IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 1, vastauksia kysymyksiin opiskelumateriaalin osasta 2:

a) Mitä tarkoittaa ohjelmointiparadigma?

Ohjelmointiparadigma on ohjelmointikielen taustalla oleva tapa ajatella ja mallintaa annettuun ongelmaan ratkaisu.

b) Miten olio-ohjelmoinnin paradigma eroaa proseduraalisesta paradigmasta?

Olio-ohjelma muodostuu kokoelmasta yhteistyössä toimivia oliota, kun taas perinteinen proseduraalinen tietokoneohjelma on funktioiden, muuttujien ja tietorakenteiden joukko.

c) Mitkä ovat olio-ohjelmoinnin neljä tärkeintä perusperiaatetta?

Olio-ohjelmoinnin neljä tärkeintä perusperiaatetta: kapselointi, tiedon kätkentä, perintä ja polymorfismi.

d) Mitä tarkoittaa käsite luokka?

Luokka on yhteenliitetty joukko erilaisia jäsenmuuttujia, luokan toiminnallisuudesta vastaavia jäsenfunktioita ja lisäksi luokalla on yhteyksiä muihin luokkiin ja olioihin erilaisilla olio-ohjelmointiin kuuluvilla yhteysmuodoilla.

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

e) Mitä tarkoittaa käsite olio?

Olio on luokan ilmentymä, eli olio luodaan luokasta. Olio on se joka suorittaa kaikki toiminnallisuudet sovelluksessa, käyttäen niitä rakenteita/määrityksiä joita luokassa on (tiedot, toiminnot, yhteydet …).

f) Mitä tarkoittavat käsitteet kapselointi ja tiedon kätkentä?

Kapselointi on ohjelmointitekniikka, joka määrää tietojen ja toteutustavan piilotuksen periaatteet. Sen tarkoitus on **uudelleenkäytettävyyden** lisääminen. Esimerkiksi valmis ohjelmistokomponentti (**esim. kirjautumisliittymä**) voidaan ottaa mukaan mihin tahansa sovellukseen, kunhan ohjelmoija tietää miten komponentin luokkia ja olioita käytetään. Komponentin sisäistä toteutusrakennetta ei tarvitse tietää. **Kapseloinnin** avulla voidaan sovelluksessa koota yhdeksi kokonaisuudeksi eli yhdeksi ohjelmistokomponentiksi yhteen kuuluvat tiedot ja toiminnot.

Tiedon kätkennällä tarkoitetaan sitä, että kapseloitua osaa (esim. luokkaa) voidaan käyttää tuntematta luokan sisäistä rakennetta. Luokan jäsenmuuttujien osalta tarkoitetaan sitä, että luokan jäsenmuuttujat tehdään sellaiseksi, että niitä voidaan käsitellä vain luokan jäsenfunktioissa.

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

g) Mitkä ovat C++ ohjelmointikielessä tiedon kätkentään liittyvät näkyvyystasot?

C++ -ohjelmoinnissa tiedon kätkentä esitetään luokassa jakamalla luokan näkyvyystasot osiin public, protected ja private.

h) Mille tasolle jäsenmuuttujat lähtökohtaisesti laitetaan ja mille tasolle jäsenfunktiot lähtökohtaisesti laitetaan?

*jäsenmuuttujat ovat lähtökohtaisesti **private** tasolla, näin jäsenmuuttujiin ei pääse käsiksi olion ulkopuolelta. Tämä tarkoittaa sitä, että vain luokan jäsenfunktioiden avulla voidaan muuttaa jäsenmuuttujien tietoja.*

*Jäsenfunktiot ovat lähtökohtaisesti julkisia (eli ovat **public** tasolla).*

i) Mitä aliluokka perii yliluokalta?

Periytymisessä aliluokka perii yliluokan jäsenmuuttujat, -funktiot ja yhteydet muihin luokkiin. Aliluokassa voidaan yliluokasta perittyjä ominaisuuksia kumota eli määritellä uudestaan.

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 2, vastaus: Jäsenfunktioiden ylikuormittaminen, polymorfismi, MyOverloading projekti

main.cpp

```
#include "printdata.h"

int main()
{

    PrintData objectPrintData;

    objectPrintData.print(10);
    objectPrintData.print(20.10);
    objectPrintData.print("moi");

    return 0;
}
```

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 2, vastaus: Jäsenfunktioiden ylikuormittaminen

printdata.h

```
#ifndef PRINTDATA_H
#define PRINTDATA_H

#include <iostream>
using namespace std;

class PrintData
{
public:
   void print(int parameter);
   void print(double parameter);
   void print(char *parameter);
};

#endif // PRINTDATA_H
```

printdata.cpp

```
#include "printdata.h"

void PrintData::print(int parameter)
{
   cout << "Kokonaisluku" << endl;
   cout << parameter << endl;
}

void PrintData::print(double parameter)
{
   cout << "Desimaaliluku" << endl;
   cout << parameter << endl;
}

void PrintData::print(char *parameter)
{
   cout << "Merkkijono" << endl;
   cout << parameter << endl;
}
```

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 3, vastaus: Nelilaskin

main.cpp

```
#include "standardcalculator.h"

int main()
{
    StandardCalculator objectStandardCalculator;

    objectStandardCalculator.sum(10,5);
    objectStandardCalculator.subtraction(10,5);
    objectStandardCalculator.multiplication(10,5);
    objectStandardCalculator.division(10,5);

    return 0;
}
```

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 3, vastaus: Nelilaskin, **MyStandardCalculator** projekti

standardcalculator.h

```
#ifndef STANDARDCALCULATOR_H
#define STANDARDCALCULATOR_H

#include <iostream>        // tarvitaan, koska tulostusolio cout on määritelty tässä kirjastossa
using namespace std;

class StandardCalculator
{
public:
   void sum(double parameterNumberOne, double parameterNumberTwo);
   void subtraction(double parameterNumberOne, double parameterNumberTwo);
   void multiplication(double parameterNumberOne, double parameterNumberTwo);
   void division(double parameterNumberOne, double parameterNumberTwo);

   void showCalculationAndResult(char parameterCalculation);
private:
   double numberOne, numberTwo, calculationResult;
};

#endif // STANDARDCALCULATOR_H
```

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 3, vastaus: Nelilaskin

standardcalculator.cpp

```
#include "standardcalculator.h"

void StandardCalculator::sum(double parameterNumberOne, double parameterNumberTwo)
{
    numberOne = parameterNumberOne;
    numberTwo = parameterNumberTwo;
    calculationResult = numberOne+numberTwo;
    showCalculationAndResult('+');
}

void StandardCalculator::subtraction(double parameterNumberOne, double parameterNumberTwo)
{
    numberOne = parameterNumberOne;
    numberTwo = parameterNumberTwo;
    calculationResult = numberOne-numberTwo;
    showCalculationAndResult('-');
}

void StandardCalculator::multiplication(double parameterNumberOne, double parameterNumberTwo)
{
    numberOne = parameterNumberOne;
    numberTwo = parameterNumberTwo;
    calculationResult = numberOne*numberTwo;
    showCalculationAndResult('*');
}
```

Jatkuu seuraavalla sivulla …

IN00BQ93 Laite- ja tuotesuunnittelun syventävät opinnot: ohjelmoinnin jatkokurssi

Osa 2: kotitehtävien vastauksia

Tehtävä 3, vastaus: Nelilaskin

standardcalculator.cpp

```
void StandardCalculator::division(double parameterNumberOne, double parameterNumberTwo)
{
   numberOne = parameterNumberOne;
   numberTwo = parameterNumberTwo;
   calculationResult = numberOne/numberTwo;
   showCalculationAndResult('/');
}

void StandardCalculator::showCalculationAndResult(char parameterCalculation)
{
   cout << numberOne << parameterCalculation << numberTwo << "=" << calculationResult << endl;
}
```