Vad är en Tråd (eng. Thread)?

- Exekveringsflöde i ett program
- System med en processor använder schemaläggning av trådexekvering
- En tråd i Java skapas genom att subklassa klassen **`Thread`** eller implementera interfacet **`Runnable`**

Klassen `Thread`

- Används för att bl.a. skapa en tråd och kontrollera trådar
- Subklasser åsidosätter (eng. override) metoden **`run()`**
- Metoden `run()` innehåller den kod som kommer att exekveras när tråden startas
- Metoden **`start()`** kör igång tråden

```
public class Spinner extends Thread {

      public void run(){
          System.out.println("Spinning....");
      }
}

//Skapar en ny Spinner-tråd
Thread spinner = new Spinner();

//Startar tråden spinner
spinner.start();
```

Interfacet `Runnable`

- Implementeras av klasser som skall kunna exekveras av en separat tråd
- Klasser som implementerar `Runnable` **är i sig själv inte en tråd**
- Definerarar bara en metod: **`run()`**
- Klasser som implementerar `Runnable` låter metoden `run()` innehålla den kod som skall exekveras när denna klass körs av en tråd
- Klass som implementerar `Runnable` skickas som argument när en ny `Thread` skapas

```
public class Messenger implements Runnable {
      public void run(){
          System.out.println("This is a message.");
      }
}

//Skapar en ny tråd innehållandes ett Messenger-objekt och startar den
Thread messenger = new Thread(new Messenger());
messenger.start();
```

Viktiga metoder i klassen `Thread`

- **`run()`** - Anropas automatiskt när tråden startas
- **`start()`** - Startar exekveringen av en tråd
- **`sleep(long millis)`** - Ser till att den exekverande tråden sover för angivet antal millisekunder
- **`yield()`** - Får den tråd som för tillfället exekverar att stanna upp och ge andra trådar möjligheten att exekvera
- **`interrupt()`** - Avbryter en exekverande tråd
- **`isInterrupted()`** - Kontrollerar om denna tråd är avbruten
- **`isAlive()`** - Kontrollerar om denna tråd är vid liv, dvs har påbörjat men inte avslutat sin exekvering
- **`setPriority(int p)`** - Sätter prioritetsvärdet för denna tråd

Metoder i klassen `Object` som har med trådar att göra

- **`wait()`** - Får tråden att vänta till dess att notify eller notifyAll anropas för tråden
- **`notify()`** - Väcker en av trådarna som väntar på detta objekt
- **`notifyAll()`** - Väcker alla trådar som väntar på detta objekt

Trådars tillstånd

- En tråd kan vara i ett av följande tillstånd:

  New - När tråden är instansierad men start() inte har anropats

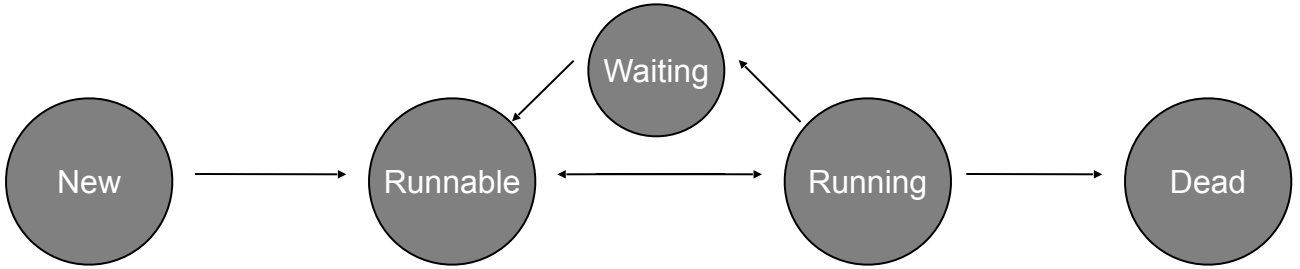
  Runnable - När tråden är berättigad att köras men har ännu inte valts att vara den exekverande tråden

  Running - Tillståndet när tråden exekveras

  Waiting - Tråden kanske väntar på att en resurs skall frigöras, dess run() kan ha anropat sleep() eller wait()

  Dead - När trådens run() har exekverat klart

- Tråden anses vara **alive** när den är i: **Runnable, Running, Waiting**



Synkronisering

- Används för att bara ge en tråd i taget tillgång till en viss kod
- För att markera att en metod är synkroniserad används nyckelordet **`synchronized`**
- Om en tråd börjar exekvera en metod som är synchronized kommer ingen annan tråd att kunna exekvera den metoden eller någon annan metod som är markerad synchronized för det objektet
- Metoder som inte är synchronized kommer fortfarande att gå att exekvera av andra trådar under tiden en annan tråd exekverar en synchronized-metod
- Det går också att synkronisera ett kodblock
- Även static-metoder går att synkronisera

```
//Metod som är synchronized
public synchronized void sendMail(){
}

//Kodblock som är synchronized
public void sendMail(){
       synchronize(this){
       }
}

//Static-metod som är synchronized
public static synchronized void sendMail(){
}
```

Synkronisering forts.

- Varje objekt har ett s.k. lås (eng. **lock**)
- När en tråd exekverar en synchronized-metod får den tråden objektets lås
- Bara den tråd som har låset kan exekvera en synchronized-metod på aktuellt objekt
- Om låset redan är upptaget av en tråd får andra trådar vänta på att den första tråden återlämnar låset. Detta sker när exekveringen av metoden är klar

Allmänt om Threads

- Nyckelordet synchronized anses inte vara en del av en metods signatur därför ärvs den inte av subklasser
- Metoder i ett interface kan inte deklareras som synchronized
- Konstruktorer kan inte deklareras som synchronized eller innehålla block-synkroniseringskod

Övning

`Thread`:

1. Skapa en klass som subklassa `Thread`
2. I subklassens public void `run()` -metod skall den aktuella tiden skrivas ut en gång per sekund

`Runnable`:

1. Skapa en klass som implementerar interfacet Runnable
2. I `public void run()` skall den aktuella tiden skrivas ut en gång per sekund
3. Skapa en ny tråd som instasieras med ett objekt av klassen som implementerar `Runnable`