```
import processing.data.JSONObject;
import processing.data.JSONArray;
import java.util.HashMap;

HashMap<String, JSONObject> students = new HashMap<String, JSONObject>();
String searchQuery = "";
JSONObject result;

void setup() {
  size(600, 600);
  background(255);


  loadAndStoreJSON("elev_1.json");
  loadAndStoreJSON("elev_2.json");
}
```

Jeg startede med at importer processing bibloteker, for at kunne parse jsonarray og json objekter.

Derefter opretter jeg en hashmap som jeg har kaldt for students. En hashmap er en måde at gemme sine objekter og give dem en nøgle værdi for hurtig parsing.

Derefter opretter jeg en tom searchQuery string, Dette string opdateres hvergang vi skriver noget. Fx hvis vi skriver E, så vil dens værdi være E, hvis vi skriver Em, så vil dens værdi være Em osv.

Derefter opretter jeg et json objekt som hedder result.

Derefter loader jeg mine to json filer.

```
void loadAndStoreJSON(String fileName) {
  JSONObject jsonobj = loadJSONObject(fileName);

  if (jsonobj != null) {
    String studentID = jsonobj.getString("studentID", "");
    students.put(studentID, jsonobj);
    println("Loaded student: " + studentID + " - " +
jsonobj.getString("fornavn") + " " + jsonobj.getString("efternavn"));
  }
}
```

Her laver jeg en funktion LoadAndStorejson(String fileName). Hvor jeg bruger en string som hedder fileName til at parse min json filer når info bliver kaldt. fileName string bruges som input når filerne skal læses. Fx hvis jeg læser elev_1.json så er dens værdi elev_1.json og det er samme situation hvis jeg læser elev_2.json.

Derefter opretter jeg et nyt objekt og kalder den for jsonobj hvor jeg gemmer datan i den.

I if sætning der tjekker jeg om min jsonobject er læst korrekt, (!= betyder hvis den er ikke ligemed null). Jeg trækker elevens unikke `studentID` ud fra JSON-dataene og gemmer det i en streng, som hedder StudentID. Derefter gemmer vi vores jsonobject i vores hashmap og bruger studentID som nøgle værdi.

```
void draw() {
  background(255);
  displaySearchBar();
  displaySearchResults();
}
```

Her på min void draw metode

Opretter en hvid baggrund

Opretter min displaysearchbar function for at vise searchbar

Opretter en funktion som viser searchresults.

```
void displaySearchBar() {
  fill(255);
  rect(50, 50, 400, 30);
  fill(0);
  textSize(16);
  text("Search: " + searchQuery, 60, 70);
}
```

fill(255): Denne funktion sætter farven til hvid (255) til at udfylde former med, i dette tilfælde søgefeltet.

rect(50, 50, 400, 30): Tegner en hvid rektangel på skærmen med startposition i (50, 50) og med en bredde på 400 pixels og en højde på 30 pixels. Dette er baggrunden for søgefeltet.

fill(0): Sætter farven til sort (0) til at tegne teksten med.

textSize(16): Sætter tekststørrelsen til 16 pixels.

text("Search: " + searchQuery, 60, 70): Tegner teksten "Search: " efterfulgt af det, der er indtastet i searchQuery, på positionen (60, 70) på skærmen. Dette viser, hvad brugeren har indtastet i søgefeltet.

```
void keyPressed() {
  if (key == BACKSPACE) {
    if (searchQuery.length() > 0) {
      searchQuery = searchQuery.substring(0, searchQuery.length() - 1);
      result = searchStudent(searchQuery, students);
    }
  } else if (key >= ' ' && key <= '~') {
    searchQuery += key;
    result = searchStudent(searchQuery, students);
  }
}
```

void keyPressed(): Dette er en standard Processing-funktion, der registrerer, når en tast på tastaturet bliver trykket. Hver gang en tast bliver trykket, kører denne funktion automatisk.

if (key == BACKSPACE): Denne betingelse tjekker, om brugeren har trykket på BACKSPACE. Hvis det er tilfældet: Hvis søgestrengen searchQuery har en længde større end 0, vil den sidste karakter blive fjernet fra strengen ved hjælp af substring().

Derefter opdateres resultatet ved at kalde searchStudent(searchQuery, students), som søger i elevdataene baseret på den opdaterede søgestreng i vores hashmap.

else if (key >= ' ' && key <= '~'): Dette tjekker, om den indtastede tast er en almindelig tegn (fra mellemrum til tilde ~). Hvis brugeren trykker på en gyldig tast:

Tasten tilføjes til searchQuery, hvilket opdaterer søgestrengen.

Søgefunktionen searchStudent(searchQuery, students) kaldes igen for at finde et match med den opdaterede søgestreng.

```
void displaySearchResults() {
  if (result != null) {
    printPersonInfo(result);
  } else if (searchQuery.length() > 0) {
    fill(0);
    textSize(16);
    text("No results found", 60, 150);
  } else {
    fill(255);
    rect(50, 120, 500, 400);
  }
}
```

if (result != null): Dette tjekker, om der er et søgeresultat i variablen result. Hvis et resultat er fundet, kaldes funktionen printPersonInfo(result), som viser elevens data på skærmen.

else if (searchQuery.length() > 0): Hvis der ikke er et søgeresultat, men søgestrengen searchQuery indeholder noget tekst, betyder det, at der er søgt efter noget, men intet blev fundet.

fill(0): Sætter farven til sort for at tegne teksten.

textSize(16): Sætter tekststørrelsen til 16 pixels.

text("No results found", 60, 150): Tegner teksten "No results found" på skærmen, der indikerer, at ingen elev matchede søgningen.

else: Hvis ingen søgestreng er indtastet (dvs. søgestrengen er tom), udfyldes en rektangel med hvid farve:

fill(255): Sætter farven til hvid.

rect(50, 120, 500, 400): Tegner en hvid rektangel på skærmen med dimensionerne 500x400 pixels på positionen (50, 120). Dette rydder skærmen, når der ikke er nogen søgestreng eller resultat

```
void printPersonInfo(JSONObject jsonobj) {
  if (jsonobj != null) {
    String fornavn = jsonobj.getString("fornavn", "Ukendt");
    String efternavn = jsonobj.getString("efternavn", "Ukendt");
    int alder = jsonobj.getInt("alder", -1);
    boolean erStudent = jsonobj.getBoolean("erStudent", false);
    JSONArray fag = jsonobj.getJSONArray("fag");
    JSONObject adresse = jsonobj.getJSONObject("adresse");
    JSONArray telefonnumre = jsonobj.getJSONArray("telefonnumre");

    textSize(16);
    fill(0);

    text("Navn: " + fornavn + " " + efternavn, 60, 150);
    text("Alder: " + alder, 60, 170);
    text("Er student: " + (erStudent ? "Ja" : "Nej"), 60, 190);

    if (fag != null) {
      text("Fag: " + fag.join(", "), 60, 210);
    } else {
      text("Fag: Ukendt", 60, 210);
    }

    if (adresse != null) {
      text("Adresse: " +
           adresse.getString("gade", "Ukendt") + ", " +
           adresse.getString("by", "Ukendt") + ", " +
           adresse.getInt("postnummer", -1), 60, 230);
    } else {
      text("Adresse: Ukendt", 60, 230);
```

```
    }

    text("Student ID: " + jsonobj.getString("studentID", "Ukendt"), 60,
250);

    if (telefonnumre != null) {
      for (int i = 0; i < telefonnumre.size(); i++) {
        JSONObject telefon = telefonnumre.getJSONObject(i);
        text("Telefon (" + telefon.getString("type", "Ukendt") + "): " +
             telefon.getString("nummer", "Ukendt"), 60, 270 + i * 20);
      }
    }
  } else {
    println("printPersonInfo: jsonobj is null");
  }
}
```

void printPersonInfo(JSONObject jsonobj): Dette er en funktion, der viser en elevs oplysninger på skærmen, hvis der er et søgeresultat. Den tager et JSONObject som argument, som indeholder elevens data.

if (jsonobj != null): Først tjekkes det, om jsonobj ikke er tom (null). Hvis det er tomt, udskrives en fejlmeddelelse i konsollen, men hvis det ikke er tomt, behandles elevens data.

Oplysninger fra JSON:

Fornavn og efternavn: Hentes fra jsonobj ved hjælp af nøglerne "fornavn" og "efternavn". Hvis disse værdier mangler, vises teksten "Ukendt".

Alder: Hentes som en heltal med nøglen "alder", med -1 som standard, hvis det mangler.

Er student: Hentes som en booloeon værdi (true/false) fra nøglen "erStudent", som angiver, om personen er en student.

Fag: Hvis fag-arrayet eksisterer, vises de fag, som eleven har.

Adresse: Hvis adressen er tilgængelig, vises den med gadenavn, by og postnummer.

Telefonnumre: Hvis telefonnumre findes, vises de sammen med deres type (mobil, hjem, etc.).

Tekst og formatering:

textSize(16): Sætter tekststørrelsen til 16 pixels for alle oplysninger.

fill(0): Sætter farven til sort, så oplysningerne kan ses tydeligt.

text(...): Viser elevens oplysninger på skærmen i rækker med en passende afstand mellem dem (y-koordinaten for teksten øges for hver oplysning).

Loop for telefonnumre: Hvis der er flere telefonnumre, looper vi gennem dem og viser hver af dem med deres type og nummer.

```
JSONObject searchStudent(String query, HashMap<String, JSONObject>
students) {
  if (query.length() < 3) {
    return null;
  }

  String lowerCaseQuery = query.toLowerCase();

  for (String key : students.keySet()) {
    JSONObject student = students.get(key);

    if (matchesQuery(student, lowerCaseQuery)) {
      return student;
    }
  }

  return null;
}
```

JSONObject searchStudent(String query, HashMap<String, JSONObject> students): Denne funktion søger efter en elev i en liste baseret på en søgeforespørgsel (query). Den returnerer en JSONObject, hvis et match findes, ellers returnerer den null.

if (query.length() < 3): Første tjek er, om søgeforespørgslen (query) er kortere end 3 tegn. Hvis den er det, afsluttes funktionen med at returnere null. Dette sikrer, at vi ikke søger, hvis forespørgslen er for kort, og undgår dermed mange irrelevante resultater.

String lowerCaseQuery = query.toLowerCase();: Forespørgslen konverteres til små bogstaver for at sikre, at søgningen er case-insensitive. Det betyder, at det ikke gør en forskel, om brugeren søger med store eller små bogstaver.

For-loop gennem students:

for (String key : students.keySet()): Dette loop går igennem alle elever i HashMap'en students ved hjælp af deres nøgle (som er elevens studentID).

JSONObject student = students.get(key): Henter elevens JSONObject fra HashMap'en baseret på deres studentID.

if (matchesQuery(student, lowerCaseQuery)): For hver elev bliver funktionen matchesQuery() kaldt. Den tjekker, om eleven matcher søgeforespørgslen. Hvis et match findes, returneres den pågældende elevs JSONObject.

return null: Hvis loopet afsluttes uden at finde et match, returneres null, hvilket betyder, at ingen elever matcher søgeforespørgslen.