Person and Turma:

simple example (with ChatGPT support)

ChatGPT

“create java code for a Person and a Turma where Turma has Persons - use a Hash to index the Person but also provide a collection to store the Person. Assume a Person an unique id called nmec and is used an equality criteria”

```
class Person {
    private int nmec;
    private String name;

    public Person(int nmec, String name) {
        this.nmec = nmec;
        this.name = name;
    }

    public int getNmec() {(…) }
    public String getName() { (…) }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Person person = (Person) obj;
        return nmec == person.nmec;
    }

    @Override
    public int hashCode() {   return Objects.hash(nmec);    }

    @Override
    public String toString() {
            return "Person{" + "nmec=" + nmec + ", name='" + name + "'}";    }
}
```

```
class Turma {
    private Map<Integer, Person> personMap;
    private Collection<Person> personCollection;

    public Turma() {
        this.personMap = new HashMap<>();
        this.personCollection = new ArrayList<>();
    }

    public void addPerson(Person person) {
        if (!personMap.containsKey(person.getNmec())) {
            personMap.put(person.getNmec(), person);
            personCollection.add(person);
        }
    }

    public Person getPerson(int nmec) {
        return personMap.get(nmec);
    }

    public Collection<Person> getAllPersons() {
        return Collections.unmodifiableCollection(personCollection);
    }

    public void removePerson(int nmec) {
        Person person = personMap.remove(nmec);
        if (person != null) {
            personCollection.remove(person);
        }
    }
}
```

```java
class Turma {
    private Map<Integer, Person> personMap;
    private Collection<Person> personCollection;

    public Turma() {
        this.personMap = new HashMap<>();
        this.personCollection = new ArrayList<>();
    }

    public void addPerson(Person person) {
        if (!personMap.containsKey(person.getNmec())) {
            personMap.put(person.getNmec(), person);
            personCollection.add(person);
        }
    }

    public Person getPerson(int nmec) {
        return personMap.get(nmec);
    }

    public Collection<Person> getAllPersons() {
        return Collections.unmodifiableCollection(personCollection);
    }

    public void removePerson(int nmec) {
        Person person = personMap.remove(nmec);
        if (person != null) {
            personCollection.remove(person);
        }
    }
}
```

Map for faster search

```
class Turma {
    private Map<Integer, Person> personMap;
    private Collection<Person> personCollection;

    public Turma() {
        this.personMap = new HashMap
        this.personCollection = new
    }

    public void addPerson(Person per
        if (!personMap.containsKey(p
            personMap.put(person.get
            personCollection.add(per
        }
    }

    public Person getPerson(int nmec
        return personMap.get(nmec);
    }

    public Collection<Person> getAll
        return Collections.unmodifia
    }

    public void removePerson(int nme
        Person person = personMap.re
        if (person != null) {
            personCollection.remove(person);
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Turma turma = new Turma();

        Person p1 = new Person(101, "Alice");
        Person p2 = new Person(102, "Bob");
        Person p3 = new Person(103, "Charlie");

        turma.addPerson(p1);
        turma.addPerson(p2);
        turma.addPerson(p3);

        System.out.println("All persons: " + turma.getAllPersons());
        System.out.println("Retrieve person with nmec 102: " +
turma.getPerson(102));

        turma.removePerson(102);
        System.out.println("After removal, all persons: " + turma.get
    }
}
```

jfernan@ua.pt

ChatGPT

“add the support to keep management of lecture presences of Persons in Lectures of a given Turma”

Question: Where do we need changes?

jfernan@ua.pt

```
class Lecture {
    private String topic;
    private Date date;
    private Set<Integer> presentPersons;

    public Lecture(String topic, Date date) {
        this.topic = topic;
        this.date = date;
        this.presentPersons = new HashSet<>();
    }

    public String getTopic() {        return topic;    }
    public Date getDate() {        return date;    }

    public void markPresence(Person person) {
        presentPersons.add(person.getNmec());
    }

    public boolean isPersonPresent(Person person) {
        return presentPersons.contains(person.getNmec());
    }

    @Override
    public String toString() {
        return "Lecture{" + "topic='" + topic + "', date=" + date + ",
presentPersons=" + presentPersons + "}";
    }
}
```

New concept

“Usually” means a new class

```java
class Turma {
    private Map<Integer, Person> personMap;
    private Collection<Person> personCollection;
    private List<Lecture> lectures;

    public Turma() {
        this.personMap = new HashMap<>();
        this.personCollection = new ArrayList<>();
        this.lectures = new ArrayList<>();
    }

    public void addPerson(Person person) { (…) }
    public Person getPerson(int nmec) {        (…) }
    public Collection<Person> getAllPersons() { (…) }
    public void removePerson(int nmec) {  (…) }

    public void addLecture(Lecture lecture) {
        lectures.add(lecture);
    }

    public List<Lecture> getAllLectures() {
        return Collections.unmodifiableList(lectures);
    }

    public void markPresence(int nmec, Lecture lecture) {
        Person person = personMap.get(nmec);
        if (person != null) {
            lecture.markPresence(person);
        }
    }
}
```

Different and unrelated features

=>

Only need to add ... No changes
In remaining code

ChatGPT

“allow to see person that where present than 1/3 of the lectures “

In Turma.java

```java
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
    int totalLectures = lectures.size();
    Map<Integer, Integer> attendanceCount = new HashMap<>();

    for (Lecture lecture : lectures) {
        for (Integer nmec : lecture.getPresentPersons()) {
            attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0) + 1);
        }
    }

    Set<Person> result = new HashSet<>();
    for (Map.Entry<Integer, Integer> entry : attendanceCount.entrySet()) {
        if (entry.getValue() > (fraction * totalLectures)) {
            result.add(personMap.get(entry.getKey()));
        }
    }
    return result;
}
```

jfernan@ua.pt

ChatGPT

adapt the getPersonsWithAttendanceAboveThreshold method to use streams

jfernan@ua.pt

In Turma.java

```java
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

jfernan@ua.pt

In Turma.java

```
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

the attendance counter for each nmec registered in lecture

jfernan@ua.pt

In Turma.java

```
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

Similar to d.get(nmec, 0) in Python dictionaries

In Turma.java

```java
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

Filter all nmec that hava the counter above % over total lectures

jfernan@ua.pt

```
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

Get from that list the actual Person reference

jfernan@ua.pt

In Turma.java

```
public Set<Person> getPersonsWithAttendanceAboveThreshold(double fraction) {
        int totalLectures = lectures.size();
        Map<Integer, Long> attendanceCount = new HashMap<>();

        for (Lecture lecture : lectures) {
            for (Integer nmec : lecture.getPresentPersons()) {
                attendanceCount.put(nmec, attendanceCount.getOrDefault(nmec, 0L) + 1);
            }
        }

        return attendanceCount.entrySet().stream()
            .filter(entry -> entry.getValue() > (fraction * totalLectures))
            .map(entry -> personMap.get(entry.getKey()))
            .collect(Collectors.toSet());
    }
```

Place in a Set to avoid repetitions

```
public class Main {
    public static void main(String[] args) {
        (…)

        Set<Person> frequentAttendees =
              turma.getPersonsWithAttendanceAboveThreshold(1.0 / 3);
        System.out.println("Persons present in more than 1/3 of the lectures: "
             + frequentAttendees);
    }
}
```

Note that changes within Turma do not imply changes in the rest of the code…
The method `getPersonsWithAttendanceAboveThreshold` abstrats for the implementation and focus on the purpose

jfernan@ua.pt