1. Архитектура на проекта

Проекта е изграден от 4 класа:

\****Agent***: Клас на агентите. В този клас са описани всички полета и методи на агентите. Всеки агент разполага с поле *name*(името на агента), *secLevel*(променлива от тип *SecurityLevel*), референции към *Elevator* и *Building*. Всеки агент си има и масив от всички етажи- *accessableFloors*, които може да достъпва, който се напълва в конструктора, спрямо зададения *secLevel*. Променливата *workDone* е цяло число, чрез което се осъществява дневната квота от работа, изразена в проценти, когато числото прехвърли 100 като стойност, това е сигнал за агента че трябва да приключи работа. За сигнал се използва булевата променлива *GoHomeTimeFlag*. Всеки агент си има и *atHomeEvent*- обект от класа *ManualResetEvent*, който сигнализира на булевата променлива *AtHome*, чрез която нишката сигнализира на главната нишка че е приключила успешно своята работа. Всеки агент се обслужва от собствена нишка, която се стартира с параметър- метода *Roam*, който капсулира целия жизнен цикъл на агента. За сигнализация между нишките на агента и нишката на асансьора се използват 2 променливи от тип *AutoResetEvent*- *elevatorSignal* и *agentSignal*. Агента синхронизира тези променливи със променливите със същите имена от *Elevator*. Както в реалния живот, агентите се подреждат на опашка на етажа, на който се намират, ако по това време има и други агенти, чакащи асансьора на същия етаж. Опашките са реализирани в клас *Building*. Полето *queueMutex* от клас *Mutex* се грижи агентите да се изчакват, когато са на опашка пред асансьора. До асансьора се допуска агентът, който е в началото на опашката(проверката става в метода *CallElevator* чрез метода *TryPeek* на конкурентната опашка). Жизнения цикъл на агента протича в следните състояния:

- Проверява се дали агента е свършил своята рабора за деня

- Ако не е свършил, нарежда се на опашка на съответния етаж и чака

- Когато дойде неговия ред, той извиква асансьора и чака той да пристигне

- Следва цикъл на натискане на бутон за желания етаж, движение на асансьора, и *SecurityCheck* проверка, която, ако върне *false*, това е сигнал за агента да избере друг бутон от списъка с етажи

- Ако агента може да слезе на съответния етаж, той слиза, и освобождава асансьора за следващия агент

- Този цикъл се повтаря докато агента не свърши работата си за деня

- Когато свърши своята работа, извиква се метода *GoHome*, който е конкретно дефинирана серия от следните действия:

~ Ако агента се намира на етаж *G*, той напуска сградата, като *atHomeEvent* се сигнализира чрез метода *Set*

~ Ако агента не се намира на етаж *G*, то той се нарежда на опашка пред асансьора, с цел да стигне до етаж *G*

~ При напускане на сграда се извиква метода *Set* на *atHomeEvent*, който сигнализира на променливата *AtHome* да приеме стойността *true*.

\****Program***: Главния клас на програмата. Той създава обектите, стартира методите им, след което изчаква нишките да приключат своята работа, преди да приключи главната нишка.

\****Elevator***: Клас на асансьора. Съдържа лист от всички етажи, променлива от тип *Floor*- *currentFloor*, която съхранява етажа, на който асансьора се намира в даден момент, и променливата *destination*, която съхранява дестинацията на асансьора в дадения момент. Съдържа и 2 променливи от тип *AutoResetEvent*, чрез които се осъществява синхронизацията с нишките на агентите. Дефинирана е и нишката *workerThread*, която изпълнява работата на асансьора. Асансьора има метод *SecurityCheck*, който изпълнява проверката за достъп на агента до съответния етаж. Асансьора има и метода *GetAvailableButtons*, който връща лист със всички възможни етажи, до които агента може да достигне( т.е. всички етажи, без етажа, на който асансьора в момента се намира). Метода *ElevatorThreadWorker* се стартира от агент, който стартира процеса на движение през етажите на отделна нишка. Тъй като само един агент може да влезе в асансьора в даден момент, това поведение се контролитра от обекта *mutex* на клас *Mutex*.

\****Building***: Клас на сградата. Този клас съдържа 2 променливи: референция към клас *Elevator*, и речник, който пази четирите опашки за всеки етаж. Ключа на речника е от тип *Floor*, а стойностите- *ConcurrentQueue* опашки (използвани са *ConcurrentQueue* опашки, тъй като са *thread*-*safe*). Класа иам 2 метода, които реализират добавянето и премахването на елемент от опашките.

2. Клас Program

В клас *Program* са създадени 12 обекта от клас *Agent*, един обект от клас *Elevator* и един от клас *Building*. Временно изходът на програмата се извежда в конзолата. При нужда, моля откоментирайте в класовете *Agent* и *Elevator* всички закоментирани редове, за да се изписва изхода във текстовия файл. В случая, добре е *Thread.Sleep* извикванията да се премахнат, тъй като в текстовия файл съдражанието на изхода е видимо чак след края на програмата.