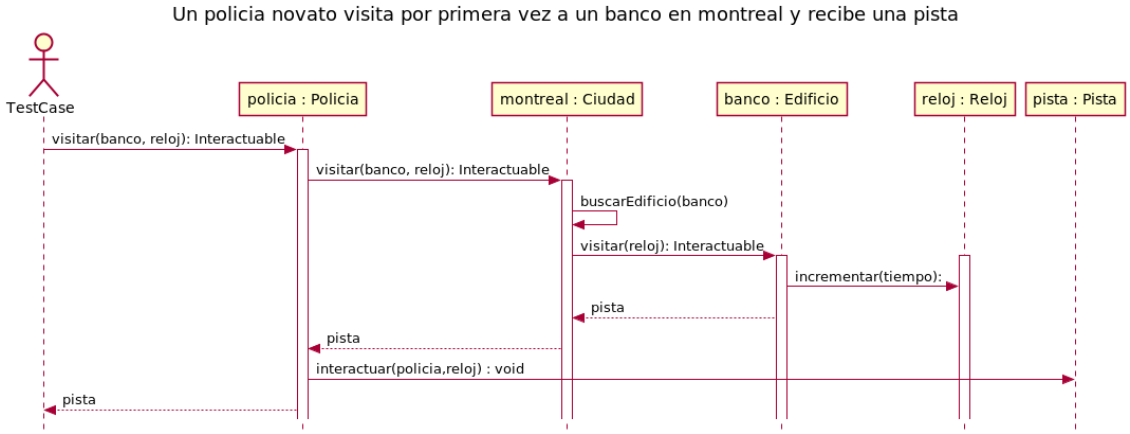

Figura 8: Un policía visita por primera vez un Banco y recibe una pista.

6.1.2. Caso en el que el edificio contiene un cuchillo

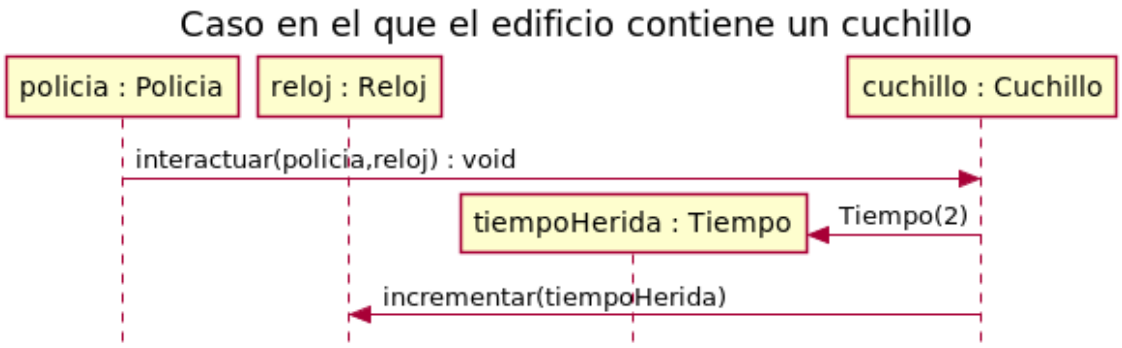

Figura 9: Se Visita un edificio con un cuchillo por primera vez.

6.1.3. Caso en el que el edificio contiene un arma de fuego

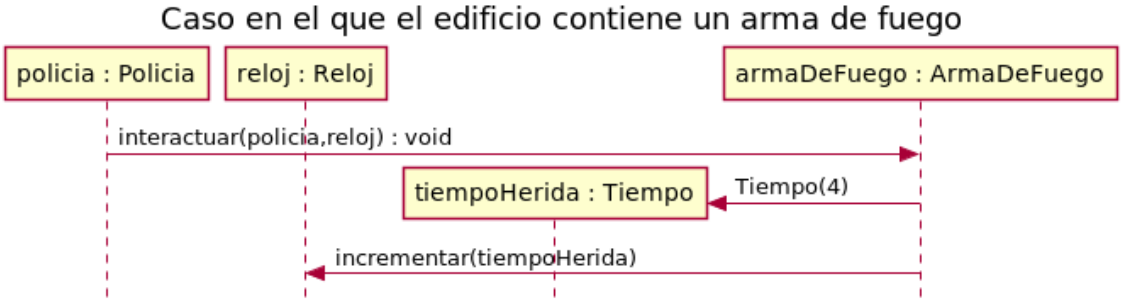

Figura 10: Se Visita un edificio con un cuchillo por primera vez.

6.1.4. Caso en el que el edificio contiene un ladrón y el policía tiene un orden de arresto emitido

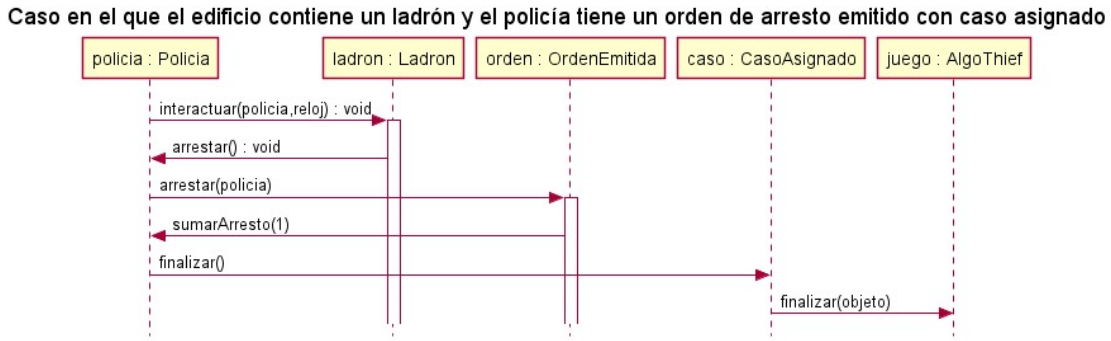

Figura 11: Se Visita un edificio con un el ladrón y el policía tiene una orden de arresto emitida.

6.1.5. Caso en el que el edificio contiene un ladrón y el policía no tiene un orden de arresto emitido

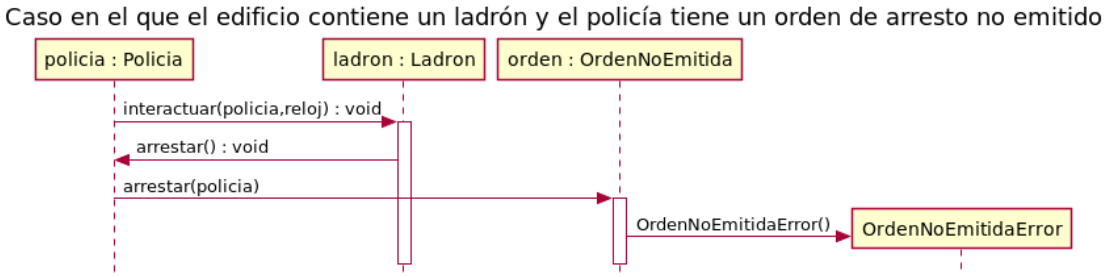

Figura 12: Se Visita un edificio con un el ladrón y el policia no tiene una orden de arresto emitida.

6.2. Diagrama para viajar a una ciudad

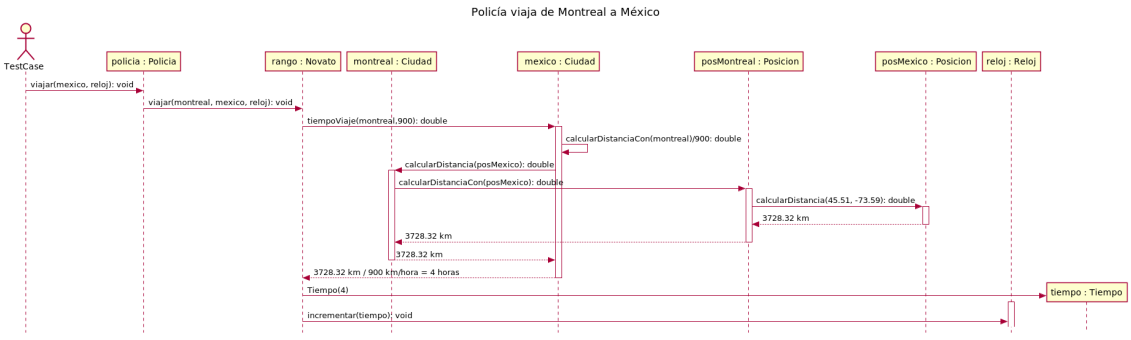

Figura 13: Un policia viaja de Montreal a México

6.3. Diagrama de secuencia para incrementar el reloj un cierto tiempo

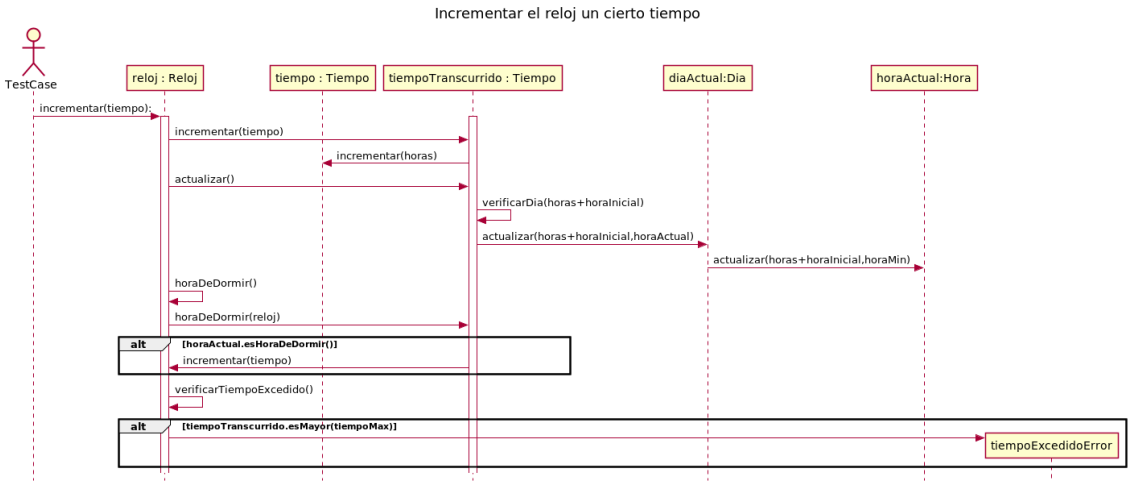

Figura 14: Diagrama de secuencia para incrementar el tiempo de un reloj

6.4. Policia emite una orden de arresto

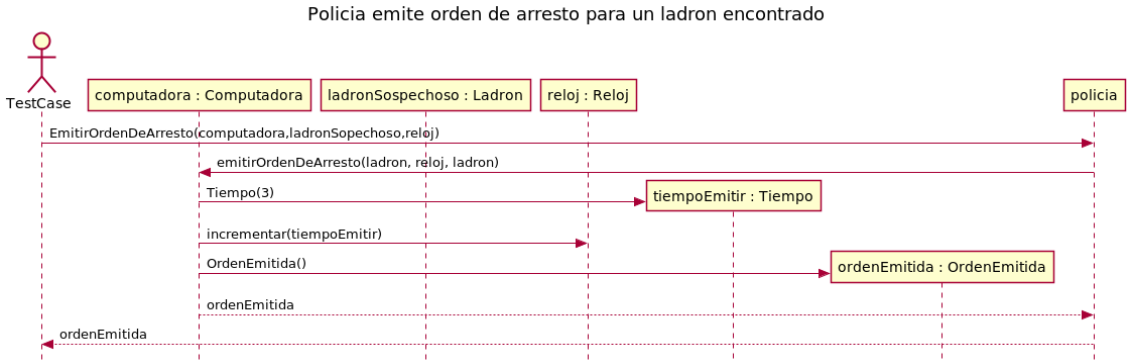

Figura 15: El policía emite una orden de arresto

7. Diagramas de Paquetes

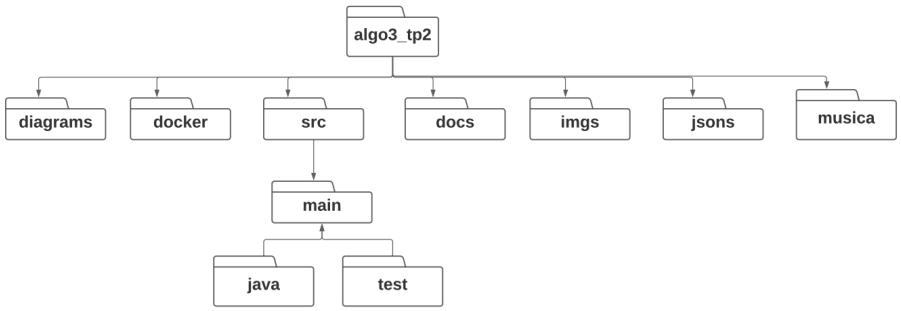

Figura 16: Diagrama de paquetes general.

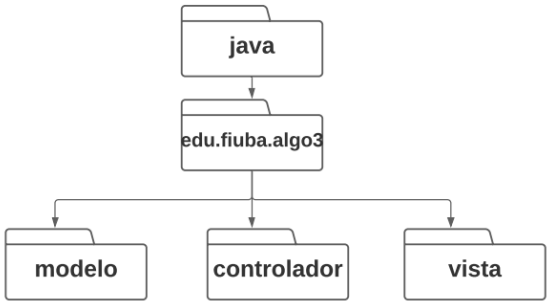

Figura 17: Diagrama de paquetes de java contemplando la estructura modelo-controlador-vista.

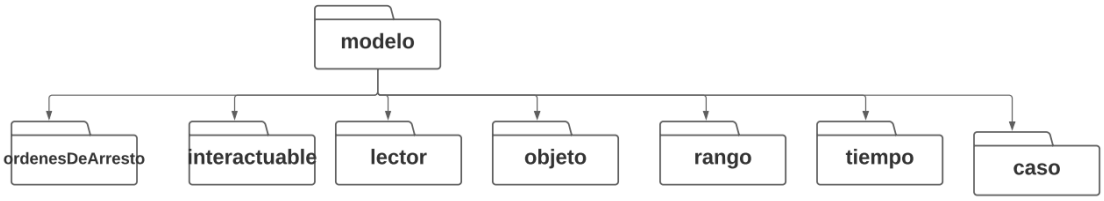

Figura 18: Diagrama de paquetes del modelo, con la agrupación de clases usadas por paquetes.

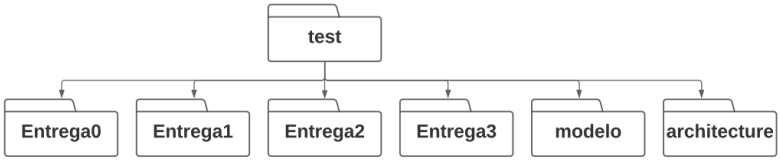

Figura 19: Diagrama de paquetes de los test

8. Diagramas de Estados

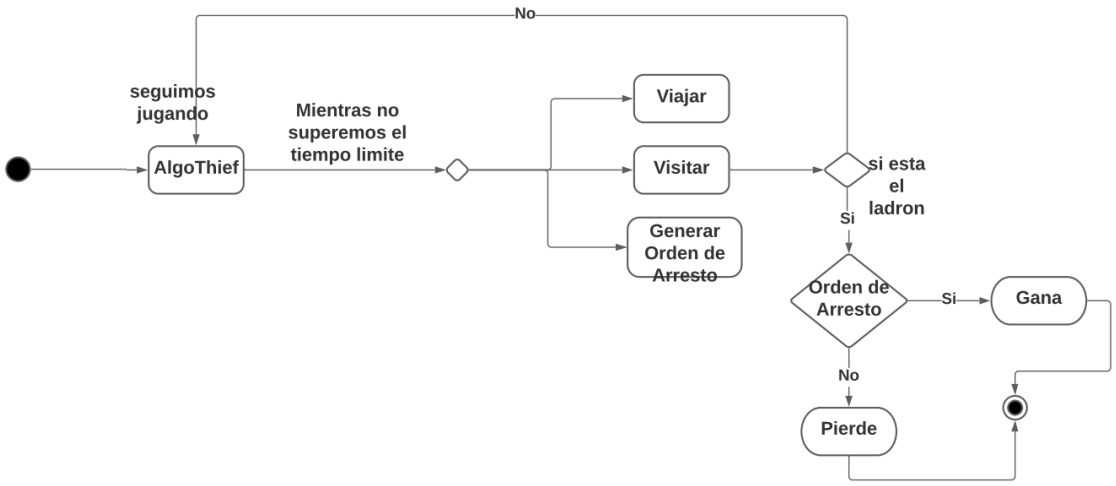

Figura 20: Diagrama de estado que representa las acciones a traves del tiempo al visitar un edificio.

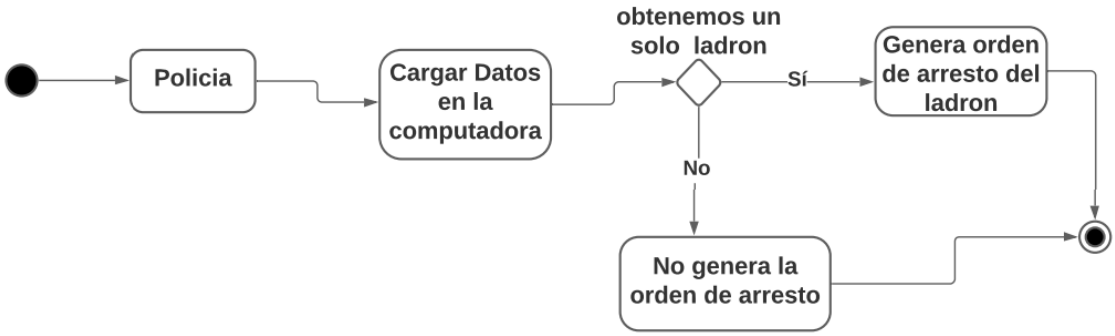

Figura 21: Diagrama de estado que representa las acciones a traves del tiempo para generar la orden de arresto.

9. Excepciones

Se consideraron lanzar excepciones en los siguientes casos:

- Cuando se supera el tiempo límite máximo para poder completar la misión (154 horas) se lanza la excepción: "TiempoExcedidoError", este se lanza al visitar un edificio, viajar o generar una orden de arresto cuyo tiempo que termina es mayor que el domingo a las 17pm.
- Cuando no se encuentra los archivos json para poder leer la informacion de las ciudades, objetos o las descripciones de los ladrones se lanzaran la excepcion: .ªrchivoNoEncontradoError".
- Cuando nos encontramos al ladron y tenemos el estado de la orden del policia como OrdenNoEmitida tambien habremso perdido , lanzando la excepcion:.ºrdenNoEmitidadError".
- Se usa la excepcion: ÇasoFinalizadoSinExitoErrorçuando atrapamos la excepcion “TiempoExcedidoError", usamos la excepcion “CasoFinalizadoSinExitoError"para activar los controladores y mostrar las siguientes escenas en la vista.