Main : MainRabbit : call the main of RabbitGrassModel

RabbitGrassModel : Use to make the model of the simulation. The main is just launching the model that is implement by normal repast method :
variable :

_schedule : use to make the grass and rabbits do what they have to do
_grid : representation of the grid display on the screen
_displaySurf : surface used to display the simulation
_rabbit : list of all the rabbits alive
_graph : graph of the number of rabbits and grasse in function of the time
From there all the variables are input of the simulation that you can change before its beginning in the program, so for that they all have the get and set method for themselves :
_rabbit : number of rabbit at the beginning of the simulation
_birthThreshold : threshold after which a rabbit can give birth
_grassGrowRate : number of grasses that grow by iteration
_grassEnergy : energy given by eating a grass
_xSize : size of the window along the X axis
_ySize : size of the window along the Y axis

-getName : give just the name of the program

-setup: which prepare the program before beginning of the simulation : it reinitialize the display surface (displaySurf), the list of rabbit (rabbitList), the schedule (that is use to give instruction to the grass and rabbits in the simulation)(schedule), the display of the graph that show the number of grasses and rabbits at each iteration (graph). And it create the slider for the in put of the simulation.
-begin : method called at the beginning of the simulation it call 3 method :

- -buildModel which create the grid on which the rabbits will move and fill it with the number « number » of rabbit, those rabbit are put randomly on the grid with a random level of energy from 1 to « energyToBirth » - 1.

- -buildSchedule who create 2 schedule : PlotUpdate that update the graph just calling the step() of the graph created and NatureStep which implements the behavior of the rabbits and grasses :
- - -randomly creates grass on the grid
- - - call RabbitReaper that delete all the rabbits whose energy <1 of the grid and of the rabbitList
- - -call RabbitCreater : for all rabbit with energy < « birthThreshold » it take them « birthThreshold » energy and create a rabbit no the grid
- - -finally it updates the display surface

- - BuildDisplay create 2 display for displaySurf :
- - -one for the grass (displayGrass) that take the grid of the grass coloring green when there is grass and black when there is not
- - -another for the rabbit (displayRabbit) that took the rabbit grid from »grid » and drawing the symbol of the rabbit with the draw function of RabbitGrassAgent (here a white circle) when there is a rabbit in the rabbit's grid.
And adding to the graph the display for the rabbits and grass with 2 classe : rabbitInSpace and grassInSpace that both implements DataSource and Sequence

begin then display the graph and the surface and make the graph make a step so that we have the beginning situation.

RabbitsGrassSimulationSpace : Class for the space on which there is rabbit and grasses.
variable :

_gridGrass : grid which represent where there is grass. There is 2 state possible by cell in the grid : 0 when there is no grass and 1 when there is.

_rabbitGrass : grid which represent where there is rabbit. There is 2 state possible by cell in the grid : either there is a RabbitsGrassSimulationAgent when there is a rabbit, or there is nothing (null) where there is none.

_countGrass : count the number of grass

L'initialization create the 2 grids for the rabbits and grasses and fill the grass's grid with zero. A lot of method are self explained by their name : the get method return the variables the isCellOccupiedRabbit and isCellOccupiedGrass check if a cell is occupied by a rabbit or grass by checking the value of the cell (isCellOccupiedGrass destroy the grass when it find it because the only time it is used is when we have to check if there is grass to eat for the rabbit.
oneStepAddGrass randomly create « nbGrass » grasses on the Grass grid and create randomly a rabbit on the rabit grid. Both of those stop only when they created enough or when they loop enough time that the program decided there is not enough place on the grid to place something.
TryMove try to make the rabbit move to the new location, if there is nothing he goes if there is he stay still.

RabbitsGrassSimulationAgent : calls for the agent rabbit :

_x and y are the coordinate of the location of the rabbit.

_ID is the unique ID of the rabbit and numberRabbit is used to give it to the ID.

tryMove create a random direction for the rabbit to go and send it to the grid so tat it test if it can go that way. step is used by the schedule : it decreases the energy of the rabbit and check if he can eat something. report is a string that gives the position and ID of the rabbit.