BOKMÅL

Universitetet i Sørøst-Norge

EKSAMENSFORSIDE

Skoleeksamen med tilsyn

Emnekode: **PB1090**	Emnenavn: Objektorientert programmering	
Dato: 12. mai 2023	Tid fra / til: 09:00	Ant. timer: 4
Ansvarlig faglærer: Lars Erik Opedal (Vestfold) & Joakim Bjørk (Kongsberg)		
Campus: Vestfold og Kongsberg	Fakultet: TNM	
Antall oppgaver: 1	Antall vedlegg: 0	Ant. sider inkl. forside og vedlegg: 4

Tillatte hjelpemidler (jfr. emneplan):

Ingen

Opplysninger om vedlegg:

Merknader:

Studenter på Campus Bakkenteigen skriver Java syntaks

Studenter på Campus Kongsberg skriver C++ syntaks

Tips.

I Wiseflow; bruke vedlegg -> kode og velge Java eller C++ og evt. et farge-tema for syntaks.

Ved eksamen på papir:

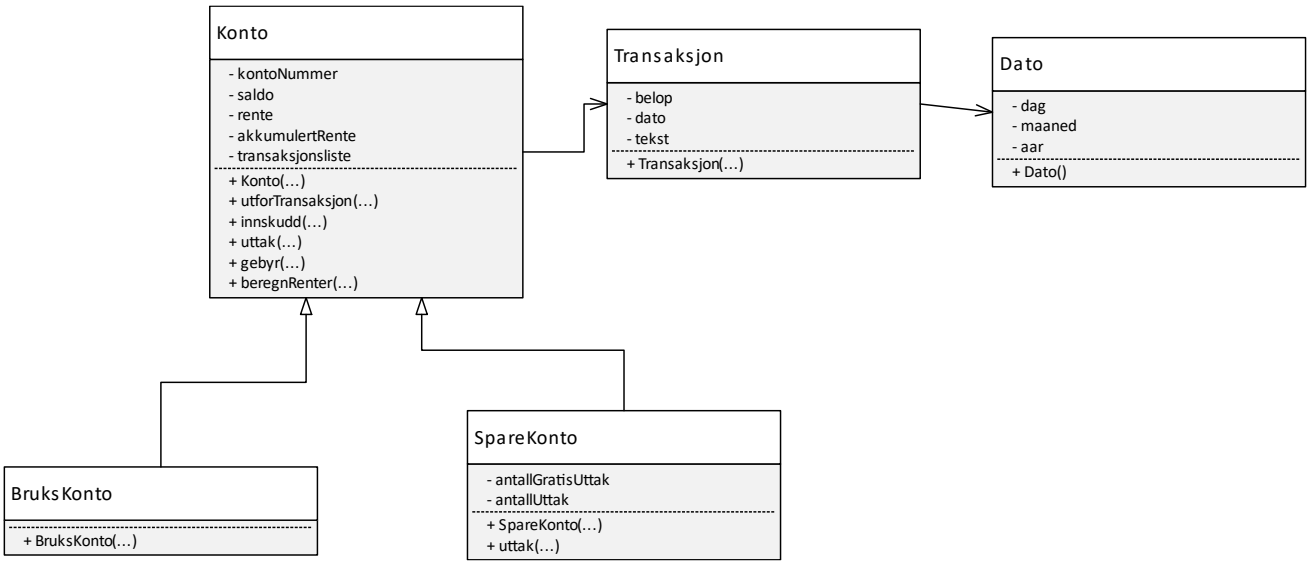
Kryss av for type eksamenspapir

Ruter ☐ Linjer ☐

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG

OPPGAVE 1 – BankKraft konto behandler (100%)

Banken «BankKraft» trenger et nytt bærekraftig kontobehandlingssystem som skal sørge at nye og innovative kontosystemer lett kan implementeres. En systemarkitekt har kommet opp med designet i diagrammet under og det er din jobb å implementere systemet. Funksjoner og variabler som er private er markert med – og det som er public er markert med +.
I funksjoner med … som parameterliste må du finne ut hvilke parametere du trenger. Du må også velge en passende datatype for medlemsvariablene.
Du kan lage funksjoner som ikke står oppført i diagrammet om du trenger det.



a) **(10%)** Implementer klassen Dato med konstruktør. Konstruktøren skal gi alle medlemsvariablene verdier. Konstruktøren skal ikke ta parametere, men hente dato fra systemet. Kode for dette er vist nedenfor.

Java	**C++**

Import java.time.LocalDate;
//----------
LocalDate dato = LocalDate.*now*();
 	
// Lagrer nåværende dato i medlemsvariablene
dag = dato.getDayOfMonth();
maaned = dato.getMonthValue();
aar = dato.getYear();
	#include <ctime>
	//----------
	Tm date: // Lager variabel til lokal tid
	time_t now = time(0); // Hent nåværende
	localtime_s(&date, &now); // Konverter til lokal tid
	
	// Lagrer nåværende dato i medlemsvariablene
	dag = date.tm_mday;
	maaned = date.tm_mon + 1; // add 1 to get correct month
	aar = date.tm_year + 1900; // add 1900 to get correct year

b) **(8%)** Implementer klassen Transaksjon med konstruktør. Konstruktøren skal gi medlemsvariablene verdier og opprette et objekt av klassen Dato.

c) **(7%)** Om du skriver Java:
- Implementer en metode toString i Transaksjon-klassen som returnerer en string som inneholder en forklarende tekst og verdien til alle medlemsvariablene.

Om du skriver C++:
- Overlast << operatoren for Transaksjon-klassen slik at en forklarende tekst og verdien til alle medlemsvariablene blir lagt inn i en stream.

d) **(3%)** Opprett Klassen IkkeDekningException. Denne skal være en utvidelse av exception / Exception.

e) **(3%)** Opprett Klassen IkkeMuligMedNegativtBelopException. Denne skal være en utvidelse av exception / Exception.

f) **(7%)** Implementer klassen konto med en konstruktør. Konstruktøren skal ta kontonummer og rente per år som parametere og disse skal lagres i medlemsvariablene kontoNummer og rente. Medlemsvariablene saldo og akkumulertRente skal settes til 0.0. medlemsvariabelen transaksjonsliste skal kunne inneholde pekere/referanser til alle transaksjoner knyttet til denne kontoen.

g) **(7%)** Implementer medlemsfunksjonen utforTransaksjon. Funksjonen skal ta et beløp og en tekst som parameter. Om beløpet er negativt og større en saldo skal det kastes et IkkeDekningException. Om beløpet er positivt så skal saldo oppdateres og det skal opprettes et transaksjonsobjekt som legges til i transaksjonslisten.

h) **(3%)** Implementer medlemsfunksjonen innskudd. Den skal ta et flyttall som parameter. Om Tallet er negativt skal det kastes et IkkeMuligMedNegativtBelopException. Funksjonen utforTransaksjon skal kalles fra innskudd for å lagre transaksjonen og oppdatere saldo. Stringen som sendes til utforTranskasjon skal være «innskudd».

i) **(3%)** Implementer medlemsfunksjonen uttak. Den skal ta et flyttall som parameter. Om Tallet er negativt skal det kastes et IkkeMuligMedNegativtBelopException. Funksjonen utforTransaksjon skal kalles fra gebyr for å lagre transaksjonen og oppdatere saldo. Stringen som sendes til utforTranskasjon skal være «uttak».

j) **(3%)** Lag medlemsfunksjonen gebyr. Den skal ta et beløp som parameter. Om beløpet er negativt skal det kastes et IkkeMuligMedNegativtBelopException. Funksjonen utforTransaksjon skal kalles fra gebyr for å lagre transaksjonen og oppdatere saldo. Stringen som sendes til utforTranskasjon skal være «gebyr».

k) **(6%)** Implementer klassen BruksKonto med konstruktør. Klassen skal arve fra Konto og renten skal settes til 0.25% per år.

l) **(15%)** Implementer klassen SpareKonto med konstruktør. Klassen skal arve fra Konto og renten skal settes til 1.6%. En sparekonto skal kun ha 5 gratis uttak i løpet av et år. Om man gjør flere uttak må man for hvert uttak betale et gebyr på 1% av uttakssummen, men minimum 250kr. Overlast uttaksfunksjonen for sparekontoklassen.

m) **(5%)** Lag main og opprett en sparekonto og en brukskonto. Sett inn 100’000 på hver av kontoene før du gjør 7 uttak og skriver ut en kontoutskrift.

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG

n) **(10%)** Systemet må beregne renter hver dag ved midnatt. Implementer funksjonen beregnRenter. Renteinntektene skal legges til variabelen akkumulertRente, husk at renter skal beregnes av det som er på saldoen pluss akkumulerte renteinntekter. Dersom måneden har forandret seg fra forrige kjøring skal akkumulert rente utbetales og variablene akkumulert rente skal settes til 0.0. For å finne den daglige renten kan du bruke rente/365.

o) **(10%)** Lag en tråd som tar en liste med kontoobjekter og kaller funksjonen beregnRenter for alle objektene ved midnatt hvert døgn. Hint: du kan bruke koden i oppgave 1a som et utganspunkt for å sjekke hva klokken er.

Lykke til og god sommer!