Scenario 1, Scenario 2, Scenario 7

```
//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;        //q //%

//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!
```

Scenario 1

```
public void add_heat_constant_p(double j) //TODO: implement iteration step
{
  //no difference between regions

  //default guess
  double new_v = volume;
  double new_u = internalenergy;   use h2 = h1 + Q*time
  //ITERATE TO SOLVE      not necessary to iterate! :)
  {
    new_v = ThermoMath.v_given_pu(pressure,new_u);
    new_u = internalenergy+(j/mass) - pressure*(new_v-volume);
  }

  //at this point, we have enough internal state to derive the rest
  volume = new_v;
  internalenergy = new_u;
  temperature = ThermoMath.t_given_pv(pressure, volume);
  enthalpy = ThermoMath.h_given_pu(pressure,internalenergy);   use 95: enthalpy(rho,T)
  entropy = ThermoMath.s_given_pu(pressure, internalenergy);   use 95: entropy(rho,T)

  transform_to_state();
}
```

Scenario 2

```
public void add_heat_constant_v(double j)
{
  //no difference between regions

  double new_u = internalenergy+(j/mass);

  //at this point, we have enough internal state to derive the rest
  internalenergy = new_u;
  pressure = ThermoMath.p_given_vu(volume, internalenergy);
  temperature = ThermoMath.t_given_pv(pressure, volume);
  enthalpy = ThermoMath.h_given_pu(pressure,internalenergy);   use 95: enthalpy(rho,T)
  entropy = ThermoMath.s_given_pu(pressure, internalenergy);   use 95: entropy(rho,T)

  transform_to_state();
}
```

```
public void add_pressure_insulated(double p, bool insulated) //TODO: implement iteration
{   Scenario 7
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
  switch(region)
  {
    case 0: //subcooled liquid
    {
      //default guess
      double new_t = temperature;
      double new_u = internalenergy;

      double new_p = pressure+p; //no significant change to other variables!
      //TODO: ITERATE TO SOLVE
      {                                                  Ignore this case - no significant change of state
      new_t = ThermoMath.t_given_pu(new_p, new_u);
      new_u = ThermoMath.u_given_pt(new_p, new_t);
      }

      //at this point, we have enough internal state to derive the rest
      pressure = new_p;
      temperature = new_t;
      internalenergy = new_u;
      volume = ThermoMath.v_given_pt(pressure, temperature);
      enthalpy = ThermoMath.h_given_pu(pressure,internalenergy);
      entropy = ThermoMath.s_given_pu(pressure, internalenergy);
    }
    break;
    case 1: //two-phase region
    {
      //IGNORE BECAUSE UNSURE HOW TO CALCULATE!
    }
    break;
    case 2: //superheated vapor
    {
      //default guess
      double new_u = internalenergy;
      double new_v = volume;

      double new_p = pressure+p; //no significant change to other variables!
      //TODO: ITERATE TO SOLVE                                       fix k = 1.27
      {
       //new_u = internalenergy-pressure*volume^k/(k-1)*(volume^(1-k)-new_v^(1-k)); //variable insulation
//TODO: if you want to implement variable insulation, change "insulated" from bool to float, then use this eqn
(and obv alter the calling functions to pass in variable)
       if(insulated)
         new_u = internalenergy; //unchanged? I got this by subbing k = 1 (sets eqn to u-p*inf*0, which I
interpreted as equal to u-p*0, ie, just u)
       else
         new_u = internalenergy-pressure*(volume-new_v); //seems coherent?
       new_v = ThermoMath.v_given_pu(new_p, new_u);
      }

      //at this point, we have enough internal state to derive the rest
      pressure = new_p;
      volume = new_v;
      internalenergy = new_u;
      temperature = ThermoMath.t_given_pu(pressure, internalenergy);
```

```
      enthalpy = ThermoMath.h_given_pu(pressure,internalenergy);
      entropy = ThermoMath.s_given_pu(pressure, internalenergy);
    }
    break;
  }

  transform_to_state();
}
```

use 95: enthalpy(rho,T)

use 95: entropy(rho,T)