Plenum: Beholdere og generiske klasser 2

Ahmed Waseem Saeed

Mars 2024

Introduksjon

Velkommen til ny plenumstime! Denne uken skal vi fortsette på forrige ukes tema, nemlig beholdere og generiske klasser. Vi skal videreutvikle det vi begynte på, der vi skal legge til mulighet for prioritering. I tillegg skal vi gjøre AListe-klassen vår *itererbar*, det vil si at vi blant annet kan benytte en for-each løkke for å gå gjennom datastrukturen vår. I tillegg må vi implementere interfacet Comparable ettersom vi skal prioritere objektene i lista vår. For å friske opp forrige ukes plenumstime, så kan du ta en titt på ressursene på plenumsiden.

Iterable

Modifiser AListe-klassen slik at den er itererbar. Dette innebærer å implementere Iterable-interfacet. Merk at det er nøklene (altså typeparameteren K) som skal itereres over. Ta en titt på forelesningsressursene eller dokumentasjonen til iterable og iterator for å se hvilke metoder som må implementeres.

Test

Modifiser testprogrammet (eller opprett et nytt) der du benytter en for-each løkke til å iterere nøklene i lista og skrive ut innholdet til den tilhørende verdien til hver enkelt nøkkel.

hentPaaIndeks

Utvid AListe.java med metoden hentPaaIndeks, som skal ta inn en indeks (int) og hente verdien på gitt indeks og returnere denne. Dersom listen er tom eller indeksen ikke er innenfor størrelsen til lista skal null returneres.

Sorterbar liste

Vi ønsker nå å implementere funksjonalitet som lar oss sette inn elementer i en sortert rekkefølge, f.eks. etter prioritet. For å få til dette må vi først sørge for

at typeparameteren K implementerer interfacet Comparable. Skriv derfor en ny klasse AListeSortert som arver fra AListe, men også spesifiserer at typeparameteren K må implementere interfacet Comparable (av typen K).

Deretter skal du overskrive metoden leggTil, slik at den setter inn elementet riktig i listen (etter prioritet). Vi definerer "riktig" i vår sammenheng ved at noder med **høyest** prioritet skal fremst i listen, altså jo høyere prioritet, desto lenger frem skal noden være i lista (frem tilsvarer en tidlig index). Merk at du må endre fra private til protected i AListe for Node-klassen og andre instansvariabler.

Som forrige time, anbefaler vi at du tegner mye underveis og kartlegger ulike scenarier for innsettingen.

Teste mer

På dette stadiet kan du benytte filen TestAListe.java for å se om testene fremdeles passerer. Gå tilbake og modifiser koden dersom noen av testene ikke består.

Passasjer

For å teste ut AListeSortert litt grundigere så ønsker vi å opprette en klasse Passasjer. Klassen skal implementere Comparable-interfacet. Videre har klassen tre instansvariabler som tas inn i konstruktøren: navn (String), alder (int) og en prioritet (int).

Metoden compareTo() skal gjøre følgende for å avgjøre prioriteten til en passasjer: først skal prioriteten sjekkes (den med høyere prioritet skal selvfølgelig bli prioritert først), hvis denne er lik mellom to passasjerer, så skal ansiennitet bli en avgjørende faktor, altså skal den eldste personen bli prioritert. Hvis dette også er likt så skal metoden returnere 0.

Overskriv også equals() metoden, slik at du kan sjekke om to passasjerer er like. Dette skal du gjøre ved å benytte compareTo()-metoden, samt sjekke at navnet er likt mellom to passasjerer.

Til slutt skal du overskrive toString-metoden slik at den skriver ut innholdet om en passasjer (navn, alder, prioritet).

Utfordring: Fremfor å bruke et heltall som prioritet, kunne vi brukt en enum som spesifiserer hvilken (boarding)gruppe en passasjerer tilhører, A, B, eller C. Implementer denne enum-en i en egen fil, og endre compareTo og konstruktøren til å benytte denne enum-en.

Utvid testprogrammet

Opprett et AListeSortert-objekt og sett inn 3-4 Passasjer-objekter (Passasjer som nøkkel K, og String (f.eks. navnet) som verdi V. Gå gjennom lista med en for-each løkke og verifiser at utskriften blir riktig.