1DV404 – Iterativ Mjukvaruutveckling

Laboration 1

Av: Mattias Wikström – mw222rs – grupp E2

Tidslogg:

Tid förbrukad	Dag	Vad?
1,5 h	2014-11-17	Skapade detta dokument och läste igenom uppgiften.
5 h	2014-11-22	Uppgift 1a – 1b
5 h	2014-11-23	Uppgift 1c
8h	2014-11-23	Uppgift 2, 3, 4

Uppgift 1a – RaknaA

1. Jag planerar att denna uppgift kommer att ta mindre än en timme för mig att genomföra, då jag i stort sett har gjort en likadan uppgift i JavaScript-kursen jag läser parallellt med denna. På grund av denna redan gjorda uppgift blir det heller inte särskilt många moment att gå igenom här, jag tänker helt enkelt gå tillbaka till den kod jag skrev i den tidigare uppgiften, kopiera de delar jag kan återanvända här och sedan lägga till de rader kod jag behöver lägga till för att räkna antal A och a i strängen.

   Detta har jag enkelt tänkt lösa genom att först omvandla strängen till en array med .split-metoden och stegar sedan igenom arrayen med .map och räknar då enkelt hur många A'n och a'n arrayen innehåller.

2. **Genomförande:**
   1. Skapade repositorie på github och kopplade mot cloud9 – 15 min.
   2. Skapade ett antal filer för denna uppgift och b- och c-uppgifterna – 10 min.
   3. Kopierade de rader kod jag hade användning för från tidigare uppgift och ändrade funktionen så att den istället för att manipulera en sträng (vilket var uppgiften i den tidigare uppgiften) till att helt enkelt räkna antal stora och små A i en sträng – 15 min.
   4. Provkörde programmet och allt verkar funka som det ska – 5min.

3. Jag vet inte riktigt hur mycket jag kan reflektera kring denna planering och mitt genomförande då jag egentligen visste exakt hur jag skulle lösa uppgiften direkt jag läste vad det var jag skulle göra – således behövdes alltså ingen planering. Själva genomförandet gick även det utan några större problem, jag skrev de rader kod jag behövde och det fungerade som väntat.

Uppgift 1b – RaknaSiffror

1. Jag planerar att även denna uppgift kommer att ta mindre än en timme att genomföra då jag i stort sett kan använda exakt samma kod som från a-uppgiften

men i .map-metoden istället för att kolla om varje bit av en array är ett stort eller litet a kolla om det är en nolla, en jämn siffra eller en udda siffra.

De jag moment jag måste genomföra är endast att byta ut några rader kod i .map-metoden, som i a-uppgiften med hjälp av ett par if-satser räknade antal a och A, för att istället få den att gå igenom arrayen (som jag skapar från den inmatade strängen med .split-metoden) och räkna siffror istället med hjälp av ett par lite annorlunda if-satser.

2. **Genomförande:**
    1. Gjorde en kopia på koden från a-uppgiften och ändrade i den funktion som där gick igenom arrayen och räknade a och A så att den istället gick igenom arrayen och med hjälp av tre if-satser räknade hur många nollor, jämna och udda siffror det inmatade heltalet innehåller. Hade några små problem med typomvandling som gjorde att jag missade nollorna, något som jag kom på efter lite testande. – 40 min.
    2. Provkörde och allt verkade funka. – 5min.
3. Likt a-uppgiften fanns det inte så mycket att planera för denna uppgift då jag i stort sett förstod hur jag skulle gå till väga för att lösa uppgiften redan då jag läste vad det var jag skulle skapa för program. Jag gjorde helt enkelt som jag planerat och ändrade i .map-metoden för att få det resultat jag eftersträvade.

Uppgift 1c – NastStorsta

1. Planeringen för denna uppgift är lite svårare, då jag inte riktigt vet hur jag skall lösa denna uppgift. Jag hade enkelt kunnat lösa den om det inte hade varit för notisen i uppgifts-förklaringen om att man ej får använda arrayer eller liknande. Hade jag fått använda arrayer hade jag ju bara använt .sort-metoden på arrayen och fått den fint sorterad efter storlek. Nu antar jag att jag måste sortera varje inmatat nummer. Jag planerar att det här kommer att ta ganska lång tid då jag inte riktigt vet vad jag skall göra och inte vet vad jag skall kolla upp innan, det är nog bara att sätta sig ner och försöka.
2. **Genomförande**
    1. Börjar med att lösa problemet med en array, sedan läser jag att man inte får använda arrayer i uppgiftsförklaringen. Börjar om från början. 30 min
    2. Gör om funktionen så att varje nummer som läses in sparas i en variabel som sedan jämförs med en biggestNumber-variabel, är det inmatade numret större så får biggestNumber värdet av det inmatade värdet. Hur man håller reda på det näst största numret är helt enkelt att om det inmatade numret är större än biggestNumber så tar helt enkelt secondBiggestNumber värdet av biggestNumber och biggestNumber får det inmatade värdet istället. Det är dock inte så enkelt, för ett nummer kan ju vara mindre än det största men ändå större än det tidigare sparade näst största, så en ELSE IF-sats läggs till där vi undersöker om det imatade värdet är större än det tidigare sparade secondBiggestNumber. – 3,5 h
    3. Problem jag kom i kontakt med handlar helt enkelt om att jag inte fick det att fungera som det skall om man bara matar in negativa siffror, något jag fick lösa med att instansiera biggestNumber- och secondBiggestNumber-variablerna med värdet av NEGATIVE_INFINITY, för om man instansierar utan värde blir det bara

undefined av det hela, och om man – som jag gjorde i början – instansierade till 0 så kommer det inte att funka att testa storlek mot negativa tal. Lite halvtrassligt som JavaScript ofta verkar vara. – 1 h

3. Att det tog längre tid än väntat att lösa den här uppgiften beror inte så mycket på min bristande planering, även om jag hade kunnat spara lite tid genom att läsa igenom uppgiften bättre och inte lagt ner onödig tid på att lösa problemet med arrayer först, utan mera i att jag är väldigt ny med JavaScript och inte riktigt van vid den milda galenskap som fria typer är. Jag vet dock inte om det är något som jag hade kunnat lösa genom bättre planering eller förundersökning inför denna uppgift, det känns mera som något som jag lärde mig genom lösandet av uppgiften och de problem jag stötte på.

   Visserligen hade jag väl kunnat läsa på mera allmänt om JavaScript inför som en del av planeringsarbetet, men jag kände inte riktigt att det skulle varit möjligt att få det mera tidseffektivt i och med att jag inte visste vad jag skulle göra för fel innan jag så att säga gjorde de. Jag hade inte tänkt på att jag bara kunde börja med att instansiera variablerna till NEGATIVE_INFINITY för att vara säker på att de inmatade talen alltid blev större i de större än- mindre än-undersökningarna jag genomför (och på så sig försäkra mig om att även negativa tal hänger med i beräkningarna). Hade jag satt mig ner och bara bläddrat i JavaScript-bibeln hade det nog bara tagit ännu längre tid. Även om det kanske inte var den mest effektivt lösta uppgiften i historien så tror jag att det var mer eller mindre den snabbaste lösningen jag kunde genomföra.

Uppgift 2 – Förändring och förbättring

2a. Jag vet inte riktigt hur strategier spelade in i planeringen av dessa programmeringsuppgifter då de var så pass simpla att jag kunde lösa med en sorts råstyrka, slå sönder dörren istället för att leta reda på nyckeln-lösning – det vill säga att jag bara kunde sätta mig ner och försöka och försöka tills jag löste det hela istället och ändå göra det på samma tid som om jag först hade gjort en rad för rad-planering. Divide and conquer känns liksom inte så relevant som strategi när koden som löser problemet knappt är 10 rader lång.

2b. Det jag tror att man kan göra för att minska konsekvenserna av eventuella oplanerade fel eller andra saker är nog dels att dela upp problemet i mindre delproblem och lösa de eftersom, men även att planera in lite buffert-tid för att kunna klara av några smällar utan att planeringen helt fallerar. Jag vet inte om jag kan ge så många konkreta exempel baserat på uppgifterna hittills i denna uppgift då de som sagt varit lite för simpla för att kunna vinna på att delas upp till mindre problem som sedan löses för sig innan man går på alltsammans, men visst hade jag nog vunnit på att inte vara så sent ute med själva laborationen i sig.

2c. Jag förstår inte riktigt vad som menas med ”förbättringsåtgärder” men det jag hade kunnat göra bättre är nog bara att jag borde ha börjat med denna laboration tidigare än några få dagar innan deadline.

Uppgift 3a – Palindrom

1. Jag planerar att denna uppgift kommer att ta ungefär 45 minuter att genomföra. Detta tack vare att jag enkelt kan använda JavaScripts inbyggda array-metoder för att vända på och sedan göra om arrayen till en sträng igen för att sist men inte minst jämföra med den oförändrade inmatade strängen. Är de exakt likadana så är det en palindrom, är de olika är det inte en palindrom.
2. **Genomförande:**
    1. Tar den inlästa strängen och lägger till .split(””).reverse().join(””); för att väldigt effektivt på bara en rad först splitta strängen till en array, vända på ordningen i arrayen och sedan sedan sätta samman arrayen till en sträng igen. - 15 min
    2. På den andra raden i funktionen skriver jag helt enkelt `return (reversed === str) ? str + " är en palindrom" : str + " är inte en palindrom";` vilket då returnerar den inmatade strängen plus *”är en palindrom”* om den inmatade strängen är exakt likadan som den omvända varianten eller *”är inte en palindrom”* om den inte är det. 20 min.
3. Precis som uppgifterna 1a-c så finns det här inte särskilt mycket att reflektera över, då uppgiften löstes så relativt enkelt. Vissa kanske skulle tycka att min rätt så minimalistiska kodstil är svårläst - som att jag valt att splitta, vända på och sedan sätta ihop den imatade strängen på bara en rad istället för att dela upp det över tre - men jag tycker att när man jobbar med inbyggda metoder som har så förklarande namn som split, reverse och join så borde det inte vara några problem.

Uppgift 3b – Fraction

1. Denna uppgift har jag lite problem med att planera då jag inte riktigt tror att den är skriven med JavaScript (som jag ju programmerat hela laborationen i) i åtanke, utan det känns väldigt C#. Men jag tänkte väl helt enkelt bara skapa en konstruktor som heter Fraction och slänga in en massa grejer i sen så blir det ju typ som en klass i C#.
2. **Genomförande:**
    1. Skapar filen Fraction.js i vilken jag skapar en konstruktor-funktion som skapar och initialiserar ett nytt bråktal som det skall. 20 min
    2. Lägger till metoderna getNumerator och getDenominator som properties på konstruktorn. 30 min
    3. Lägger till isNegative-metoden som kollar om det är negativt. 10 min.
    4. Metoderna add och multiply skapas och plussar ihop, respektive multiplicerar det ursprungliga bråktalet med det bråktal man skickar med som argument. 30 min
    5. isEqualTo och toString skapas utan större problem. 40 min
3. Reflektionen över den här uppgiftens planering och genomförande blir nog inte heller så lång. Jag satte mig helt enkelt ner och skrev kod som jag tycker fyller de krav jag uppfattar finns i uppgiftsbeskrivningen. De problem jag mötte under kodandet hade precis som tidigare mest att göra med min lilla erfarenhet med JavaScript och var väl därför inte direkt något jag tycker att jag hade kunnat komma förbi genom att planera mitt genomförande på något annat sätt – det var bara att sätta sig ner och försöka och sedan lösa de problem som uppenbarade sig.

Uppgift 4 – Planering

4a.

Vi måste först och främst bestämma vem som skall göra vad. Förmodligen är vi fem i gruppen alla någorlunda dugliga på att programmera, men kanske är någon bättre på något annat än de andra. Vi pratar igenom våra tidigare erfarenheter och styrkor/svagheter, i vilket område av skapandet av tjänsten som vi tror att vi kommer göra mest nytta och så vidare.

Efter det delas det ut roller till de fem deltagande i gruppen, en får axla ansvaret att hålla koll på de andra och på något sätt se till att allt inte bara spretar ut allt för mycket. Dock så är vi ju så få att herr ”chef” inte bara kan sitta och ta det lugnt och kolla på de andra jobba – även hen måste programmera men får då kanske en lite lättare roll i själva programmeringsbiten.

Arbetet delas därefter upp i mindre delar och en tidsplan för varje del skapas, sedan läggs dessa ihop och ger en större tidsram för hela projektet. ”Chefen” har haft kontakt med kunder och säger att det kommer att ta högst två månader att klara av projektet – men efter sammanträde med de fyra andra i gruppen kommer alla överens om att det nog minst behövs 3 månader.

4b.

Den största svårigheten för mig att planera en uppgift som denna är att jag inte har en aning om hur man i själva verket skulle göra för att genomföra ett liknande projekt. Jag har hållit på med programmering sedan i september, och har verkligen inte kommit så långt att jag kan estimera hur mycket tid det kommer att behövas för att skriva ett program som håller koll på administratörsrättigheter och åtkomstlistor. Jag vet inte ens vad OOXML-dokument innebär.

Bortser man från det och antar att frågan är mera allmän – vad är svårt med att planera ett stort projekt som detta? Ja det är väl just det som vi pratat om väldigt, väldigt kort och lätt om i någon av föreläsningarna – att det är svårt att veta vilka problem man kommer möta innan man möter dem. Lösningen är väl att divide and conquer så mycket man bara kan, och dela upp arbetet i små mindre delar som man kan ta sig igenom med jämn takt.

Det är dock också väldigt svårt för mitt att svara på den mera allmänna frågan också då jag inte känner att jag fått lära mig någonting alls hittills om olika modeller eller vad det nu kan tänkas heta, UML eller SCUM eller något liknande? Det är ju bara ord som svischat förbi på en slide under en föreläsning, ingenting vi har gått igenom med något som helst djup. Hade jag fått några läsanvisningar till kurslitteraturen eller annat hade jag väl kunnat ta till mig det på så sätt, men det har ju varit radiotystnad på kursens hemsida i två veckor.