OS PROJECT INTERFACE: Java

STRUCTURE OF yourfile.java :

```
#include's        //  These declarations (e.g. jobtable) exist only in your file, but
#define's         //  since they lie outside of function definitions
typedef's         //  they are global to the whole file and retain their
static variables  //  values over the course of program execution
                  //  (i. e., between function invocations).

void siodisk(int jobnum);
void siodrum(int jobnum, int jobsize, int coreaddress, int direction);

    //  Channel commands siodisk and siodrum are made available to you by the simulator.
    //  siodisk has one argument: job number, of type int and passed by value.
    //  siodrum has four arguments, all of type int and passed by value:
    //      first argument is job number;
    //      second argument is job size;
    //      third argument is starting core address;
    //      fourth argument is interpreted as follows:
    //          1   => move from core (memory) to drum
    //          0   => move from drum to core (memory)


void ontrace();          // called without arguments
void offtrace();         // called without arguments

    //  The 2 trace procedures allow you to turn the tracing mechanism on and off.
    //  The default value is off.   WARNING: ontrace produces a blow-by-blow description
    //  of each event and results in an extremely large amount of output.
    //  It should be used only as an aid in debugging.
    //  Even with the trace off, performance statistics are
    //  generated at regular intervals and a diagnostic message appears in case of a crash.
    //  In either case, your OS need not print anything.


void startup()
{
    //  Allows initialization of static system variables declared above.
    //  Called once at start of the simulation.

}

// INTERRUPT HANDLERS
    //  The following 5 functions are the interrupt handlers.   The arguments
    //  passed from the environment are detailed with each function below.
    //  See RUNNING A JOB, below, for additional information
```

```
void Crint (int []a, int []p)
{
        //    Indicates the arrival of a new job on the drum.
         //    At call: p [1] = job number
         //                    p [2] = priority
         //                    p [3] = job size, K bytes
         //                    p [4] = max CPU time allowed for job
         //                    p [5] = current time

}


void Dskint (int []a, int []p)
{
        //    Disk interrupt.
        //    At call: p [5] = current time
}

void Drmint (int []a, int []p)
{
        //    Drum interrupt.
        //    At call: p [5] = current time
}

void Tro (int [], int []p)
{
      //    Timer-Run-Out.
      //    At call: p [5] = current time
}

void Svc (int []a, int []p)
{
      //    Supervisor call from user program.
        //    At call: p [5] = current time
        //                    a = 5 => job has terminated
        //                    a = 6 => job requests disk i/o
        //                    a = 7 => job wants to be blocked until all its pending
        //                                    I/O requests are completed
}
```

Additional functions local to OS (scheduler, swapper, etc.)

RUNNING A JOB:

```
//    Before leaving each interrupt handler (with the return statement)
//    you must call the dispatcher to send info about which job to run.
//    The dispatcher should set the a and p arguments as follows:
```

```
//          a = 1   CPU is idle, p is ignored
//          a = 2   CPU is in user mode,
 //                      p [0], p [1], and p [5] are ignored
 //                      p [2] = base address of job to be run
 //                      p [3] = size (in K) of job to be run
 //                      p [4] = time quantum
```

NOTES:

- time is in milliseconds.
- core addresses are in K (0 - 99).
- priority ranges from 1 (highest) to 10 (lowest).
- assume interrupts are inhibited while OS is executing.

TO RUN SOS WITH YOUR OS:

compile yourfile.cpp separately and link with sos.obj (PC) or sos.o (Unix).
main() is defined in sos.

Look on the project Web site for individual "how-to" files for your compiler.