Aplicación SmartHunter

Programación concurrente y Distribuída

Curso 2011-12

Miguel Telleria, Laura Barros, J.M. Drake

`telleriam AT unican.es`

Computadores y Tiempo Real

http://www.ctr.unican.es

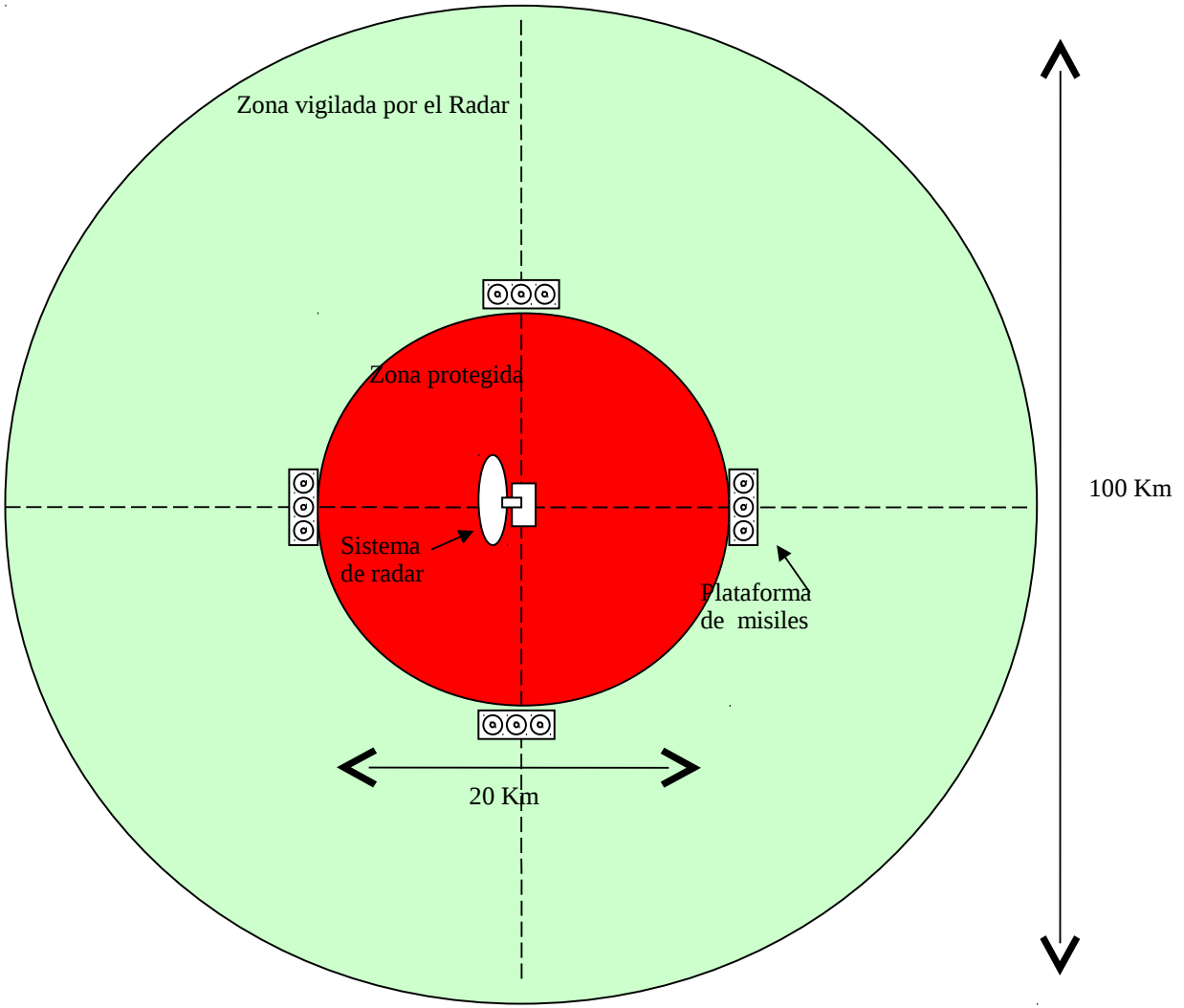
Contenido

- Aplicación SmartHunter

- Particionados para las prácticas

Aplicación SpeedFalcon

Esquema gráfico



Principales objetos

- **DefenseBase**: Coordinador de todo el sistema
  - Mantiene las lista de jets activos y amenazantes.
- **AirRaid** (Interfaz RadarSystem): Provee escaneos de Jets
  - También genera los Jets de manera aleatoria
  - También lleva la GUI
- **JetData**: Operaciones sobre Jets amenazantes.
  - Calcula la posición del jet según la dirección y el tiempo
  - Decide si tiene que asignar un misil al jet
- **4 MissileBattery** (N, S, E, O)
  - Crean e interaccionan con los misiles
- **20 Missiles** (5 por cada MissilBattery)
  - Accionan un motor (MissilHardware) cambiando su orientación
- **20 SpeedFalcon** (Interfaz MissilHardware)
  - Hacen mover al misil

GUI

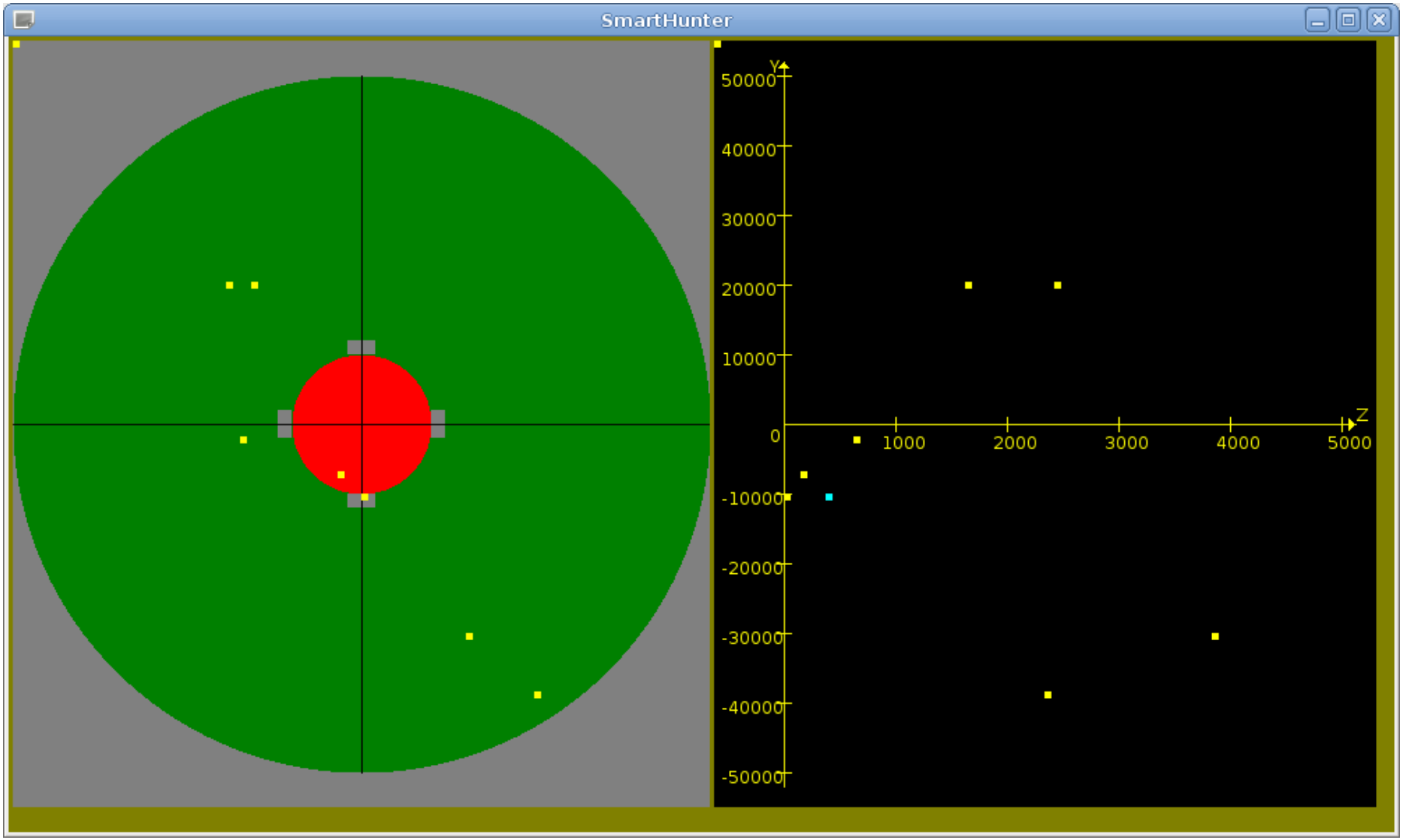
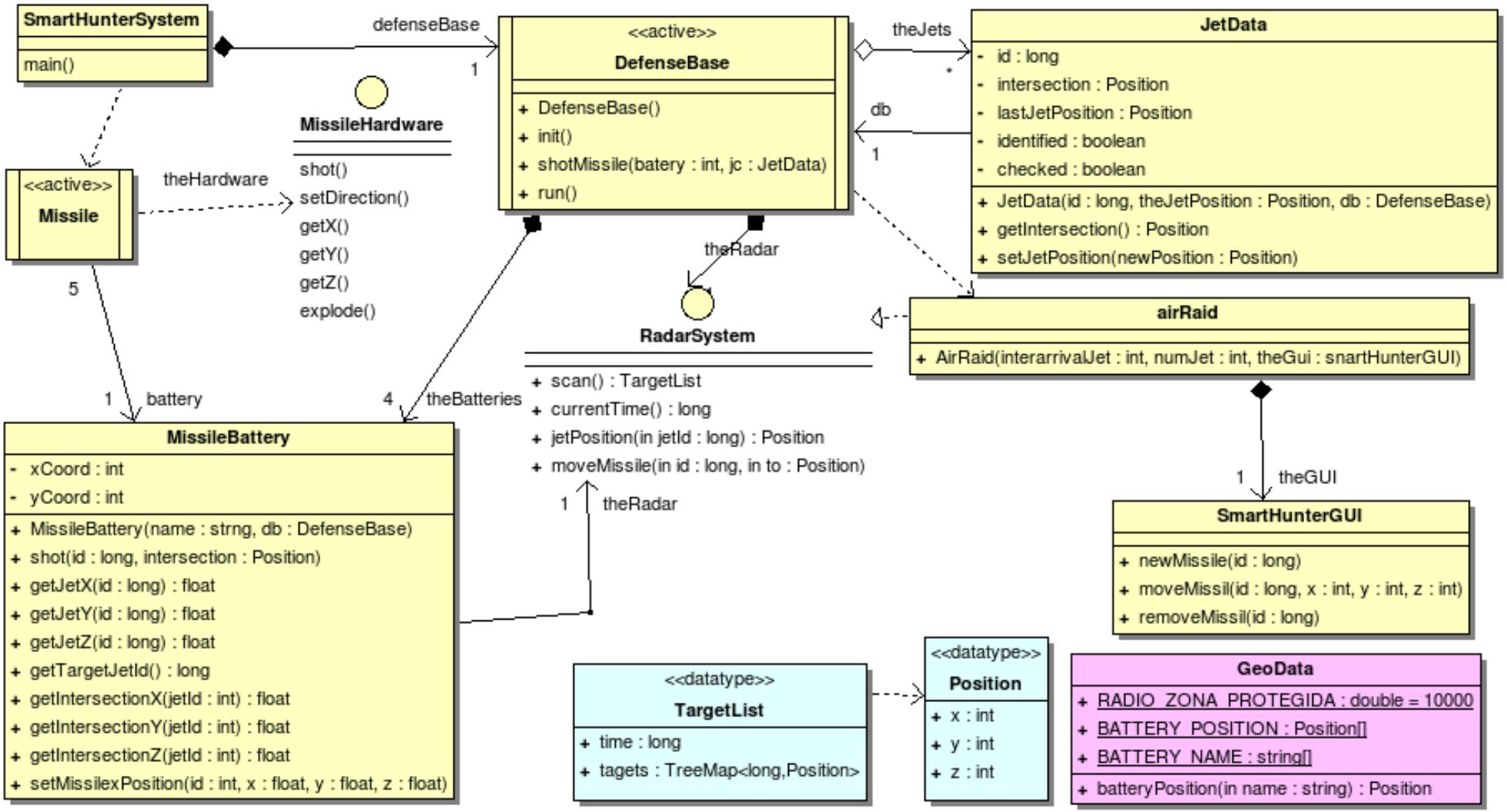
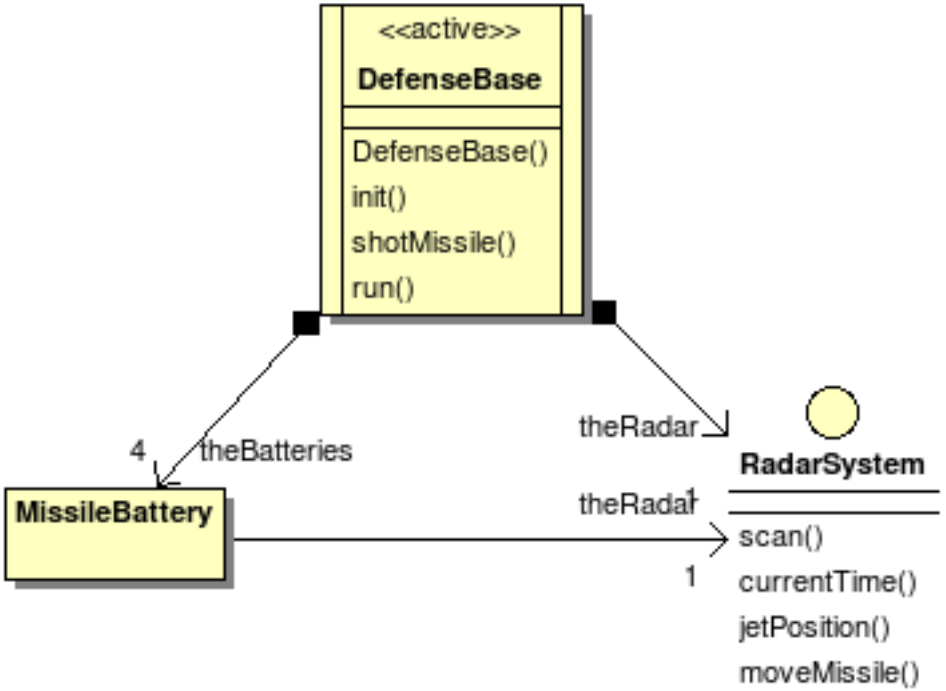

SmartHunter
Y
50000
40000
30000
20000
10000
0
-10000
-20000
-30000
-40000
-50000
Z
1000
2000
3000
4000
5000

Diagrama de clases

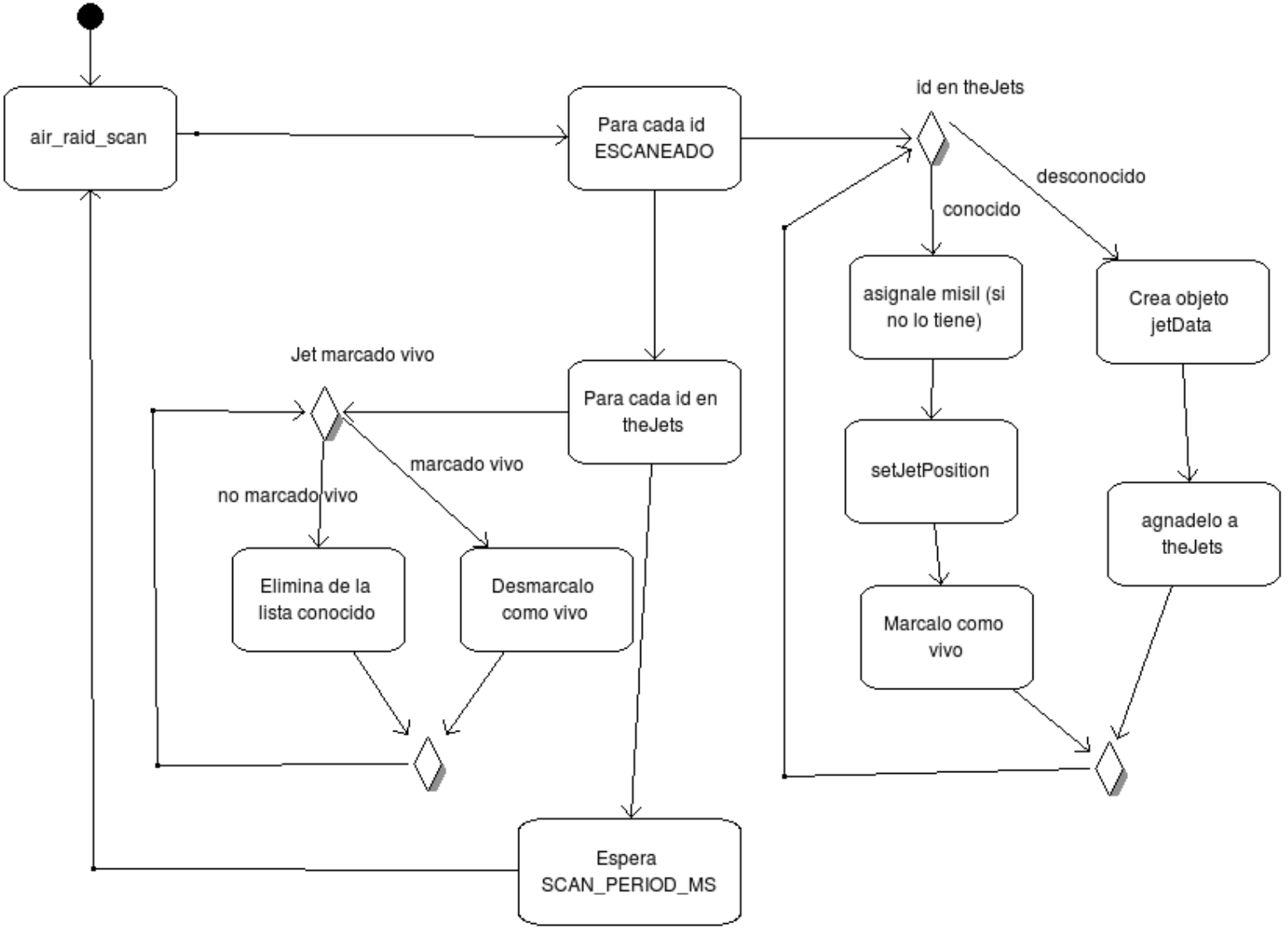


Clase DefenseBase

- Misión
  - Recoger periódicamente escaneos del radar
  - Mantener la lista de jets conocidos
  - Es un thread
- Atributos
  - 4 MissileBattery
  - Interfaz RadarSystem
  - theJetList: JetData
- Métodos
  - **run()**:
    - Su ciclo de vida
  - **shotMissile(battery_nr, jetData)**
    - Ordena a una MissileBattery lanzar un misil

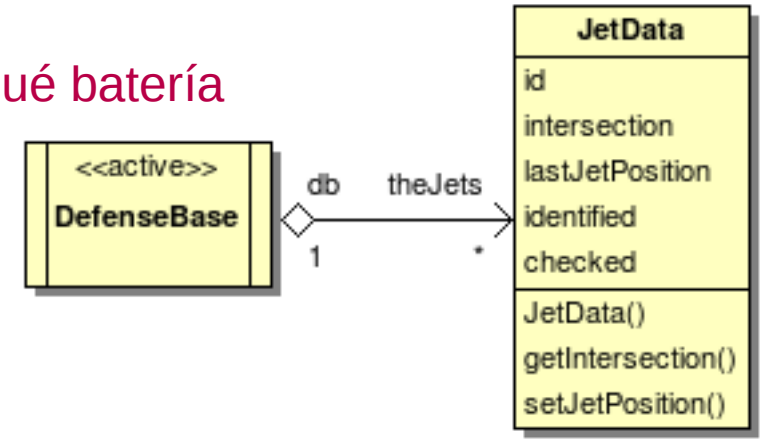


Ciclo de vida de DefenseBase



Clase JetData

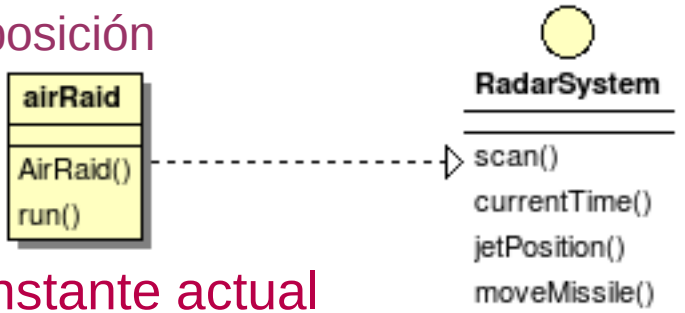
- Misión: Llevar datos de cada jet enemigo
  - Discierne si el jet ataca a la zona o no
  - Ordena el lanzamiento de un misil y desde qué batería
  - Calcula el punto de intersección



- Atributos
  - **id** (long): id del jet
  - **db**: DefenseBase
  - **intersection** (posicion): Punto de intersección con el cilindro rojo
  - **identified** (boolean): Si ya está analizada su trayectoria y asignado misil
  - **checked** (boolean): Si sigue siendo confirmado por el radar
- Métodos
  - **setJetPosition**(): Actualiza la posición y los datos del jet
  - **getIntersection**(): Devuelve el punto de intersección con el cilindro

Clase AirRaid

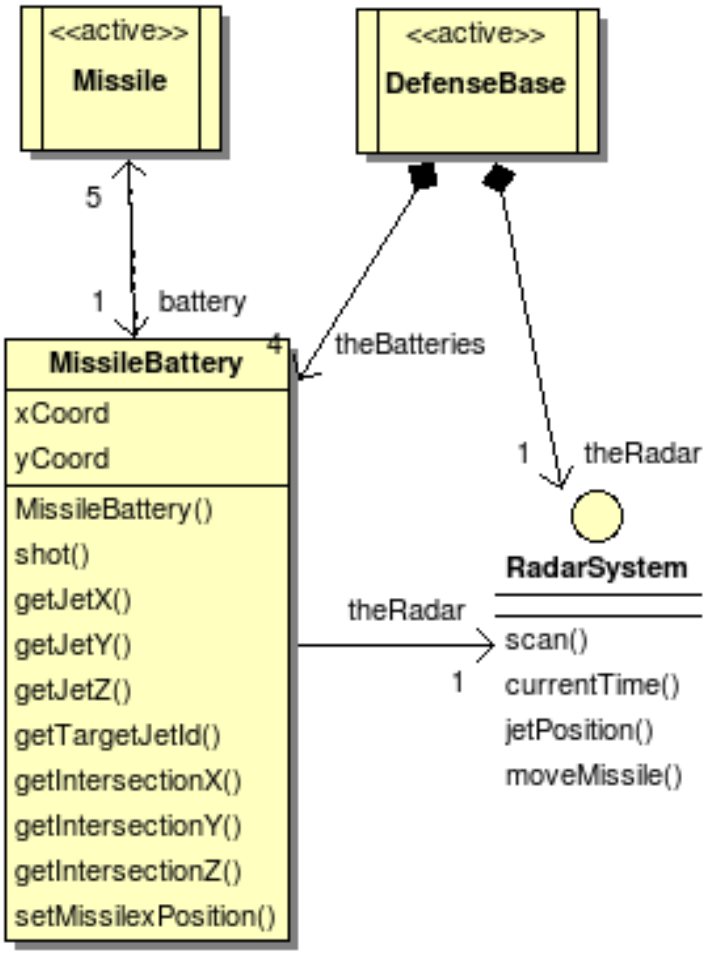
- Misión
  - Por un lado implementa la interfaz RadarSystem
    - Da un escaneo periódico de los jets existentes
    - Informa de la posición en tiempo real de cualquier jet
  - Por otro lado
    - Genera aleatoriamente los jet's y actualiza su posición
    - Maneja la GUI
- Métodos
  - **scan**(): Hace un barrido de los misiles en el instante actual
  - **jetPosition**(): Devuelve la posición de un jet
  - **moveMissile**(): Actualiza en la GUI la posición del misil
  - **run**(): Ciclo de creación de jets



Clase MissileBattery

- Misión: Comunicación entre DefenseBase, radar y los misiles
  - La DefenseBase le asigna Jets y ptos de intersección
  - Los misiles le preguntan si hay algún target disponible
  - Informa a los misiles (via radar) de la posición de los jets
  - Informa a los misiles de su pto de intersección con los jets
- Atributos
  - **radar**: Referencia al radar
  - **unAssignedTargets**: Lista de Jetid's sin asignar
  - **intersectionMap**: Hash que encuentra el pto de intersección a.p.d. JetId
- Métodos
  - **shot**(): Usado por defenseBase para asignar un jet
  - **getTargetId**(): Usado por el misil para pedir un objetivo
  - **getIntersection**(), **getJet**(): El misil pide información de su objetivo

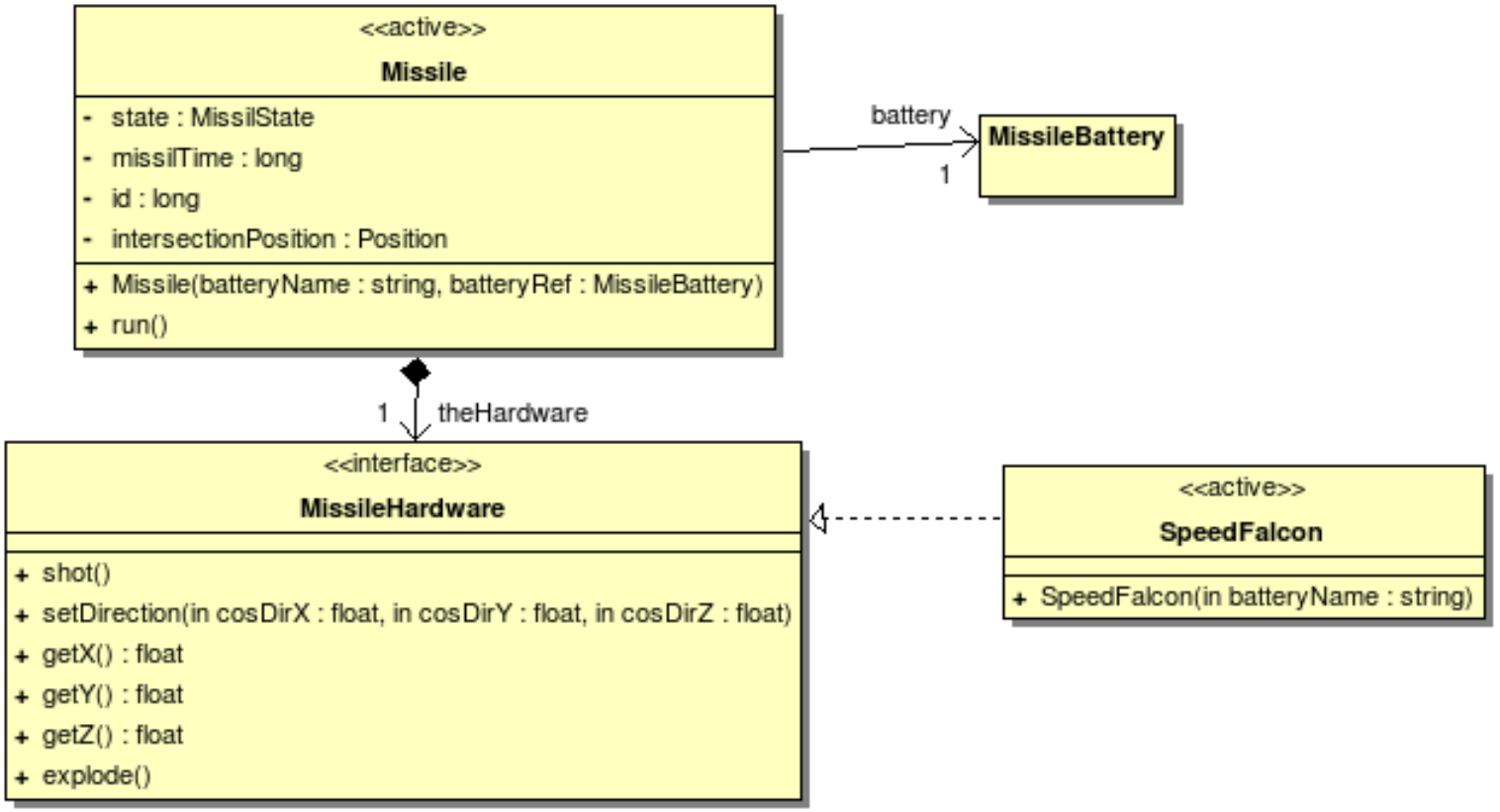
MissileBattery con DefenseBase, Radar y Misil



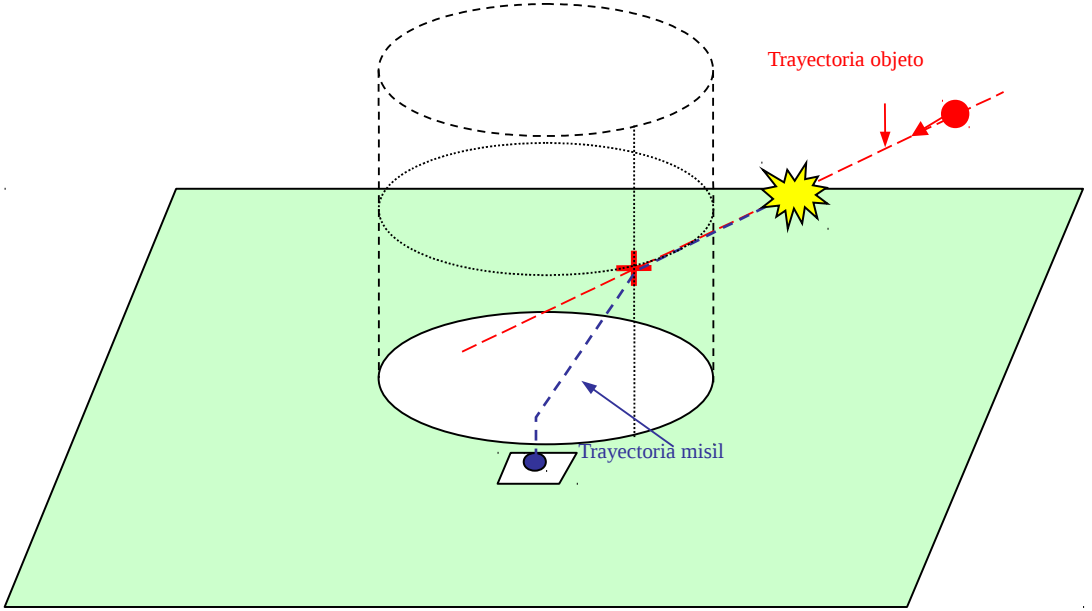
Clase Misil

- Misión: Comandar el motor del misil
  - Calcula el ángulo de trayectoria
  - Manda al motor explotar
  - Es un thread
- Atributos
  - **battery**: La bateria de la que depende
  - **theHardware**: El motor (SpeedFalcon) que comanda
  - **currentPosition**: Posición del misil (obtenida del speedFalcon)
  - **jetPosition**: Posición del jet
  - **intersectionPosition**: Pto de intersección con el cilindro
- Métodos
  - **getPosition**(): Pide la posición del misil al MisilHardware
  - **distancia**(): Calcula la distancia que le queda para llegar a un punto
  - **getJetPosition**(), **getInstersectionPosition**(): Pregunta a MissilleBattery

Missil con MissilHardware y SpeedFalcon

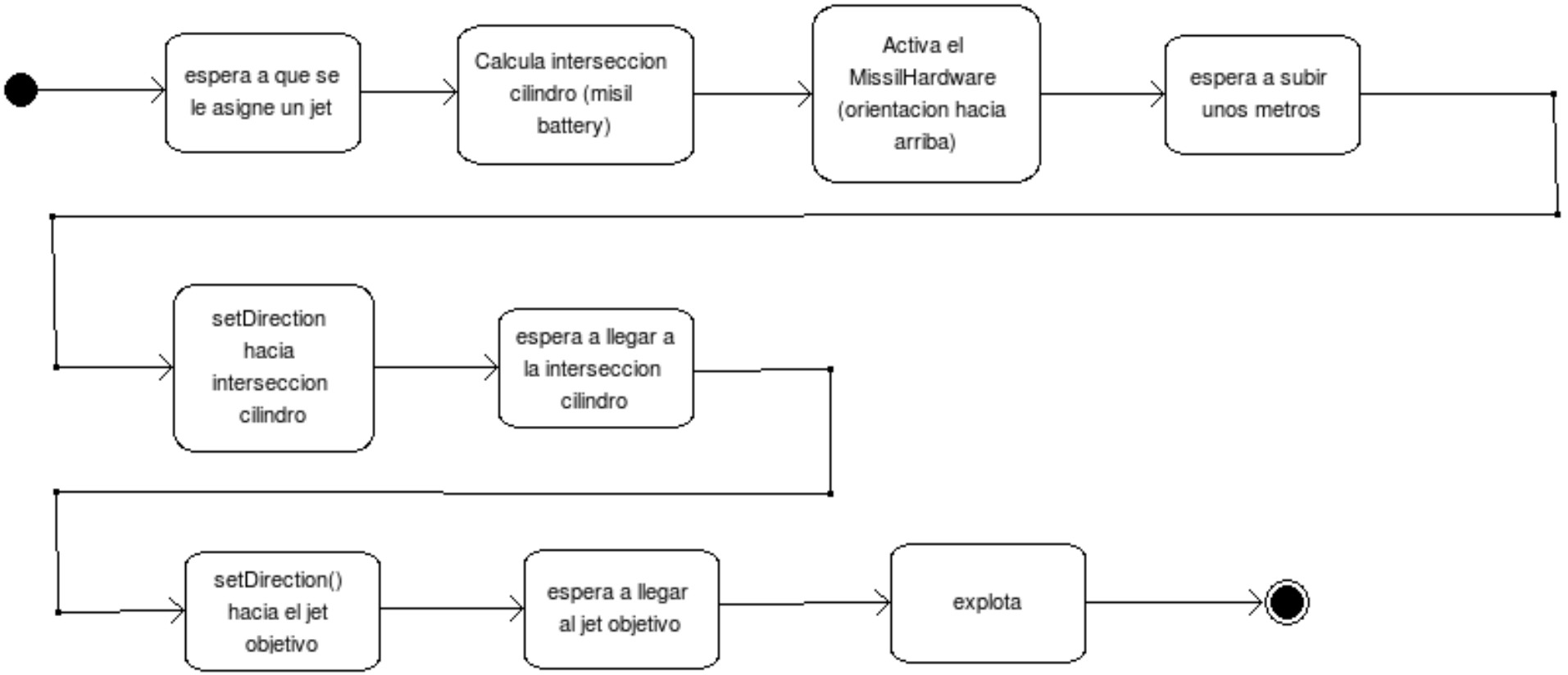


Trayectoria del Misil



- El misil hace su trayectoria en 3 fases
  - Levantamiento vertical inicial
  - Llega al punto de intersección con el misil
  - Encara al jet y va hacia él

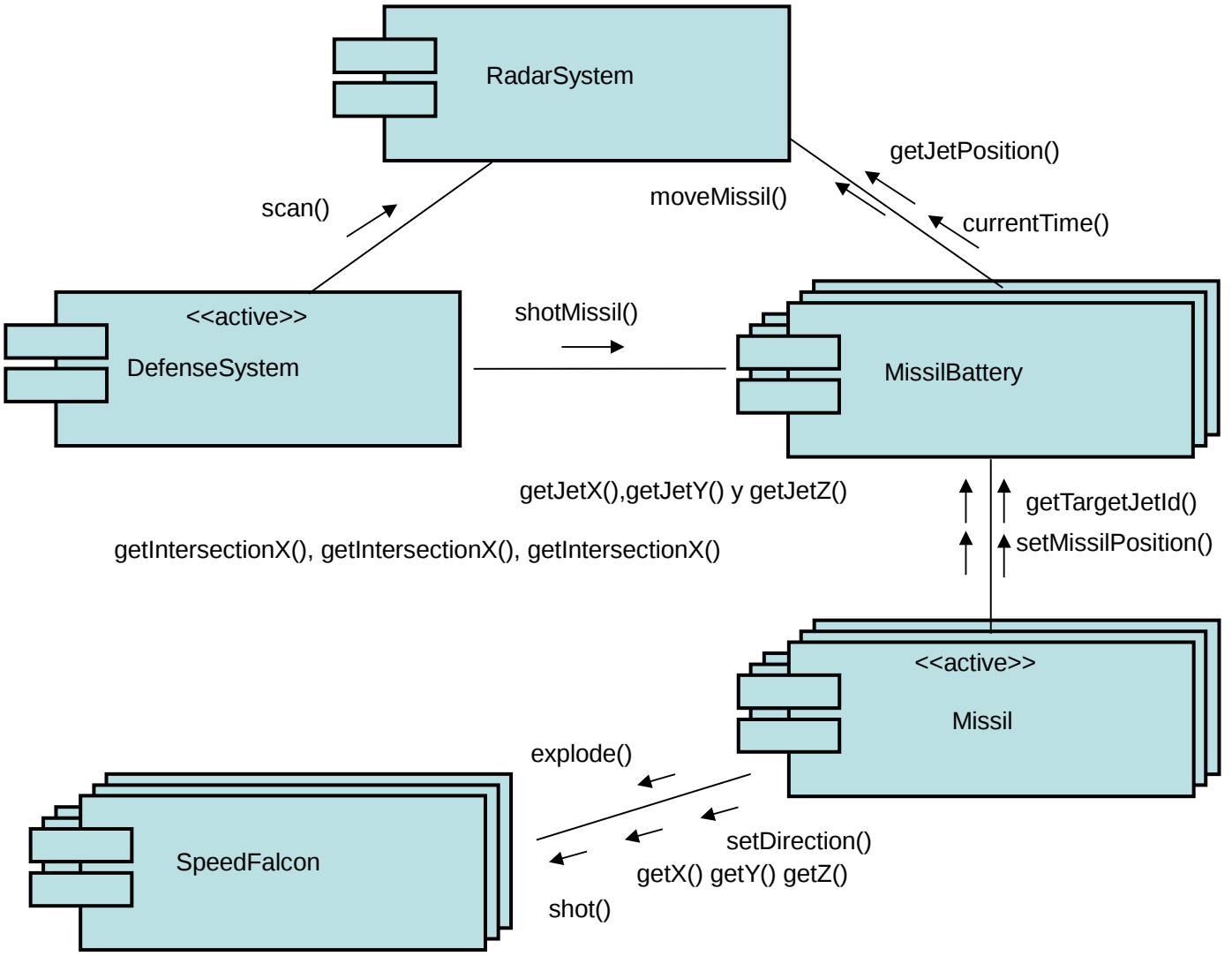
Diagrama de actividad del misil



Clase SpeedFalcon

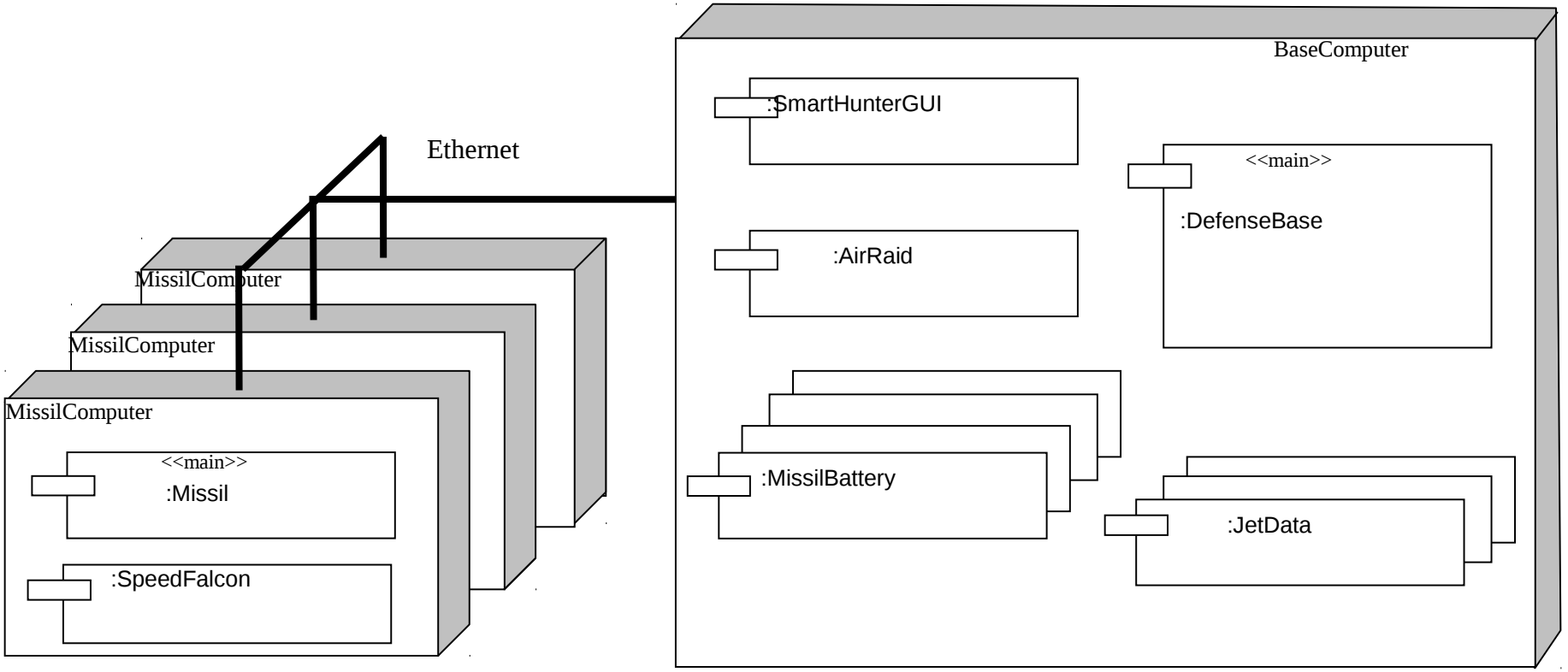
- Mision: Implementa la interfaz MissilHardware (motor del misil)
  - Es un thread que actualiza la posición del misil en línea recta
  - Cambia la dirección del misil
  - Explota
- Atributos
  - posicion (x, y, z): Posición actual del misil
  - Coseno director (ux, uy, uz): Dirección de avance del misil
- Métodos
  - **shot**(): Arranca
  - **getX**(), **getY**(), **getZ**(): Obtén la posición actual
  - **setDirection**(): Cambia la dirección de avance
  - **explode**(): Explota

Uniendo todo



Particionado de las prácticas

Miércoles 23 y Jueves 1



Jueves 24 y Miércoles 30