

Laboration 3: *GUI och CORBA*

Bakgrund

Grafiska gränssnitt blir allt vanligare. Ofta programmeras dessa med objektorienterade tekniker. Java har ett rikt stöd för att konstruera grafiska tillämpningar genom att använda antingen Java-AWT eller Java-Swing som båda ingår i standardklassbiblioteket.

Idag ser vi också allt fler lösningar där användare samarbetar över nätverk. En del av detta samarbete yttrar sig i applikationer av mer asynkront slag, via tex WWW-sidor och databaser. Men vi ser också ett behov av mer intensivt och direkt samarbete personer emellan. I detta sammanhang finns det behov av programmeringstekniker som ger stöd för samarbete mellan programdelar som körs på olika plattformar, skrivna i olika programmeringsspråk, där också vissa anpassningar till olika personers eller organisationers specifika behov kan göras.

I detta sammanhang har OMG:s CORBA trätt fram som en de facto standard vad det gäller distribuerad programmering för datorstött samarbete. CORBA har idag bindningar till flera olika språk, såsom C++, Smalltalk, C och Java, och tillåter att kod och objekt skrivna i olika språk samtidigt kan användas och samarbeta.

Mål

Ge övning i och kunskaper om att skriva mer realistiska distribuerade grafiska interaktiva program.

Uppskattad tidsåtgång

20 timmar per gruppmedlem.

Förberedelse

Studera gränssnitten `Collection`, `Map`, `Set` och `Iterator` samt bla klasserna `Dictionary` och `HashMap` i paketet `java.util`.

Läs igenom materialet om CORBA och konstruktion av grafiska gränssnitt från föreläsningarna samt materialet som delats ut i anslutning till dessa föreläsningar.

Tips

Gränssnitt med Swing:

<http://java.sun.com/docs/books/tutorial/uiswing/mini/index.html>

CORBA med JDK:

<http://java.sun.com/docs/books/tutorial/idl/index.html>

Collections mm:

<http://java.sun.com/docs/books/tutorial/collections/index.html>

Uppgifter

Konstruera en bank men bankkonton och bankomater genom att utföra dom två följande uppgifterna.

1. Konstruera bank med konton och bankomater

Konstruera en enkel Bank med olika bankkonton och bankomater för att göra penningtransaktioner.

Banken skall kunna hantera:

- Bankkonton
Med kontonummer och inestående saldo.
Det skall gå att ta ut, sätta in och föra över pengar mellan konton.
- Bankkunder
Som kan ha ett eller flera konton.

Vidare skall ni konstruera *bankomater* med möjlighet att ta ut pengar, fråga efter saldo på konto eller föra över pengar till annat konto.

Distribuera bankomaten med hjälp av av *CORBA* så att den körs på en server med möjlighet för (bankomat-) klienter att ansluta och interagera med den.

Konstruera ett *enkelt grafiskt gränssnitt* för bankomaten och om ni vill för andra delar av applikationen (men det är inget krav). För detta kan ni koda för hand med förslagsvis Swing eller, exempelvis, använda JBuilders möjligheter att snabbt och enkelt konstruera gränssnitt av åtminstone formulärtyp.

Om ni behöver hantera *mängder*, tex för aggregat av konton eller kunder så skall klasser som implementerar gränssnittet *Collection* eller *Map* användas. Dvs vissa klasser i paketet `java.util`, som exempelvis `HashSet` och `Dictionary`'s subklasser. Om ni behöver iteratorer så ska ni använda *Iterator*'s API.

Ni måste också *använda JUnit* för att enhetstesta den kod ni skriver.

Testa alltid först

Så långt det är praktiskt möjligt ska ni som vanligt använda JUnit för att skriva tester innan ni skriver produktionskod.

FÖRSLAG TILL GENOMFÖRANDE

Skapa bankkonton

Skapa och testa (eller var det tvärt om testa och skapa...) bankkonton.

Konstruera bankomat

Konstruera en enkel bankomat med möjlighet att göra dom olika transaktionerna som krävs i uppgiften. Gör inget grafiskt gränssnitt än. Glöm dock inte att testa först!

Distribution med hjälp av CORBA

Distribuera bankomaten med hjälp av av *CORBA* så att den körs på en server med möjlighet för klienter att ansluta och interagera med den. Glöm inte att enhetstesta distributionen!

Grafiskt gränssnitt

Skapa grafiskt gränssnitt för bankomat. Ni behöver inte automatiskt gränssnittet, om ni nu inte kommer på något smart och enkelt sätt att göra detta på.

Tänk på: Uttag eller insättning sker på av användaren angivet konto. Dvs man "loggar in" och anger det konto man vill bli ansluten till. Alla transaktioner sker sedan mot det aktuella kontot till dess att "kunden" avslutar eller, som extra frivillig finness, en viss tid av inaktivitet har överskridits.

2. Implementera också följande "roliga finesser"

Om flera personer kopplar upp sig mot samma konto skall dom andra göras medvetna om detta. Tillför också en möjlighet för klienter som använder samma konto att skicka textmeddelanden till varandra.