1. Henkilötiedot

   Ohjelmoinnin peruskurssi Y2, projektityön dokumentti

   Jani Hyrkäs, 348348, EST, 12.5.2017

2. Yleiskuvaus

   Yleinen kuvaus siitä, mitä on luotu. Tähän osioon voi monissa tapauksissa pitkälti lainata suunnitelmasta. Jos osioon on tullut olennaisia muutoksia suunnitelmaan nähden, niistä tulee mainita tässä. Jos aihe on mahdollista toteuttaa usealla eri vaikeusasteella, ilmaise myös, minkä tasoisena työ on lopulta mielestäsi toteutettu.

3. Käyttöohje

   Opastus ohjelman käyttöön: miten ohjelma käynnistetään? Mitä sillä voi tehdä? Mitä komentoja käyttäjällä on valittavanaan? Jne.

4. Ohjelman rakenne

   Ohjelman erottelu tärkeimpiin osakokonaisuuksiinsa, toteutuneen luokkajaon esittely. Minkälaisilla luokilla kuvaatte ohjelman ongelma-aluetta? Mitä ongelman osaa kukin luokka mallintaa? Mitkä ovat luokkien väliset suhteet? Entä millaisia luokkia käytätte ohjelman käyttöliittymän kuvaamiseen? Valmiita Python-luokkia ei tarvitse esitellä samalla tavalla kuin omia, mutta tehdyt valinnat on hyvä perustella.

   Tässä voi esittää myös mahdollisia muita ratkaisumalleja ja näin perustella valittu ratkaisu. Jos suinkin mahdollista, liittäkää mukaan jonkinlainen graafinen luokkakaavio (voitte käyttää esim. UML-luokkakaavionotaatiota, mutta se ei ole millään muotoa pakollista). Esitelkää omien luokkien keskeiset metodit. Huom. oleellista on vain se, mitä metodeilla tehdään, ei se, miten ne sisäisesti toimivat.

5. Algoritmit

   Sanallinen kuvaus käyttämistänne algoritmeista, eli siitä miten ohjelma suorittaa tarvittavat tehtävät. Esim. miten tarvittava matemaattinen laskenta tapahtuu? (kaavat mukaan) Miten algoritminne löytää lyhimmän tiereitin kahden kaupungin välille? Miten toteuttamanne pelin tekoäly toimii? Kaavioita tms. voi käyttää apuna tarpeen mukaan. Mitä muita ratkaisuvaihtoehtoja olisi ollut? Perustelkaa valintanne: Verratkaa toteutusta johonkin toiseen ratkaisuun, ja selittäkää miksi päädyitte juuri tähän.

   Tässä kohdassa on siis tarkoitus selostaa ne periaatteet, joilla ongelmat on ratkaistu, ei sitä, miten algoritmit koodataan. Siis ei luokkien tai metodien kuvauksia tai muitakaan Pythoniin tai ohjelmakoodiin liittyviä seikkoja tänne. Pseudokoodiesitys keskeisimmistä ei-tunnetuista algoritmeista on kuitenkin hyvä olla sanallisen kuvauksen tukena. HUOM!Jokaisessa työssä on aina algoritmeja, toiset ehkä yksinkertaisempia kuin toiset, moni aivan itse alusta saakka keksittyjä. Kuvaa tässä niistä muutama kaikkein olennaisin.

6. Tietorakenteet

   Minkälaiset kokoelmatyypit/tietorakenteet soveltuvat parhaiten ohjelmassa tarvittavan tiedon varastoimiseen ja käsittelyyn? Miksi? Mitä muita valintamahdollisuuksia olisi ollut? Käytittekö muuttuvatilaisia (mutable) vai muuttumattomia (immutable) rakenteita? Jos käytitte Pythonin valmiita tietorakenteita, ei niiden tarkkaa määrittelyä tarvitse esittää. Jos taas ohjelmoitte itse jonkin tietorakenteen, on sen toimintatapa selostettava.

7. Tiedostot

   Selostakaa tässä osiossa myös millaisia tiedostoja ohjelmasi käsittelee, jos mitään. Esim. ovatko ne tekstitiedostoja vai binaaritiedostoja, ja miten tieto on niissä esitetty? Kuvatkaa lopullinen tiedostoformaatti sillä tasolla, että assistentti voi halutessaan helposti luoda ohjelmalle testidataa. Jos ohjelma tarvitsee toimiakseen käyttäjän luomia asetustiedostoja tms. laittakaa ne lähdekoodin mukaan liitteeksi. Tarkoitus on palauttaa koodi sellaisessa muodossa

että assistentti voi helposti kokeilla sitä käyttämättä paljon aikaa ohjelman käyttökuntoon virittelyyn.

8. Testaus

   Kertokaa miten ohjelmaa testattiin ja kuinka se vastasi suunnitelmassa esitettyä.

   Läpäiseekö ohjelma kaikki suunnitelmassa esitetyt testit? Kuinka ohjelmaa testattiin sitä rakennettaessa? Oliko testauksen suunnittelussa jotain olennaisia aukkoja? Yksikkö;testausta harjoiteltiin kurssin alkupuolella. Jos teit yksikkötestejä jollekin koodin osalle, niin kerro testauksesta tässä.

9. Ohjelman tunnetut puutteet ja viat Kuvaa tässä osiossa kaikki tuntemasi puutteet ja viat ohjelmassasi. Kerro miten korjaisit nämä ongelmat jos jatkaisit projektia. Mitä vähemmän assistentti löytää puutteita kohdista joiden väität toimivan sen parempi. Ole siis rehellinen. Lisäksi hyvin jäsennetyt kehitystarpeet kertovat perehtymistä ongelmaan ja sen ratkaisuun.

10. 3 parasta ja 3 heikointa kohtaa Assistentti käyttää runsaasti aikaa tutustuessaan ohjelmaasi, mutta ei välttämättä näe ja tunne toteutustasi samalla tavoin kuin sinä. Jos ohjelmassa on joitakin kohtia joita itse pidät erityisen hyvinä, mainitse tässä niistä 1-3 kappaletta lyhyen perustelun kera. Jos ohjelmassa on kohtia jotka itsekin tiedät heikoiksi, voi mainita myös nämä. Tällöin mahdollisuus että nämä heikot kohdat dominoivat arvostelua vähenee huomattavasti. Tässä voi myös esittää sanallisesti kuinka olisi nämä asiat halutessaan korjannut.

11. Poikkeamat suunnitelmasta

    Teitkö jotain toisin kuin olit suunnitellut? Miksi? Osuiko suunnitelmaan laatimasi ajankäyttöarvio oikeaan? Miksei? Entä toteutusjärjestys?

12. Toteutunut työjärjestys ja aikataulu Kerro tässä yleisellä tasolla missä järjestyksessä projekti lopulta toteutettiin (mielellään myös päivämäärät). Missä poikettiin suunnitelmasta?

13. Arvio lopputuloksesta

    “Yhteenveto” ja itsearviointi joka voi toistaa yllämainittujakin asioita.

    Arvioikaa ohjelman laatua, kertokaa sen hyvistä ja huonoista puolista. Onko työssä oleellisia puutteita ja mistä ne johtuvat (mahdollinen hyvä perustelu dokumentissa voi korvata pienet puutteet)? Miten ohjelmaa olisi voinut tai voisi tulevaisuudessa parantaa? Olisiko ratkaisumenetelmien, tietorakenteiden tai luokkajaon valinnan voinut tehdä paremmin? Soveltuuko ohjelman rakenne muutosten tai laajennusten tekemiseen? Miksi tai miksi ei?

14. Viitteet

    Mitä kirjoja, nettisivuja tai muuta materiaalia olette käyttäneet? Kaikki lähteet tulee ilmoittaa, vaikka niihin kuuluisivat pelkkä kurssilla käyttämänne oppikirja ja perusluokkakirjastojen API-kuvaus.