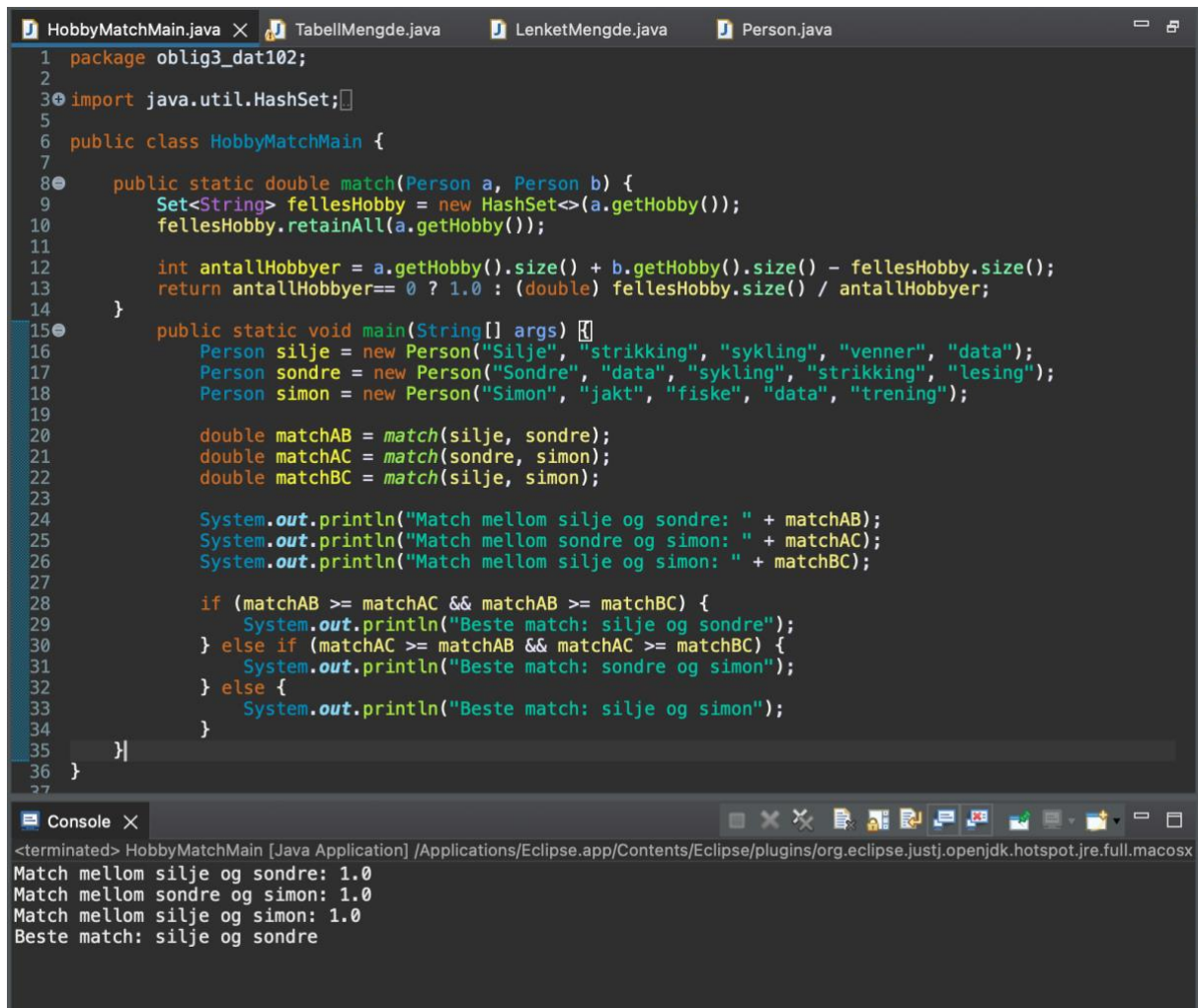


# Obligatorisk innlevering 3, dat102

Julie Elea Fjellstad  
Marius Larsen Arnevik

## Skjermbilder av kjøring av programmene



The screenshot displays the Eclipse IDE with four open Java files: HobbyMatchMain.java, TabellMengde.java, LenketMengde.java, and Person.java. The main focus is on HobbyMatchMain.java, which contains the following code:

```
1 package oblig3_dat102;
2
3 import java.util.HashSet;
4
5
6 public class HobbyMatchMain {
7
8     public static double match(Person a, Person b) {
9         Set<String> fellesHobby = new HashSet<>(a.getHobby());
10        fellesHobby.retainAll(a.getHobby());
11
12        int antallHobbyer = a.getHobby().size() + b.getHobby().size() - fellesHobby.size();
13        return antallHobbyer == 0 ? 1.0 : (double) fellesHobby.size() / antallHobbyer;
14    }
15
16    public static void main(String[] args) {
17        Person silje = new Person("Silje", "strikking", "sykling", "venner", "data");
18        Person sondre = new Person("Sondre", "data", "sykling", "strikking", "lesing");
19        Person simon = new Person("Simon", "jakt", "fiske", "data", "trening");
20
21        double matchAB = match(silje, sondre);
22        double matchAC = match(sondre, simon);
23        double matchBC = match(silje, simon);
24
25        System.out.println("Match mellom silje og sondre: " + matchAB);
26        System.out.println("Match mellom sondre og simon: " + matchAC);
27        System.out.println("Match mellom silje og simon: " + matchBC);
28
29        if (matchAB >= matchAC && matchAB >= matchBC) {
30            System.out.println("Beste match: silje og sondre");
31        } else if (matchAC >= matchAB && matchAC >= matchBC) {
32            System.out.println("Beste match: sondre og simon");
33        } else {
34            System.out.println("Beste match: silje og simon");
35        }
36    }
37 }
```

The console output at the bottom shows the results of the program execution:

```
<terminated> HobbyMatchMain [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx
Match mellom silje og sondre: 1.0
Match mellom sondre og simon: 1.0
Match mellom silje og simon: 1.0
Beste match: silje og sondre
```

```
JavaSetToMengde.java  Oppgave_4.java X
1 package oblig3_dat102;
2
3 import java.util.Arrays;
4
5
6
7 public class Oppgave_4 {
8
9     public static void main(String[] args) {
10
11         final int antallElement = 100000;
12         final int maksElement = 1000000;
13         final int antallSok = 10000;
14
15         HashSet<Integer> hashSet = new HashSet<>();
16         int[] tabell = new int[antallElement];
17
18         int tall = 376;
19         for(int i = 0; i < antallElement; i++) {
20             hashSet.add(tall);
21             tabell[i] = tall;
22             tall = (tall + 45713) % maksElement;
23         }
24
25         Arrays.sort(tabell);
26
27         Random tilfeldig = new Random();
28         int[] soketall = new int[antallSok];
29         for(int i = 0; i < antallSok; i++) {
30             soketall[i] = tilfeldig.nextInt(maksElement);
31         }
32         long startHashSet = System.nanoTime();
33         int funnetIHashSet = 0;
34         for (int tallSok : soketall) {
35             if(hashSet.contains(tallSok)) { // sjekker om tallet finnes
36                 funnetIHashSet++;
37             }
38         }
39     }
40 }
```

```
<terminated> Oppgave_4 (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.ju
HashSet treff : 1016
HashSet tid: 4.305326ms
Binærsøk treff : 1016
Binær tid: 4.712411ms
```

## Oppgave 4.d.

- i. Boolean inneholder(T element)
  - LenketMengde: itererer gjennom hele listen til elementet er funnet eller evt kommer til slutten, best case er  $O(1)$  dersom elementet er først, og  $O(n)$  dersom elementet ikke finnes eller evt er sist.
  - TabellMengde: for-løkke som sjekker hvert element i tabellen, samme som LenketMengde, best case er  $O(1)$  dersom elementet er først, og  $O(n)$  hvis ikke finnes eller sist.

- ii. Boolean erDelmengdeAv(MengdeADT<T> annenMengde)
  - LenketMengde: sjekker hvert element i mengden og om det er likt annenMengde,  $O(1)$  hvis vi finner det med en gang som er best case, dersom annenMengde har  $m$  elementer og vi ikke finner det med en gang er tiden i verste fall  $O(n*m)$
  - TabellMengde: samme som LenketMengde
- iii. Boolean erLik(MengdeADT<T> annenMengde)
  - LenketMengde: logikken blir lik som i erDelMengdeAv, vi itererer gjennom tabellen og sammenligner opp mot annenMengde, best case vi finner den med en gang  $O(1)$ , worst case vi må gå gjennom alle elementene  $n$  i annenMengde  $m$   $O(n*m)$ .
  - TabellMengde: samme som LenketMengde
- iv. MengdeADT<T> union(MengdeADT<T> annenMengde)
  - LenketMengde: siden hvert element blir kopier til en ny lenket liste har vi  $O(n+m)$
  - TabellMengde: samme som Lenket
- v. Fjern(T element)
  - LenketMengde: best case er at elementet blir funnet først og fjernet  $O(1)$ , worst case er  $O(n)$ , hvis elementet ikke finnes eller er sist.
  - TabellMengde:  $O(1)$  her også dersom elementet er først i tabellen slik at vi får fjernet det med en gang, også  $O(n)$  som worst case her, bruker en for-løkke og i verste fall må man sjekke alle elementene  $n$

## Oppgave. 4. e.

Hashset er raskere, noe som gir mening siden søking i hashset skjer i  $O(1)$ , mens binærsøk bruker  $O(\log n)$ .