1DV404 Laboration 1

Planering och förbättring

av Oskar Klintrot, oklib08

Uppgift 1

Planering av uppgift 1a

	Steg	Tidsåtgång: 1 h 30 min
1.	Inmatning till sträng	15 min
2.	Skapa en metod för att jämföra hur många ”a” och ”A” det finns i strängen.	45 min
3.	Skriv ut resultatet till användaren.	15 min
4.	Reflektion	15 min

Utförande av uppgift 1a

Steg	Fel vid programmering	Brister i planeringen	Tidsåtgång: 45 min
1.	Inga	Nej	5 min
2.	Skapade en klass för metoderna, vilket var mer overkill än det lilla dry som blev i denna lilla applikation.	Behövde skapa två olika metoderna.	20 min
3.	Inga	Nej	10 min
4.	N/A	N/A	10 min

Reflektion kring uppgift 1a

Planeringen gick någorlunda bra, enda tabben var att jag behövde göra två metoder och inte bara en. I utförandet kunde jag från första början hållit mig till min ursprungliga tanke att skapa metoderna direkt under Main. Hade jag repeterat mer från C#-kursen hade jag mer troligen gjort rätt från början. Detta misstag tog dock så lite tid i anspråk att det i praktiken antagligen hade gott på ett ut, om inte att det tagit längre tid att läsa på innan jag började koda.

Att första steget skulle gå fortare än planerat var väntat men det var sagt att planeringen skulle vara i steg om 15 min. Jag gillar att ha lite slack i planeringen men i detta fall var det kanske lite onödigt att ha med det i ett eget steg på 15 min. Att skapa metoderna gick mycket snabbare än planerat, mycket tack vare att jag snabbt hittade strings replace-funktionen i kurslitteraturen till C#-kursen. Hade jag behövt leta mer, i boken eller på nätet, hade det troligtvis genast tagit mer tid i anspråk.

Planering av uppgift 1b

	Steg	Tidsåtgång: 2 h 15 min
1.	Inmatning till int.	15 min
2.	Skapa klass för nedanstående metoder.	15 min
3.	Skapa metod för att se hur många ”0” som finns i talet.	30 min
4.	Skapa metod för att se hur många udda tal som finns i talet.	30 min
5.	Skapa metod för att se hur många jämna tal som finns i talet.	15 min
6.	Skriv ut resultatet till användaren.	15 min
7.	Reflektion	15 min

Utförande av uppgift 1b

Steg	Fel vid programmering	Brister i planeringen	Faktisk tidsåtgång: 2 h 50 min
1.	Tankevurpa med klassens egenskaper. Anropade objektet felaktigt. Problem med att ta hand om felaktig inmatning.	Gjordes som #2.	35 min
2.	Inga fel men behövde läsa på en del för att få allt rätt.	Implementerades som #1.	15 min
3.	Problem med omvandlingen från int till string, börjar talet på 0 försvinner första nollan vid omvandlingen.	Problem med felhantering. Miss att inte tänka på att inmatningen måste vara string för att få med inledande nolla.	35 min
4.	Problem med omvandlingarna mellan olika arrayer och problem med gammal kod som störde.		40 min
5.	Inga. Ändrade dock om i koden så att alla metoderna bakades ihop till en enda då koden för udda tal och jämna tal var mer eller mindre exakt samma, enbart ett ”!” skiljde dem åt.	Koden var i stort sett identisk med #4, vilket inte är bra med tanke på DRY.	30 min
6.	Inga.	Inga	2 min
7.	N/A	N/A	13 min

Reflektion kring uppgift 1b

Största misstaget som gjordes var att jag hade glömt hur man använder sig av klasser. Det tog ett tag att få den korrekt och anropade den och använda den på rätt sätt. Jag hade även kunnat fundera kring DRY redan vid planeringsstadiet. Som det blev nu så upptäckte jag först när jag kom till steg 5 att koden för steg 4 och 5 kommer att bli i stort sett identisk. Därför fick jag lägga tid på att få ihop metoderna till en för att undvika onödig kod. Hade jag planerat från början att de skulle vara en metod istället för två (eller tre, nu bakade jag in alla metoderna i en enda) hade jag kunnat spara en del tid.

En annan miss var att jag inte tänkte på att 0 kan finnas först i talet och att int inte sparar 0:an i så fall. Där fick jag lägga tid på att göra om min metod så att den använde sig av en string istället för int. Dock blev en string enklare att hantera i nästa metoderna då en string väldigt enkelt går att göra om till en char[], något jag utnyttjade för att kontrollera varje tecken i arrayen för sig i en for-loop.

Ett problem som inte var ett misstag men som tog lång tid att lösa var att kontrollera att strängen som matades in bara bestod av siffror och inte av några andra tecken. Lösningen blev till slut ganska

elegant. Ingen try-catch behövdes användas, något jag undviker efter att Eddy i Webbteknik I sa att man kastar fel i första hand till andra utvecklare. Eftersom användaren knappast är en annan utvecklare vill jag lösa felhanteringen utan att kasta ett fel och då kom TryParse till användning.

Planering av uppgift 1c

	Steg	Tidsåtgång: 2 h
1.	Läs in 3 heltal till var sin int.	15 min
2.	Skapa en metod som jämför talen med varandra och retunerar det näst största talet.	45 min
3.	Skriv ut det näst största talet till användaren.	15 min
4.	Utöka till 10 heltal	30 min
5.	Reflektion	15 min

Utförande av uppgift 1c

Steg	Fel vid programmering	Brister i planeringen	Faktisk tidsåtgång: 1 h 30 min
1.	Inga	Inga	15 min
2.	Började med att försöka sortera alla talen, som man hade gjort med en array. När jag bytte taktik till att enbart jämföra det senast inmatade talet med det tidigare största och näst största talet gick det fort att skriv metoden.	Borde ha tänkt igenom vilka alternativ som fanns för att kunna lösa problemet innan jag började koda. Snöade in på fel taktik.	60
3.	Inga	Inga	-
4.	Inga	Inga	-
5.	N/A	N/A	15 min

Reflektion kring uppgift 1c

Jag hade snöat in på att lösa uppgiften på samma sätt som man sorterar en array, fast sortera enskilda int. Det gick hyfsat med tre helttal men hade blivit alldeles för bökigt att sortera 10 heltal. Jag hade tänkt att man i en while-loop jämför de talen närmast varandra och byter plats på dem om de är i fel ordning. Gör man det tillräckligt många gånger kommer största talet ha vandrat längst upp och minsta längst ner. Det var dock väldigt omständigt och jag lyckades inte klura ut hur loopen skulle skrivas. Tillslut kom jag på att det räcker att jämföra det senast inmatade talet med det tidigare största och näst största talet. Då spelar det ingen roll om jag ska göra 3 eller 100 inmatningar, det går lika smidigt ändå. När metoden sen skulle ändras från 3 inmatningar till 10 behövde jag bara ändra antalet loopning på for-loopen där inmatningen och jämförelsen gjordes från 3 till 10.

Uppgift 2

2a: Ge exempel på några alternativa strategier som du kan använda i din planering av programmeringsuppgifterna!

Det finns framförallt två punkter som jag kan ta hänsyn till för att lyckas bättre med min planering, vilka dessutom går hand i hand med varandra:

- Avsett ordentligt med tid för planeringen av arbetet
    - En grov planering fungerar bra vid större projekt över en längre tid men vid de här små projekten (eller vid planeringen av mindre bitar av ett större projekt) hade det varit bättre att planera mer i detalj.
- Planera redan från början för alternativa strategier
    - När man ska planera arbetet mer i detalj kan det vara lämpligt att titta på flera olika alternativa strategier. Detta blev tydligt i uppgift 1c där jag förlorade mycket tid på att jag hade en dålig strategi. Hade jag lagt mer tid i planeringen och funderat igenom flera olika strategier hade jag troligtvis kunnat ändra strategi betydligt tidigare i arbetet när jag upptäckte att det började bli en väldigt omständlig kod som skulle bli svår att utöka till att testa fler tal.

2b: Två av anledningarna till att din planering avviker från verkligheten är dels felen du gör dels alla andra saker som inträffar. Hur kan du ta hänsyn till eller minska konsekvenserna av dessa? Ge konkreta exempel baserat på dina erfarenheter!

Som jag skrev under 2a tror jag att noggrannare planering samt att i förväg kolla vad jag kan göra om en strategi inte fungerar så att jag är förbered vid eventuella fel och problem som uppstår. Att vid planeringen kolla på gammal kod och exempel tror jag också kan minska risken för att göra fel är välja fel väg att gå. Till exempel hade jag nog kunnat förutse att 1b.4-5 borde ha varit en metod om jag hade planerat noggrannare. 1c.2 hade antagligen tagit mindre än halva tiden att genomföra om jag från början hade planerat för två olika strategier. Mer tid i planeringen sparar tid senare!

2c: Implementera två ”förbättringsåtgärder” i ditt planeringsarbete!

Mina två förbättringsåtgärder bli att:

1. Avsätt tid till planeringen, beroende på svårighetsgrad kan 15-30 minuter vara lämpligt. Gå igenom gammal kod för att se vad som kan ”återanvändas”. Försök ta hänsyn till DRY redan vid planeringen och anteckna detta vid punkter där det kan vara bra att ta hänsyn till DRY redan från första början.
2. Skriv, om möjligt, ner flera olika strategier och försök uppskatta tidsåtgången för dem. Börja arbeta med den som borde ta kortast tid, visar den sig vara mindre lyckat så byt tidigt till nästa strategi istället för att lägga ner onödig tid på en dålig strategi. Det går alltid att gå tillbaka om den visade sig vara den bästa trots allt.

Uppgift 3

Planering av uppgift 3a

Steg	Alt.	Moment	Tidsåtgång: 1 h 45 min – 2 h 15 min
1.		Planera uppgiften	30 min
2.	a	Ta bort alla mellanslag i den inmatade strängen med String.Replace(), återanvänd kod från uppgift 1a	15 min
2.	b	Ta bort alla mellanslag med String.Join() och String.Where() och leta efter char.IsWhiteSpace()	30 min
3.		Stoppa in inmatade strängen i en array med String.ToCharArray	15 min
4.	a	Kopiera arrayen och vänd byts plats på arrayen med Array.Reverse()	15 min
4.	b	Byt plats på arrayen genom att skapa en tom array och kör en foreach ”baklänges”	30 min
5.		Jämför varje position i arrayerna i en foreach	15 min
6.		Om arrayerna inte matchar, retunera till användaren att inmatningen inte är en palindrom.	15 min

Utförande av uppgift 3a

Steg	Alt.	Fel vid programmering	Brister i planeringen	Faktisk tidsåtgång: 1 h 15 min
1.	-	-	-	20 min
2.	a	Inga	Nej	15 min
3.		Tänkte att jag behövde göra om char[] till string[].	Kunde tänkt på vad jag faktiskt behövde för typ av array.	5 min
4.	a	Inga	Nej	5 min
5.	-	Behövde använda en for-loop och inte en foreach. Tänkte inte på att sätta två arrayer lika med varandra gör att de refererar till samma objekt. Skapade en copy-metod.	Borde tänkt på att i alla fall ha en for-loop som ett alternativ. Borde tidigare i planerande ha gjort ett eget steg med att kopiera arrayen.	20 min
6.	-	Inga	Bakades in i steg 5.	-
7.	-	Inga	Lade till att programmet loopar om inte användaren trycket på esc och att versaler fungerar.	15 min

Reflektion kring uppgift 3a

Uppgiften var förhållandevis enkel varpå inga större missar gjordes, den enda nämnvärda var att jag råkade skapa två referenser till samma array när jag egentligen vill arrayen. Detta löstes dock snabbt och smärtfritt. En mindre tabbe var att jag behövde använda en for-loop istället för en foreach, något som blev uppenbart väldigt tidigt eftersom jag hade tänkt använda mig av ”i”.

Förarbetet i planeringen gjorde att arbetet flöt på väldigt bra, behövde knappt stanna upp och tänka efter en enda gång! Även att ha olika alternativ klara för sig på förhand var väldigt smidigt, även om jag inte behövde byta alternativ i just denna uppgift. Jag kommer absolut fortsätta med samma arbetsgång i kommande uppgifter.

Uppgift 3b

Planering av uppgift 3b

Steg	Alt.	Moment	Tidsåtgång: 3 h 15 min
1.		Planera uppgiften	30 min
2.		Skapa egenskaperna getNumerator och getDenominator och skapa felhantering då getDenominator är = 0	15 min
3.		Skapa konstruktorn	15 min
4.		Skapa metoden isNegative	15 min
5.	a	Skapa metoden add	30 min
5.	b	Överlagra +-operatorn istället för att skapa en ny metod	15 min
6.	a	Skapa metoden multiply	30 min
6.	b	Överlagra *-operatorn istället för att skapa en ny metod	15 min
7.	a	Låt getDenominator hantera N = 0 istället för add och multiply	15 min
7.	b	Implementera felhantering i metoderna add och multiply vid N = 0	15 min
8.		Skapa metoden isEqualTo	15 min
9.		Skapa metoden toString	15 min
10.			

Utförande av uppgift 3b

Steg	Alt.	Fel vid programmering	Brister i planeringen	Faktisk tidsåtgång: 1 h 30 min
1.	-	-	-	30 min
2.		Nej	Inga	10 min
3.		Nej	Inga	5 min
4.		Nej	Inga	5 min
5.	b	Nej	Inga	5 min
6.	b	Nej	Inga	5 min
7.	a	Nej	Inga	-
8.		Ful lösning! Problem med att få double på rätt ställe	Borde ha letat upp fler alternativ	25 min
9.		Nej	Skapade en överlagrad ToString() istället för toString()	5 min

Reflektion kring uppgift 3b

Eftersom en struktur för hur man skulle lösa uppgiften redan fanns och MSDN hade ett väldigt bra och enkelt exempel på hur en klass för att hantera bråktal skulle kunna se ut gick arbetet väldigt smidigt framåt. Jag frångick dock instruktionerna vid tre tillfällen. För det första valde jag att överlagra + och * operatorerna istället för att skapa nya metoder som skulle göra samma sak fast på ett minde tilltalande sätt. Exempel på hur man skulle göra fanns på MSDN och jag såg ingen anledning till att uppfinna hjulet en gång till utan använde mig av de exempel som fanns då de ändå var en sådan lösning jag hade tänkt mig i själva metoden. Jag valde också att överlagra ToString(), jag misstänker att det var så det egentligen var tänkt. I vilket fall var det den snyggaste och mest logiska lösningen.

I planeringen borde jag ha buntat ihop fler moment till samma steg eftersom jag inte kunde planera kortare tid än 15 min, vilket blev på tok för långt i de flesta fallen. Jag överskattade dock uppgiftens

komplexitet, när jag väl började koda gick det mesta väldigt mycket smidigare än jag hade tänkt mig. Dock blev isEqualTo() mycket fulare än jag hade hoppats på, jag borde ha letat upp efter flera olika alternativ redan vid planeringen, precis som jag egentligen hade sagt tidigare att jag skulle gjort, men jag trodde den skulle vara enklare än vad den var. Vid planeringen tänkte jag att jag bara skulle jämföra täljare och nämnare men så kom jag på när jag väl skulle skriva koden att exempelvis $\frac{1}{2}$ är samma bråk som $\frac{2}{4}$ och då blev det genast klurigare. Antingen skulle jag fått försöka hitta minsta gemensamma nämnaren eller helt enkelt bara dividera bråken och få ut svaret i en double som sedan konverterades till en int för att kunna jämföras med det andra bråket. Jag valde den senare lösningen då jag trodde den skulle gå snabbast. Dock blev resultatet som sagt mindre bra.

Uppgift 4

<brasklapp> Jag kan inget av det här så alla tider är högst ungefärliga. "Applikationen fungerar som ett gränssnitt mot ett dokument "moln" där registrerade användare lagrar och delar dokument." *tolkar jag som att serversidan redan är programmerad och fullt fungerande med vad allt det innebär med inloggning, lagring och så vidare, alltså är det bara är klientsidan som ska programmeras. Programmerarna heter, Kalle, Sara, Gösta, Sandra och Herbert. </brasklapp>*

Vilka uppgifter måste ni genomföra?

- Gränssnitt
    - Filhanterare
        - Visa filer och mappar á la Explorer och Finder
        - Möjlighet till grundläggande funktionalitet såsom att flytta, byta namn på och radera filer och mappar.
    - Textredigerare
        - Hantera OOXML formaterade dokument
        - Visuellt tilltalande och lättförståeligt utseende
        - Funktioner för enklare redigering
    - Senaste ändringar
        - Färgkodning efter typ av modifiering och av vilken användare
    - Behörigheter/Åtkomstkontrollista
        - Visuellt tilltalande, visas till höger om filhanteraren
        - Möjlighet att ändra behörigheter
        - Administratörsbehörighet krävs för att ändra andras åtkomst till filen
    - Delning
        - Möjlighet att lägga till att andra, antingen alla eller några utvalda användare, ska kunna läsa, ändra eller ta bort filer. Konsensus hos administratörer krävs dock för att ta bort en fil eller mapp.
    - Avsluta medlemskap
        - Radera användaren
        - Radera användarens innehåll
            - Kontrollera att filerna inte delar administrativa rättigheter med andra, fråga i så fall de andra om de vill att filen ska vara kvar eller tas bort. Konsensus krävs för att radera en fil.

Hur lång tid tar det och vem ska utföra det?

Kalle: Filhanteraren: Visa filer och mappar á la Explorer och Finder, beräknad tidsåtgång: 4 veckor.

Sara: Filhanteraren: Möjlighet till grundläggande funktionalitet, beräknad tidsåtgång: 2 vecka. Senaste ändringar: Färgkodning och så vidare, beräknad tidsåtgång: 1 vecka.

Gösta: Textredigeraren: Visuellt tilltalande utseende, beräknad tidsåtgång: 3 veckor. Hantera OOXML formaterade dokument, beräknad tidsåtgång: 1 vecka.

Sandra: Textredigeraren: Funktioner för enklare redigering, beräknad tidsåtgång: 4 veckor.

Herbert: Behörigheter, delning och avsluta medlemskap, beräknad tidsåtgång: 4 veckor.

Reflektera kring svårigheterna i att planera en uppgift som denna!

Den första och kanske mest uppenbara svårigheten när det kommer till att planera projektet är att det innehåller så många moment som en novis inom webbprogrammering aldrig har kommit i kontakt med, till exempel hur man gör en enkel WYSIWYG textredigerare eller hur lång tid det kan tänkas ta. Det blir direkt lättare heller av att inte veta hur många delmoment som krävs eller om man gör bäst i att låta en person arbeta med den eller dela upp den på flera.

Oerfarenheten gör det också svårt att estimera hur stor del av projektet som kommer flyta på bra och var som det kan behövas planera in extra tid för problem som troligtvis kommer uppstå. För en novis verkar allt vara problem men en erfaren programmerare skulle mycket väl redan så här tidigt i projektet kunna förutsäga vad som kommer vara svårast att lösa.

Omfattningen på arbetet gör det också svårt att försöka beräkna tiden då en försening hos en programmerare påverkar flera andra programmerares arbete. Exempelvis ska Sara implementera de grundläggande funktionerna i filhanteraren som Kalle skriver och Sandra ska implementera funktioner i Göstas textredigerare. Ett sätt att hantera detta är att låta programmerarna ha olika delmål att jobba med som inte är beroende av varandra. Sara kan till exempel arbeta på färgkodningen om Kalle har problem med att komma igång med filhanteraren.

En sak som kan hjälpa till att få projektet att gå fortare att utföra är att använda ramverk och tidigare lösningar från andra projekt. Webbyrån kanske en färdig modell för att hantera användare? Kanske jQuery och/eller Ajax kan förkorta utvecklingstiden avsevärt? Finns det färdiga Open Source applikationer att använda sig av?

Vilka krav som ställs på kompabilitet inverkar också på tiden projektet kommer ta. Om applikationen ska fungera på enbart senaste versionen av Chrome behöver inte hänsyn tas till att applikationen ska fungera tillfredsställande i exempelvis Firefox. Om å andra sidan krav ställs på att applikationen ska fungera från till exempel IE7 och uppåt kommer enbart testandet i olika webbläsare att ta upp en stor del av tiden.

Hur mycket som faktiskt kan göras parallellt är också svårt att veta. I projektplanen delas några av uppgifterna upp mellan olika personer men om båda verkligen kan påbörja sina olika delar samtidigt är inte helt självklart. Även om de kan arbeta parallellt kan man ställa sig frågan huruvida det hade gått fortare att göra en sak i taget och kanske hjälpas åt med samma uppgift.

Bilaga A

Tidslogg

Uppgift	Tid	Dag	Arbetsuppgift
0	1 h 30 min	2014-11-13	Strukturera upp och planera labboration 1. Se till att dokumentet finns med rubriker, tidslogg med mera. Skapa nytt rep på github med branches till de första uppgifterna. Skapa solutions i Microsoft Visual Studio.
1a	45 min	2014-11-13	Programmera och reflektera kring uppgift 1a
GitHub	10 min	2014-11-13	Bråkade med GitHub. Alla branches försvann från Microsoft Visual Studio.
1b	2h 30 min	2014-11-15	Arbetade med uppgiften
1b	20 min	2014-11-16	Städade i koden och mergade brancherna.
1b	20 min	2014-11-17	Slutförde uppgiften.
1c	1 h 45 min	2014-11-17	Påbörjade och slutförde uppgiften.
2.1-3	20 min	2014-11-17	Påbörjade och slutförde uppgiften.
3a	1 h 35 min	2014-11-23	Påbörjade och slutförde uppgiften.
3b	1 h 45 min	2014-11-23	Påbörjade och slutförde uppgiften.
4	3 h 45 min	2014-11-24	Påbörjade och slutförde uppgiften.
Labb 1	30 min	2014-11-24	Paketera och skicka in laborationen.