Inlämningsuppgift

Reflektera över trådhantering

Tim Dahl - 2024-03-21.
Syne23.
TUC Yrkeshögskola.
Linköping.

A. Du har skrivit ett program som skriver ut talen 1-500 genom olika trådar.

1. **Hur blir utskriften?**
   Utskriften blir oordnad och varierar varje gång programmet körs för att flera trådar arbetar parallellt utan synkronisering.

2. **Får du samma resultat varje gång du kör programmet?**
   Resultat kommer skilja sig åt varje gång programmet körs eftersom olika trådar kan få tillgång till resurserna i olika ordning. Så nej, resultatet kommer inte vara samma varje gång programmet körs.

3. **Varför blir resultatet så?**
   Resultatet är oordnat eftersom flera trådar arbetar samtidigt och konkurrerar om resurser utan att ha någon specifik ordning för att skriva ut talen.

B. Du har också skrivit ett program som asynkront hämtar data från olika webbsidor.

1. **Vad är skillnaden mellan att använda async/await och att använda multitrådning?**
   Async/Await användning är mest lämplig för att hämta till exempel webbsidor, databaser eller filsystem. När man använder Async/Await blockerar inte tråden medan en uppgift ska slutföras, istället kan tråden fortsätta att utföra andra uppgifter eller vara ledig att hantera andra förfrågningar.
   Multitrådning innebär att flera trådar körs parallellt för att utföra uppgifter samtidigt. Vilket kan vara användbart för beräkningsintensiva uppgifter eller när du behöver utnyttja flera processorkärnor för att förbättra prestanda. Med **Multitrådning** måste man hantera synkronisering och konkurrens manuellt för att undvika problem.

2. **När anser du att man ska använda async/await, och när ska man istället använda multitrådning?**

Async/Await kan man använda när man gör uppgifter som till exempel att hämta data från webbsidor, databaser eller filsystem.

Multitrådning kan man använda när man har uppgifter som kan dra nytta av att köra parallellt för att förbättra prestandan, eller om man vill utnyttja flera processorkärnor för att utföra uppgifter samtidigt.

C. Slutligen fick du till uppgiften att skriva ett program som adderar ihop talen 1-500.

1. **Varför fick du ett System.IO.IOException första gången du körde programmet?**
   Fick System.IO.IOException första gången jag körde den för att flera uppdrag försöker komma åt samma operation eller fil samtidigt.

2. **Vilket tal fick du ut i filen när programmet väl körde klart?**
   Talet jag fick ut när jag körde programmet var 124 750, då jag körde PrintNumbers(int a, int b) som kör talen mellan 1-500, vilket betyder att den inte tar med 500. Jag testade även om programmet skulle ta med 500 och då blev talet 125 000.

3. **Vad är skillnaden mellan dessa?**
   Lock är en lättviktig konstrukt som bara fungerar inom en applikation domän och används för enklare synkroniserings behov, medans **Mutex** är en mer kraftfull synkroniserings mekanism som fungerar över process gränserna och kan användas när flera processer behöver synkronisera åtkomst till gemensamma resurser.

4. **Ge ett exempel vardera på när man bör använda lock respektive Mutex.**
   Lock används när synkronisering krävs för att hantera åtkomst till delad resurs inom en trådgrupp, till exempel när flera trådar behöver skriva till eller läsa från en delad lista i en flertrådig applikation, medans **Mutex** används när synkronisering krävs för att hantera åtkomst till delade resurser mellan olika processer, till exempel när flera instanser av en applikation måste synkronisera åtkomst till en gemensam fil på disk.

5. **Utöver dessa finns också klassen Semaphore. Beskriv vad den gör och hur man använder den?**
   Semaphore är en synkronisering mekanism som begränsar antalet trådar som kan få tillgång till en resurs samtidigt. Den används för att implementera en gräns för antalet trådar som kan få komma åt en resurs samtidigt**.**