Programmering 2: övningsuppgift 2

Version: 1.3
Uppdaterad: 2021-04-09, se dokumenthistoriken sist i dokumentet för info.
Författare: Patrick Wentzel

Detta är den andra övningsuppgiften på kursen PROG2 VT2021.

Uppgiften är giltig under kursomgången fram till uppsamlingstillfället i augusti 2021 och upphör därefter att gälla och kan komma att ersättas kursomgången 2022.

Innehållsförteckning

Inledning

Efter uppgraderingarna i butikens IT-stöd, med nya klasser och sånt går affärerna bättre än någonsin, men en trist konsekvens är att en del saker har börjat fungera allt sämre i takt med att kunderna har blivit fler.

Butikens enda sätt att hitta saker är att gå igenom en lång osorterad lista med information om inspelningar och varje gång det kommer in en kund med specifika önskemål tar det en evighet att hitta rätt saker. När man har sorterat listan på år kommer det någon och frågar efter att skivor av en viss artist, och när man har sorterat listan på artister vill någon veta vilka skivor som finns i en viss genre…

Exempel på saker som kunderna vill veta är:

- Finns det några skivor av Bruce Springsteen?

- Vilka genrer finns det skivor inom?

- Finns albumet Thriller?

- Vad finns det som är nyare än 2010?

- Vilka skivor finns det av Miles Davis? Gärna sorterade på ålder

- Vilka skivor finns det i genren rock?

- Finns det några jazzskivor mellan 1950 och 1960?

Ibland kommer det också kunder och frågar om

- Vilka skivor som finns från 1983?

- Finns det några skivor från innan 1960?

- Vilka skivor finns med John Coltrane? Gärna sorterade på titel

Det här fungerar inte, och därför vill affärens ägare Johnny Rocker att du ska fixa till systemet och göra så att det går fort att hitta saker. Till din hjälp finns det en början till en klass och dessutom ett par gränssnitt med de operationer klassen ska stödja.

En del saker måste göras, men om du har tid och lust så finns det några extra önskemål i slutet.

Och, till sist en sak till…

Ibland händer det att någon kommer in med en samling skivor och undrar om vi vill köpa den - det behövs ett sätt att snabbt kolla vad i samlingen som vi inte redan har - och det spelar ingen roll om det är CD eller LP, vi är bara intresserade av skivor som vi inte har alls (vare sig som CD eller LP).

Uppgiften

Uppgiften är alltså att anpassa klassen `Recording` så att den fungerar som nyckel i en avbildningstabell eller mängd och implementera de två gränssnitten `SearchOperations` och `Stats` i klassen `Searcher` och lägga till lämpliga datasamlingar som medlemsvariabler och fylla de med data så att det olika metoderna fungerar effektivt.

I iLearn finns följande filer att utgå ifrån:

- Database.java - ett register av inspelningar
- Exercise2.java - ett enkelt testprogram
- Recording.java - en representation av inspelningar
- SearchOperations.java - gränssnittet med sökmetoderna
- Stats.java - gränssnittet med statistikmetoder
- Searcher.java - klassen som används för sökningar

Som en hjälp finns det i iLearn ett enkelt testprogram (`Exercise2.java`) som visar hur klassen Searcher är tänkt att användas, samt några enkla tester.

Metodnamn kan också ge vägledning om vad för datastruktur som kan passa för just den metoden.

Krav & restriktioner

Eftersom syftet med uppgiften i första hand är att öva på användning av olika typer av avbildningar och mängder ska uppgiften lösas helt utan användning av någon subklass till `List` (t.ex. `ArrayList` och `LinkedList`).

Utöver koden som läser in data till de olika datasamlingarna bör uppgiften i princip kunna lösas utan linjära sökningar[1] och lösningar som använder datastrukturer fel kommer sannolikt inte klara testerna och kan komma att underkännas.

En godkänd lösning kommer helt säkert kräva minst två avbildningar som medlemsvariabler.

Samlingar som returneras ska inte kunna modifieras (byta innehåll). Klassen `Collections` har flera användbara hjälpmetoder för det.

Inga metoder ska returnera null, förutom default-metoderna i `SearchOperations`.

Metoder som returnerar samlingar ska returnera tomma samlingar om det inte finns något av det som efterfrågas.

---

[1] Linjära sökningar går igenom en samling från start till slut steg för steg när något ska lokaliseras.

Inlämningsdatum (deadline)

Rekommenderat inlämningsdatum är den 7/4. Uppgiften behöver vara avklarad innan slutuppgiften med deadline 5/6 kan lämnas in.

Nyckelord[2]

HashMap, HashSet, Collection, TreeMap, TreeSet, Comparator, hashCode, equals, headMap, tailMap, subMap, SortedSet, Optional,
Collections.unmodifiableXXX (Collection , SortedSet , List, etc.),
Collections.emptyXXX (List , Set , SortedSet, etc.)

[2] Nyckelorden är de delar av Java som förmodligen behöver finnas med i en godkänd lösning, eller som kan vara till hjälp.

Klasserna

Database

Klassen `Database` är ett litet register av inspelningar (Recording).

Klassen är given på förhand och **ska inte förändras** på något vis.

Klassen har en metoder:

- `getRecordings` ger tillbaka en oföränderlig `Collection<Recording>`.

Recording

Klassen `Recording` representerar en inspelning (antingen CD eller LP).

Klassen ska kunna fungera som nyckel i en avbildingstabell eller mängd.

Likhet (equality) ska vara definierad som att två olika objekt av klassen betraktas som lika om de har samma titel, artist och år. Typ eller genre ska inte spela någon roll.

Objekt av typen `Recording` har ingen naturlig sorteringsordning, men det är ok att definiera en sorteringsordning om det krävs för att lösa någon del av uppgiften (även om andra lösningar förmodligen är bättre).

Searcher

Klassen `Searcher` är den klass som ska modifieras och som ska implementera de två gränssnitten `SearchOperations` och `Stats`.

Klassen ska ha de publika metoder som krävs av gränssnitten, och inga andra publika metoder. Klassen kan ha privata metoder och medlemsvariabler.

När klassen instansieras ska data läsas in till lämpliga datastrukturer så att metoderna som gränssnitten kräver fungerar som tänkt.

Klassens konstruktor tar emot en `Collection<Recording>` som för testningen kan vara samlingen som kommer från `Database.getRecordings`, se exempelkoden i `Exercise2.java` för hur den är tänkt att användas.

Gränssnitten

Stats

Gränssnittet `Stats` innehåller följande metoder:

- **`numberOfArtists`**

  Metoden ger antalet unika artister.

- **`numberOfGenres`**

  Metoden ger antalet unika genrer.

- **`numberOfTitles`**

  Metoden ger antalet unika titlar.

SearchOperations

Gränssnittet innehåller 11 metoder, varav åtta saknar default-implementation och alltså måste implementeras. Tre metoder har default-implementationer och behöver inte implementeras men kan göras som extra övning, de testas bara om de inte returnerar `null`. Metoder som ska returnera samlingar behöver inte ge resultatet sorterat om det inte anges explicit i uppgiftsbeskrivningen eller i JavaDoc-kommentar i gränssnitten.

Obligatoriska metoder

- **`doesArtistExist`**

  Metoden tar in ett namn och returnerar true om artisten finns, false om inte.

- **`getGenres`**

  Metoden ger en omodifierbar samling med genrer.

- **`getRecordingByName`**

  Metoden ger ett objekt av typen `Optional<Recording>` som innehåller inspelningen med den sökta titeln om den finns, eller är tomt annars.

- **`getRecordingsAfter`**

  Metoden ger en omodifierbar samling med inspelningar från och med det angivna året.

- **`getRecordingsByArtistOrderedByYearAsc`**

  Metoden tar in ett namn på en artist och returnerar en omodifierbar samling med inspelningar sorterade på år i stigande ordning.

- **`getRecordingsByGenre`**

  Metoden tar in namnet på en genre och ger en omodifierbar samling med inspelningar i genren.

- **`getRecordingsByGenreAndYear`**

  Metoden tar in ett namn på en genre och två årtal och ger en omodifierbar samling med inspelningar i genren gjorda mellan de angivna åren.

- **`offerHasNewRecordings`**

  Metoden tar emot en samling med inspelningar och ger tillbaka en omodifierbar samling med de objekt som inte redan fanns i databasen.

Frivilliga utökningar

- **`optionalGetRecordingsBefore`**

  Metoden ger en omodifierbar samling med inspelningar gjorda innan det angivna året.

  Metoden har en default-implementation och testas bara ifall den överskuggas och inte returnerar `null`.

- **`optionalGetRecordingsByArtistOrderedByTitleAsc`**

  Metoden tar in ett namn på en artist och ger en omodifierbar samling med inspelningar av den sökta artisten.

  Samlingen ska vara sorterad på titel i stigande ordning A -> Z

  Metoden har en default-implementation och testas bara ifall den överskuggas och inte returnerar `null`.

- **`optionalGetRecordingsFrom`**

  Metoden tar in ett år och ger en omodifierbar samling med inspelningar från det angivna året.

  Metoden har en default-implementation och testas bara ifall den överskuggas och inte returnerar `null`.

Instruktion för inlämning

För att kunna lämna in behöver man ha valt grupp för övningsuppgift 2.

I VPL för övningsuppgiften laddas de två klasserna `Recording` och `Searcher` upp i separata filer. Klassen `Database` och gränssnitten `SearchOperations` och `Stats` finns redan i VPL och ska inte laddas upp. Testprogrammet `Exercise2.java` ska inte lämnas in.

Koden testas automatiskt med hjälp och kommer att bli godkänd med automatik när alla tester uppfylls.

Överst i samtliga inlämnade filer ska det finnas en kommentar med namn och användarnamn för samtliga gruppmedlemmar. Exempelvis:

```
// PROG2 VT2021, Övningsuppgift 2
// Grupp 004
// Dewey Duck dedu1111
// Huey Duck hudu1234
```

Dokumenthistorik

Version	Datum	Ändringar
1.0	2021-03-27	Första version.
1.1	2021-03-28	Rättat några stavfel.
1.2	2021-03-30	Flera förtydliganden, samt ändrat information om Database och Searcher.
1.3	2021-04-09	Förtydligade krav på användning av datastrukturer.