58160 Ohjelmoinnin harjoitustyö

Johdanto
9.3.2009

Tuntiop. Sami Nikander
sami.nikander@helsinki.fi

Aloitusluento

I Kurssin sisältö: Mitä vaaditaan?

II Työvaiheet: Mitä tehdään?

III Työvälineet: Millä tehdään?

IV Java-vinkkejä: Miten tehdään?

V Ryhmiinjako ja jonottajien karsinta

Ohjelmoinnin harjoitustyö

- **Suunnittele, toteuta ja dokumentoi vuorovaikutteinen sovellus (~vapaavalintaisesta aiheesta)**
- **Laajuus n. 500-2000 koodiriviä, dokumentaatio n. 10-30 sivua (ml. kuvat)**
- **Suoritustapa: n. 6 viikon itsenäinen työskentelyjakso (myös ryhmätapaamisia)**

Asema opinnoissa

- **Ohjelmoinnin harjoitustyö kuuluu tietojenkäsittelytieteen perus- ja aineopintoihin**
- **Pääaineopiskelijalle pakollinen**
- **Pakolliset esitiedot:**
  (tarkistetaan opintosuoritusotteesta)
  - **Ohjelmoinnin perus- ja jatkokurssi**
  - **Ohjelmistojen mallintaminen (tai vastaava)**

Kurssin tavoitteet

- **Ohjelmointityöskentelyn harjoittelu**
  - suunnittelu
  - dokumentointi
  - ohjelmointi
  - testaus
  - ”tuotteen” viimeistely
- **Projektityöskentelyn harjoittelu**
  - ratkaisuista neuvotteleminen ja niiden demoaminen
  - aikatauluissa pysyminen
- **Itsenäisen työskentelyn harjoittelu**
  - Ohjelmoinnin jatkokurssin ja Ohjelmistojen mallintamisen oppien omatoiminen soveltaminen

Arvosteluperusteet

Dokumentaatio 40%

12p Tekninen dokumentti
6p Käyttöohje
6p Kieliasu, jäsentely

Työprosessi 10%

6p Aikataulussa pysyminen, säntillisyys, kommunikointi

Yht. 60 p

Toteutus 40%

12p Toiminnallisuus, luotettavuus, käytettävyys
12p Tehokkuus, ylläpidettävyys, siirrettävyys

Testaus 10%

6p Systemaattisuus, kattavuus

Pisterajat	30	35	40	45	50
Arvosanat	1	2	3	4	5

Tuotokset

- **Valmis, asennuskelpoinen ajettava ohjelma**
  - Ohjelman toimittava TKTL:n Linux/Windows-koneissa
  - Palautetaan lähdekoodi
  - Paketoidaan tarvittavat tiedostot yhdeksi tiedostoksi (esim. zip, gz, tar, jar...)
- **Dokumentaatio (15-40 s?)**
  - Aiheen rajaus, määrittelydokumentti
  - Käyttöliittymäsuunnitelma
  - Ratkaisujen perustelut, algoritmien ym. kuvaus
  - Luokkakaavio, kutsukaavio, luokkien tarkempi kuvaus
  - Testaussuunnitelma ja testiraportti
  - Käyttöohje
  - Asennusohjeet, ylläpito-ohjeet ym.

Dokumentoinnista...

- **Ohjaajasi päättää vaadittavan dokumentaation muodon (ryhmäkohtaisia pieniä eroja)**
- **Dokumentaatiosta suuri osa kannattaa tehdä suunnitteluvaiheessa**
- **Dokumentaatiolla on suuri vaikutus arvosteluun. Jätä dokumentin kirjoittamiselle riittävästi aikaa!**
- **Palautettava paperimuodossa (arkistointia varten)**

Ohjaus ja ryhmäopetus

- **Henkilökohtainen ohjaus**
  - N. 15 min / vko, pakollista (sovi poikkeukset erikseen)
  - Työ tulkitaan keskeytetyksi, jos et ota yhteyttä viikkoon
  - Jos työ ei ole edistynyt, tule silti vastaanotolle
  - Ohjaajalla 2 roolia: asiakas / opettaja. Asiakas valvoo, että
    - ohjelma täyttää vaatimukset
    - projekti pysyy aikataulussa
- **Työn esittely ryhmässä**
  - vko 3 (suunnitelmat) ja vko 6 (valmis ohjelma)
  - omassa harjoitusryhmässä, n. 10 min/esitys
- **Neuvontapäivystys (työviikot 4-6)**
  - Ks. ajankohdat --> http://www.cs.helsinki.fi/opintoneuvonta/
  - 2. krs, B- ja C-siipien välisessä syvennyksessä
  - “drive in”-neuvontaa, tule ja kysy vapaasti

Rankka projekti!

- **4 op (2 ov) = 80 h työtä!**
- **80 h / 6 vkoa = 3 h / vrk (arkisin)**
- **Työmäärä vaihtelee yksilöllisesti, n. 40-120h**
  - **Jos esitiedot hatarat/ruosteessa, työtä on enemmän**
  - **Jos osaat koodata ennestään, työtä *voi olla* vähemmän**
- **Jos pääosa työstä jää viimeisen 2 vkon ajalle: 8h päivässä viikon ympäri!**
- **Terveisiä kurssin suorittaneilta:**
  - **”Aloita nyt ihmeessä se homma ajoissa!”**
  - **”Ei tosiaan kannata jättää viime tinkaan!”**
  - **”Tein puolet tunneista yhdessä viikonlopussa”**

Rankka projekti!

- **Harjoitustyön keskeyttäneille uusi mahdollisuus vain jonotuslistan kautta!**
- **Jos voimat loppuvat kesken, keskeytä jokin muu kurssi, jolla ei ole keskeytyssanktioita**
- **Älä ”katoa” – aina voidaan neuvotella!**
- **Limeksen Älä Hätäile -opas: ”Harjoitustyön keskeyttäminen on ehkä tyhmin asia, minkä voit opiskeluaikanasi tehdä”**

Malliaikataulu

Vko	Työvaihe
1	Aihevalinta, määrittelyä, Java-kertausta
2	Oliosuunnittelua, koodauksen harjoittelua
3	Suunnittelua, alustavaa koodausta, demo
4	Koodausta, testausta, dokumentointia
5	Koodausta, testausta, dokumentointia
6	Viimeistelyä, testausta, demo

Aikataulu todellisuudessa? ;-)

Vko	Työvaihe
1	Yleistä pihallaoloa; aihevalinta helppo mutta miten jatkaa tästä eteenpäin?
2	Suunnittelu tökkii, Javaakaan ei ole ehtinyt opetella...
3	Oman raakileen demo hermostuttaa, ohjaaja pistää olio-suunnitelmat uusiksi, koko työ tuntuu käsittämättömältä
4	Koodaus osoittautuu luultua hankalammaksi...
5	Koodi ei toimi, dokumentti ei vastaa todellisuutta, kantapään kautta alkaa oppia. Kunpa olisi vielä 3 vkoa aikaa!
6	Proto ei toimi demossa. Viim. viikolla 3 yötä epätoivoista bugikorjausta ja Open Officen kanssa taistelua. Voitto!

Ohjeita verkossa

http://courses.cs.helsinki.fi/

-> Ohjelmoinnin harjoitustyö, kevät 2009

Ohjeita ja linkkejä, aihekuvaukset

Työn palautus (joissain ryhmissä)

http://www.cs.helsinki.fi/kurssit/perus/58160-8/

-> linkki Syksy 2005–

Kurssin viralliset tiedot:

neuvontapäivystysajat,

arvosteluperusteet

aihe-ehdotuksia ym.

Aloitusluento

I Kurssin sisältö: Mitä vaaditaan?

II Työvaiheet: Mitä tehdään?

III Työvälineet: Millä tehdään?

IV Java-vinkkejä: Miten tehdään?

V Ryhmiinjako ja jonottajien karsinta

Työvaiheet

- **Ohjelmistotuotannon perinteiset työvaiheet (vesiputousmalli):**
  1. (Kartoitus) ~ Aiheenvalinta
  2. Määrittely ~ Aiheen rajaus ja tarkennus
  3. Suunnittelu
  4. Toteutus
  5. Testaus
  6. (Käyttöönotto, ylläpito) ~ Käyttöohje, asennuspaketti
- **Todellisuudessa työvaiheita tehdään yleensä limittäin ja iteratiivisesti (myös tässä työssä)**
- **Dokumentointi osana jokaista työvaihetta**

Työvaiheiden ajoituksesta

- **Tyypillinen kokeneen ohjelmoijan virhe on sännätä heti koodaamaan, koska siihen on rutiinia**
  - Ohjelma voikin valmistua nopeasti, mutta se ei välttämättä täytä vaatimuksia ja sen jälkikäteen dokumentoiminen ei suju
- **Tyypillinen aloittelevan ohjelmoijan virhe on lykätä koodausta, koska siihen ei ole rutiinia**
  - Suunnitteluvaihe on oikea vaihe tehdä tutkimustyötä
  - On hyödyllistä kirjoittaa pieniä testikoodinpätkiä, joilla varmistaa ymmärtäneensä oppimansa oikein
- **Jos dokumentointi ei ole tuttua tai se on vastenmielistä, sitäkään ei kannata lykätä...**

Aiheenvalinta

- **Aihe-ehdotuksia**
  http://www.cs.helsinki.fi/group/alabra/Aiheita.html
  (linkki myös Moodlessa)
  - **Ajanviete- ja pulmapelejä**
    Miinaharava, Master Mind, muistipeli, laivanupotus...
  - **Hyötyohjelmia**
    Budjetinseuranta, HOPS-laskuri, CD-kortisto...
  - **Oma aihe**
    Tarkista ohjaajalta, soveltuuko ideasi harjoitustyöaiheeksi

- **Jos epäilet taitojasi, ota helppo aihe!**
  - Hyvin tehty helppo aihe tuottaa yleensä paremman arvosanan kuin viimeistelemätön, liian kunnianhimoinen työ

Aiheen rajaus

- **Täydennä valmiiksi annettu aihekuvaus itse: ne ovat (tarkoituksella!) hiukan ympäripyöreitä**
- **Kuvaa tehtävä/ongelma omin sanoin ja määrittele itse, mitä toimintoja, ominaisuuksia ja piirteitä ohjelmaan tulee mukaan**
  - Jos teet esim. laivanupotuksen, täsmennä millaisilla säännöillä ja minkä kokoisella ruudukolla peliä pelataan, onko kyseessä 1 vai 2 pelaajan peli jne.
  - Jos teet budjettiohjelman, päätä millaisia raportteja ohjelmasta pitää saada, mitä tietoa siihen syötetään jne.

Aiheen rajaus

- **Graafinen vai tekstipohjainen käyttöliittymä?**
  - Vapaavalintaista; molemmilla voi saada täydet pisteet
  - Tekstipohjainen vaatii vähemmän uuden opiskelua, mutta mielekkään/laajan tekstipohjaisen käyttöliittymän suunnittelu ja koodaus on Javalla työlästä
  - GUI vaatii Javan Swing-komponenttien opiskelua, mutta käyttöliittymäratkaisut ovat (osin) helpompia toteuttaa
- **Sovellus vai sovelma?**
  - Molemmat periaatteessa sallittuja, mutta...
  - Kurssivaatimuksiin kuuluu tiedostojen lukemista ja kirjoittamista, joissa sovelmalla on rajoitteita!
  - On mahdollista tehdä myös ohjelma, joka toimii kumpana tahansa, käynnistystavasta riippuen

Määrittely
(vaatimusanalyysi, vaatimusmäärittely, speksaus...)

- **Vastaa kysymykseen: Mitä projektissa tehdään?**
  - Mikä on ohjelman käyttötarkoitus?
  - Kuka sitä käyttää?
  - Missä (laite)ympäristössä sitä käytetään?
  - Minkälainen käyttöliittymä siinä on?
  - Mitä ohjelmalla tulee voida tehdä?
- **Voit lähteä liikkeelle esim. sanaston laatimisesta**
  - Sanastossa (glossary) määritelty tärkeimmät käsitteet mahdollisimman yksikäsitteisesti ja ristiriidattomasti
  - Ei välttämätön, mutta voi helpottaa omaa työtäsi
- **Älä kiinnitä suunnitteluratkaisuja tässä vaiheessa!**

Määrittely
(vaatimusanalyysi, vaatimusmäärittely, speksaus...)

- **Sopimus ohjelman tilanneen asiakkaan ja toteuttajan välillä**
- **Jos (ja kun!) ohjelma poikkeaa määrittely-dokumentissa luvatusta, lupa muutoksiin on kysyttävä asiakkaalta (tässä: ohjaajalta!)**
- **Vesiputousmallin ongelmat näkyvät tässä**
  - Määrittelyjä täytyy käytännössä lähes aina palata muuttamaan, oli se ”virallista” tai ei
- **Ns. ketterät prosessimallit ottavat muuttuvat vaatimukset huomioon**
  - Käytännössä tälläkin kurssilla on runsaasti joustonvaraa

Suunnittelu

- **Vastaa kysymykseen: Miten tehdään?**
- **Säästää toteutuskustannuksia: mitä paremmin suunniteltu, sitä suoraviivaisempi toteutus**
- **Kannattaa aikatauluttaa muita vaiheita pidemmäksi**

- **Konkreettisen toteutuksen suunnittelu:**
  - Algoritmit: miten ristinolla-peli päättelee voittorivin?
  - Tietorakenteet: miten budjettiohjelma organisoi tiedot?
  - Ajonaikainen toiminta: miten tieto kulkee ohjelmassa?
  - Käyttöliittymä: miten käyttäjä komentaa ohjelmaa?
  - Valmiin koodin käyttö (sallittua ohjaajan hyväksyessä ja oikeellisesti dokumentoituna)

Suunnittelu

- **Muista suunnittelun käytännöt:**
  näistä puhuttiin jo Johdatus TKT:hen -kurssilla!
- Kehitettävyys – mitä oletuksia teet; miten varaudut ohjelman muuttamiseen jälkeenpäin?
- Yksinkertaisuus – pyri siistiin rakenteeseen, vältä trikkejä
- Luotettavuus – vähintään ohjelmasi tulee kaatua nätisti
- Suorituskyky – tässä työssä korkeintaan harjoittelumielessä
- Tietoturva – tässä työssä lähinnä tiedon eheys (mitä se on?)

Suunnittelu

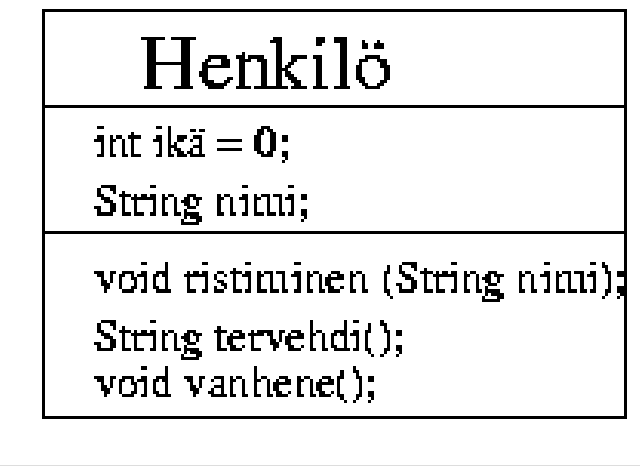
- **Kertaa Ohjelmistojen mallintaminen -kurssin materiaali!**
  - Erityisesti UML-tekniikka, sitä tarvitaan suunnittelussa:
    - Käyttötapaukset
    - Luokkakaaviot ***!***
    - Sekvenssikaaviot
- **Voit aloittaa esim. kevyillä käyttötapauskuvauksilla**
  - Ohjaaja ei välttämättä vaadi UML-käyttötapauksia, mutta ne voivat auttaa aihepiirin ja vaatimusten hahmottamista
  - Käyttötapauksia ei näin pienissä aiheissa (varsinkin pelit) tule yleensä kuin muutama

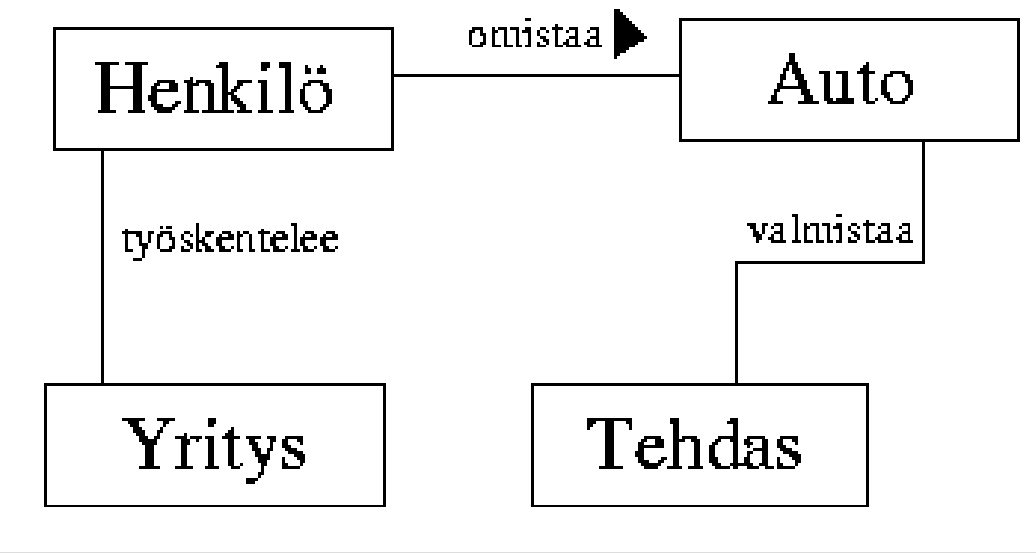
Suunnittelu

- **Luokat, metodit ja muuttujat määrätään, tuloksena luokkakaavio (ja API-kuvaus)**
- **Noudata oliolähestymistapaa:**
  - abstraktit tietotyypit: olio tarjoaa sovittuja palveluita muille
  - kapselointi: muuttujien ja metodien näkyvyys
  - perintä
- **Mieti suunnitellessasi, miten jatkokehittäjä vaihtaisi luokan toiseen**
- **Tee tarvittaessa abstrakteja yliluokkia ja rajapintaluokkia**

Suunnittelu

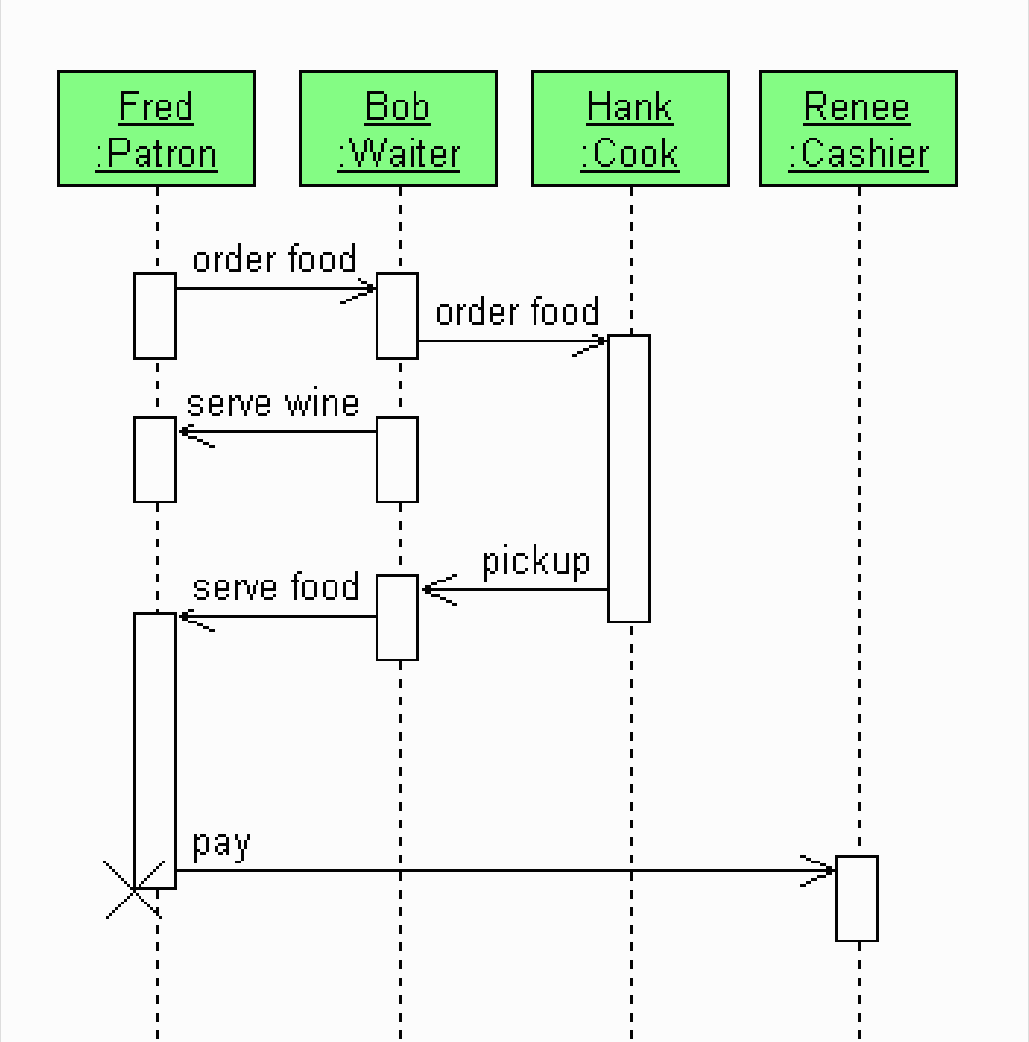
- **Tietorakenteiden kuvaus: UML:n luokkakaavio**
- **Visualisoi ohjelman rakennetta ja helpottaa sen hahmottamista**
- **Lähinnä itse tehdyistä luokista**
  - myös Swing-luokkia ym. tilanteen mukaan





Suunnittelu

- **Olioiden yhteistyön kuvaus: UML:n sekvenssikaavio**
- **Kertoo ajonaikaisesta toimintalogiikasta**
- **Erityisesti monimutkaisten metodikutsujen dokumentointiin**



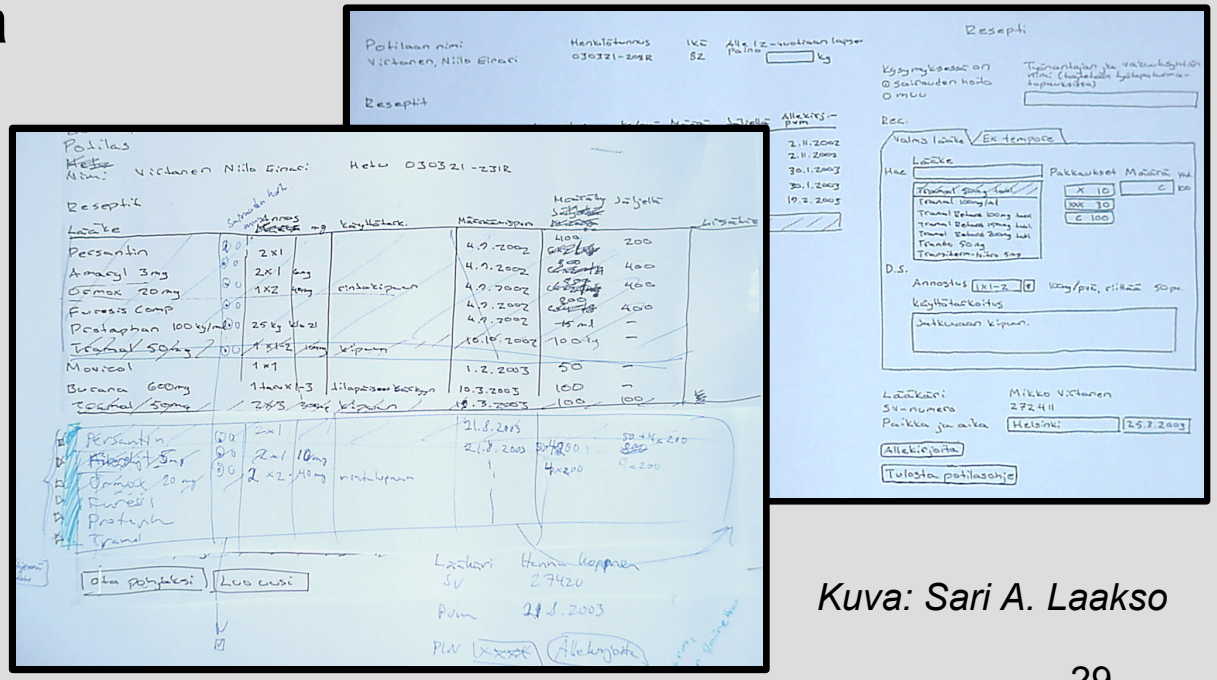
Kuva: Wikimedia Commons / GFDL license

Suunnittelu

- Luokkien sisäisen toiminnan kuvaus:
  esim. pseudokoodia, vapaamuotoisia kaavioita...

```
Käy läpi koko pelilauta
   Jos ruudussa on oma nappula
      ...
   Jos ruudussa on vastustajan nappula
      ...
   Jos ruutu on tyhjä
      ...
```

- Käyttöliittymäsuunnitelma kannattaa piirtää vapaamuotoisesti (esim. käsin paperille) kuvitteellisena ”screenshottina”



Kuva: Sari A. Laakso

Kaavioiden piirtäminen

- **Kynä ja paperi ovat hyödyllisiä**
  - Luonnosvaiheessa suttupaperi ja kynä erittäin hyvä työkalu
  - Puhtaaksipiirtovaiheessa kannattaa siirtyä piirto-ohjelmaan...
- **Suositellaan ”aitoa” kaavionpiirto-ohjelmaa**
  - Esim. Visual Paradigm, ArgoUML, Umbrello, Dia...
  - *http://www.cs.helsinki.fi/u/ruohomaa/jss/jss_kaaviot.html*
  - Ilmaisohjelmat ovat hieman kankeita/bugisia (tallenna usein!)
- **Toimisto- ja piirto-ohjelmat eivät sovellu**
  - Toimisto-ohjelmien tai piirto-ohjelmien perustoiminnot: viivoja ja pallukoita
  - Ei tukea esim. luokkakaavion piirrolle (perintä, assosiaatiot...)
  - Piirrosten muokkaus työlästä (miten siirrät laatikkoa?)

Toteutus

- **Aloita Javalla leikkiminen ja koodauskokeilut heti työn alussa**
  - Kokeile ja muokkaa esim. Java Tutorialien esimerkkejä
  - Saat tuntumaa ohjelmointiin, koodaustyökalut tulevat tutuksi
- **Älä kirjoita ohjelmaa ”valmiiksi” ennen kuin ohjaaja on hyväksynyt oliosuunnitelmat**
- **Alkuvaiheen koodikokeiluista osa joutaa roskiin**
  - Tämä on aivan normaalia ohjelmistonkehitystä!
  - Vanhaan koodiin takertuminen vaatii usein enemmän työtä kuin sen uusiksi kirjoittaminen: muista kehitettävyys/yksinkertaisuus

Toteutus

- **Jos suunnittelu on tehty huolellisesti, toteutusvaihe on lyhyt ja sisältää vain valmiin oliosuunnitelman kirjoittamisen Java-koodiksi**
- **Yleensä kuitenkin toteutuksen aikana huomataan suunnittelussa ongelmia = palataan korjaamaan suunnitelmaa**
    - ”Valmistakin” koodia joudutaan usein kirjoittamaan uudestaan, mahdollisesti useaan otteeseen
    - Tähän kannattaa varautua sekä henkisesti että aikataulullisesti...

Koodaustyyli

- **Noudata ohjaajasi antamia tyyliohjeita**
  - olio-ohjelmoinnista
  - koodinkirjoituksesta
  - kommentoinnista
- **Arvostelussa koodin toimivuuden lisäksi vaikuttaa myös ainakin:**
  - luettavuus (selkeä jäsentely, järkevä nimeäminen, ...)
  - johdonmukaisuus (oikea asia tehdään oikeassa paikassa)
  - kommentoinnin laajuus ja informatiivisuus (suurin osa kommentaatiosta metodien ja luokan esittelyn yhteyteen, muuten vain tarpeen mukaan)

Koodaustyyli

- **Ohjeita**

  Jari Juslinin Java-ohjeita (vanhoja mutta toimivia!)
  http://zds.iki.fi/zds/java/
  http://zds.iki.fi/zds/java/tyyliohje.shtml <- koodin kirjoitus
  http://zds.iki.fi/zds/java/oliotyyliohje.shtml <- oliosuunnittelu

  Arto Wiklan tyyliohje
  http://www.cs.helsinki.fi/u/wikla/JohdOhj/Tyyli.html

Koodaustyyli

- **Defensiivinen ohjelmointi: Metodeita kirjoittaessa varaudu epäkelpoihin parametreihin, vaikka olisitkin varma ettei todellinen parametri saa epäkelpoa arvoa.**
- **Esim. jos parametri on:**
  - ***int***-tyyppinen: varaudu nollaan, negatiivisiin ja positiivisiin arvoihin sekä mahdollisiin ongelma-arvoihin
  - ***String***-tyyppinen: varaudu "normaaliin" sekä tyhjään merkkijonoon, null-arvoon, sekä mahdollisiin ongelma-arvoihin
  - ***olio***tyyppinen: varaudu aina null-arvoon *(Javan API:sta löytyy useimmiten tieto, voiko käyttämäsi Javan valmiin luokan metodi palauttaa null-arvon)*

Kommentointi ja JavaDoc

- **Suositeltavin kommentointitapa on javadoc**
  - /** Kuten java-kommentit, mutta avataan 2 tähdellä */
  - Em. kommenteista voit generoida javadoc-ohjelmalla automaattisesti API-kuvauksen (HTML-muotoisena)
  - API-kuvaus on hyvä liite tekniseen dokumenttiin: kertoo mm. metodien parametreista ja toimintalogiikasta
  - API-kuvausta varten kirjoittaminen ”pakottaa” miettimään luokkien ja metodien kommentteja
  - API-kuvaus ei riitä yksinään tekniseksi dokumentiksi!

http://java.sun.com/j2se/javadoc/writingdoccomments/index.html

Kommentointi ja JavaDoc

```
/**
 * luo Supertankkerin, jolla on kpl kappaletta tyhjiä öljytankkeja, joiden
 * kunkin vetoisuus on tankinKoko. Jos kpl on ykköstä pienempi,
 luodaan
 * yksitankkinen alus. Jos tankinKoko on pienempi kuin 100.0, tankkien
 * vetoisuudeksi asetetaan 100.0.
 */
public SuperTankkeri(int kpl, double tankinKoko) {  ...
```

javadoc

Package Class Tree Deprecated Index H

PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD

Class SuperTankkeri

```
java.lang.Object
  extended by SuperTankkeri
```

```
public class SuperTankkeri
extends java.lang.Object
```

Constructor Summary

SuperTankkeri(int kpl, double tankinKoko)	luo Supertankkerin, jolla on kpl kappaletta tyhjiä öljytankkeja, joiden kunkin vetoisuus on tankinKoko.

Method Summary

void	lastaa(int mihin, double määrä) lisää tankkiin mihin öljyä määräverran.
static void	main(java.lang.String[] args) Main-metodissa luokan yksikkötestaus (Jaakko Nenonen)
double	pura(int mistä, double määrä) poistaa tankista mistä määrän verran öljyä.

Testaus

- **Käsitellään erillisellä luennolla 3. työviikolla**

Aloitusluento

I Kurssin sisältö: Mitä vaaditaan?

II Työvaiheet: Mitä tehdään?

III Työvälineet: Millä tehdään?

IV Java-vinkkejä: Miten tehdään?

V Ryhmiinjako ja jonottajien karsinta

Sovelluskehitin

- **Eclipse:** *http://www.eclipse.org*
- **NetBeans:** *http://www.netbeans.org*

- **Sovelluskehittimen etuja:**
  - **uusien luokkien generointi sujuu helposti**
  - **koodin syntaksitarkistus lennossa**
  - **automaattitäydennys metodien ym. nimille**
  - **automaattitäydennys parametrilistoille**
  - **navigointiapuja koodissa kahlaamiseen**
  - **debug-työkalut sisäänrakennettuna**
  - **kunnon koodieditori osaa sisennykset, sulkujen täsmäykset, avainsanojen korostuksen ym.**

Editori vai sovelluskehitin?

- **Jos haluat maksimoida työtehosi...**
  - Kunnon sovelluskehittimen (esim. Eclipse, NetBeans) opettelu vie joitakin tunteja, mutta maksaa itsensä takaisin jo tämän harjoitustyön aikana
  - Jos jatkat TKT:n opintoja yhtään pidemmälle, tämä on suositeltavin vaihtoehto
- **Jos haluat minimoida uuden opettelemisen...**
  - Koodauksesta selviää myös käyttämällä samaa tuttua tekstieditoria kuin tähänkin asti
  - Jos et aio opiskella TKT:tä kuin minimimäärän (perusopinnot), tämä voi olla realistinen vaihtoehto

Aloitusluento

I Kurssin sisältö: Mitä vaaditaan?

II Työvaiheet: Mitä tehdään?

III Työvälineet: Millä tehdään?

IV Java-vinkkejä: Miten tehdään?

V Ryhmiinjako ja jonottajien karsinta

Java-ohjeita

- **TTY:n Java-sivut**
  http://javala.cs.tut.fi/
- **Jaakko Nenosen linkkilista**
  http://www.cs.helsinki.fi/u/jnenonen/alabra/linkit.html
- **Ohj. perusteet / Java-ohjelmointi -kurssien sivut**
  - *http://www.cs.helsinki.fi/u/wikla/Ohjelmointi/*
  - *http://www.cs.helsinki.fi/u/tapasane/kurssit/Java-Ohjelmointi/kevat2008/*
- **Ohjelmointitekniikka (Java) -kurssin sivu**
  http://www.cs.helsinki.fi/u/wikla/OTJ/
- **Sunin Java-ohjeiden koostesivu**
  http://java.sun.com/javase/6/docs/

Aloitusluento

I Kurssin sisältö: Mitä vaaditaan?

II Työvaiheet: Mitä tehdään?

III Työvälineet: Millä tehdään?

IV Java-vinkkejä: Miten tehdään?

V Ryhmiinjako ja jonottajien karsinta