ZH3 – Siege Simulator

Bevezető

Egy ostrom alatt álló vár védelmét szimuláljuk a lehető legegyszerűbb módon. Az ellenfél csupán egyetlen számként van reprezentálva. Az ellenség nem támad, a várvédők viszont nyilakat lőnek a támadókra. Minden nyílvessző eggyel csökkenti a támadókat szimbolizáló értéket. Ha ez eléri a nullát, akkor véget ér a szimuláció.

Mellékletként megtaláljátok a fentebb leírt pofon egyszerű program egyszálas implementációját. Az ***enemy*** változó megmondja hány nyílvesszőre van még szükség a támadók legyőzéséhez, míg a ***tickRate*** az egy egységnyi időt jelzi ezredmásodpercekben kifejezve. Ez utóbbira épül a ***delay(int ticks)*** metódus, mely adott egységnyi ideig megvárakoztatja a hívót.

Egyszerre 10 nyílvesszőt tudunk kilőni. Ezek mindegyike 1 egységnyi idő alatt ér célba. Két nyílzápor között 5 egységnyi időnek kell eltelnie. Mivel a program szekvenciális, ezért összesen 10*1+1*5=15 egységnyi ideig tart egy nyílzápor. Ez lassú!

Az első feladat megoldása után a további feladatok tetszőleges sorrendben megoldhatóak. A feladatok nem épülnek az előző feladatokra.

Az egyes feladatok pontozása nem a nehézséget, hanem a megoldáshoz szükséges időt mutatja. A feladatok olyan sorrendben szerepelnek, ahogy a megoldásukhoz szükséges ismeretanyagot sajátítottátok el a félév során. Tehát a későbbi feladatokhoz bonyolultabb, a félév vége fele tanult koncepciók és nyelvi elemek szükségesek. Emiatt érdemes – bár nem kötelező – a feladatokkal sorrendben haladni.

Feladat 1: Multithreaded Volley – 10 pont (könnyű)

Minden egyes nyílvessző külön szálon repüljön, majd ugyanez a szál vonjon le az ***enemy***-ből egyet, illetve biztosítsa a képernyőre való kiiratást a nyílvessző célba érésekor, hasonlóan mint ahogy az egyszálas implementációban szerepel.

Pontozás

5 pont: *Továbbra is 5 egységnyi idő kell egy nyílzápor megkezdéséhez, és továbbra is 1 egységnyi idő kell 1 nyílvessző célba juttatásához, de az egyes nyílvesszők immáron a háttérben, külön szálakon repdesnek ezért 15 egységnyi idő helyett 5 egységnyi idő kell egy újabb nyílzáporhoz.*

5 pont: *A* ***tickRate*** *és az* ***enemy*** *közös erőforrás a szálak között. Oly módon végezzük a párhuzamosítást, hogy továbbra is thread-safe maradjon az alkalmazás. Ehhez szükség lehet a* ***final***, ***volatile***, ***synchronized*** *kulcsszavakra, valamint az* ***Atomic*** *osztályokra is. Természetesen a felsorolásból nem szükséges mindent használni mindaddig amíg az implementáció helyes.*

Feladat 2: BlockingQueue – 10 pont (közepes)

Az első feladat megoldását követően feltűnhet, hogy a párhuzamosítás hatására kissé megkeveredett a kiiratási sorrend. Ezt tegyük rendbe!

A kiiratási munkákat bízzuk egy külön, kifejezetten erre kitalált szálra. Ez egy ***BlockingQueue***-n keresztül kommunikáljon azokkal a szálakkal, melyek üzenetet szeretnének a képernyőre vinni. Tehát

azok a szálak melyek eddig egyből a képernyőre írtak mostantól tegyék az üzenetüket ebbe a konténerbe.

A kiiratós szál induljon el a program legelején és a program futásának végéig vegye ki a közös konténerből az üzeneteket és írja ki azokat a képernyőre!

Pontozás

5 pont: *Megfelelően működik a queue és a kiiratós szál, azaz a program elején elindul, a végén terminál, a kettő között pedig a más szálak által átadott üzeneteket képernyőre viszi.*

2 pont: *Elintéztük továbbá, hogy az egyes üzenetek időrendi sorrendben kerüljenek a queue-ba, ezért most már szigorúan monoton csökkenő sorozatot látunk futtatáskor a képernyőn.*

1 pont: *A kiiratós szál leállásához nem támaszkodunk más szálra (pl. nem kell interrupt vagy flag átbillentés a main-ből). Legfeljebb egy egységnyi idő alatt terminál, ha vége a szimulációnak.*

1 pont: *Üresjáratban nem égeti a CPU-t.*

1 pont: *Reszponzív, azaz ha nem üres a queue, akkor azonnal kiírja a következő üzenetet.*

Feladat 3: wait-notify – 10 pont (nehéz)

Titkos alagútrendszeren keresztül 10 egységnyi időnként sikerül egy ágyúgolyót és egy kis puskaport a várba csempészni. A várban van ágyú, ezért ezt azonnal használni is tudjuk.

A csempészésért és az ágyú kezeléséért egy-egy szál feleljen. Az ágyúgolyókat és puskaport felhalmozni nem lehet, ezért elég egy 2 állású flag berakása a kódba. A csempész 10 egységnyi időnként ezt állítsa igazra, majd jelezze (notify) az ágyút kezelő szál felé, hogy tud tüzelni.

Nincs elég ember ahhoz, hogy állandóan az ágyú mellett álljon valaki, ezért amikor épp nincs ágyúgolyó és puskapor akkor pihentetjük a védekezés ezen elemét (wait). Egy 1 ágyúgolyó 10 nyílvesszőnek felel meg.

Tüzelést követően a nyílvesszőkhöz hasonlóan írjunk a képernyőre! A nyílvesszőkkel ellentétben az ágyúgolyónak nincs repülési ideje, azonnal célba ér, ezért egyből le lehet vonni az értékét.

Pontozás

5 pont: *Megfelelően működik a csempész és ágyúkezelő szál, azaz a program elején elindulnak, a végén terminálnak, a kettő között pedig az egyik 10 egységnyi időnként igazra billent egy flag-t, majd szól a másiknak, aki erre azt visszabillenti és tüzel az ágyúval.*

2 pont: *A csempésznek bár 10 egységnyi idő kell, soha ne várjon egyhuzamban ennyi ideig. Legfeljebb 1 egységnyi idő után vegye észre, ha vége a szimulációnak, de a CPU-t se égesse folyamatosan.*

2 pont: *Az ágyúkezelő néha magától is vessen pillantást az ágyú helyzetére. Legfeljebb 1 egységnyi idő késéssel akkor is vegye észre, hogy lehet tüzelni, ha a csempész elfelejtene neki szólni. Tehát ne arra alapozzuk a kódunkat, hogy „majd úgyis szólnak”, a wait nem erre való, hanem a flag értékét vizsgáljuk meg bizonyos időközönként. A wait-notify csupán a CPU égetés (folyamatos polling) és rossz reszponzivitás (sleep) elkerülése miatt szükségeltetik.*

1 pont: *Mindkét szál egyedül, más szálak segítsége nélkül legyen képes 1 egységnyi idő alatt leállni.*

Feladat 4: CyclicBarrier – 5 pont (nehéz)

Az egyes nyílvesszők ne egyesével vonogassanak le az ***enemy***-ből! helyette egyetlen nagy nyílzáporként gyűljenek össze a támadók fejei felett, majd együtt vonjuk le a nyilak utáni értékeket.

Ehhez használjunk ***CyclicBarrier***-t! Maguk a nyílvesszők repülhetnek a megszokott módon, de levonás és kiiratás helyett a barrier előtt gyülekezzenek, majd áttöréskor az utoljára érkező szál egyben vonjon le 10 nyílvesszőnyi értéket és erről egyetlen kiiratásban tudósítson.

Pontozás

3 pont: *feladat részleges megoldása az egyszerűbb* ***CyclicBarrier(int parties)*** *konstrukorral. Elég, ha a nyílvesszők levonás és kiiratás előtt megvárják egymást, nem kell közös levonás és kiiratás.*

2 pont: *A részleges megoldás befejezése a* ***CyclicBarrier(int parties, Runnable barrierAction)*** *konstruktorral. Ekkor a* ***barrierAction*** *segítségével a teljes feladatot meg tudjuk valósítani, azaz a közös levonást és kiiratást is.*

Feladatsorral kapcsolatos kérdés esetén az elérhetőségem: menczer.andor@gmail.com