Test2 – Kevin and the burglar

The story is about a burglar who wants to break in a house. Luckily Kevin is at home and sets trap for him.

#For mark 2:

There are 3 subtasks to do for 2:

- A protected printing (`Printer`);
- A task (`Door`);
- The task (`Burglar`).

The `Printer` protected should have a `Print`, procedure which is used for printing the events.

The task `Door` should have two entry point: `Open` and `Close`, which can be called many times. The fact that the door is open or closed should be saved, and use conditional entries accordingly.

The `Burglar` should wait until it gets dark (`delay 1.0`), opens the door, waits 3 seconds then closes the door and leaves (terminates).

 # For mark 3:

The burglar did not expect to meet someone in the house. Kevin sets trap at the door which is activated after few seconds when the door opens.

New elements:

* `Trap` tasktype;

* `Hit` entry point in Burglar;

* Extension of door opening;

* Random generator in a protected.

Create a task type (`Trap`), which asks for a random number (float between 0 and 4), prints this number and waits as long time, after that calls the `Hit` entry point for which is willing to wait 0.01 seconds.

The burglar has to be extended by selective waiting, during the 3 seconds time he is in the house he can get a hit call, but after the 3 seconds he will close the door and terminates.

At the opening of the door creates dynamically (using `access`) a trap task.

The random generator `Safe_Random` is a protected, which has a (`Generate`) procedure which randomly generates a `Duration` value (float between 0.0 and 4.0).

#For mark 4:

The burglar did not observe that there are more doors at the house.

The `House` should be protected, while the `Door` a task type, with an integer discriminant (`ID`) a value from 1 to 5. In the house store pointers to doors, which are created by an `Init` procedure. The init by a for loop creates the pointers in an array such that the i-th door has discriminant i.

The house can be accessed by a `Get_Door` procedure which has an out parameter – a pointer to a door. You can randomly generate a door id inside the procedure.

Modify the burglar accordingly. The main program should call the `House.Init`.

 # For mark 5:

Luckily Kevin is at home and sets traps for the burglar.

New elements:

* `Kevin`, a task;

* The door task type has a `Set_Trap` entry point;

* The house has a new procedure:`Get_Door(Door_ID, out Door_Access)`, which provides not a random door, but the one with ID number.

Kevin is a task who with a loop on doors calls the `Set_Trap` entry point, at each door this trap setting is of 1 second. He has to call `House.Get_Door`, overloaded version with the `Door_ID` in parameter, then sets the trap (`Door.Set_Trap`).

Kevin has an entry point `Catch`, which can be called while he is setting the traps. After sets the door's trap he waits to be caught, then terminates happily. If the burglar does not get hit in 3.0 seconds, then he is catching Kevin (`Kevin.Catch`).

The door of course has to be modified so that beside open and close it has a `Set_Trap` entry point too. The door should set a trap only after this entry was called.

The burgler should start after 3.5 seconds so that Kevin has time to be prepared.