InIn
IntersectionInstance

World
`Object objects[]`
`WorldState lastWorldState`

`constructor(map, initial state)`
parsing and interpreting a chosen map format done in this function?

`evolve(delta-t)`

```
For object i in objects:
  offspringObjects=objects[i].evolve(delta-t,
    intersectionResult[i]))
  self.addObject(offspringObjects)

run()

determine delta-t from all Object.get-required-delta-t()
While true
  intersectionResult=worldClass.intersect()
  t=t+delta_t
  worldClass.evolve(intersectionResult)
  lastWorldState.recordState(objects)
```

`InIn intersectionResult[]=intersect()`
intersectionResult is an array of array of InIn, where IntersectionResult[i] is an array of InIn pertaining to object i

```
for every object i:
  for every other object j:
    if not cubesIntersect(objects[i].boundingBox()
        ,objects[j].boundingBox()):
      intersectionResult[i]=empty
    else:
      intersectionResult[i][j]=no idea what to do, an instance of InIn
      #they still might have no intersection.
      #But if they do we need some details
```

WorldState
use: e.g. debug
the simulator works in delta-t steps. A worldState can be recorded either before or after evolution of all the objects. A partial worldState is invalid unless every object state is flagged as completed or not. The reason is that the world cannot be initialized with a snapshot of a partially completed worldState without knowing which objects have already evolved in that snapshot.

A data structure which keeps a set of states for every objectID

`load(file)`
`dump(file)`
`extractState(objectID)`
`recordState(objects[])`
For object in objects:
Get and record object.dumpState()

Map

XMLmap

Position
position and orientation
`x,y,z`
`phi,theta`

Object
`objectID`
every object needs an ID to be tracable in map and in state
`shapeClass shape`
`positionClass position`
position of an anchor point regardless of shape
`motionClass motion`

`offspringObjects=evolve(delta-t)`
changes the state
offsprings are the possibly created objects. How these possibly imaginary objects interact with the world and how long they live is not an architectural concern
pass for a trivial evolution. Otherwise, the object requires information. For instance, intersection with other objects. What to pass to evolve()?
`visualize()`
returns the information required for graphics boundingBox()
simply calls that of the shape? TBD. if world needs shape, we can omit this anyway
`dumpState()`
return value must have the objectID + a list of states. Maybe with variables names and values. All textual
`get-required-delta-t()`
calculates the delta-t it requires to operate right

Motion
Velocity and acceleration
`vel-x,vel-y,vel-z`
`acc-x,acc-y,acc-z`

Shape
some uniform definition? See the comment on the right
`boundingBox()`
returns a x-y plane bounding box. Can be done using a generalized algorithm, no implemented only in the parent class.

Sensor
`sense()`
I guess pointer to all world objects is required + world intersect method

LidarSensor
essentially a light beam rotating at an x-y plane and measuring distance with surrounding objects

BumperSensor
detects at a hit upon intersection, does not involved in shape

Robot
`Brain brain`
`evolve(delta-t)`
Call sensors if any
Call brain.stateTranform()

VaccumCleaner
`Brain brain`
`Sensor laser`
`Sensor bumper`
`Wheel wheelRight`
`Wheel wheelLeft`

DiscretizedShape
takes a mesh as input. Mesh given in a file or something?

Disk
`axis1`
`axis2`
`Coordinate center`

DiscretizedShape2D
takes bitmap and if required extends it to a mesh of infinitely long pieces along z-axis

Cylinder
`Disk disk1`
`Disk disk2`

DiningTable
`Cylinder legs[]`
`Cube topSurface`

Wheel
note that we could but don't have to model a wheel much. Just an object which evolves in its position
`angularSpeed`
`evolve(delta-t)`