BOKMÅL

Universitetet i Sørøst-Norge

EKSAMENSFORSIDE

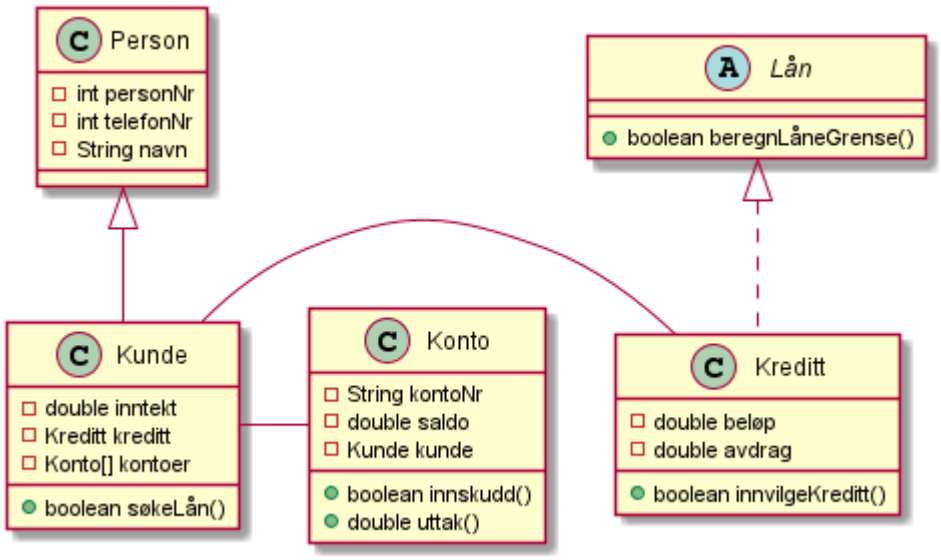
Skoleeksamen med tilsyn

Emnekode: PB1090	Emnenavn: Objektorientert programmering	
Dato: 7.juni, 2021	Tid fra / til: 09:00-13:00	Ant. timer: 4
Emneansvarlig: Lars Erik Opdal, Joakim Bjørk		
Campus: Vestfold og Kongsberg	Fakultet: Fakultet for teknologi, naturvitenskap og maritime fag	
Antall oppgaver: 4	Antall vedlegg: 0	Ant. sider inkl. forside og vedlegg: 5
Tillatte hjelpemidler (jfr. emneplan): Ingen hjelmemidler tillatt.		
Opplysninger om vedlegg:		
Merknader: Studenter ved campus Vestfold skal levere all kode i Java. Studenter ved campus Kongsberg skal levere all kode i C++.		

Ved eksamen på papir: Kryss av for type eksamenspapir Ruter ☐ Linjer ☐

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG

OPPGAVE 1 Klassehierarki – 20%



Du skal implementere klassene i diagrammet ovenfor. Du skal ikke implementere metoder/medlemsfunksjoner, kun skrive signaturen. Alle medlemsvariabler skal være private og alle metoder/medlemsfunksjoner skal være public. Kunde arver fra Person og Kreditt arver fra Lån. Lån er en abstrakt klasse. Du kan bruke pekere/referanser der du mener det passer.

Oppgave 2 Klasser og objekter – 40%

I denne oppgaven skal du skrive to klasser, **Medlem** og **Idrettslag**, som skal brukes av et idrettslag til å holde orden på hvilke medlemmer som har betalt kontingent.

I klassen Medlem skal medlemsnummer, navn og telefonnr lagres som string-verdier. Videre skal det lagres informasjon om hvorvidt kontingenten er betalt. Klassen skal inneholde en konstruktør med parametre string medlemsnr, string navn og string telefonNr, og som initialiserer de respektive verdiene i objektet.

Du kan lage egne hjelpefunksjoner om du trenger det.

I klassen Idrettslag skal det være datastruktur for å lagre Medlem-objekter her kan du bruke referanser/pekere om du vil. Klassen skal i tillegg inneholde følgende metoder:

- En konstruktør som initialiserer datastrukturen.

- **void** leggTilMedlem(Medlem medlem)
  Skal legge til Medlem *medlem* i datastrukturen.

- **Medlem** finnMedlem(string medlemsnr)
  Skal returnere medlemmet med medlemsnummer medlemsnr. Hvis medlemmet ikke finnes, skal det returneres null

- **void** slettMedlem(string medlemsnr)
  Skal slette medlemmet med medlemsnummer medlemsnr. Hvis medlemmet ikke finnes, skal det ikke gjøres noen ting.

- **void** registrerBetaling(String medlemsnr)
  Skal registrere at medlemmet med medlemsnr har betalt kontigenten. Hvis medlemmet ikke finnes, skal en passende feilmelding skrives ut på skjermen fra metoden.

- **void** skrivPurreliste()
  Skal skrive ut en liste på skjermen med medlemsnr, navn og telefonnr til de medlemmene som *ikke* har betalt kontigenten.

- **void** harLikeTlfnr()
  Skal sjekke om noen medlemmer har like telefonnr. Hvis ingen medlemmer har like telefonnr, skal en passende melding skrives ut på skjermen. Hvis det finnes medlemmer med samme telefonnr, skal det skrives ut en linje med telefonnr og deretter medlemsnr og navn på de medlemmene som har dette telefonnr. Det skal skrives ut én slik linje for hvert telefonnr som deles av flere medlemmer.

NB.
Merk at du kun skal skrive kode for klassene og metodene som er nevnt i teksten over. Du skal *ikke* skrive main.

Oppgave 3 Sortering av data – 20%

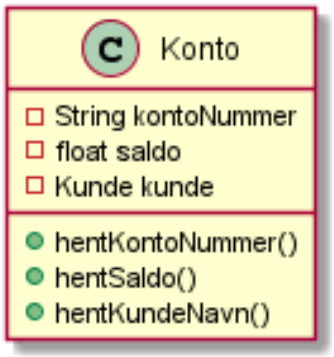
Vi skal nå bruke Idrettslag-klassen i Oppgave 2. Vi kan anta at det er maksimalt 2000 medlemmer som ligger i medlemslisten. Vi kan anta at alle medlemsnumrene ligger i intervallet 1000,...,2999 og at alle medlemmer har forskjellig medlemsnummer.

Du skal nå skrive en metode/medlemsfunksjon i Idrettslag-klassen; sorterMedlemmer som returnerer en liste over alle Medlems-objekter fra medlemslisten. Du kan selv velge om du vil bruke pekere/referanser i denne oppgaven. Listen du skal returnere skal være sortert etter stigende medlemsnnummer, dvs. fra 1000 til 2999. Du skal bruke følgende algoritme for sorterMedlemmer:

- Opprett en array som heter temp med plass til 2000 Medlems-objekter.
- Gå gjennom arrayen med Medlems-objekter i medlemslisten og legg alle medlemmene inn i temp arrayet. En person med medlemsnummer 1385 skal ligge på indeks 385 i arrayet osv.
- Mens du gjør dette, hold orden på hvor mange objekter som ligger i temp arrayet. La en **int**- variabel antall lagre dette antallet.
- Gå gjennom temp fra indeks 0 og utover. En del av plassene i temp vil være null, men en del vil også referere til medlemsobjekter. Flytt medlemsobjektene fra temp over til en ny array for medlemsobjekter, med lengde lik antall som er telt opp. Arrayen du skriver over til, fra temp vil være korrekt sortert fra 0 til 1999.

Oppgave 4 Tråder - 20%

Denne gangen skal vi bruke en *oppdatert* versjon av Konto-klassen fra UML diagrammet i oppgave 1. Vi har nå følgende Konto-klasse.



Vi har fått en ny metode/funksjon tilgjengelig i main: *hentAlleKontoerFraDatabasen();*

Vi har nå 10000 konto-objekter med *saldo*. Du skal **ikke** lage metoden over. Lag et program som bruker tråder til å finne den høyeste *saldoen* blant alle kunder i den *hypotetiske* databasen.

- Først må du finne ut hvor mange kjerner din CPU har. Dersom du har *2*, skal du lage *2* tråder som tar hver sin del av konto-arrayet; 10000 / 2 = 5000 pr.stk. Har du 4 kjerner, blir det 10000 / 4 = 2500, osv.

  (Dvs. løsning skal fungere for antall kjerner som er tilgjengelig på den PC som kjører programmet)

  Du kan bruke metoden/funksjonen *int availableProcessors()* får å tilgang til antall kjerner. Du skal maksimalt starte 16 tråder.

- Start med å skrive en trådklasse/ arbeidsklasse (Java) eller en funksjon (C++) som tar inn et array med konto-objekter, i tillegg start og stop verdi for indeks til arrayet, og antall elementer som skal behandles. Du skal også ha en variabel for å lagre indeksen til kunden med høyest saldo. Den skal vi senere bruke til å finne maksimal verdi for alle kundene.

- Skriv kode i main som oppretter tråder. Start trådene og la alle bli ferdig. Lag et array for å samle inn resultatene fra alle trådene. Til slutt finner du den globalt maksimale verdien for *saldo*, fra alle resultater som trådene har behandlet.

- Når resultatet er funnet skal du skrive ut navnet på kunden og beløpet. Da brukes *hentSaldo* og *hentKundeNavn-metoden* fra konto-objektet. Dette kan gjøres direkte i main.

Lykke til og god sommer