Heist to the museum*

A group of *M thieves* plans to steal the paintings in exhibition at a museum in Aveiro. The paintings are in display in *N* different rooms, each having Q_i paintings hanging on the walls, with $i = 0, 1, \ldots, N\text{-}1$.

The operation is directed by the *master thief* who organizes her companions, standing in queue while waiting to be summoned, in assault parties of *K* elements and assigns to each party a target room in the museum. Different parties may proceed to different rooms at the same time. In the end, the master thief calls a meeting and informs the rest of the thieves about the sack earnings.

To prevent detection by the museum guards, the thieves forming a party crawl in line, as fast and as silently as they possibly can, along a previously established path between the outside gathering site and the target room. They must ensure that contiguous elements in the crawling line are not separated by a distance larger than a previously fixed value. Upon arrival at a museum room, each party member looks for a painting still hanging on the wall and, if there is one, he takes it down, detaches the canvas from the framing, rolls it over and inserts it in a cylinder container he carries on his back. He then prepares to leave the room. The way out is the same as the way in and the crawling procedure adopted before is adopted again. When a thief reaches the outside location where the master thief is hiding, he takes the canvas out of the cylinder and hands it to the master thief who stores it in the back of a van, or tells her he is coming empty-handed. Since the master thief does not know beforehand how many paintings are hanging in the walls of each room, she goes on promoting incursions to the same room until she is sure the room is empty.

The crawling movement of party G_j, with $j = 0, 1, \ldots, (M\text{-}1)/K\text{-}1$, requires successive increments of position that obey the following rules

- the ingoing movement (from the outside gathering site to the museum room) is performed by taking positive position increments and the outgoing movement (from the museum room to the outside gathering site) is performed by taking negative position increments
- the distance between the outside gathering site and the museum room i is D_i length units, with $i = 0, 1, \ldots, N\text{-}1$
- the ingoing movement only starts when all group members have been selected and are ready to proceed, the outgoing movement only starts when all group members have taken a canvass or are empty-handed, because no more paintings are hanging in the room walls
- the thieves in a party crawl in line, can overtake one another, but can never stay side by side, nor be separated by a distance larger than *S* length units
- at each iteration step, the thief t_j, with $j = 1, \ldots, M\text{-}1$, can change his position from 1 to MD_{t_j} length units, always moving as fast as he possibly can without violating the constraints imposed by the previous rule
- the maximum displacement, MD_{t_j}, is specific to each thief t_j, the thieves are not all equal, some are more agile and faster than others.

Assume there are 7 thieves in the whole, master included, the maximum displacement of the ordinary thieves is a random number between 2 and 6, the number of exhibition rooms having paintings in display is 5 with random distances to the outside concentration site between 15 and 30, the number of paintings hanging in each room is a random number between 8 and 16 and that the assault parties have 3 elements. Also assume that the maximum separation limit between thieves crawling in line is 3 length units.

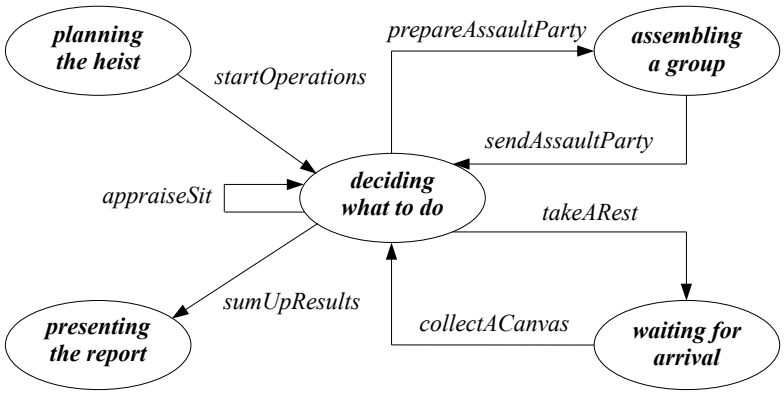
Write a simulation of the life cycle of the thieves using one of the models for *thread* communication and synchronization which have been studied: monitors or semaphores and shared memory.

One aims for a distributed solution with multiple information sharing regions that has to be written in Java, run in Linux and terminate. A *logging* file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

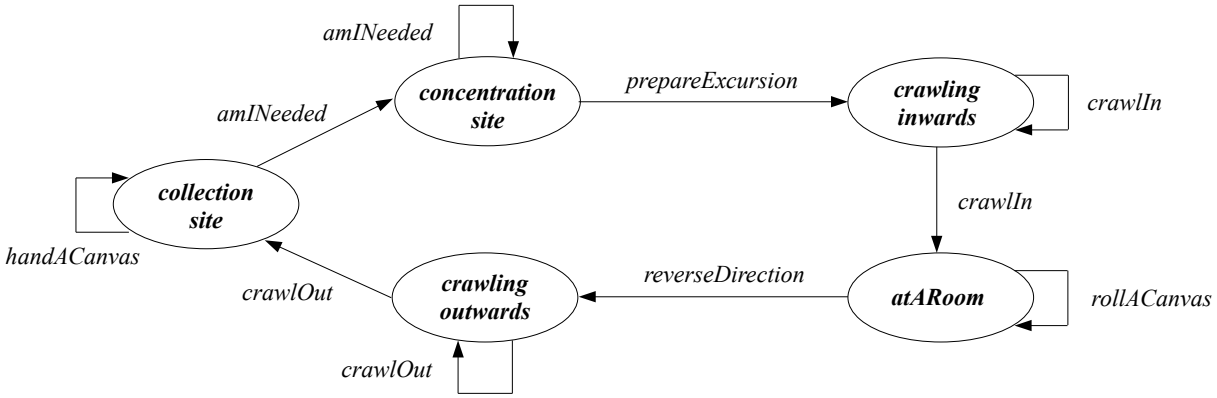
* Concept by Pedro Mariano

Suggestion to solution

Master thief life cycle



Ordinary thieves life cycle



Characterization of the interaction

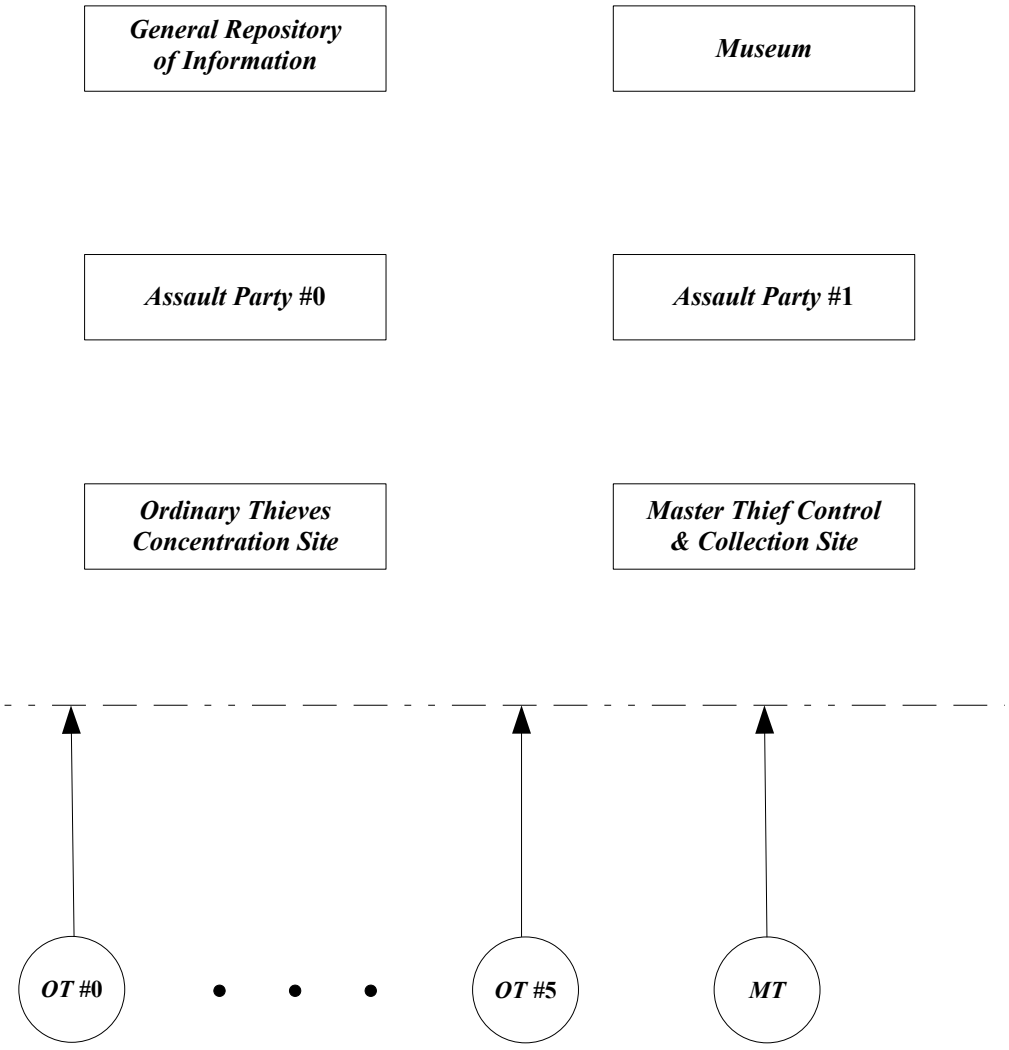
Master thief

PLANNING_THE_HEIST – initial state (*transitional*)
DECIDING_WHAT_TO_DO – transitional state with eventual waiting
master thief proceeds if the next operation is *takeARest* and blocks if it is one of the other two and there is not a sufficient number of ordinary thieves available (the totality for *sumUpResults* and enough to create an assault party for *prepareAssaultParty*)
when master thief blocks, she is waken up by the operation *amINeeded* of an ordinary thief
ASSEMBLING_A_GROUP – blocking state
master thief is waken up by the operation *prepareExcursion* of the last of the ordinary thieves to join the party
WAITING_FOR_GROUP_ARRIVAL – blocking state
master thief is waken up by the operation *handACanvas* of one of the assault party members returning from the museum
PRESENTING_THE_REPORT – final state

Ordinary thieves

CONCENTRATION_SITE – blocking state (*initial / final state*)
the ordinary thief is waken up by one of the following operations of the master thief: *prepareAssaultParty*, during heist operations, or *sumUpResults*, at the end of the heist
CRAWLING_INWARDS – transitional state with eventual waiting
for the crawling in movement to start, the first party member is waken up by the operation *sendAssaultParty* of master thief
the ordinary thief proceeds until the target room at the museum is reached and blocks if he can not generate a new increment of position (before blocking, he wakes up the fellow party member that is just behind him in the crawling queue, or the first one still crawling, if he is the last)
when blocking occurs, the ordinary thief is waken up by the operation of *crawlIn* of a fellow party member
AT_A_ROOM – transitional state
CRAWLING_OUTWARDS – transitional state with eventual waiting
for the crawling out movement to start, the first party member is waken up by the operation *reverseDirection* of the last party member to decide to leave the room
the ordinary thief proceeds until he reaches the outside gathering site and blocks if he can not generate a new increment of position (before blocking, he wakes up the fellow party member that is just behind him in the crawling queue, or the first one still crawling, if he is the last)
when blocking occurs, the ordinary thief is waken up by the operation of *crawlOut* of a fellow party member
COLLECTION_SITE – blocking state
the ordinary thief is waken up by the operation of the master thief: *collectACanvas* when she receives the canvas taken from the museum, or when she gets the report that the room is already empty

Information sharing regions



Guidelines for solution implementation

1. Characterize interaction at the state level.

2. Specify the life cycle and internal properties of each of the *intervening entities*.

3. Specify for each *information sharing region* the internal data structure, the operations which will be invoked, identifying their signature, functionality and who is the calling entity, and the synchronization points.

4. Sketch the *interaction diagram* which describes in a compact, but precise, way the dynamics of your solution. Go back to steps 1 and 2 until you are satisfied the description is correct.

5. Proceed to its coding in Java as specific reference data types.

6. Write the application main program which should instantiate the different *information sharing regions* and the different *intervening entities*, then start the different entities and finally wait for their termination.

7. Validate your solution by taking several runs and checking for each, through the detailed inspection of the logging file, that the output data is indeed correct.