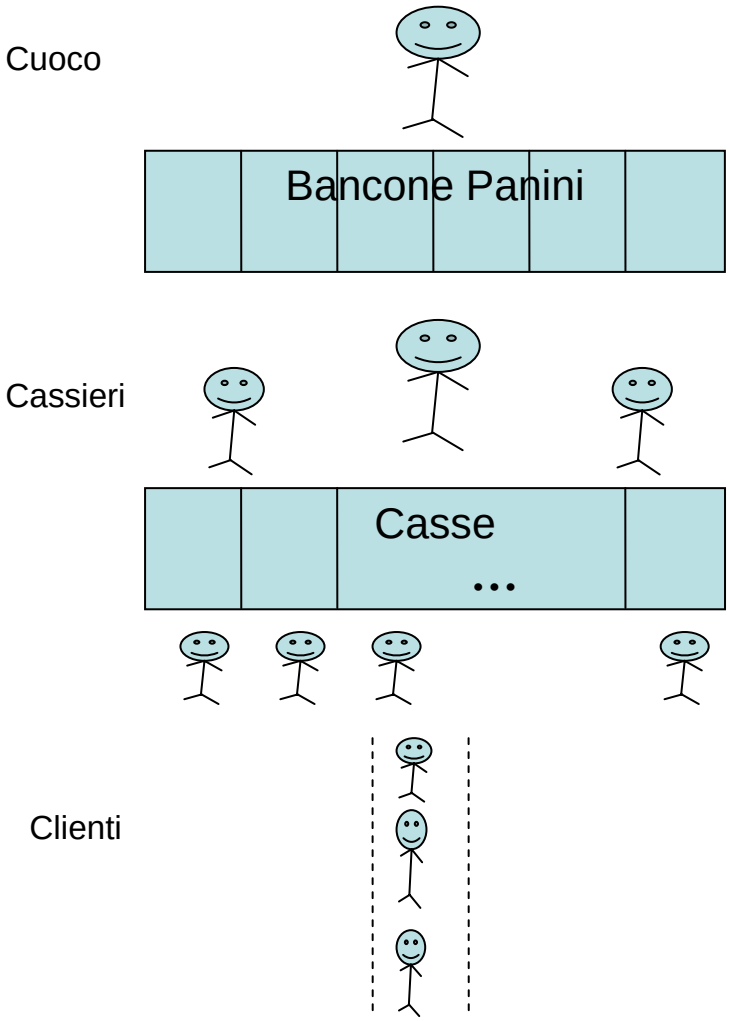
Esercizio N. 1

La paninoteca "`McHammer`" necessita di un metodo per lo smaltimento dei propri ordini secondo lo schema di figura:




La paninoteca funziona in base alla presenza di 4 tipi di entità:

**Paninoteca.** E' composta da due sotto-strutture dati:

1. Le **casse** (CS), dove è possibile ordinare un certo numero di panini. Ogni **ordine** è un insieme di **elementi** <`tipologia,quantità`>, che rappresentano il numero di panini richiesti di un certo tipo (tra quelli offerti dalla paninoteca). La paninoteca dispone di un'unica fila di accesso alle casse. Un cliente effettua un ordine presso le casse attraverso il metodo `putOrdine(O)` (O rappresenta l'ordine del cliente costituito da un insieme di elementi <tipologia,quantità>). Il metodo `putOrdine` occupa la prima cassa disponibile, sottopone l'ordine al cassiere e termina quando l'ordine è stato COMPLETAMENTE SERVITO CON I PRODOTTI RICHIESTI. Inoltre il metodo `putOrdine` deve bloccare il thread che lo invoca se non ci sono casse disponibili.
   Un cassiere preleva un ordine mediante il metodo `getOrdine(C,O)`. Il metodo `getOrdine` deve bloccare il thread chiamante se non ci sono ordini da servire. Il valore C è passato per riferimento e ritorna il numero della cassa da cui è stato prelevato l'ordine.

Infine il cassiere serve il cliente mediante l'invocazione del metodo `putOrdinazione(C)` (C rappresenta la cassa servita). Il metodo `putOrdinazione(C)` libera la cassa e sblocca il cliente in attesa.

2. Il **bancone dei panini** (BP), di dimensione P (la paninoteca offre P differenti tipologie di panini), dove il cuoco deposita i panini appena preparati. I panini vengono depositati dall'**unico** cuoco mediante il metodo `putPanini(T,N)` che colloca sul bancone N panini di tipo T (si assuma illimitata la capacita del bancone dei panini).
   I panini vengono prelevati dai cassieri attraverso invocazioni successive del metodo `getPanini(T,N)` (preleva N panini di tipo T da BP), fino a costituire l'ordine richiesto dal cliente. Il metodo `getPanini` si blocca nel caso in cui il bancone non contenga il numero di panini richiesti (ad esempio se ci sono 2 panini di tipo T e al cassiere ne servono 5, il cassiere si blocca senza prelevare nessun panino, in attesa che ce ne siano almeno 5).

**Cliente.** Il cliente è un tipo di thread il cui ciclo di vita è il seguente:

a) Inattività. Il cliente non fa nulla di particolare;
b) Sottomissione ordine. Il cliente genera un ordine e lo sottomette alla paninoteca eseguendo `putOrdine(O)`, il cliente rimane in attesa finché non viene servito;
c) Attesa non bloccante. Il cliente consuma i panini ordinati, il tempo di attesa non bloccante deve essere simulato in maniera casuale, ma ragionevolmente proporzionale al numero di panini ordinati;
d) Ritorno al punto a.

**Cassiere.** Il cassiere è un tipo di thread con il seguente ciclo di vita:

a) Prelievo di un ordine dalle casse CS, eseguendo `getOrdine`. Si blocca nel caso non ci siano ordini disponibili in CS.
b) Elaborazione dell'ordine, il cassiere preleva i panini dal bancone BP, mediante una sequenza di chiamate di `getPanini`.
c) Erogazione dell'ordine, il cassiere consegna i panini al cliente, eseguendo `putOrdinazione`.
d) Ritorno al punto a.

**Cuoco.** Il cuoco è un tipo di thread che ad intervalli di tempo regolari produce un certo numero di panini di un certo tipo e li colloca sul bancone BP mediante il metodo `putPanini`.

Si progettino le classi `Paninoteca`, `Cliente`, `Cassiere`, `Cuoco`, nonché un programma `main` che avvii la paninoteca `McHammer`, con `C` clienti, `K` casse/cassieri e `1` cuoco.

La soluzione proposta dallo studente dovrebbe garantire (in rigoroso ordine di priorità)

1. Accesso in mutua esclusione alle strutture dati della paninoteca;
2. Prevenzione di qualsiasi possibile situazione di deadlock;
3. Prevenzione di possibili situazioni di starvation;
4. Utilizzo ottimale delle risorse/scelta delle strutture dati adeguate.

E' possibile l'uso di qualsiasi tipo di appunti o libro di testo.

Esercizio N. 2

E’ Dato il seguente frammento di FAT:

| Settore | Settore Successivo |
|---|---|
| 310 | 318 |
| 311 | 315 |
| 312 | 314 |
| 313 | 317 |
| 314 | <EOF> |
| 315 | 316 |
| 316 | <EOF> |
| 317 | 311 |
| 318 | 312 |
| 319 | 320 |
| 320 | 310 |
| ... | |

Assumendo che i file A e B iniziano rispettivamente nel blocco 319 e 313, e che i blocchi abbiano dimensione 4KB, calcolare il blocco corrispondente ai seguenti offset (espressi in byte) nei file A e B:

1. 11260 file A
2. 20480 file A
3. 4095 file A
4. 6144 file B
5. 12288 file B
6. 16384 file B