Rökstudd forritun

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-02-10

Skjölun forrita

Tvær tegundir af skjölun:

- Ytri skjölun er ætluð notendum
- Innri skjölun er ætluð forriturum

Ytri skjölun:

- "Hvernig nota ég þennan svarta kassa?"
- Inniheldur nægilegar upplýsingar til að nota "svarta kassann"
- Inniheldur ekki upplýsingar um innviði "svarta kassans"
- Leggur áherslu á hvað, en ekki hvernig

Innri skjölun:

- "Hvernig virkar svarti kassinn?"
- Skjölun á innviðum – gagnaskipan og kóða

Framsetning á skjölun

Ytri skjölun:

- Notkunarlýsingar

Innri skjölun:

- Gagnaskipan (fastayrðing gagna)
- Fastayrðing lykkju – tekið fyrir í næstu viku

Notkunarlýsingar

Grf. eftirfarandi aðferð:

```
public static void qsort(int[] f, int i, int j)
```

Hvað gerir hún? Hvernig má kalla á hana?

Er hægt að nota aðferðina án þess að vita hvernig hún er útfærð?

Notkunarlýsing

Notkunarlýsing fyrir aðferð samanstendur af:

- Forskilyrði: lýsing á stöðunni fyrir framkvæmd
- Eftirskilyrði: lýsing á stöðunni eftir framkvæmd

Dæmi um notkunarlýsingu:

```
// Notkun: qsort(f,i,j);
// Fyrir:  f[i..j-1] er löglegt svæði í f
// Eftir:  Búið er að raða f[i..j-1] í vaxandi röð
public static void qsort(int[] f, int i, int j)
```

Notkunarlýsing

Notkunarlýsing inniheldur nægar upplýsingar til að notandi geti:

- Kallað rétt á aðferðina
- Vitað nákvæmlega hvaða áhrif kallið hefur

Notkunarlýsing er einskonar samningur á milli notanda og forritara.

Forskilyrði inniheldur kröfu forritara. Sá sem kallar á aðferðina verður að uppfylla þá kröfu.

Eftirskilyrði inniheldur kröfu notandans. Sá sem útfærir aðferðina verður að uppfylla þá kröfu.

Forsendur

Forritun út frá forsendum.

Notandi aðferðar gerir ráð fyrir að eftirskilyrði hennar sé satt ef hann kallar á hana á réttan hátt (þ.e. forskilyrði hennar sé satt.)

Forritari aðferðar gerir ráð fyrir að forskilyrði hennar sé satt í upphafi og verður að tryggja að eftirskilyrðið sé satt í lokin.

Notkunarlýsing er **explicit samningur** um hvernig má nota aðferðina og hvað hún gerir. Það má breyta útfærslu hennar svo lengi sem hún uppfyllir ennþá notkunarlýsinguna.

Fleiri dæmi

```
// Notkun: k = leita(f,i,j,x);
// Fyrir:  f[i..j-1] er í vaxandi röð
// Eftir:  f[i..j-1] er óbreytt, i <= k <= j, og
//         f[i..k-1] <= x < f[k..j-1]
static int leita(double[] f, int i, int j, double x)
```

Fleiri dæmi

```
// Notkun: MergeSort.merge(q1,q2,q);
// Fyrir:  q er tóm biðröð, q1 og q2 eru biðraðir, sem
//         innihalda gildi, sem eru í vaxandi röð frá
//         fremsta til aftasta.
//         q hefur pláss fyrir öll gildin í q1 og q2.
// Eftir:  q1 og q2 eru tómar, gildin úr þeim eru í q í
//         vaxandi röð frá fremsta til aftasta.
static void merge(Queue q1, Queue q2, Queue q)
```

Upplýsingahuld

Upplýsingahuld er sú meginregla í hugbúnaðargerð að sérhverja veigamikla hönnunarákvörðun skuli fela í einingu.

Til dæmis, ef nota skal biðraðir í kerfinu, þá skal **fela í einingu hvernig biðraðirnar eru útfærðar.**

Tilgangur:

- Tryggir aðskilnað milli notanda einingar og forritara einingar
- Forritari getur breytt ákvörðun um útfærslu án þess að það kosti stóran uppskurð á kerfinu

Upplýsingahuld

Mikilvægt að gæta þess að notkunarlýsing brjóti ekki upplýsingahuld með of miklum upplýsingum um innviði einingar.

Dæmi um brot á upplýsingahuld:

```
// Notkun: s.push(x);
// Fyrir:  Ekkert
// Eftir:  Búið er að bæta x í nýjan hlekk sem var
//         settur fremst í keðjuna
```

Rétt væri:

```
// Notkun: s.push(x);
// Fyrir:  Ekkert
// Eftir:  Búið er að bæta x efst á hlaðann s.
```

Fleiri dæmi um brot

Fleiri dæmi um brot á upplýsingahuld:

```
// Notkun: s.push(x);
// Fyrir:  Ekkert
// Eftir:  Búið er að bæta x í nýjan hlekk efst
//         á hlaðann s
```

Hvað er að þessu?

Fleiri dæmi um brot

Annað dæmi um brot á upplýsingahuld:

```
// Notkun: s.push(x);
// Fyrir:  Ekkert
// Eftir:  Búið er að bæta x í nýjan hlekk efst
//         á hlaðann s
```

Hvað er að þessu?

Það er innra útfærsluatriði að hlaðinn sé útfærður með hlekkjum! Hann gæti alveg eins verið útfærður með fylki.

Það skiptir ekki máli fyrir notandann hvernig gildin eru geymd. Notandinn hefur áhuga á eiginleikum hlaðans.

Gagnaskipan

Gagnaskipan er lýst með s.k. fastayrðingu gagna (data invariant)

Fastayrðing gagna er stöðulýsing sem fjallar um tilviksbreytur í tilviki klasa.

Fastayrðing gagna er vanalega sett fram í athugasemd efst í klasa þar sem tilviksbreyturnar eru skilgreindar.

```
public class Stack
{
    // Gildin á hlaðanum eru geymd í keðjunni
    // top frá efsta til neðsta.
    // n er fjöldi gilda.
    private Node top;
    private int n;
```

Reglur

Reglur um fastayrðingu gagna:

1. Smiður í klasa (constructor) kemur hlutnum í rétt upphafsástand (þ.a. fastayrðing gagna sé sönn)
2. Aðferð í klasanum **gerir ráð fyrir því að ganga að gögnunum í réttu ástandi** (þ.e. fastayrðing gagna er alltaf sönn í upphafi aðferðar)
3. Aðferð í klasanum **verður að skilja hlutinn eftir í réttu ástandi** (til að næsta aðferð finni hann í réttu ástandi), þ.e. fastayrðing gagna er alltaf sönn í lok aðferðar

Fastayrðing gagna

Góð fastayrðing gagna lýsir gagnaskipan nægilega vel til að hægt sé að útfæra sérhverja aðferð í klasanum **bara út frá fastayrðingu gagna**

Með öðrum orðum, við útfærslu á einni aðferð þá þarf ekki að skoða neina aðra aðferð!

Útfærsla á einni aðferð í klasanum er óháð útfærslum á öðrum aðferðum í honum. Sérhver útfærsla er **bara háð fastayrðingu gagna**.

Fastayrðing gagna

Hvernig sannfærum við okkur um að klasi sé réttur?

Við athugum hvort sérhver aðferð sé rétt m.t.t. fastayrðingu gagna, forskilyrðis og eftirskilyrðis

Aðferð má gefa sér eftirfarandi forsendur:

- Fastayrðing gagna er sönn í byrjun aðferðar
- Forskilyrðið er satt í byrjun aðferðar

Eftirfarandi kröfur eru lagðar á aðferðina:

- Fastayrðing gagna sé aftur sönn í lok aðferðar
- Eftirskilyrðið er satt í lok aðferðar

Fastayrðing gagna

Til hvers að standa í þessu?

Ef öllum reglunum er fylgt þá má kalla á aðferðirnar í hvaða röð sem er.

Það þarf ekki að skoða hvernig klasinn er notaður, því réttleiki hans er tryggður óháð því hvernig hann er notaður.

Forritari gefur sér alltaf tiltekna gagnaskipan. Þetta snýst bara um að skrifa niður það sem er búið að ákveða og fylgja því eftir.