```
// A program to study the time-dependent temperature
// field around a nuclear waste rod in a 2D model.

import java.lang.*;
public class Nuclear {
  final static int nx = 100, n = 5, nt = 1000,
    mt = 1000;
  public static void main(String argv[]) {
    double d[] = new double[nx];
    double e[] = new double[nx];
    double c[] = new double[nx];
    double b[] = new double[nx];
    double p[] = new double[nx];
    double s[][] = new double[nt+1][nx];
    double T[][] = new double[nt+1][nx+1];
    double dt = 1.0/mt, tc = 1, T0 = 1, kappa = 2e7;
    double ra = 25, rb = 100, h = rb/nx, h2 = h*h;
    double s0 = dt*kappa*T0/(ra*ra), g = dt*kappa/h2;

 // Assign the elements in the matrix 2-H_i
    for (int i=0; i<nx; ++i) {
      d[i] = 2*(1+g);
      e[i] = -(1+0.5/(i+1))*g;
      c[i] = -(1-0.5/(i+2))*g;
    }

 // Modify the first equation from T"=0 at r=0
    d[0] -= 2*g/3;
    e[0] += g/6;

 // Assign the source of the radiation heat
    int na = (int) (ra/h);
    for (int i=0; i<=nt; ++i) {
      double t = -dt*i/tc;
      for (int j=0; j<na-1; ++j) {
        s[i][j] = s0*Math.exp(t);
      }
    }
```

nt time steps
nx space steps
mt=nt
n = printout skip

so=source parameter
ra=cylinder radius
rb=simulation radius
T0=initial temperature
dt=time step
h=space step

Result from 4-point formula at edge

```
// Find the temperature field recursively
   for (int i=1; i<=nt; ++i) {

  // Assign the elements in the matrix 2+H_0
     double d0 = 2*(1-g);
     double e0 = (1+0.5)*g;
     double c0 = (1-0.5)*g;

  // Evaluate b[0] under the condition T"=0 at r=0
     b[0] = d0*T[i-1][0]+e0*T[i-1][1]
           +c0*(4*T[i-1][0]-T[i-1][1])/3
           +s[i-1][0]+s[i][0];

  // Find the elements in the array b[i]
     for (int j=1; j<nx; ++j) {

    // Assign the elements in the matrix 2+H_0
       d0 = 2*(1-g);
       e0 = (1+0.5/(j+1))*g;
       c0 = (1-0.5/(j+1))*g;

    // Obtain the elements from the last recursion
       b[j] = d0*T[i-1][j]+e0*T[i-1][j+1]
             +c0*T[i-1][j-1]+s[i-1][j]+s[i][j];
     }

  // Obtain the solution of the temperature field
     p = tridiagonalLinearEq(d, e, c, b);
     for (int j=0; j<nx; ++j) T[i][j] = p[j];
   }

// Output the result at every n spatial data points
   for (int j=0; j<nx; j+=n) {
     double r = h*(j+1);
     System.out.println(r + " " + T[nt][j]);
   }
 }

// Method to solve the tridiagonal linear equation set.

 public static double[] tridiagonalLinearEq(double d[],
   double e[], double c[], double b[]) {...}
}
```

Loop over time

Solve for Temperature at all positions (j index) for one time step

Close time loop