

Extrauppgiftslabbar

Bakgrund

Den som vill höja sitt betyg måste göra extrauppgifter. Antingen gör man dom extrauppgifter som finns definierade i anslutning till dom ordinarie labbarna eller så gör man en helt fristående extrauppgiftslab.

Uppgifter

Du kan välja mellan ett antal olika extrauppgifter eller definiera en egen uppgift och efter överenskommelse med kursledaren värdera och genomföra den.

Extra Extrauppgifter

För att höja värdet av extrauppgiftslabben ytterligare kan du också göra något av följande.

Utvidgningarna värderas som 1-2 extrauppgifter, per moment beroende på omfattning (efter överenskommelse med kursledaren)

X1. Gör en *interaktiv textuell eller grafisk dialog* till din applikation. Dvs gör antingen ett textuellt dialoggränssnitt för användning i terminalfönster eller ett grafiskt dialoggränssnitt som använder lämpliga AWT-komponenter som knappar, inmatningsfält, menyer och listor. Se till att programmet klarar av "felaktig" inmatning utan att krascha. Konstruera också gärna en liten hjälpfunktion som ger en kort översikt av hur programmet används och dess funktioner.

X2. Gör din tillämpning *grafisk eller distribuerad*. Dvs antingen:

- gör du hela gränssnittet grafiskt så att dina objekt får en grafisk representation (båtar ritas som båtar, ett fiaspel ritas med plan och pluppar). Då ändring i spelet sker skall detta återspeglas i dess grafiska presentation (flyttas en båt eller plupp så skall detta synas på skärmen). Helst skall programmet också vara direktmanipulativt, dvs att man interagerar direkt med hjälp av musen osv (en plupp flyttas genom att man "tar tag i den och flyttar den").
- eller så gör du din tillämpning distribuerad med hjälp av av CORBA. Om det är ett spel så ser du till att flera kan spela, är det en simulering så kan du se till att den kan kontrolleras av flera klienter samtidigt, osv. Detta är ju ganska tillämpningsberoende och det går inte att exakt säga hur det hela skall gå till för alla typer av tillämpningar.

Uppgiftsförslag

1. Egen uppgift (värde 1-6 xtrauppgifter efter överenskommelse med kursledaren)

Du har möjlighet att själv hitta på en egen uppgift. Vill du göra det skall du lämna in en kort skriftlig beskrivning av uppgiften minst 2 veckor före redovisningstillfället eller muntligen prata med kursledaren. Efter godkännande av uppgiften kan du börja jobba med den.

2. Monopol (värde 3 extrauppgifter, med grafik blir det mer)

Denna uppgift går ut på att implementera spelet monopol. Du får bortse från vissa moment i reglerna om du vill, men det skall klart framgå av kompletterande UML-dokumentationen vilka regler som inte implementeras.

3. Simulering av flygplats (värde 3 extrauppgifter, med grafik blir det mer)

INTRODUKTION

Du ska simulera arbetet vid en flygplats. För att uppgiften inte ska bli alltför omfattande nöjer vi oss med att simulera startande och landande flygplan. Flygplatsen består därför av en trafikledning och en landningsbana. Varje flygplan startar regelbundet från flygplatsen under dagen. När planen har startat flyger de en sväng för att sedan komma tillbaka till flygplatsen. Efter landningen görs service på planen. När denna är avklarad startar planen igen.

UPPGIFT

Din uppgift är att göra en UML-beskrivning för ovanstående, implementera modellen i Java och simulera en del av en dag på flygplatsen (t.ex. 06.00-9.00). Om du tänker göra extrauppgift X1 och/eller X2 bör du redan nu tänka igenom dessa så att inte alltför omfattande ändringar av systemet måste göras i efterhand.

I princip får du välja själv hur du vill implementera systemet, men det måste finnas en klar koppling mellan UML-modellen och din kod (se nedan).

BESKRIVNING AV CENTRALA OBJEKT OCH FUNKTIONER

Flygplanen

När ett flygplan ska starta begär det av flygledningen att få ett starttillstånd. Flygledningen svarar med att tilldela flygplanet en tidslucka (time slot) i vilket planet ska starta. Planet väntar sedan till dess att tidsluckan infaller och startar då. På motsvarande sätt begär ett landande plan landningstillstånd. Om ett landande plan inte får landningstillstånd i tid kommer det att krascha.

Varje plan har en flygtid, ft , som talar om hur lång tid det ska gå från det att planet startar till dess att det landar igen. Men p.g.a. vindar m.m. under färden så kommer inte planen fram exakt i tid utan den verkliga flygtiden hamnar i intervallet $[ft-dt, ft+dt]$. Slumpa fram ett värde genom att använda klassen `Math` (se Budd kapitel 17). Planet antas också ha bränsle som räcker i bt minuter från starten.

När ett plan har landat behöver det service. Servicen tar tid, i medeltal tiden st . Men på grund av olika typer av problem som dyker upp varierar servicetiden. Använd samma modell som för flygtiden ovan med samma dt . Efter avslutad service är planet klart för att starta igen. Normalt ska tiden mellan avgångar vara at (t.ex. avgång varje timme om $at=1$ timme), men p.g.a. förseningar vid föregående flygning och service kan starten ibland bli fördröjd. Parametrarna ft , st , dt , bt och at ska vara lätta att ändra.

Trafikledningen

Trafikledningen har ansvaret för att tilldela flygplanen tidluckor vid start och landning. När trafikledningen får en begäran om landningstillstånd börjar den leta efter en ledig tidslucka. Denna process tar en viss tid, under vilken trafikledningen är upptagen. I extrauppgift X1 ska det vara möjligt för ett plan som måste nödlända att avbryta behandlingen av ett annat plan. Den tid det tar att hitta en ledig tidslucka ska enkelt kunna varieras i programmet.

Tidluckorna

För att planeringen av starter och landningar ska vara enkel delas tiden för en landningsbana in i tidsluckor, vilka här antas vara lika långa. Ett plan hinner precis landa eller starta i en tidslucka.

Tider

Det är enklast att arbeta internt med sekunder som tidsenhet.

STATISTIKINSAMLING

I grunduppgiften ska du samla in statistik på hur länge landande plan i snitt fick vänta från det att de begärde landningstillstånd till dess de landade. Motsvarande statistik ska också samlas in från startande flygplan. Statistiken ska skrivas ut terminalfönstret i slutet av varje simulering.

UTSKRIFTER UNDER SIMULERINGEN

Under simuleringen ska det vara möjligt att följa händelseförloppet genom att lämpliga utskrifter görs vid olika händelser.

4. Simulering av båttransporter (värde 3 extrauppgifter, med grafik blir det mer)

Du skall simulera båttransporter mellan ett antal olika hamnar. Situationen är följande. I hamnarna finns varor som skall transporteras till en annan hamn. För att transportera varorna finns ett antal båtar. Båtarna lastar varor i en hamn och fraktar varorna dit dom ska. I den nya hamnen lastas sedan nya varor och dessa transporteras till sin destinationshamn. Finns det för tillfället inga varor i en hamn ska båtarna vänta till nya varor anländer (alternativt gå till en annan hamn).

Hamnarna har bara plats för ett visst antal båtar vid kaj. Detta innebär att båtar som anländer när hamnen är full får vänta tills någon annan båt avgår. Varor som skall transporteras från en hamn ska kunna komma till hamnen efterhand som simuleringen körs, dvs alla varor som skall transporteras behöver inte finnas i hamnen från början. Varje vara har ett fraktpris, en vikt, en volym och en typ. Båtarna i sin tur kan bara lasta varor av vissa typer och inte mer än en viss vikt och volym. Det kostar också att frakta varorna, dels en kostnad som beror på avståndet mellan två hamnar, dels lastnings och lossningskostnader (du bestämmer själv hur dessa beräknas). Den vinst som båten gör genom att frakta varor beräknas som skillnaden mellan fraktpriset för varan och de kostnader som finns för frakten. När simuleringstiden är över skall statistik för varje båt skrivas ut som talar om vilka inkomster och utgifter som båten har haft.

5. Äventyrsspel (värde 3 extrauppgifter, med grafik eller distribution blir det mer)

I ett äventyrsspel går man omkring i en (från början) okänd värld. Världen kan t.ex. bestå av ett antal rum. I de olika rummen kan olika föremål finnas vilka kan plockas upp och användas på olika sätt. Det brukar också lura ett antal faror i form av troll, spöken, fladdermöss osv. Dessa faror kan vandra omkring i rummen. Vissa av föremålen man plockar upp kan användas som vapen eller på annat sätt skydda mot farorna. Det brukar vara möjligt att gå mellan de olika rummen genom att ange en riktning eller motsvarande. Det ska också vara möjligt för den som spelar att lista vilka föremål han/hon bär på samt att lista vilka föremål som finns i ett aktuellt rum. Lägg inte ner för mycket tid på att skriva en kommandotolk utan gör en relativt enkel kommandoavkodning.

Din uppgift är nu att specificera en UML-beskrivning av spelet och sedan implementera ett äventyrsspel. För att du skall få godkänt måste spelet vara tillräckligt komplicerat, detta bedöms utgående från UML-dokumentationen. Det ska vara lätt att ändra hur världen ser ut, att lägga till nya faror och att lägga till nya vapen/hjälpmiddel.

6. Fia med knuff (värde 2 extrauppgifter, med grafik blir det mer)

Din uppgift är att implementera spelet "Fia med knuff" i en objektorienterad version. Spelplanen skall ritas ut grafiskt på skärmen och man skall med hjälp av musen tala om vilken pjäs som skall flyttas. Efter att man klickat på pjäsen skall den själv flytta till sin nya position (beroende på vad programmets tärning visar). Programmet ska kontrollera att den pjäs som skall flyttas verkligen får flyttas.

Lite tips: Tänk noga igenom vilka objekt som är lämpliga att ha (spelare, spelplan, ...) och vilka data som dessa måste lagra. Tänk också igenom vilka metoder som de olika objekten behöver ha.

7. Konferenssystem (värde 3 extrauppgifter, med grafik blir det mer)

Konstruera ett konferenssystem a la ICQ. Basera det hela på CORBA (eller möjligen, lite svårare, direkt på socketar).

Tips: det finns redan en hel del kodexempel på följande adress

<http://www.nada.kth.se/kurser/kth/2D4334/99-00/contents/exempel.html>.

8. UML-editor

Gör en editor för att konstruera någon av UML:s olika diagramtyper. Värde av labben beror av vilken editor som görs. Användningsfall- och klassdiagram är troligen enklare än tillstånds-, aktivitets- eller sekvensdiagram.