SNUS – Prvi kolokvijum

Napisati konzolnu aplikaciju koja omogućava simulaciju zagrevanja nuklearnog reaktora usled hemijske reakcije, kao i spuštanja temperature reaktora putem vodenog hlađenja.

- Kreirati klasu **Reaction** koja predstavlja hemijsku reakciju i sadrži svojstva **double HeatingRate** i **double TimeSpan**. HeatingRate predstavlja količinu toplote koju reakcija proizvodi u vremenskom periodu definisanom sa TimeSpan.
- Kreirati klasu **Valve** (ventil) sa svojstvima **double InRate** i **bool IsOpen**. InRate predstavlja promenu količine tečnosti u jednoj sekundi, a IsOpen indikator otvorenosti ventila.
- Kreirati klasu **Pump** (pumpa) sa svojstvima **double InRate** i **bool isOn**. InRate predstavlja količinu vode koju pumpa unosi u sistem u jednoj sekundi, a IsOn indikator rada pumpe.
- Kreirati klasu **WaterCooling** koja predstavlja vodeno hlađenje, i sadrži listu ventila **Valves** i objekat pumpe (**Pump**).
- Kreirati klasu **NuclearReactor** sa sledećim svojstvima: **Reaction Reaction**, **WaterCooling WaterCooling, double CurrentTemperature**, **double MinTemperature**, **double MaxTemperature**, kao i konstruktorom koji kao parametare prima objekte klasa Reaction i WaterCooling, a postavlja granične vrednosti temperature na 50 i 300. Unutar ove klase implementirati i metodu **RandomiseHeatRate()**, koja vrši promenu svojstva HeatingRate reakcije za neki procenat između **-30** i **30**.
- Kreirati klasu **SystemState** sa svojstvima **DateTime TimeStamp**, **double CoreTemperature, Pump pump, HeatingRate hearingRate** i kopijom liste ventila - **ValvesSnapshot**. Opis ovih svojstava je dat u nastavku zadatka.
- Kreirati delegate **CoreTemperatureHandler** i **CriticalTemperatureHandler** prema potrebama zadatka.
- Kreirati klasu **NuclearReactorMonitoring** sa poljem **NuclearReactor nuclearReactor**, događajima **coreTemperatureChanged** i **criticalValueOccured**, listom stanja sistema **systemStates**, kao i nitima **Thread reactionThread, Thread waterCoolingThread** i **randomiseThread**.
- Kreirati konstruktor klase NuclearReactorMonitoring koji inicijalizuje sva polja ove klase, a kao parametar prima objekat klase NuclearReactor. Kreirati tri objekta ventila i dodati ih u listu ventila, kao i jedan objekat pumpe.
- Kreirati metodu **OpenCloseValves()** koja vrši zatvaranje jednog nasumično odabranog ventila i otvara sve preostale ventile.
- Kreirati metodu **StopReactionAZ5()** koja naglo zaustavlja hemijsku reakciju, tako što postavlja HeatingRate reakcije na 0, otvara sve ventile i pali pumpu.
- Implementirati metodu **StartReactionSimulation(int seconds)**, koja pokreće nit reactionThread, a kao ulazni parametar prima trajanje reakcije u sekundama.
- Implementirati metodu **ReactionSimulation** koja vrši simulaciju promene temperature jezgra nuklearnog reaktora usled hemijske reakcije i vodenog hlađenja. Hemijska reakcija izaziva uvećanje temperature jezgra za količnik HeatingRate/TimeSpan na svaku sekundu. Međutim, usled vodenog hlađena jezgra, temperatura se na svaku sekundu smanjuje za sumu InRate-ova svih otvorenih ventila. Prema tome, formula po kojoj se menja temperatura na svaku sekundu je: $\Delta t = \frac{heatingRate}{timeSpan} - \sum_{open\ valves} inRate$. Ukoliko temperatura jezgra reaktora dostigne kritičnu vrednost (MaxTemperature), potrebno je zaustaviti hemijsku

reakciju pozivom metode **StopReactionAZ5**, i nastaviti sa spuštanjem temperature jezgra. Simulacija promene temperature traje **seconds** sekundi ili dok temperatura reaktora ne dostigne donju graničnu vrednost.

- Implementirati metodu **StartWaterCoolingSimulation()**, koja pokreće nit waterCoolingThread.
- Implementirati metodu **WaterCoolingSimulation** koja vrši simulaciju promene stanja ventila u toku vremena. Ova metoda na svake dve sekunde poziva metodu OpenCloseValve i time zatvara jedan, a otvara sve preostale ventile kojima se reaktor hladi.
- Aktivirati događaj **coreTemperatureChanged** na svaku promenu temperature jezgra reaktora. Događaj **criticalValueOccured** aktivirati u trenutku kada temperatura reaktora dostigne kritičnu vrednost (MinTemperature ili MaxTemperature).
- Implementirati metodu **StartRandomisation(int seconds)** koja pokreće nit randomiseThread, unutar koje se na seconds sekundi poziva metoda RandomiseHeatRate(). Ova nit ostaje pokrenuta dok god je pokrenuta nit reactionThread.
- Kreirati metodu **private void OnCoreTemperatureChanged(double temperature)** koja ispisuje poruku o trenutnoj temperaturi jezgra reaktora, a poziva se prilikom promene temperature. Pored ispisa potrebno je i kreirati instancu SystemState klase.
- Kreirati metodu **private void OnCriticalTemperatureOccured()** koja ispisuje poruku o kritičnoj temperaturi jezgra reaktora.
- Kreirati metodu **public void Report()**, koja nakon završetka simulacije na konzolu ispisuje sve trenutke u kojima je bilo otvoreno barem 2 ventila, pumpa je bila upaljena, pumpa je bila upaljena, ukupan protok ventila bio je veći od 20 a temperatura jezgra je bila manja od 120 stepeni.
- U Main-u kreirati objekte svih navedenih klasa. Svi ventili su zatvoreni. Pokrenuti hemijsku reakciju pozivom metoda StartReactionSimulation i StartRandomisation, sačekati dve sekunde, a potom pozvati metodu StartWaterCoolingSimulation, čime započinje vodeno hlađenje jezgra.
- Nakon gašenja reactionThread niti, pozvati Report() metodu.