* 1. **Class StarSystem**
     1. +turns = 100 // == years\_till\_death\_star /10 (counts down)
     2. +index\_of\_mainp\_lanet = NONE
     3. +MIN\_PLANETS = 5
     4. +MAX\_PLANETS = 7
     5. +planets\_list // All active planets (5-7)
     6. +distance\_list // list with all non GZ rings
     7. +gz\_distance\_list // list with all GZ rings

In constructor:

* + 1. +set\_up() // set\_up method to generate planets must generate at least 1 goldilocks’ zone planets + all other planets
    2. +show\_planets() // show the user all generated planets
    3. +set\_main\_planet() // choose MainPlanet from planets\_list, use copy-constructor to copy all information to a new object of type MainPlanet; initialise index\_of\_main\_planet
    4. check\_winning\_condition() // **if** MainPlanet.engineering == 30 : Simulation won (final message) **else** **if** (turns != 0 && MainPlanet.engineering != 30) || MainPlanet.actual\_population <= 1 : Simulation over (final message)
    5. next\_turn() // decreases turns by 1 and offers the user information and the option to continue to the next turn
    6. +generate\_event(MainPlanet) // create object of type Event and returns that object // frequency of generating an event per turn is random (e.g. 1/3)
    7. Event event = generate\_event();
    8. MainPlanet.change\_base\_values(event)
  1. **Class Planet()** 
     1. +Planet(distance) // constructor
     2. +landmass // value between 10 and 100
     3. +distance // distance from star to planet (get from set\_up)
     4. +atmosphere // composition atmosphere between 1 and 100 (100: perfect)
     5. +radius // radius of the planet
     6. +max\_population // true maximum amount of species that a planet can hold; calculated via landmass & technologies & food\_per\_organism; Upper limit
     7. +FOOD\_PER\_ORGANISM // constant, minimal need for one organism to be alive = 1
     8. +get\_area() // calculates the area of a planet and returns it
     9. +get\_usable\_landmass() // returns (landmass\*get\_area())/100 ; is needed to calculate max\_population
     10. +get\_temperature () // calculate temperature from **distance**, **atmosphere**
     11. + life\_quality() // calculate QOL from return value get\_temperature and get\_usable\_landmass() and affection\_factor
     12. +show\_information()
  2. **Class MainPlanet(Planet)**
     1. +actual\_population // amount of species
     2. +population\_health // state of organism
     3. +medicine;
     4. +agriculture;
     5. +architecture;
     6. +engineering;
     7. +spend\_points() // user spends points on technologies(4)
     8. +calculate\_progression() // via tech & actual\_population (Jelle)
     9. +change\_base\_values(generate\_event() = NONE) // updates:

*If* event.type\_event == 0 : disaster: check\_technologies() -> changes event multipliers positively if technology allows this

*If* event.type\_event == 1 : breakthrough update\_technologies() // via Robin (multiplier or immediate progress)

set\_research\_focus() // choose your tech focus and generate +1 every 5 turns + creative input + show how many turns left till +1

+update\_variables() // multipliers\*variables // (variables = atmosphere, landmass, population\_health)

landmass -> affects get\_usable\_landmass

atmosphere -> affects get\_tempeature();

life\_quality() -> affected by get\_usable\_landmass() and get\_temperature() and population\_health

max\_population -> affected by update\_technologies()

actual\_population -> affected by life\_quality()

(if actual\_population > max\_population : affection\_factor between 0 and 1 ? affection factor 1 or higher

* 1. **Classs Event**
     1. +type\_event = NONE
     2. +atmosphere\_multiplier
     3. +landmass\_multiplier
     4. +population\_health\_multiplier
     5. +disaster\_list // dict of lists (list contains all texts) – optional: read through file
     6. +breakthrough\_list // dict of lists (list contains all texts) – optional: read through file
     7. +Event(MainPlanet, turns) // constructor // checks turns (type of disaster depends on turn); generates either a disaster or a breakthrough // type\_event = 0 (disaster) or 1 (breakthrough)
     8. set\_multipliers // only if disaster // if unchanged (== not relevant) then multiplier = 1, else change multipliers
  2. **Class Star(StarSystem) // optional: only when adding visual**
     1. +star\_size