```
//state
public double pressure;       //p //pascals
public double temperature;    //t //kalvin
public double volume;         //v //M^3/kg
public double internalenergy; //u //J/kg
public double entropy;        //s //J/kgK
public double enthalpy;       //h //J/kg
public double quality;        //x //%
```

Kelvin, not Kalvin

```
//static properties of system
public double mass = 1; //kg
public double radius = 0.05; //M
//public double surfacearea = Math.Pow(3.141592*radius,2.0); //M^2 //hardcoded answer below
public double surfacearea = 0.024674011; //M^2 //hardcoded answer to eqn above
public double surfacearea_insqr = 38.2447935395871; //in^2 //hardcoded conversion from m^2 to in^2

//assume starting/ending point consistent for whole API!

public void add_heat_constant_p(double j)
{
  double new_h = enthalpy+j;

  //at this point, we have enough internal state to derive the rest
  enthalpy = new_h;
  volume = ThermoMath.v_given_ph(pressure, new_h);
  temperature = ThermoMath.t_given_ph(pressure, new_h);
  entropy = ThermoMath.s_given_vt(volume,temperature);
  internalenergy = ThermoMath.u_given_vt(volume, temperature);
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
  switch(region)
  {
    case 0: quality = 1;                                   break; //subcooled liquid
    case 1: quality = ThermoMath.x_given_pv(pressure, volume); break; //two-phase region
    case 2: quality = 0;                                   break; //superheated vapor
  }

  transform_to_state();
}
```

quality = 0 corresponds to saturated liquid

quality = 1 corresponds to saturated vapor

quality is only defined inside and on the vapor dome. Outside the vapor dome it does not make sense to talk about quality.

```
public void add_heat_constant_v(double j)
{
  double new_u = internalenergy+j;
  double new_t = ThermoMath.iterate_t_given_v_verify_u(temperature,volume,new_u);

  //at this point, we have enough internal state to derive the rest
  internalenergy = new_u;
  temperature = new_t;
  pressure = ThermoMath.p_given_vt(volume,temperature);
  enthalpy = ThermoMath.h_given_vt(volume,temperature);
  entropy = ThermoMath.s_given_vt(volume,temperature);

  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
```

```
  switch(region)
  {                    quality = 0 corresponds to saturated liquid
    case 0: quality = 1;                                   break; //subcooled liquid
    case 1: quality = ThermoMath.x_given_pv(pressure, volume); break; //two-phase region
    case 2: quality = 0;                                   break; //superheated vapor
  }                    quality = 1 corresponds to saturated vapor

  transform_to_state();
}

public void add_pressure_uninsulated(double p)
{
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
  double new_p = pressure+p;

  switch(region)
  {
    case 0: //subcooled liquid
    case 1: //two-phase region
    {
      //AVOID THESE SCENARIOS
      return;
    }
    break;
    case 2: //superheated vapor
    {
      //default guess
      double new_u = internalenergy;
      double new_v = volume;

      //already done!
      new_v = ThermoMath.v_given_pt(new_p,temperature);
      new_u = internalenergy - pressure*volume*Math.Log(new_v/volume);
      //at this point, we have enough internal state to derive the rest     new_u = ThermoMath.u_given_pt(new_p, temperature)
      pressure = new_p;
      volume = new_v;
      internalenergy = new_u;
      enthalpy = ThermoMath.h_given_vt(volume,temperature);
      entropy = ThermoMath.s_given_vt(volume,temperature);
    }
    break;
  }

  transform_to_state();
}

public void add_pressure_insulated(double p)
{
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
  double new_p = pressure+p;

  switch(region)
```

```
 {
  case 0: //subcooled liquid
  case 1: //two-phase region
  {
   //AVOID THESE SCENARIOS
  }
  break;
  case 2: //superheated vapor
  {
   //default guess
   double new_t = temperature;
   double new_u = internalenergy;
   double new_v = volume;

   {                          to avoid having to iterate here, we can use:
    double k = 1.27;          new_v = volume*[(pressure / new_p)^(1/k)]
    new_u = internalenergy-((new_p*new_v-pressure*volume)/(1-k));
    new_t = ThermoMath.iterate_t_given_p_verify_u(temperature,pressure,new_u);
    new_v = ThermoMath.v_given_pt(new_p, new_t);
   }

   //at this point, we have enough internal state to derive the rest
   pressure = new_p;
   volume = new_v;
   temperature = new_t;
   internalenergy = new_u;
   enthalpy = ThermoMath.h_given_vt(volume,temperature);
   entropy = ThermoMath.s_given_vt(volume,temperature);
  }
  break;
 }

 transform_to_state();
}
```