Olio-ohjelmointi: prosessi ja säie

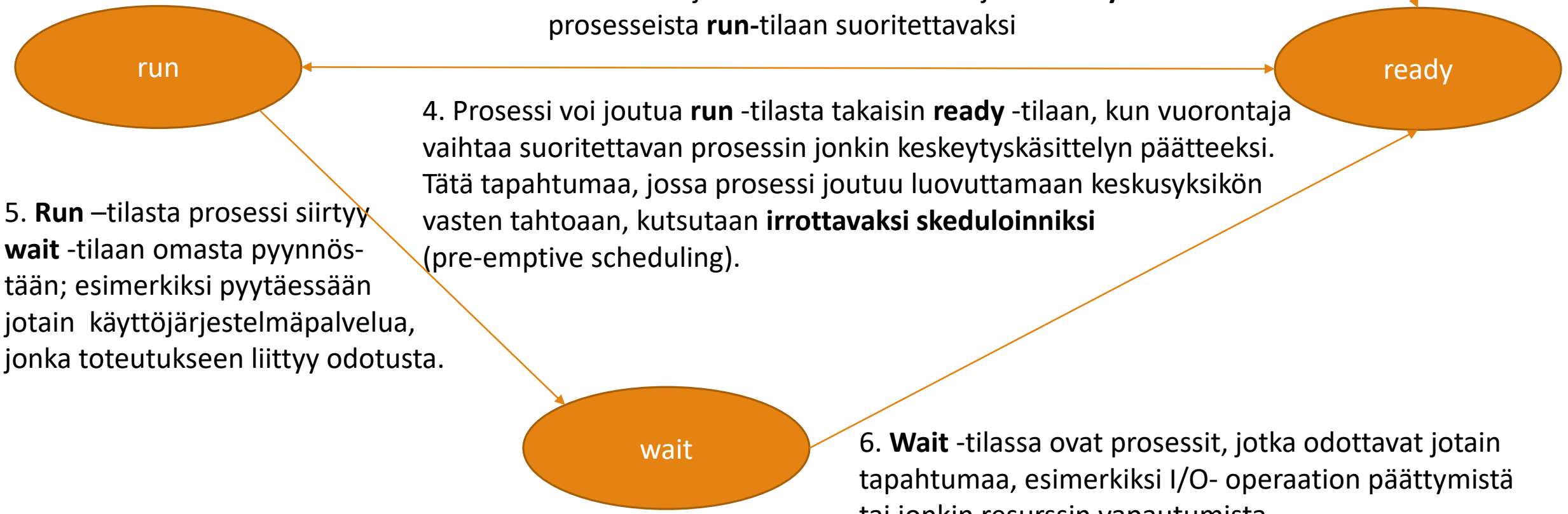
- Tietotekniikassa prosessi on tietokoneessa ajossa oleva ohjelma.
- Jokaisella prosessilla on yleensä käytössä oma muistialue ja muita resursseja, joihin muut prosessit eivät pääse käsiksi.
- Useimmat käyttöjärjestelmät ylläpitävät näitä tietoja prosessitaulussa.
- Usein prosessit koostuvat yhdestä tai useammasta säikeestä.
- Säie eroaa prosessista siten, että sillä ei ole omia resursseja, vaan se käyttää sen prosessin resursseja, johon se kuuluu.
- Säikeiden avulla voidaan tietokoneessa ajaa moniajoa(=rinnakkaisuus), jolloin eri prosessien eri säikeet tekevät jokainen omaa tehtäväänsä

Olio-ohjelmointi: prosessi ja säie

1. Prosessi luotu/käynnistetty
Odottaa pääsyä run tilaan

3. **Run** tilassa ollaan saatu oma koodin pätkä suoritukseen prosessorille.

2. Vuorontajan tehtävänä on valita jokin **ready** -tilan prosesseista **run**-tilaan suoritettavaksi



4. Prosessi voi joutua **run** -tilasta takaisin **ready** -tilaan, kun vuorontaja vaihtaa suoritettavan prosessin jonkin keskeytyskäsittelyn päätteeksi. Tätä tapahtumaa, jossa prosessi joutuu luovuttamaan keskusyksikön vasten tahtoaan, kutsutaan **irrottavaksi skeduloinniksi** (pre-emptive scheduling).

5. **Run** –tilasta prosessi siirtyy **wait** -tilaan omasta pyynnöstään; esimerkiksi pyytäessään jotain käyttöjärjestelmäpalvelua, jonka toteutukseen liittyy odotusta.

6. **Wait** -tilassa ovat prosessit, jotka odottavat jotain tapahtumaa, esimerkiksi I/O- operaation päättymistä tai jonkin resurssin vapautumista

Olio-ohjelmointi: virtuaalifunktiot, prosessi ja säie

- Avaa selaimeen sivu www.qt.io ja siirry sivun loppuun ja klikkaa otsikon **Developers** alapuolella olevaa linkkiä **Documentation**

- Kirjoita hakukenttään sana **QThread**

- Avaa hakulistalta linkki (**QThread Class | Qt Core …**)

- Mitä otsikkotiedosto kuuluu luokkaan, mikä paketti täytyy olla mukana **.pro** tiedostossa, minkä luokan **QThread** perii?

- Käydään opettajan ohjaamana luokan **QThread** perusajatus ja seuraavat osat:
  - Public Functions
  - Protected Functions, erityisesti funktio **virtual void run()**
  - Public Slots, erityisesti funktio **void start(Priority priority = InheritPriority)**
  - Static Public Members, erityisesti funktio **void msleep(unsigned long msecs)**

Olio-ohjelmointi: virtuaalifunktiot, prosessi ja säie

- Tee uusi projekti **MyThreadApp (**tyyppi **Application->Qt Console Application)**
- Lisää ohjelmaan luokka **MyFirstThreadClass**
- Kun luokka on lisätty projektiin, niin avaa projektin tiedosto **myfirstthreadclass.h**, ja laita luokka perimään **public** tasolla **QThread** luokka. *Huom. Muista* ***#include*** *lause!*
- Kun olet tarvittavat koodit kirjoittanut ohjelmaan, niin suorita **Build** (älä aja ohjelmaa!).
- Lisää luokan **protected** osaan funktio **virtual void *run*() override;** (**run()** funktio kuuluu luokassa **QThread** osaan **protected**)
- Lisää funktion **run()** toteutuksen tyhjä runko tiedostoon **myfirstthreadclass.cpp.** Suorita **Build** (älä aja ohjelmaa!).
- Lisää ohjelmaan luokka **MySecondThreadClass** ja lisää siihen samat rakenteet kuin luokkaan **MyFirstThreadClass**
- Suorita **Build** (älä aja ohjelmaa!).

Olio-ohjelmointi: virtuaalifunktiot, prosessi ja säie

- Avaa **main.cpp**
- Esimerkissä oliot luodaan harjoituksen vuoksi pinomuistiin.
- Luo **main()** funktiossa luokasta **MyFirstThreadClass** olio **objectMyFirstThreadClass** käyttäen pinomuistia.
- Käynnistä olion avulla luokan **run()** funktion toteutus kutsumalla oliolla **start()** jäsenfunktiota.
- Luo **main()** funktiossa luokasta **MySecondThreadClass** olio **objectMySecondThreadClass** käyttäen pinomuistia.
- Käynnistä olion avulla luokan **run()** funktion toteutus kutsumalla oliolla **start()** jäsenfunktiota.

Olio-ohjelmointi: virtuaalifunktiot, prosessi ja säie

- Tee luokan **MyFirstThread** funktiosta **run()** alla olevan mukainen

```
void MyFirstThreadClass::run()
{
   for (int i=0;i<10;i++)
   {
      qDebug() << "MyFirstThread =" << i;
      // Staattista jäsenfunktiota voidaan kutsua aivan kuten tavallisiakin jasenfunktioita, tai sitten voidaan
      // kutsua alla olevan esimerkin mukaisesti
       QThread::msleep(1000);
   }
   qDebug() << "MyFirstThread finished";
}
```

Olio-ohjelmointi: virtuaalifunktiot, prosessi ja säie

- Tee luokan **MySecondThread** funktiosta **run()** alla olevan mukainen

```
void MySecondThreadClass::run()
{
   for (int i=0;i<10;i++)
   {
      qDebug() << "MySecondThread =" << i;
      // Staattista jäsenfunktiota voidaan kutsua aivan kuten tavallisiakin jasenfunktioita
     msleep(2000);
   }
   qDebug() << "MySecondThread finished";
}
```

- Suorita build ja aja ohjelmaa. Seuraa mitä ohjelma tulostaa näytölle! Kumpi säie pääse suorituksensa loppuun ensin ja miksi?

- ***HUOM! Ohjelmaan ei rakenneta tällä kertaa SIGNAL/SLOT toiminnallisuutta, joten prosessi ei pääty oikein!***

- Osan 8 kotitehtävässä rakennetaan sovellus, jossa säikeiden avulla suoritetaan ohjelmassa toimintoja ja mukana on myös SIGNAL/SLOT toiminnallisuus, jotta ohjelma/prosessi päättyy oikein.