CDIO - Projektoplæg 02362

Forår 2017
Udleveres: 21/2 – 2017
DTU Compute

1 Generel information

Projektet danner formelt grundlag for eksamen og gennemføres i grupper af 3-5 personer. Hver gruppe skal udvikle et program på baggrund af den nedenstående opgaveformulering og dokumentere programmet i en rapport, der skal afleveres senest **fredag den 5. maj 2017 kl 23:59.** Formalia omkring afleveringen: Se afsnit 5.

1.1 Bedømmelsesgrundlag

Det skal tydeligt fremgå, hvem der har været hovedansvarlig for de forskellige dele af både rapport og program. Der kan godt angives flere hovedansvarlige på udvalgte dele.

Det skal tydeligt fremgå, hvis der er dele af det afleverede – både rapport og program, der ikke er udarbejdet af projektgruppen selv – og hvilke dele, det i givet fald drejer sig om.

Bedømmelsen vil hovedsagligt ske på baggrund af selve rapporten samt en mundtlig prøve. Det er således ikke nok at lave et godt og veldesignet program. Rapporten skal

- dokumentere programmet
- være let at orientere sig i, læse og forstå.
- benytte relevant fagligt sprog og diagrammer

Mht. programmet lægges der vægt på, at det

- implementerer en betydelig mængde af den beskrevne funktionalitet
- er opbygget efter en flerlagsmodel og opdelt i relevante interfaces og klasser
- undgår gentagelse af kode ved anvendelse af hjælpe-metoder og nedarving
- indeholder eksempler på exceptions, collections, inputvalidering med Regex, osv.

2 Krav til rapporten

Rapporten forsynes med en forside med navne, studie-numre, og underskrifter (se bilag A).

Rapportens omfang: Max 40 normalsider.

Sidenumre på alle sider (nederst i midten på hver side).

Forslag til afsnit i rapporten:

1. **Indholdsfortegnelse**
2. **Forord og indledning:** Skriv i hvilken sammenhæng projektet er udført, og giv en kort beskrivelse af rapportens indhold. Læseren skal få et indtryk af, hvad det handler om.
3. **Problemstilling og baggrund:** Beskriv kort kravene til programmet (~ opgaven).
4. **Analyse:** Beskriv for hver af de centrale problemstillinger evt. alternative løsningsmuligheder og begrund derefter den valgte løsning. Dette kunne blandt andet vedrøre design, datastrukturer eller databasen.
5. **Implementation:** Her beskrives både kodning af manglende java-komponenter og databasen til persistering af et igangværende spil.
6. **Software dokumentation:** Beskriv selve **java**-programmet herunder det overordnede design og de vigtige klassers ansvar. Udarbejd et UML designklassediagram og evt. et UML Sekvensdiagram over et centralt program-flow. Endvidere forventes eksempler på anvendelse af Javadoc.
   Angiv strukturen af databasen dvs. tabeller inkl. nøgler. (E/R-diagram – logisk skema osv.). Fysisk database dokumenteres også med et eller flere **mysql**-scripts
7. **Test:** Beskriv, hvordan programmet er afprøvet, og hvad afprøvningen viste, udmøntet i flg. 3 afsnit: a. Strategi for test (hvad testes og hvad testes ikke), b. Selve testen og c. dokumentation for testen.
8. **Brugervejledning og eksempel:** Beskriv, hvordan man installerer programmet – herunder, hvad man skal gøre for at få oprettet de nødvendige tabeller, der skal bruges af programmet. Beskriv kort, hvordan man bruger programmet efter installation. Vis relevante kommenterede skærmdumps fra en programkørsel med typiske data.
9. **Konklusion:** Opsummér arbejdet og giv en vurdering af resultatet. Angiv herunder kort og præcist, hvad det afleverede program kan (implementeret funktionalitet) samt evt. væsentlige mangler.
10. **Litteraturliste**
11. **Appendiks:** SQL-scripts (CRUD i det omfang det skønnes relevant)

3 Vejledning og Review/Demo

Der kan hentes vejledning i løbet af projektperioden med assistance af en DB-hjælpelærer.

Før I går i gang med *4.1.5 Delopgave 4 (se nedenfor)* skal jeres E/R diagram godkendes af Bjarne og/eller Finn.

Det færdige E/R-diagram, beskrivelse af mapning til logisk model osv. skal også medtages i rapporten, sammen med de databasemæssige læringsmål I har anvendt (f.eks.: bevis for normalform, vews, triggers, ”Stored Procedures”, osv.).

Efter nogle uger gennemføres et review for hver gruppe (5/4). Her vil gruppens medlemmer præsentere kort det foreløbige resultat af deres arbejde (diagrammer og kode). Der forventes her tillige en demo-kørsel af en prototype / det delvist færdige program. Gruppen får herefter feedback på status. Demo er udskudt til eksamensdagen.

4 Opgaven

Overordnet: En færdiggørelse af Matador-spillet med mulighed for at gemme ”et igangværende spil”.

4.1 Matador

Der ønskes udarbejdet et program til simulering af et Matador spil (manglende moduler og komponenter)

4.1.2 Delopgave 1: Datastruktur

Et matadorspil består, som alle ved, af en spilleplade, et sæt af terninger samt 2-6 antal spillere. Endvidere består en spilleplade af en sekvens af felter af forskellig type. Design de overordnede datastrukturer og klasser, der repræsenterer en spilleplade, et felt, en spiller og et sæt terninger. Det et krav, at I tegner et UML diagram, som viser designet.

4.1.3 Delopgave 2: Valg af funktionalitet

Det er tanken, at så mange af spillets funktioner som muligt skal implementeres. Dog kan det være hensigtsmæssigt (evt. pga. kompleksitet) at fjerne nogle af spillets funktionaliteter eller sætte begrænsninger på dem, f.eks. bytning af grunde på kryds og tværs.
Se evt. kopi af Matador-spilleregler og spillepladen.

4.1.4 Delopgave 3: Design tilstandsmaskine

Når det bliver en spillers tur, står han/hende over for en række valg, muligheder og begrænsninger - f.eks. kan en spiller, der lander på en grund, som ikke ejes af andre spillere, købe grunden, pantsætte egne grunde og så købe grunden, eller undlade at gøre noget. Design dén tilstandsmaskine, som håndterer disse valg, muligheder og begrænsninger. Den minimale opfyldelse af dette krav – er et tilstandsdiagram.

4.1.5 Delopgave 4: Indlæsning og udlæsning af spil (gemme og hente et spil)

Da et matadorspil kan tage lang tid, er det hensigtsmæssigt, at man kan hente og gemme et spil. Det er et krav, at denne funktionalitet implementeres ved at gemme alle relevante tilstande i en lokal eller central database (mysql).
Afvikling af SQL-statements kan ske ved selv at skrive Java-kode, der benytter **JDBC**.

4.1.6 Delopgave 5: Brugergrænseflade

Anvend den GUI I fik i efteråret, så spillerne kan:

1. spille matador, dvs. slå med terninger, rykke brik, købe grunde, osv.
2. gemme spil, (kræver 4.1.5)
3. hente spil, (kræver 4.1.5)
4. afslutte spil

Brug evt. grafik-pakken fra efteråret til visning af spilleplade mm.

4.1.7 Delopgave 6: Hvad skal med?

De ovenfor skitserede emner (se afsnit 1.1, afsnit 2 ”Indholdsfortegnelse” og afsnit 3). Jo flere læringsmål (fra 02327 0g 02362) I kan finde anvendelse for – jo bedre. Der vil nok altid være nogle, som ikke passer på opgave-typen, men det må man også meget gerne skrive.

5 Formalia vedr. afleveringen

Rapporten, Eclipseprojektet og SQL-scripts afleveres kun **elektronisk på campusnet**:

Digital aflevering - via CN som en zipfil (med rapport + bilag + kode mv) det vil sige alt.

- Eclipseprojektet skal navngives efter gruppenummer. (Ex: Matador_grp_A)

Husk at oprette jeres gruppe på CN med alle studienumre under menupunktet opgaver.

Bilag A: Standardforside

Projektopgave 2016.

02362 Projekt i Softwareudvikling

Afleveringsfrist: fredag den 5/5 2017 Kl. 23:59

Gruppe nr.: **B**

Gruppedeltagere:
Studie nr.,Efternavn, Fornavne Underskrift

s000001, Høgh, Stig _____
Kontakt person (Projektleder)

s000002, Johansen, Gunnar _____

s000003, Elk, Klaus

_____

Denne opgave indeholder 35 sider inkl. denne side.