Tre “enkla” programmeringsuppgifter.

Skriver av: Jens Larsson UD 2014.

tidsplan:
från 2014-11-20 18.00. till 11/25 00.00. Det blir ca 6 dagars arbetstid.

Uppgifter:
3 uppgifter små uppgifter gjorda i valfritt programmeringsspråk. 2 medelstora uppgifter och en stor uppgift som bara ska planeras.

Planering:
Skriva ner hur långt tid varje uppgift skulle ta och

Genomförande:
Min strategi är att skriva kod tills jag kommer till något problem i koden där jag behöver läsa mig på för att klara den. Därefter googlar jag först lösningar och kollar material från skolan hemsida. Eventuellt frågar jag någon annan om vad dom använde sig av för att lösa uppgiften.

Deltagare:
Jens Larsson UD.

Lösning av fel:
Eftersom att jag inte har programmerat tillräckligt och stött på något problem som går någorlunda lätt att rätta till så kör jag med att testa mig fram.

uppgift 1.a

Planering: Jag planerade att den skulle ta 90 minuter. Den tog 60 minuter.

Fel:

För att jämnföra integern input med ett litet a (Char), skrev jag först “a” och trodde skulle fungera. Jag bytte istället ut “a” till ‘a’ så att kompilatorn ser att bokstaven är av typen Char.
avvikelser: Uppgiften bestod bara av ett svårt problem och behövde heller inte struktureras upp för att klaras av.

uppgift 1.b

Planering: Jag planerade att uppgiften skulle ta 90 minuter. Den tog 30 minuter.
Fel: Inget
avikelser: den tog mindre tid än vad jag planerade.

uppgift 1.c

Planering: Jag planerade att den skulle ta 90 minuter. Den tog 45 minuter.
Fel: Jag började med att “göra” en egen array som tilldelade värden till 10 stycken variabler. Jag insåg att det inte gick något vidare. Sen läste jag uppgiften en gång till och kom på den var enklare än vad jag hade förutspått.
Avikelser: Den tog mindre tid än jag planerade.

Reflektion för alla

Jag började med UML projektet sent och har nu bara några få dagar på mig. Därför planerar jag att lägga ner så många timmar som möjligt och se hur långt jag kommer innan inlämningen.

Jag började jobba på första uppgiften för att uppskatta hur lång tid den skulle ta.
Dom första uppgifterna hade jag tillräcklig kunskap om för att klara av på mindre än 3 timmar.
En större planering dom små uppgifterna är frågan då man inte behöver strukturera upp koden med klasser eller metoder.

Dom problemen jag hade med uppgiften gick att lösa genom att ställa lämplig fråga på google och se ungefär hur en liknande uppgift är löst.

Uppgift 1a. En planering för att lösa uppgift 1a skulle kunna vara att dela in uppgiften så att funktionaliteten av varje kodbit är omsluten av en metod/klass eller är indelad med en kommentar. Dock så behöver inte denna uppgift något sånt eftersom den är för liten för att bli behandlad objektorienterat.

Om man tänker att varje metod eller klass ska ta emot eller skicka saker, så kan man börja strukturera upp hela uppgiften och sedan hålla på att lösa en del åt gången. tillexempel i uppgift 1.a så skulle jag kunna göra en metod som har ett lämpligt namn och uppgiftatt utföra.tillexempel räkna a och heta “CountAs”. Låta den vara så så länge och gå vidare med att strukturera upp nästa del av projektet.

uppgiften 1.b gick att lösas på ett enkelt sätt genom att att låta ascii numret bestämma ifall tecknet var stort eller litet. Därför undrar jag om uppgiften hade kunnat lösas på ett väldigt annorlunda sätt nu i efterhand. Såsom om det skulle finnas medföljande C# metoder att använda sig av.

Uppgift 1.C Jag tänkte att det ända sättet att lösa uppgiften på var att använda sig av en array. Då man inte fick använda sig av en så tänkte jag mig då att man försöker göra någon imitation av en.
Bara 2 nummer behöver sparas ner åt gången för att kunna presentera näst sista siffran. Därefter ändras grundkraven från att behöva spara ner 10 siffror till 2 siffror.

2 förändring och förbättring

2.a En alternativ planering för att klara av mina uppgifter hade varit att estimera ungefär hur lång tid varje uppgift hade tagit. Börja med att läsa material kring uppgiften och skriva ner hur långt tid den tar. Nästa moment är ställa frågor kring koden och estimera tid för varje större problem.
Dela in lämplig minimum tid för varje problem så att man har marginal tills att ska var inlämnat.

ett annat alternativ är att börja med att göra lätta delar av uppgiften i så stor utsträckning som möjligt. Sedan skriva ner alla delar av uppgiften som är svåra att genomföra och estimera en större portion tid för att klara av dom.

2.b Eftersom att jag inte startade med uppgifterna direkt så hade jag istället kunnat börja med uppgifterna vid leveransdatumet och klarat av delar av per dag. Då hade jag kunnat skriva en implementationslista där varje implementation hade kunnat ha en estimerad tid. Tillexempel på uppgift 2b så höll på att försöka göra en array. Jag fråga sedan andra och dom hade klarat av uppgiften med en foreach sats. Därefter valde jag att försöka lösa uppgiften med en foreach sats istället.

Åtgärder: När jag försökte lösa 1.b uppgiften så tog jag första bästa metod för att lösa uppgiften och det kan ställa till det för en om den inte fungerar. en annan metod fär att minska konsekvernerna är kolla upp olika sätt att lösa uppgiften på innan man snöar in sig på en.
Som att istället välja foreach satsen istället för att försöka imitera en array.

2.c 2 förbättringsåtgärder för uppgifterna 1.a 1.b och 1.c

.

1. göra snabba prototyper: skriva kod snabbt för att se vad som behövs i arbetet.
2. använda sig av divide and conquer.

3 förbättrad planering av programmering

Planering för uppgift 3a.

moment 1 tids planering 15 min. jag tänker att det tar 60 minuter att läsa om uppgiften. 30 min för en prototyp.. läsning och lösning av det svåraste problemet i uppgiften 60 min. 60 min tesning och bugfixande.
sammanlag: 3.5 timmar per uppgift. 3.5 * 3 = 10.5 timmar för alla uppgifter

moment 2. 60 minuter. läsa igenom och förstå uppgiften.
gjort: läst om uppgiften, palindrom och hur man gör en palindrom i c#.

slutsats:
Hur man omvänder en sträng
http://stackoverflow.com/questions/228038/best-way-to-reverse-a-string

sida om hur man kollar ifall uppercase.
https://social.msdn.microsoft.com/Forums/vstudio/en-US/0e24579c-3c38-4a0d-aa5f-6732495c2c9b/check-whether-string-contains-uppercase-letters

moment 3. 60 min strukturera upp bra kod.

slutsats: eftersom att det stod lämpliga namn på alla metoder som skulle vara med i uppgiften, kunde man strukturera upp koden genom det och sedan lösa dom bit för bit.

moment 2. 60 minuter gör prototyp.
Prototyping tänket blir till att varje metod löses och att man sedan går vidare när den fungerar.

moment 4. 120 min hitta lösningar på alla problem i uppgiften.

moment 5. 60 min debugging.

Fel: jag började med att kolla på metoder som kunde kolla omvända ett ord men slutade med en for loop som omvände ordet.

“else” if (read.Any(char.IsUpper)) var tvungen och ha else framför.

jag var tvungen att lägga till -1 till for loopens i.length för att read[i] inte skulle gå utöver index.

Reflektion:

När jag gjorde denna uppgift ledde den till som dom andra ett svårare problem i uppgiften. I detta fallet var det att kunna se om ett ord var ett palindrom. Därför tänkte jag mig att det var huvudproblemet i uppgiften. Jag gjorde därför allt annat innan jag gav mig på det stora problemet. Vilket var introduktion, Inmatning av orden och felmeddelande vid felinmatning.

Planering för uppgift 3b.

moment 1. 15 min för att tidsplanera. tid estimerad för uppgiften: 6 timmar. Den tog: 4 timmar.

moment 2. 60 minuter. läsa igenom och förstå uppgiften.
Klart: Jag läste om divisions formler.

moment 3. 60 minuter gör prototyp.
Klart: Jag började skriva ner allt innehåll som fanns med i uppgiftens presentation
Strukturerade upp klassens metoder och kunde på så viss se vad som behöver göras.

moment 4. 60 min strukturerar upp bra kod.
Eftersom att första prototypen var tillräcklig så fick den bli mallen på programmet.

moment 5. 120 min hitta lösningar på alla problem i uppgiften.

moment 6. 60 min debugging. Felmeddelande om inmatningen är 0, var inte implementerat.

moment 7. 15 min granskning. i uppgiften står det att man bara ska göra en klass men frågan är om koden ska gå att kompileras. jag gör ett program som kan exekveras istället.

att göra :
2 tal ska gå att mata in i början.
dom 2 talen ska sparas ner i divisionsform. Möjligtvis en metod som kan spara ner tal hade varit praktiskt. *SaveInput*.
Dom 2 talen ska sedan Presenteras.

klart: jag sklippade *SaveInput* metoden och gjorde så att programmet kan mata in 2 tal som kan multipliceras och adderas.

moment 7. detta moment tillägger jag då jag måste göra ett program som kan kompilera och testa klassen *Fraction*.

Uppgiften: tog 2.5 timmar kodning. och 3 timmar läsning.

avikelser:

Eftersom att det är svårt att veta hur långt tid var och en del av uppgiften tar att göra är det också svårt att bedöma tiden.

Fel:

tänkte först att *add* och *multiply* metoderna skulle ta in en int type som argument men såg senare att argument från objekt var bättre.

Skrev först att additions formeln skulle bli (tal1 + tal2)/tal. men ändrade den till den rätta formeln.

Eftersom att inte om hur jag skulle strukturera upp *Main* klassen så testade jag mig fram.

jag var tvungen att ändra returtyper och argument till mer passande några gånger.

Reflektion:

Denna uppgift var mycket lättare att implementera divide and conquer på. Man kunde börja med att skriva ut alla metoder och sedan lösa dom en efter en. Eftersom att det bara står att man ska skriva en klass och inte att hela programmet ska kunna köras så körde jag först på bara en klass. Sedan insåg jag att det var bäst och kunna köra programmet för att på så vis testa det fullt ut.

Uppgift 4 - planering

Uppgifter.

UI

1. skall vara ett gränssnitt.
2. inloggning
3. filutforskare
4. lista med senast ändrade dokument.
5. listan ska ha färgkodning efter sorts modifiering och enligt användare användare. vem som ändrar dokumentet måste visas.
6. inbyggd textredigerare.

användare.

1. Skall kunna ladda upp befintligt formaterade OOXML formaterade dokument.
2. skapa dokument med en inbyggd textredigerare.
3. administera sin portfölj. Flytta, byta namn, radera dokument och mappar.
4. kan ändra åtkomst på sina mappar.
5. kan göra mappar eller dokument publika så att andra kan ändra på dokumenten. meddelande skickas då
6. vid radering av konto raderas också alla mappar/dokument. Om andra har fått rättigheter till dokumenten så kommer dom få meddelande om samtyckte. Alla som har rättigheter måste samtycka för att dokumenten ska raderas.

Dokument

1. är i OOXML-Format
2. ska ha åtkomstlista och nyttjanderätt.
3. lagras i ett moln.

Administratör

1. kan ändra åtkomst på allas mappar.

Tid per uppgift.

1. inloggning. 100 timmar.
2. filutforskare. 100 timmar.
3. Lista. 50 timmar

4. färgkodning 50 timmar
5. 5 inbyggd textredigerare.. 2000 timmar
6. uppladdning av OOXML dokument. 50 timmar.
7. administering av dokument 50 timmar.
8. rättighets hantering av dokument 200 timmar
9. meddelandekorg 50 timmar.
10. moln lagring av dokument 300 timmar.
11. adminstratör hantering 50 timmar.
12. design av UI 300 timmar.

Sammanlagt blir det 3000 timmar med 2000 av dom på textredigeraren.

Vem gör vad.

om person 1 är en designer kan han designa UIN. Såsom inloggningen filutforskaren, listan och designa ikoner. m.m.

person 2 kan göra inloggning. färgkodning och användar administering av dokument

person 3 kan göra rättighets hantering och administratörs hanteringen..

person 4 filutforskaren och listan.

person 5 kan göra molnlagringen.

Reflektion

Svårigheter:
storhet
antal personer
3 stora program

Denna uppgiften är väldigt svår att estimera då man skulle kunna säga att den består av 3 delar som skulle kunna bara program i sig.

Den inbyggda textredigeraren som ska göras står det inget om. Först och främst ska man kunna skriva dokument i den för att kännetecknа sig som en textredigerare. många

dagens textredigerar har mängder med funktioner såsom rättstavning, markringsverktyg, layout design, färgblandings alternativ, utskrift redigerare. Den inbyggda textredigeraren skulle kunna var hur sofistikerad som helst. Den skulle ta flera år att utveckla eller bara bara kunna göra det mest grundliga. Ett annat alternativ kanske är att beställa någonslags färdig variant.

Nästa bit som är svår att estimera tid och skulle vara ett projekt i sig är molnlagringen. Mängder med servrar, hårdiskar och nätverkskonfigurationer ligger nog bakom. Också undehållning då lagringsutrymmet måste utökas.

Själva webbsidan som användaren ska surfa in på är kan ses som ett projekt i sig.

För att verkligen få reda på hur lång tid programmet skulle ta att göras så skulle varje inviduell uppgift brytas ner i antal klasser/objekt som måste skapas.