UMEÅ UNIVERSITET
Institutionen för datavetenskap

1999-08-23

Kalle Prorok,
Thomas Johansson

Tentamen(+lösningsförslag) i Objektorienterad programmering för ingenjörer (TDBB09) 1999-08-23, kl 09.00 - 15.00

Lokal: Skrivsal 4, Östra Paviljongerna, Ålidbacken 23

Hjälpmedel: inga (förutom penna, suddgummi och matsäck)

Kursutvärdering: Sist bland papperen finns en kursutvärderingsblankett. Ta med den hem och fyll i den. Lämna in den till Thomas när du hämtar ut den rättade tentan.

Börja varje uppgift på ett nytt blad.
Använd bara en sida av papperet.
Skriv namn på varje blad.
Lägg uppgifterna i rätt ordning.
Observera att uppgifterna inte är numrerade efter svårighetsgrad och kom ihåg att även delvis lösta uppgifter kan ge poäng.
Kommentera noga all källkod så att vi kan se hur du har tänkt.

Maxpoäng är 40, fördelade på 5 frågor.

Betygsgränser:

5	32p
4	26p
3	20p

Fråga om det är något som är oklart ! Vi kommer till skrivsalen ca **10.30** och **12.00**.

Lycka till !

Thomas Johansson
tel 786 62 59

Kalle Prorok
tel 786 50 18

UPPGIFT 1 – OO (9p)

a) Objektorienteringens tre grundprinciper talar man ibland om, vilka är de? Beskriv varje princip kortfattat (1-2 meningar). (3p)
Inkapsling, Arv och Dynamisk bindning
b) Vad kännetecknar (är) ett objekt och ge några tumregler för att hitta dem! (2p)
VAD vi hanterar, har tillstånd/minne, beteende. Substantiv i spec, verkliga objekt, enheter, centrala begrepp.
c) Vilka 5 typer av relationer brukar vi prata om? (2p)
Klass-objekt(instansiering), Användning, Association, Aggregat, Arv
d) Varför vill man maximera inkapslingen och minimera beroenden? (2p)
Ger större säkerhet och enklare återanvändning,underlättar utbyggnad&förändring.

UPPGIFT 2 – C++ (7p)

a) Om du använt dynamiska strukturer (allokerat minne med new) i en klass så är det ofta nödvändigt att ta med två speciella metoder. Vilka (1p) och varför bör du ta med dem? (1p)
Initierings(kopierings)konstruktorn och tilldelningsoperatorn.
(konstruktor/destruktor resp new/delete har gett 1p pga luddig fråga)
b) *Referenser* används med fördel vid två tillfällen i ett program. Vilka tillfällen? (2p) och varför är referenser fördelaktigt just här? (1p)
Praktiskt när man skall kedja anrop och då man behöver ett ”vänstervärde”. Då man vill ändra parameterns värde (typ swap(int&i,int&j)) Sparar tid vid parameteröverföring av stora objekt, ev via const referens.
c) Varför bör utskriftsoperatorn göras som en vän? Ge exempel på hur den kan se ut eller hänvisa till uppgift 5! (2p)
För att man ska slippa ändra i standardbiblioteket (klassen är i iostream). Kanske bör den kunna komma åt medlemmar i applikationens klass. class Minklass { friend ostream& operator<<(ostream&os, const Minklass& m);private: int medlem;}; ostream& operator<<(ostream&os, const Minklass& m){ os<<m.medlem;return os;}.

UPPGIFT 3 – Struktur (10p)

a) Varför är *exceptions* ett bra sätt att hantera fel på? Visa hur det görs i C++ (2p) Vilket är det stora problemet med felhantering? (2p)
*Separerar kod och felhantering. try{ func(); throw ”Fel nu igen!”;forts();} catch(char*s){cerr<<s;exit(-6);}.Ofta vet det kodavsnitt som upptäcker felet inte vad som skall göras – kommunikationsproblem.*
b) Varför övertar *templates* ofta rollen från Arvsbegreppet? Och hur går det till? Ge ett enkelt exempel! (2p)
T är mer flexibelt och ger snabbare exekv. Exempel finns t.ex i form av container-klasser i algoritmbiblioteket (STL) som kan lagra godt. element. Om arv användes vore alla element tvungna att ärva från ”object”, Lagringsbar el.dyl.

c) Vad är namnrymder, *namespaces*, bra för? (1p)
Medger modulär programmering och undviker namnkrockar (att samma namn används för skilda saker i olika programdelar).
d) Vad menas med inkrementell systemutveckling? (1p)
Dela upp programutvecklingen i flera faser(ranker) med gradvis ökad funktionalitet. Ger snabbt ett enkelt, körbart system för utvärdering med kund.
e) Vad är en statisk medlemsvariabel och ge ett exempel på vad man kan ha den till ! (2p)
Gemensam för alla objekt i klassen. Kan t ex användas som räknare av antal objekt eller för klassglobal data som t ex ränta. Initieras globalt.

UPPGIFT 4 – Java (4p)

a) I Java finns det (ännu) inte någon motsvarighet till templates. Vad kan man använda i stället ? (1p)

b) Java-program skall gå att köra på olika datorer och under olika operativsystem utan att behöva kompileras om. Hur åstadkoms detta ? (1p)

c) I C++ anropas destruktorer för att frigöra resurser som t.ex. minne. Motsvarigheten finns i Java men används sällan. Vad händer i ett Java-system när minnet börjar ta slut och du vill skapa ett nytt objekt ? (1p)

d) I Java finns inga pekare, men det går ändå att referera till objekt via en speciell sorts 'variabel'. Vad kallas sådana ? (1p)

UPPGIFT 5 – Programmering (10p)

Du har just blivit chef för Umeås nyinrättade Botaniska Trädgård, och en av dina första uppgifter är att göra ett datorbaserat klassificeringssystem för växterna. Eftersom du aldrig hört talas om Carl von Linne´ bestämmer du dig för att det finns två sorters växter: Smalbladiga och Bredbladiga. Skissa på ett system (gör ett klassdiagram med UML) som uppfyller följande krav-specifikation:

Klassen *Botaniska* skall kunna innehålla upp till 100 Växter, som kan vara antingen Bredbladiga eller Smalbladiga. För varje växt skall det gå att få reda på (via metoder) vilket artnamn den har samt hur hög den just nu är. För bredbladiga växter skall det också gå att se hur breda bladen är.

Implementera dina klasser (du bör ha minst fyra stycken) med alla metoder som behövs och skriv också ett testprogram som skapar 10 växter av slumpmässig typ, med olika namn och andra attribut. Skriv också utskriftsrutiner för varje växttyp som skriver ut information om aktuell växt. Det ska gå att skriva ut alla växter som ingår i Botaniska via virtuella funktioner. Överlagra << -operatorn !

< SLUT PÅ UPPGIFTER >