```
public class JuegoCSP implements CSProcess {
  // valores simbolicos para las materias primas
  final int CEREAL = 0;
  final int AGUA = 1;
  final int MADERA = 2;

  // Canales para pedir al servidor
  private Any2OneChannel chCargar;
  private Any2OneChannel chAvanzar;
  private Any2OneChannel chReparar;

  private JuegoCSP () {
    this.chCargar = Channel.any2one();
    this.chAvanzar = Channel.any2one();
    this.chReparar = Channel.any2one();
  }
  public void cargar(int m) {
    chCargar.out().write(m);
  }
  public void avanzar() {
    chAvanzar.out().write(null);
  }
  public void reparar() {
    chReparar.out().write(null);
  }
```

```
  public void run() {
    // declaramos aqui el estado del recurso
    int[] materias = {0, 0, 0};

    // soporte para recepcion alternativa condicional
    // Nombres simbolicos para los indices de servicios
    final int CARGAR = 0;
    final int AVANZAR = 1;
    final int REPARAR = 2;
    // Entradas de la select
    final AltingChannelInput[] entradas =
    {chCargar.in(), chAvanzar.in(), chReparar.in()};
    // Recepcion alternativa
    final Alternative servicios = new Alternative (entradas);
    // Sincronizacion condicional en la select
    final boolean[] sincCond = new boolean[3];

    // el servidor ejecuta un bucle de servicio infinito:
    while (true) {
      // Preparacion de las precondiciones
      sincCond[CARGAR] = materias[CEREAL] + materias[AGUA] +
                         materias[MADERA] < 9;
      sincCond[AVANZAR] = materias[CEREAL] > 0 && materias[AGUA] > 0;
      sincCond[REPARAR] = materias[AGUA] > 0 && materias[MADERA] > 0;
      switch (servicios.fairSelect(sincCond)) {
        case CARGAR:
         int queMateria = (Integer) chCargar.in().read();
         materias[queMateria]++;
          break;
        case AVANZAR:
         chAvanzar.in().read();
         materias[CEREAL]--;
         materias[AGUA]--;
          break;
        case REPARAR:
         chReparar.in().read();
         materias[AGUA]--;
         materias[MADERA]--;
          break;
      }
    } // fin bucle servidor
  } // fin servidor
}
```

```
public class MateriasPrimas {

  private Monitor mutex;
  private Monitor.Cond condCargar;
  private Monitor.Cond condAvanzar;
  private Monitor.Cond condReparar;

  private int cereal = 0;
  private int agua = 0;
  private int madera = 0;

  public MateriasPrimas(){
    mutex = new Monitor();
    condCargar = mutex.newCond();
    condAvanzar = mutex.newCond();
    condReparar = mutex.newCond();
  }

  public void cargarCereal() {
    mutex.enter();
    if (cereal + agua + madera == 10) {
      condCargar.await();
    }
    cereal = cereal + 1;
    desbloqueoSimple();
    mutex.leave();
  }

  public void cargarAgua() {
    mutex.enter();
    if (cereal + agua + madera == 9) {
      condCargar.await();
    }
    agua = agua + 1;
    desbloqueoSimple();
    mutex.leave();
  }

  public void cargarMadera() {
    mutex.enter();
    if (cereal + agua + madera == 10) {
      condCargar.await();
    }
    madera = madera + 1;
    desbloqueoSimple();
    mutex.leave();
  }
```

```
  public void avanzar() {
    mutex.enter();
    if (cereal == 0 || agua == 0) {
      condAvanzar.await();
    }
    cereal = cereal - 1;
    agua = agua - 1;
    desbloqueoSimple();
    mutex.leave();
  }

  public void reparar() {
    mutex.enter();
    if (agua == 0 || madera == 0) {
      condReparar.await();
    }
    agua = agua - 1;
    madera = madera - 1;
    desbloqueoSimple();
    mutex.leave();
  }

  private void desbloqueoSimple() {
    if (cereal > 0 && agua > 0 && condAvanzar.waiting() > 0) {
      condAvanzar.signal();
    }
    else if (agua > 0 && madera > 0 && condReparar.waiting() > 0)
{
      condReparar.signal();
    }
    else if (cereal + agua + madera < 10) {
      condCargar.signal();
    }
  }
}
```