**Objectorienterad Programering**

**Laboration 4**

**Pontus Axgart**

**Objektorienterad programmering – Laboration 4**

Laboration 4 består av tre sammanhängande uppgifter som kan ge Godkänt eller Väl godkänt. För godkänt krävs att alla uppgifter är lösta med minst godkänt resultat. För väl godkänt krävs dessutom väl godkänt på minst två uppgifter. Redovisningen görs muntligt för din lärare/handledare och skriftligt i form av en labbrapport.

**Kaffemaskin**

|  |
| --- |
| Image courtesy of Stuart Miles/ FreeDigitalPhotos.net |

Du arbetar hos företaget Go-kaffe AB. Ni ska utveckla en ny kaffemaskin vid namn Barista 3000 som era kunder ska kunna hyra och placera i sina fikarum.

Kunderna kommer att betala dels en fast månadsavgift och dels en rörlig avgift som baseras på förbrukning. För att kunna ta ut rätt avgift från varje kund är det viktigt att maskinen loggar information om hur många och vilka drycker som serverats.

De drycker som maskinen ska kunna laga till och servera är följande:

* Bryggkaffe med/utan socker/mjölk
* Café au lait
* Varm choklad
* Tevatten

Maskinen ska alltid portionera ut en kopp dryck åt gången. Den ska ha behållare för kaffepulver, chokladpulver, mjölk och socker. Den ska anslutas till en kallvattenkran samt ett vanligt eluttag.

Maskinen kommer till en början att ha ett enkelt konsolbaserat användargränssnitt. Längre fram kommer detta att bytas ut mot ett mer användarvänligt grafiskt gränssnitt.

All utveckling ska göras med objektorienterad analys och design samt implementation med de objektorienterade tekniker som täckts hittills i kursen.

**Uppgift 1: Klassdiagram (G – VG)**

Börja med att identifiera de olika entiteter som ingår enligt ovanstående beskrivning. Rita ett UML-klassdiagram eller annan skiss som visar entiteterna som klasser och deras inbördes relationer. Identifiera några attribut och operationer för varje klass.

**Uppgift 2: Sekvensdiagram (G – VG)**

Utifrån de klasser som du identifierat i uppgift 1, rita sekvensdiagram för följande händelser:

* En kund väljer och får en kopp kaffe med socker och mjölk.
* En kund väljer och får en kopp tevatten. Maskinen fyller först på vattenbehållaren.

**Uppgift 3: Implementation (G – VG)**

Implementera din lösning i form av en C#-konsolapplikation.

Tänk på att hålla GUI-delen så skild som möjligt från själva programlogiken, eftersom den kommer att bytas ut så småningom.

Vid bedömningen tittas särskilt på

* användning av objektorientering (interface, arv, polymorfism etc)
* vilka entiteter som finns med (begripliga, identifierbara)
* om entiteterna motsvarar fysiska företeelser eller processer
* ansvarsområden (inte för stort, inte för litet)
* kopplingar mellan klasserna
* att användargrässnittet är separerat från logiken i så hög grad som möjligt
* att själva klassdiagrammet är ritat på rätt sätt
* Sekvensdiagram: för enkelhetens skull går det bra att enbart rita flöden mellan olika klasser, inte alla enskilda objekt.

Liksom tidigare läggs stor vikt vid upplägg och struktur, bland annat

* att alla specificerade funktioner finns implementerade,
* att tidigare funktionalitet fortfarande fungerar korrekt,
* att det inte finns några uppenbara logiska fel eller brister i lösningen,
* att studenten använder antingen svenska eller engelska konsekvent,
* uppdelning i klasser och metoder,
* tydliga variabelnamn,
* beskrivande felmeddelanden,

För det högre betyget krävs:

* felhantering med exceptions
* kommentering av klass och funktioner samt inuti koden där det skapar större tydlighet.
* större vikt läggs också vid separation mellan GUI och logik.

**Laboration 5**

**Pontus Axgart**

**Objektorienterad programmering – Laboration 5**

Laboration 5 består av två sammanhängande uppgifter som kan ge Godkänt eller Väl godkänt. För godkänt krävs att båda uppgifterna är lösta med minst godkänt resultat. För väl godkänt krävs väl godkänt på båda uppgifterna. Redovisningen görs muntligt för din lärare/handledare och skriftligt i form av en labbrapport.

**Kaffemaskin, version 2**

Första versionen av kaffemaskinen blev en veritabel succé. Go-kaffe AB har därför bestämt sig för att fortsätta utvecklingen av Barista 3000.

Version två av maskinen ska utökas med två nya drycker:

* Kallt vatten (direkt från kranen)
* Kolsyrat vatten (kranvatten till vilket kolsyra tillsätts från en inbyggd patron)

Utöver detta ska maskinen fungera på exakt samma sätt som tidigare.

För att programkoden ska bli mer underhållsbar i framtiden finns önskemål om att bättre använda sig av de verktyg som finns i och med objektorienterad programmering, till exempel arv och polymorfism, där detta är lämpligt.

**Uppgift 1: Klassdiagram (G – VG)**

Se över klassdiagrammet från förra laborationen. Identifiera möjligheterna att använda objektorientering fullt ut. Börja med att identifiera en övergripande struktur, därefter kan du utforma klasser med attribut och operationer, samt inbördes relationer. Inför ändringarna i klassdiagrammet.

**Uppgift 2: Implementation (G – VG)**

Implementera din lösning i form av en C#-konsolapplikation. Utgå från implementationen av laboration 4 och gör de förändringar i koden som behövs.

Tänk på att liksom tidigare hålla GUI-delen så skild som möjligt från själva programlogiken, eftersom den kommer att bytas ut så småningom.

Vid bedömningen tittas särskilt på

* användning av objektorientering (interface, arv, polymorfism etc)
* vilka entiteter som finns med (begripliga, identifierbara)
* om entiteterna motsvarar fysiska företeelser eller processer
* ansvarsområden (inte för stort, inte för litet)
* kopplingar mellan klasserna
* att användargrässnittet är separerat från logiken i så hög grad som möjligt
* att själva klassdiagrammet är ritat på rätt sätt

Det förutsätts att studenten använder sig fullt ut av objektorienterade koncept såsom arv, interface och polymorfism där så är lämpligt.

Liksom tidigare läggs stor vikt vid upplägg och struktur, bland annat

* att alla specificerade funktioner finns implementerade,
* att tidigare funktionalitet fortfarande fungerar korrekt,
* att det inte finns några uppenbara logiska fel eller brister i lösningen,
* att studenten använder antingen svenska eller engelska konsekvent,
* uppdelning i klasser och metoder,
* tydliga variabelnamn,
* beskrivande felmeddelanden,

För det högre betyget krävs:

* felhantering med exceptions
* kommentering av klass och funktioner samt inuti koden där det skapar större tydlighet.
* större vikt läggs också vid separation mellan GUI och logik.

**Laboration 6**

**Pontus Axgart**

**Objektorienterad programmering – Laboration 6**

Laboration 6 består av en uppgift som kan ge Godkänt eller Väl godkänt. Redovisningen görs muntligt för din lärare/handledare och skriftligt i form av en labbrapport.

**Kaffemaskin, version 3**

**Implementation (G – VG)**

Säljarna Lasse och Klasse på Go-kaffe AB jublar. De har just lyckats avsluta en affär som handlar om att hyra ut kaffemaskinen Barista 3000 till ett större bemanningsföretag i IT-branschen. Alla vet ju att det går åt mycket kaffe i IT-branschen, så det här ser ut att bli en riktigt lönsam affär.

Kunden har dock ett krav som behöver tillgodoses före leverans. De anser att det gamla konsol-gränssnittet är svåranvändbart. Istället önskar de sig ett grafiskt användargränssnitt där alla val enkelt kan utföras genom att klicka på knappar.

Din uppgift är att designa ett grafiskt användargränssnitt för kaffemaskinen och att därefter

implementera din lösning i form av en C#-Windows-forms-applikation. Utgå från laboration 5 och gör de förändringar i koden som behövs. Om du har följt råden i föregående uppgifter, bör du enkelt kunna byta ut konsol-GUI-klassen mot Windows-GUI-klassen och behålla nästan all underliggande kod oförändrad.

För det högre betyget ska dessutom följande tillägg implementeras (använd Microsofts C#-dokumentation på nätet för att lösa uppgifterna):

* En illustration (till exempel en bild på en kaffekopp) visas i programfönstret
* En ljudfil (\*.wav) spelas upp i samband med att en dryck serveras

Liksom tidigare läggs stor vikt vid upplägg och struktur, bland annat

* att alla specificerade funktioner finns implementerade,
* att tidigare funktionalitet fortfarande fungerar korrekt,
* att det inte finns några uppenbara logiska fel eller brister i lösningen,
* att studenten använder antingen svenska eller engelska konsekvent,
* uppdelning i klasser och metoder,
* tydliga variabelnamn,
* beskrivande felmeddelanden

För det högre betyget krävs:

* felhantering med exceptions
* kommentering av klass och funktioner samt inuti koden där det skapar större tydlighet.
* En illustration (till exempel en bild på en kaffekopp) visas i programfönstret
* En ljudfil (\*.wav) spelas upp i samband med att en dryck serveras

Testautomatisering

**Laboration 1**

**Pontus Axgart**

 1           Introduktion

Denna uppgift består av att man ska skriva ett urval av enhetstester mot en kaffeapplikation.

För att få godkänt ska man få ett godkänt resultat på de fyra första uppgifterna.

För väl godkänt så ska man även klara alla deluppgifter under uppgift 5.

De kriterier som tas hänsyn till är:

* Korrekthet (att testerna går att kompilera och köra med förväntat resultat och att testerna faktiskt testar det som de ska).
* God struktur, namngivning och läsbarhet.

Redovisning sker dels genom inlämning av enhetstester och dels genom muntlig presentation av enhetstesterna.

**1.1          Uppgift 1 (G)**

Det allra första ni ska göra är skapa upp ett enhetstestprojekt, skapa en referens till *NUnit* samt en referens till kaffeapplikationsprojektet. Därefter ska ni fortsätta med nedanstående uppgifter. Fundera också på vilket sätt ni vill köra enhetstesterna på (genom NUnit-runnern eller internt i Visual Studio). Det är helt valfritt vilket man väljer.

**1.2          Uppgift 2 (G)**

Skriv två enhetstester mot klassen **CoffeMaker** somframgångsrikt testar att metoden **MakeDrink()**kan skapa:

* Bubbelvatten
* En kaffe med mjölk

**1.3          Uppgift 3 (G)**

Skapa två enhetstester mot klassen **SugarContainer** som testar att sockerbehållaren:

* faktiskt minskas ner korrekt efter att man tar något från sockerbehållaren.
* kastar ett **EmptyContainerException** när man får en tom sockerbehållare.

**1.4          Uppgift 4 (G)**

Utveckla ett enhetstest för klassen **WaterContainer** som verifierar att vattenbehållaren återfylls automatiskt när den blir tom.

**1.5          Uppgift 5 (VG)**

Lyft ut skapandet (Arrange-fasen) av **CoffeMaker-**klassen så att den skapas upp i en för testklassen gemensam **[Setup]**-metod.

**1.6          Uppgift 6 (VG)**

Skapa ett parametriserat positivt test för giltiga värden hos klassen **CoffeeStrengthEstimator** och dess metod **EstimateCoffeeStrength()**.

**1.7          Uppgift 7 (VG)**

Skapa ett parametriserat negativt test med ogiltiga värden för klassen CoffeeStrengthEstimator och dess metod **EstimateCoffeeStrength()**.

**Laboration 2**

**Pontus Axgart**

**1           Introduktion**

Under kursen har vi gått igenom olika testfall för att automatisera för API-gränssnitt med SoapUI och C#. Arbetet görs i par. Alternativt görs laborationen individuellt om särskilda skäl finns. Börja med att läsa igenom alla uppgifter. Betygskriterierna uppfylls genom att göra de olika uppgifterna enligt informationen i föreläsningsmaterialet.

För betyget G görs uppgift 1, 2 och 3.

För betyget VG görs alla uppgifter i 4.

Mer detaljerna om vad som ingår i arbetet beskrivs i föreläsningsmaterialet.

**1.1          Uppgift 1 (G)**

Skapa upp ett nytt *REST-projekt i SoapUI* som ska innehålla testfall för Libris XSearch. Utgå från ett av exemplen i dokumentationen som URI. Det är till detta projekt som du ska lägga till nya testfall.

* Skapa ett testfall för sökning efter titlar av författaren Robert Jordan
* Validera att svaret innehåller namnet ’Jordan’
* Validera att svaret innehåller dokument på engelska
* Skapa ett testfall som söker efter ämnet: ’testning’. Använd Json-formatering för svaret
* Validera att svaret indikerar antalet träffar är större än 100. Använd er av SoapUIs Script Assertion för att kontrollera detta. I denna bloggpost kan ni läsa mer: [http:// HYPERLINK "http://www.robert-nemet.com/2012/08/json-asserting-in-soapui.html"robert-nemet.com/2012/08/json-asserting-in-soapui.html](http://www.robert-nemet.com/2012/08/json-asserting-in-soapui.html)
* . Titta direkt under rubriken ”Asserting Json” om hur man gör.
* Skapa negativa API-tester i SoapUI
* Validera svaret när man försöker anropa XSearch med en sökfråga som inte ger någon träff, exempelvis: ’very\_long\_string\_query\_with\_no\_valid\_reply’
* Validera svaret om man INTE skickar med den obligatoriska parametern ’query’.

**1.2          Uppgift 2 (G):**

Samma testfall som Uppgift 1 men i Visual Studio C# projekt.

**1.3          Uppgift 3 (G):**

Skapa i Visual Studio upp en mängd datadrivna testfall för XSearch med olika sökparametrar samt som ger positiva och negativa utfall. Notera att även det förväntade testutfallet ska vara en av parametrarna.

**1.4          Uppgift 4 (VG):**

Hitta och testa en annan API som är REST-baserat. Här kan du hitta en mängd olika förslag: <http://www.programmableweb.com/category/all/apis?data_format=21190>

* Skapa Visual Studio C# projekt för att testa tjänsten. Skapa några testfall utan att använda ramverk eller modell för svaren.
* Bygg C# klassmodell för API-svaren. Omvandla testade API-svar till objekt enligt modellen.
* Bygg ramverk för API-anrop. Skapa en mängd positiva och negativa testfall med hjälp av ramverk strukturen.
* Skapa en mängd datadrivna testfall.