Modele Class

functions that a (sub-) model SHOULD re-implement

a constructor,
-> calling the mother constructor Modele(nbvals, nbparams)
-> filling names, extNames, paramNames
and optionally paraLowBounds, paraUpBounds, and backSimulated

virtual void derivatives(const vector<double> &x, vector<double> &dxdt, const double t);
-> should calculate dxdt[] from the value of variables x[] and the current time t

virtual void setBaseParameters();
-> fill params[] with a default set of parameters

virtual void initialise(long long _background = DEFAULT_BACKGROUND_VALUE);
-> can change init[], params[], whatever, depending on an option (_background)

pre-implemented functions :

void simulate(double _sec_max, Evaluator* E = NULL);
-> uses the solver to simulate val[] from the current t to t + sec_max
-> the variables for which over(internalVarID) = true are replaced by data
-> each wished data-point (time, variable), pre-defined in the Evaluator E is passed to E during simulation.
-> if saveKinetics = true, the tablecourse 'cinetique', is filled during simulation.

void applyOverride(vector<double> &x, double t);
-> fills the state vector x[vars] according to the overriders for time t

void clearOverride(vector<double> &x, vector<double> &dxdt);
-> fills x[]=0 and dxdt[]=0 for each overrided variable to avoid that the solver gets confused in case x[] was not following dxdt[]

void initialiseDone();
deleteCinetique();
penalities = 0;
stopCriterionReached=false;

bool checkDivergence();
-> if any of x[] is NAN, INF, or > STOP_VAL, then stopCriterionReached = true;
penalities += 100 / (0.01 + log(t/100 + 1)) for each

string print();