UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Ny og utsatt eksamen i: IN1010 – Objektorientert programmering
Eksamenssemester: Vår 2020
Tid for eksamen: Torsdag 13. august kl 9:00 til torsdag 20. juni kl 9:00
Oppgavesettet er på 5 sider

Kabal

I dette oppgavesettet skal du utforme og skrive ditt eget program. Mange valg overlates til deg som programmerer. Det er imidlertid ikke nok at programmene fungerer, det vil bli lagt vekt på at du programmerer objektorientert slik du har lært i IN1010. Det er også noen krav til grensesnitt, tråder, klasser og subklasser som skal være med.

Det er viktig at alle programmene dine kan kompileres og kjøres. Det er selvfølgelig best om du klarer å lage programmer som virker perfekt, men om du ikke klarer det, er det allikevel viktig at du lager programmer som kompilerer og kjører, selv om det ikke løser oppgaven helt på riktig måte. Om du har kode som bare nesten virker, kommenter den ut og skriv en kommentar med en begrunnelse for hvorfor du har denne koden der og hvilke problemer du opplever som gjør at koden ikke virker. Programmer som leveres men som ikke kompilerer og kjører, vil ikke gi særlig uttelling.

Husk at det ikke er lov å dele kode og det er ikke lov å bruke kode du ikke har skrevet selv. Du kan importere fra Java standard API med unntak av klasser i Javas Collections rammeverk - du skal altså ikke importere og bruke ArrayList, LinkedList eller andre implementasjoner av List interfacet. Du må gjerne bruke dine egne løsninger fra obligatoriske oppgaver om de er relevante - for eksempel Lenkeliste klasser. Ikke bruk Æ, Ø eller Å (små og store) noe sted i programmene dine.

Alle oppgavene er gitt en vekt i poeng som er den maksimale uttellingen du kan få på denne oppgaven. Du må til sammen ha minst 40 poeng for å få karakteren bestått.

I de påfølgende oppgavene skal du programmere et objektorientert system for å legge kabal (engelsk *patience* eller *solitaire*) med spillkort (vanlig / fransk kortstokk med 52 kort). Systemet består av noen grunnleggende klasser for håndtering av kort, samt klasser som støtter legging av en konkret kabal. Flere av klassene kan gjenbrukes ved implementering av andre kabaler (og kortspill med flere deltakere).

Vi skal konsentrere oss om kabalen Idioten/ Idiot i denne oppgaven. I denne kabalen er målet å sitte igjen med 4 kort på bordet: Det kortet som har høyest verdi i hver farge (fargene er spar, hjerter,

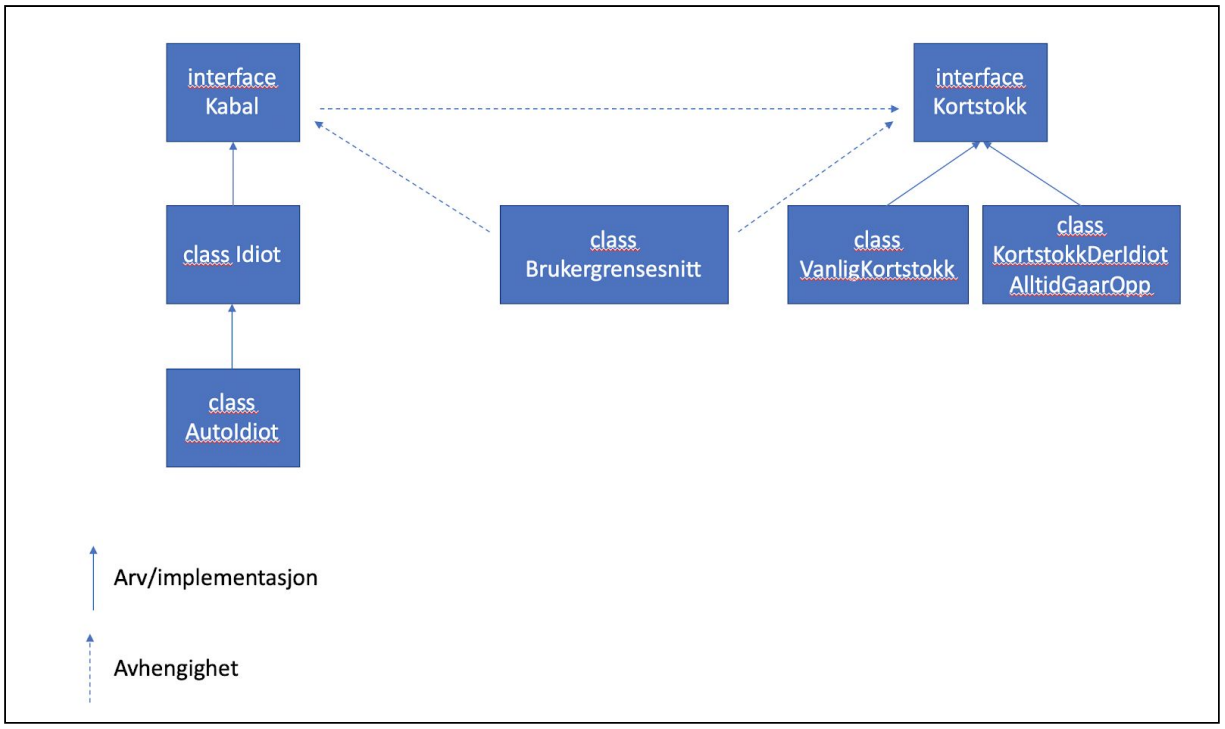
ruter, kløver). Siden essene i vårt system har verdien 1, er kortene med høyest verdi de fire kongene, og det er disse som skal ligge igjen for at kabalen "går opp".

Man starter med å plassere 4 åpne (så man ser farge og valør) kort fra kortstokken ved siden av hverandre. Dersom to eller flere av kortene har samme farge, fjernes (kastes) kortene med lavest verdi slik at det ikke finnes flere kort med samme farge.

Deretter legges 4 nye åpne kort på samme plasser - om det ligger et kort på en plass fra før legges det nye kortet delvis oppå (dette kalles en bunke). Igjen sammenligner du kortene (kun de øverste i hver bunke) og kaster ett og ett inntil øverste kort i hver bunke har forskjellig farge. Et kort kan flyttes fra øverst i én bunke til en tom plass (bunke) - da kommer et nytt kort øverst i bunken du flyttet fra, og det kan kanskje kastes flere kort. Når ingen flere kort kan kastes, og det ikke er mulig å flytte fra en bunke til en ledig plass (tom bunke), legger man ut 4 nye kort på de samme bunkene - inntil hele kortstokken er brukt og kabalen er ferdig lagt.

Dersom det da *kun* ligger 4 konger igjen på bordet til slutt har kabalen gått opp (det skal ikke ligge andre kort igjen under kongene). Denne kabalen går bare opp 1-2% av gangene den legges - noe som kanskje er årsaken til navnet Idioten.

Nedenfor ser du en forenklet figur som viser noen viktige sammenhenger mellom klasser og interface i systemet du skal skrive. Du skal følge dette designet i ditt program. Merk at ikke alle klassene i systemet er tatt med i denne figuren.
«Avhengighet» som illustrert ved de prikkete linjene angir at dersom en klasse A er avhengig av (peker på med prikkete linje) en klasse eller interface B, kan endringer i klassen/ interfacet B kreve endringer i klassen A.
De heltrukne linjene angir at en klasse A som peker på en klasse B er en subklasse av B. Dersom klassen A peker på et interface B er A en implementasjon av B.



Oppgave 1. Klassen Kort (5 poeng)

Klassen Kort representerer et spillkort med farge (spar, hjerter, ruter eller kløver) og valør (verdi 2-10, knekt (verdi 11), dame (verdi 12), konge (verdi 13) eller ess (her med verdi 1). Et kort kan være åpent (farge og valør vises) eller lukket (kun baksiden som er lik for alle kort i en kortstokk vises). Klassen skal implementere interface Comparable slik at to kort kan sammenlignes basert på verdi dersom de er av samme farge. Dersom de har ulik farge skal de ansees å ha samme verdi (uavhengig av valør på de to kortene). Klassen skal videre ha en toString metode som returnerer en tekstlig representasjon av kortet (i denne oppgaven kortets farge og valør).

a) Implementer klassen Kort inkludert konstruktør og metoder for å lese av farge og valør.

Svaret avleveres sammen med svaret for oppgave 2.

Oppgave 2. Interface Kortstokk med implementasjon (15 poeng)

a) Skriv et interface Kortstokk med metoder for å blande kortene i kortstokken ("stokke" - det vil si endre rekkefølgen på kortene i kortstokken slik at den blir tilfeldig), hente ut et tilfeldig kort, hente ut det øverste kortet i kortstokken og for å sjekke om kortstokken er tom.
b) Skriv en klasse StandardKortstokk som implementerer interface Kortstokk. Konstruktøren skal opprette en kortstokk med 52 standard spillkort representert i en lenkeliste, og klassen skal ha en toString-metode.
c) Skriv en klasse KortstokkDerIdiotAlltidGaarOpp som implementerer interface Kortstokk for en kortstokk med færre kort - der kortene kun har valører ess, 2, 3, konge (verdier 1-3 og konge) i alle fire farger. Metoden som blander kortene skal i denne klassen returnere uten å endre rekkefølgen på kortene i kortstokken.
d) Skriv et testprogram, Test2, som viser et eksempel på at dine klasser virker.

Svar på oppgave 1 og 2 leveres samlet som en .zip-fil med en java-fil for hver av klassene Kort, StandardKortstokk, KortstokkDerIdiotAlltidGaarOpp og Test2 samt interface Kortstokk og andre klasser/interface som kreves for at testprogrammet i 2d) skal kunne kjøres. I .zip-filen skal det også ligge et snapshot/ bilde av test-kjøring av Test2 fra 2d). Last opp zip-filen som svar i Inspera.

Oppgave 3. Klassen Bunke (10 poeng)

Når man spiller kort eller legger kabal arbeider man vanligvis med en eller flere bunker av kort. I dette systemet kan man kun legge til og fjerne kort på toppen av bunken.
Skriv en klasse Bunke som kan holde rede på et vilkårlig antall kort. Klassen skal ha metoder for å legge på et nytt kort (øverst), ta av et kort (øverst), se på det øverste kortet og sjekke om bunken er tom. Svaret avleveres sammen med svaret for oppgave 4.

Oppgave 4. Rekursjon med implementasjon (10 poeng)

a) Skriv en ny metode, skrivut, i klassen Bunke som skriver ut bunken. Metoden skal anvende rekursjon: et rekursivt kall for hvert kort i bunken.
b) Skriv et testprogram, Test4, som viser et eksempel på at din klasse Bunke, og spesielt skrivut, fungerer.

Svar på oppgave 3 og 4 leveres samlet som en .zip-fil med en fil for hver av klassene inklusive Test4 og andre klasser/interface som kreves for at testprogrammet i 4b) skal kunne kjøres, samt bilde av din test-kjøring i 4b). Last opp zip-filen som svar i Inspera.

Oppgave 5. Testprogram for å legge Idioten (15 poeng)

Kabalene som skal implementeres av dette systemet kan alle legges ved hjelp av et begrenset sett operasjoner. Disse er definert av et interface Kabal.

a) Design og skriv interfacet Kabal.
b) Skriv en klasse UferdigIdiot som implementerer interface Kabal, men med metoder som kun skriver ut på terminal at de kalles og returnerer en verdi av riktig type.
c) Skriv en klasse BrukerstyrtIdiot med en main-metode som lar en bruker legge Idiot-kabalen. Brukeren skal kunne gi kommandoer for å stokke en kortstokk, starte en ny runde (legge ut 4 nye kort), kaste et kort (når det finnes et kort med større verdi i samme farge øverst i en annen bunke), og flytte et kort til en tom bunke. Programmet skal sjekke at alle operasjoner er i henhold til reglene for Idioten som beskrevet tidligere, samt håndter passende Exceptions ved behov. For hver flytting av et kort skal brukeren få beskjed om hvilke kort som ligger øverst i hver bunke. Når kabalen er ferdig lagt skal programmet gi beskjed om kabalen gikk opp eller ikke.

Svar på oppgave 5 leveres samlet som en .zip-fil med en fil for hver av klassene inklusive Test4 samt andre klasser/interface som kreves for at testprogrammet i 5c) skal kunne kjøres. Last opp zip-filen som svar i Inspera.

Oppgave 6. Ferdig program for å legge Idioten (15 poeng)

a) Erstatt klassen UferdigIdiot fra forrige oppgave med en ny klasse Idiot som implementerer alle metodene som spesifisert i interface Kabal, samt eventuelle hjelpemetoder og nødvendig datastruktur basert på tidligere implementerte klasser.
b) Lag en test-kjøring av programmet og inkluder utskrift av kjøringen i din besvarelse.

Svar på oppgave 6 leveres samlet som en .zip-fil med en fil for hver av klassene inklusive Idiot samt andre klasser/interface som kreves for at testprogrammet i 6b) skal kunne kjøres. Last opp zip-filen som svar i Inspera.

Oppgave 7. Tråd med implementasjon (10 poeng)

a) Utvid programmet: Lag en tråd som skriver ut tidsforbruket hvert sekund.
Bruk gjerne: TimeUnit.SECONDS.sleep(1);
Brukeren skal kunne gi en kommando, pause, som stopper klokken – og en kommando fortsett som starter klokken igjen (teller videre).
b) Lag en test-kjøring av programmet - som inkluderer både pause og fortsett.

Svar på oppgave 7 leveres samlet som en .zip-fil med en fil for hver av klassene inklusive Idiot samt andre klasser/interface som kreves for at testprogrammet i 7b) skal kunne kjøres. Last opp zip-filen som svar i Inspera.

Oppgave 8. Grafisk brukergrensesnitt med implementasjon (20 poeng)

Klassen Idiot kan utvides for eksempel med en subklasse, Autoidiot, som selv legger kabalen. I denne oppgave skal dere utvide klassen Idiot: enten ved at lage en subklasse eller ved at omskrive den.
a) Utvid klassen Idiot slik at kabalen underveis skrives ut i et grafisk brukergrensesnitt, et GUI-vindu, programmert med JavaFX. Brukeren skal styre kabalen via knapper på skjermen. Husk knapper til «pause» og «fortsett».

b) Kjør programmet og ta bilder av skjermen. Legg bildene og løsningen på oppgave 8 sammen med de klassene du trenger fra oppgavene 1-8 i en mappe slik at alt som trengs for å kjøre programmet ligger i denne mappen. Komprimer mappen til en zip-fil. Last opp denne zip-filen som svar i Inspera.

Lykke til!

Eyvind Axelsen, Eric Jul, Stein Gjessing og Siri Moe Jensen