DA339A Inlämningsuppgift 3

Syfte

Inlämningsuppgift 3 är en del av de moment i kursen som examinerar följande lärandemål:

- Kunskap och förståelse
  - visa förståelse för ett algoritmiskt tankesätt
  - visa förståelse för strukturerad och objektorienterad programmeringsteknik
  - visa förståelse för användningen av analys och modellering i utvecklingsprocessen för att ta fram programvara
- Färdighet och förmåga
  - kunna strukturera och implementera objektorienterade program i ett programmeringsspråk
  - kunna använda strukturerad och objektorienterad programmeringsteknik vid programutveckling
  - känna till och till viss del kunna använda befintliga klasser i klassbibliotek
  - kunna använda analys och modellering vid framtagning av programvara
  - kunna dokumentera programvara

Inlämningsuppgift 3 utgör provkod 2005 *Inlämningsuppgifter del 3*. Provkoden utgör 2 hp på kursen och har betygsskala UV vilket innebär att studenten kan erhålla betyg U, G eller VG på uppgiften. För att erhålla betyget VG så krävs att alla delar för G är uppfyllda samt även de delar som krävs för VG.

Den muntliga redovisningen är också en läraktivitet i kursen då studenten får återkoppling på inlämnad lösning. Vid redovisningen närvarar en annan student som också ska redovisa och varje student lyssnar på en annan students redovisning och återkoppling. Att se andras kod och andras lösningar är en viktig del i att lära sig programmera och detta är ett tillfälle för detta.

Redovisning

Inlämningsuppgift 3 redovisas genom skriftlig inlämning på kursplatsen på Canvas samt en muntlig redovisning.

Inlämning ska endast göras om du har en komplett lösning som du anser uppfyller kraven för uppgiften.

Du får inte lämna in en lösning som saknar delar och använda redovisningstiden för hjälp. Behöver du hjälp med inlämningsuppgiften så nyttjar du handledningstider och/eller labbtider till hjälp för detta. Den som försöker redovisa en lösning som saknar delar kommer att avbrytas och hänvisas till nästa hjälptillfälle för hjälp och nästa omtillfälle för inlämning/redovisning.

Skriftlig inlämning

I den skriftliga redovisningen ska följande lämnas in:

- En zip-fil med källkod som innehåller de filer och kataloger som ingår i projektet. De filer som lämnas in ska kunna kompileras och programmet exekveras. Struktur med kataloger för paket ska behållas i zip-filen men inga andra kataloger eller filer ska finnas i inlämningen.

- En pdf-fil med namn enligt mönstret *da339a_InUpp3_förnamn_efternamn.pdf* (förnamn och efternamn ersätts med ditt namn) som innehåller bilder med diagram. Diagrammen får inte vara ritat för hand utan ska vara ritat med något verktyg exempelvis Visual Paradigm (men måste inte nödvändigtvis vara ritat med VP).

Deadline för inlämning till det ordinarie tillfället är **onsdag 15/12 23.55**. Inlämningar som lämnats in sent garanteras inte redovisning vid det ordinarie tillfället.

Om man inte lämnat in till det ordinarie tillfället eller blir underkänd vid den muntliga redovisningen för det ordinarie tillfället så ges det två omtillfällen med deadline för inlämning:

- **Söndag 11/01 - 2022 23.55**
- Augusti 2021, exakt datum ej bokat

Observera att detta är de enda examinationstillfällena som garanteras för Inlämningsuppgift 3.

Muntlig redovisning

För att göra den muntliga redovisningen så måste en skriftlig inlämning gjorts innan deadline. Endast kompletta lösningar får redovisas.

Har du gjort en inlämning av den skriftliga delen så bokar du en tid för redovisning via kalendern för kursen i Canvas. Tider för muntliga redovisningar blir tillgängliga efter deadline för den skriftliga inlämningen. Se separata instruktioner för hur du gör en bokning av muntlig redovisning.

För godkänt på uppgiften krävs att den muntliga redovisningen godkänns. Det är inte tillräckligt att endast ha lämnat in en skriftlig redovisning som löser uppgiften. Studenten måste kunna förklara sin kod och hur denna fungerar och kunna svara på frågor om vad som händer i koden under vissa förutsättningar. En icke tillfredställande muntlig redovisning kan resultera i att kompletteringar av den skriftliga redovisningen kan behöva lämnas in.

Förberedelser

Inlämningsuppgift 3 omfattar kursinnehåll som gåtts igenom till och med föreläsning F21 och laboration L22. I Inlämningsuppgift 3 så används flera olika byggstenar för problemlösning som tidigare används i laborationer och exempel.

Uppgift

Övergripande beskrivning

I uppgiften så ska du implementera ett program i Java baserat på MVC-modellen och med ett GUI (Graphical User Interface) i Swing. Ett befintligt kodskelett för GUI:t finns på kursplatsen: **Inlämningsuppgift3.zip.** Kodskelettet behöver kompletteras med Controller-klass, Model-klasser och funktionalitet. Du måste inte använda detta kodskelett utan kan skapa din helt egna lösning för detta från start eller med inspiration och utgångspunkt från Inlämningsuppgift 2.

Uppgiften innebär att du skall öva på att ta fram en objektorienterad lösning med inkapsling, generalisering och polymorfism (F19 och F20). Du skall även göra ett klassdiagram med generalisering och MVC (Model View Controller), samt ett sekvensdiagram för din lösning.

Uppgiften är inte lika hårt specificerad som de föregående så det är upp till varje student att bestämma vilka klasser din lösning behöver och hur ditt GUI (F21) skall se ut (även om det gess viss ledning nedan och i kodskelett för den som vill använda det). Det ställs dock vissa krav på arkitekturen och vad denna ska innehålla.

Ditt uppdrag i denna uppgift är att skriva en Java-applikation som kan vara förberedd för användning av en generell restaurang, men ska fungera specifikt för en pizzeria. Applikationen ska hantera beställningar hos restaurangen. Applikationen ska ha en arkitektur som stöder eventuell utbyggnad av denna till att hantera fler typer av maträtter och meny-val än pizzor.

Krav på funktionalitet för betyg G

Det program du skapar ska uppfylla följande krav på funktionalitet i programmet:

Id för krav	Kravbeskrivning
IU3FG1	Applikationen ska ha ett sammanhängande grafiskt gränssnitt, det vill säga ett huvud-fönster med GUI-delar (eventuellt enligt kodskelett) och egen design.
IU3FG2	GUI skall visa en lista på vilka pizzor man kan beställa. Varje pizza ska listas med vilken topping den innehåller (tomatsås, ost, skina som exempel på toppingar) och pris.
IU3FG3	Användaren ska kunna välja en pizza att beställa från listan med pizzor.
IU3FG4	Användaren ska kunna ange hur många pizzor av en viss typ som ska beställas.
IU3FG5	Användaren ska i en order beställa minst 1 stycken pizza av 1 sort.
IU3FG6	Applikationen ska inte begränsa hur många olika sorters pizza användaren kan beställa så länge sorten finns i menyn.
IU3FG7	För en beställning ska det visas hur många pizzor av vilken typ som beställts och vad den totala kostnaden blir för beställningen.
IU3FG8	Användaren ska kunna välja att se en lista på beställningar som listar ett id för beställningen, vad denna innehåller och den totala kostnaden.

Krav på funktionalitet för betyg VG

Det program du skapar ska uppfylla följande krav på funktionalitet i programmet om du vill ha betyg VG. GUI och kod behöver utökas eller ersättas med funktionalitet enligt nedan:

Id för krav	Kravbeskrivning
IU3FVG1	Applikationen ska lista olika drycker som kan beställas.
IU3FVG2	Drycker ska separeras i alkoholfria och alkoholhaltiga drycker.
IU3FVG3	För alkoholhaltiga drycker ska alkoholprocent visas.
IU3FVG4	Användaren ska kunna beställa drycker genom att välja dryck från en lista och ange antalet av drycken som ska beställas.
IU3FVG5	Om användaren beställer en alkoholhaltig dryck så ska användaren behöva intyga att hen är över 18 år för att beställningen ska kunna göras
IU3FVG6	Det ska finnas ett val som innebär att skapa sin egen pizza. I detta val anger användaren från en lista vilka toppingar pizzan ska ha (tillåtet att "hårdkoda" dessa) och ger pizzan ett namn. Detta sparas sedan som en ny pizza i menyn som man kan välja pizza från. Priset bestäms utifrån antalet toppingar som pizzan innehåller (du sätter själv intervall för detta i ditt program).

Krav på implementation för betyg G och VG

Följande krav ställs på din implementation av uppgiften:

Id för krav	Kravbeskrivning
IU3I1	Källkoden ska vara väl formaterad med lämplig indentering som presenterats på kursen.
IU3I2	Högst upp i alla .java-filer ska det finnas en kommentar som innehåller: • Förnamn och efternamn • Datorid • Vilket program du läser
IU3I3	Källkoden ska gå att kompilera och exekvera.
IU3I4	Uppgiften ska vara löst på rimligt sätt utifrån de verktyg för problemlösning som presenterats på kursen till och med F21.
IU3I5	Samtliga instansvariabler skall vara privata.
IU3I6	Metoderna i Controller-klassen skall dokumenteras kort ovanför metoddeklarationen gällande metodens syfte och namn, datatyp och kort beskrivning ges för varje parametrar samt datatyp och beskrivning för eventuellt returvärde (se L18 och Forumtest.java för exempel).
IU3I7	Applikationen skall programmeras med en strikt MVC-struktur där view och model inte skall känna till varandra (de ska inte importera varandras paket).
IU3I8	Minst ett interface skall ingå i lösningen.
IU3I9	Minst en abstrakt klass med minst en abstrakt operation/metod skall ingå i lösningen.
IU3I10	Lösningen ska innehålla en hierarki med generalisering i minst två nivåer (en superklass som har ett lager av en eller flera sub-klasser).
IU3I11	Arkitekturen ska vara förberedd för att programmet ska kunna utökas med fler typer av maträtter och drycker i menyn för beställningar. Detta ska göras genom att skapa en lämplig arkitektur med generalisering, abstrakta klasser och interface. Alla dessa klasser används kanske inte i applikationen i nuläget (mer av detta används i uppgiften om man gör delen för VG) men ska finnas förberedda med eget kod-skelett och synas i klassdiagram (även om man endast implementerar G-delen).
IU3I12	När programmet startas skapas minst 5 olika sorters pizzor och läggs in i programmet som pizzor man kan beställa. Detta görs **inte** i GUI utan görs lämpligtvis i konstruktorn till control-klassen.
IU3I13	Det som går att beställa ska hanteras som en eller flera arrayer via control-klassen (klassen Array, ArrayList och liknande är tillåtet att använda för den som önskar).
IU3I14	Beställningar ska sparas och hanteras som en array av beställningar via control-klassen (klassen Array, ArrayList och liknande är tillåtet att använda för den som önskar).
IU3I15	Lösningen ska använda dynamisk bindning där detta går att tillämpa.
IU3I16	Lösningen ska använda polymorfism där detta går att tillämpa.

Krav på diagram för betyg G

Din lösning ska dokumenteras med ett klassdiagram och ett sekvensdiagram. Säkerställ att diagrammen du lämnar in stämmer överens med varandra och den kod du lämnar in.

Följande krav ställs på diagrammen:

Id för krav	Kravbeskrivning
IU3D1	Klassdiagrammet ska följa den UML-notation som presenterats på kursen.
IU3D2	Klassdiagrammet ska visa huvudklassen för GUI-delen, dvs. klasser i koden som anropas av någon klass av stereotypen Control (förmodligen bara en klass). Control-klasser och alla entity-klasser skall visas i klassdiagrammet.
IU3D3	Klassdiagrammet behöver inte ta hänsyn till att klasserna ligger i olika paket. Alla klasser visas i ett och samma klassdiagram utan paket i diagrammet.
IU3D4	De klasser som ska visas i klassdiagrammet ska visas med alla attribut och namn och datatyp/klass för attributen. Operationer/metoder behöver inte visas i klassdiagrammet.
IU3D5	För alla klasser i klassdiagrammet ska det anges om klassen är av stereotypen boundary, control eller entity.
IU3D6	Klassdiagrammet ska visa om en klass är abstract. Om operationer/metoder visas i klassdiagrammet ska det anges om en operation/metod är abstract.
IU3D7	Klassdiagrammet ska visa interface som skapats och som används i koden.
IU3D8	Associationer mellan klasser i klassdiagrammet ska visas med multiplicitet i bägge ändarna och lämpliga typer av associationer (exempelvis generalisering, aggregation eller komposition - överanvänd dock inte detta).
IU3D9	Det ska finnas ett sekvensdiagram som visar vad som sker när en användare vill se en lista över alla pizzor. Sekvensdiagrammet ska starta med den metod i Controller-klassen som anropas av någon GUI-klass.
IU3D10	I sekvensdiagrammet ska alla parametrar i operationer/metoder visas med namn och datatyp/klass och returvärden ska visas med datatyp/klass.
IU3D11	Sekvensdiagrammet ska visa alla anrop som sker mellan klasserna i klassdiagrammet för den aktivitet som sekvensdiagrammet gäller. Alla iterationer och selektioner i anropade operationer/metoder ska visas i sekvensdiagrammet.
IU3D12	Sekvensdiagram och klassdiagram måste vara konsistenta – det vill säga de måste stämma överens och om ett anrop sker från ett objekt av klass A till ett objekt av klass B så måste klass A och B vara associerade i klassdiagrammet.
IU3D13	Diagrammen ska vara ritade med något verktyg som Visual Paradigm eller liknande och får inte vara en handritad skiss på papper som sedan skannats in.

Tips och ledning

Gå igenom föreläsningar F16 till F21 som handlar mest om OOP. Labbarna L16 till L22 borde vara till hjälp då de mestadels kan användas i delar av denna inlämningsuppgift.

Försök inte göra allt med en gång! Gör bit för bit och räkna med att vissa saker kommer kanske att behöva ändras i din design när du kommit en bit på väg. Bygg upp programmet steg för steg för att ha bättre kontroll på vad som ger dig problem i nuläget.

Börja med att modellera entity-klasserna (här ingår abstrakta klasser och interface) – de klasser som är av typen model i MVC-mönstret. Tänk efter vilka krav som finns på arkitekturen.

Det kan vara lämpligt att skapa interface för olika typer av saker som kan visas i en meny och som kan beställas. Allt detta behöver exempelvis ha något namn som kan visas och ett pris.

Det är lämpligt att skapa mer generella abstrakta klasser för saker som kan beställas i menyn och som har gemensamma attribut och beteende men som skiljer sig åt i vissa avseenden. Dessa klasser implementerar det interface som finns för allt i menyn. För dessa klasser så skapa en lämplig hierarki med olika sub-klasser för exempelvis drycker och olika maträtter. Abstrakta metoder kan exempelvis vara hur pris beräknas då olika kategorier kan ha olika momssatser eller rabatter på sikt.

För pizzor så bör listan med toppings hanteras som en array med strängar då pizzor kan ha olika antal toppings och kan behöva ändras.

Nyttja klassdiagram för att få översikt över dina klasser och börja implementera dem efter hand och undersök hur de behöver relatera till varandra. Vilka klasser är inte beroende av någon annan klass - börja med att implementera dessa och bygg ut en klass i taget. Skriv gärna små testprogram till din kod som skapar objekt av klasserna och ger dem värde för att testa din kod.

Eftersom du får ett kodskelett med GUI, kan du sedan skissa din utökning av användargränssnitt på ett papper eller ett ritverktyg (eller Powerpoint, Word, Paint, etc.). Gränssnittet skall visa komponenter som hanterar nödvändiga input/output. Dessa i sin tur bör ge dig en bild av vilka instansvariabler (och även metoder) du behöver ha i olika klasser. Utifrån detta kan du designa Controller-klassen och GUI-komponenter. Tänk igenom de scenarier som du behöver genomföra i programmet (tänk på hur metoden för CRC-kort fungerar). Vilken information behöver finnas i GUI? Vilken information kan en controller-klass ta fram eller räkna ut givet vad den får från GUI – och vad behöver då skickas till GUI från början? Komplettera din arkitektur efter hand med metoder som behövs för att genomföra de olika scenarierna.

Glöm inte att spara ditt arbete ofta och ta även backup av det du gör på en plats så fort något fungerar bra.