Una officina dispone di **NTAV** stazioni di lavoro (con due posti per stazione). La lavorazione è eseguita da minisistemi robotizzati chiamati *bot*. Sono usati due modelli di bot che pervengono alle stazioni di lavoro da due code distinte: i bot di tipo **TIPO0=0** possono lavorare in coppia nella stessa stazione, invece quelli di tipo **TIPO1=1** più complesso devono lavorare da soli. Per la simulazione ogni bot è rappresentato da un thread. L'allocatore assegna ai singoli bot i posti secondo le seguenti regole, in ordine di priorità decrescente:

- se c'è un bot di **TIPO1** in attesa ed una stazione è completamente libera, la assegna per intero al bot (al primo della relativa coda – n.b. il bot in questo caso occupa due posti);
- se c'è un bot di **TIPO0**, assegna al bot (al primo della relativa coda) un posto libero in una stazione già occupata (parzialmente) da un bot di **TIPO0**, ed eventualmente il primo posto di una stazione libera. Si noti che la liberazione di una stazione da parte di un bot di **TIPO1**, in assenza di un bot di **TIPO1** in attesa, può soddisfare due richieste di bot di **TIPO0**.

Realizzare la simulazione dell'allocazione dei posti con la classe Java **Officina** che utilizza il **Monitor di Hoare** come unico tipo di strumento di sincronizzazione. In particolare la classe includa tra l'altro:

- le variabili **stazioniLibere** e **postiLiberi** che rappresentano in ogni istante rispettivamente il numero di stazioni completamente libere e il numero totale di posti liberi;
- il vettore **stato[]** che rappresenta il numero di posti occupati per ciascuna stazione;
- il vettore di Condition **codabot[2]** che rappresenta le code di attesa, una per tipo di bot;
- il vettore **inAttesa[2]** che tiene il conteggio dei bot in attesa in ciascuna delle code;
- il metodo di servizio **int trovaStazione** che restituisce l'indice **0..NTAV-1** della prima stazione completamente libera;
- il metodo di servizio **int trovaPosto** che restituisce l'indice **0..NTAV-1** di una stazione che ha almeno un posto libero, dando priorità nella ricerca alle stazioni che hanno un solo posto libero (rispetto a quelle che ne hanno 2);
- il metodo **int entra(int tipo)** che rappresenta l'ingresso di un bot di tipo **tipo** e restituisce, dopo una eventuale attesa in coda, l'indice della stazione dove sono stati trovati il o i posti richiesti;
- il metodo **void esce(int tipo, int pos**) che rappresenta il completamento d'azione e uscita di un bot di tipo **tipo** che occupava uno o due posti della stazione di indice **pos**;
- la classe interna **Bot** che rappresenta un singolo bot; nel costruttore riceve un indice identificativo e la codifica del suo tipo; nel metodo esecutivo deve mettersi in coda in base al tipo chiamando **entra**, simulare la lavorazione con una attesa casuale tra **minT** e **maxT** (parametri temporali comuni a tutti i bot), uscire chiamando **esce** e poi terminare;

un metodo di collaudo che riceve sulla linea di comando il parametro **NTAV**, crea un'istanza della classe fornendo due costanti di prova **minT** e **maxT**, nonché il valore **NTAV**, e poi crea in tempi casuali thread bot di tipo scelto casualmente.