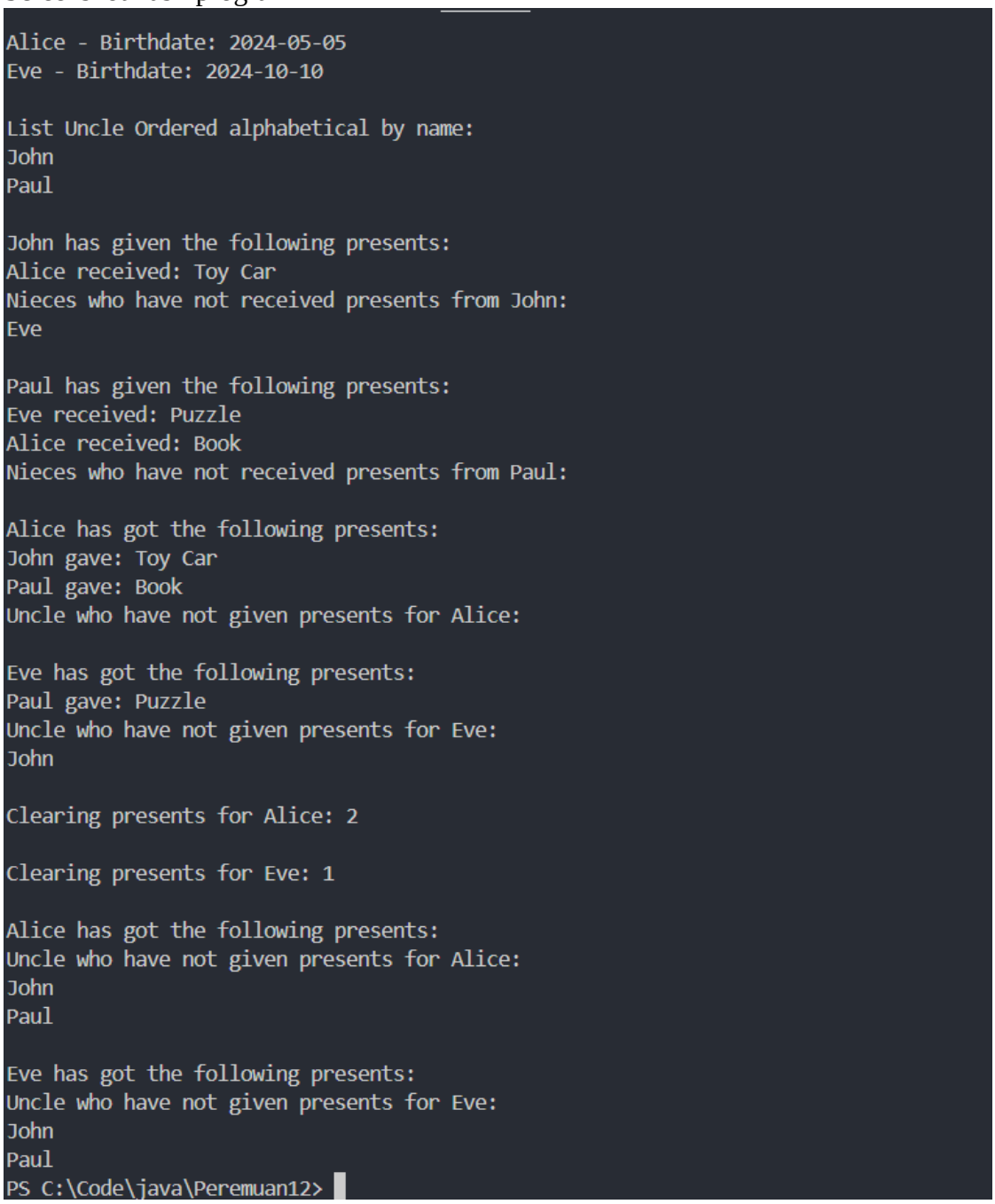
TUGAS TEORI
JAVA COLLECTION

Diajukan untuk memenuhi salat satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek

Disusun Oleh:
Daiva Raditya Pradipa (231511039)

Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
2024

1. Screenshot hasil program



2. Penjelasan

   Program ini bertujuan untuk mengetahui daftar paman dan juga keponakan dalam suatu keluarga. Selain itu program ini juga betujuan untuk mengetahui daftar hadiah yang diberikan oleh paman kepada keponakannya untuk menghindari satu keponakan menerima hadiah yang sama dari 2 atau lebih paman. Untuk mencapai hal itu, program akan meminta paman memasukkan daftar hadiah yang akan diberikan kepada keponakan yang ingin diberikan hadiah, kemudian program akan memeriksa apakah ada hadiah yang sama yang diberikan oleh paman lain untuk keponakan yang sama. Jika ada maka program tidak akan memasukan hadiah tersebut ke daftar hadiah untuk keponakan tersebut. Dengan ini kita dapat memastikan setiap keponakan menerima hadiah yang berbeda dari setiap paman. Kemudian program juga dapat menampilkan

daftar paman yang memberikan hadiah untuk suatu keponakan beserta dengan hadiah yang diberikan, dengan cara ini kita juga secara tidak langsung mendapatkan informasi siapa paman dalam keluarga yang tidak memberi hadiah. Hal ini juga berlaku sebaliknya dalam program ini kita juga dapat menampilkan daftar keponakan yang diberikan hadiah oleh suatu paman, sekaligus menampilkan daftar keponakan mana saja yang belum atau tidak diberikan hadiah oleh suatu paman.

3. Pembahasan Soal

The family decide they need a computer program to manage the giving of presents. The requirements are:

1. Uncles and nieces can be added to the system. The date of each niece's birthday is recorded.

```
public boolean addUncle(String name) {
    Uncle uncle = new Uncle(name,this);
    uncles.add(uncle);
    return true;
}
```

Method addUncle disini betujuan untuk menambahkan daftar paman pada suatu keluatga dengan menyimpannya ke dalam atribut treeset paman sebagai berikut.

```
public class Family {
    private Set<Uncle> uncles;
```

Mengapa saya memilih atribut collection set? Hal ini dikarenakan dalam requirement program terdapat ketentan bahwa nama uncle tidak boleh terdapat duplikasi pada sistem. Maka dari itu set merupakan pilihan yang tepat karena set menjaga uniqueness dari elemen elemen yang ada dalam set. Meskipun begitu, untuk menjaga nilai uniqueness yang kita maksud pada sistem, kita harus mendefinisikan sendiri maksud dari uniqueness pada sistem dengan mengoveride method equals dan hashcode pada class Uncle sebagai berikut.

```
public class Uncle  {
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Uncle uncle = (Uncle) o;
        return Objects.equals(this.name, uncle.name);
    }

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

Method equals() disini akan membandingkan dua objek Uncle berdasarkan atribut name. Pertama, ia memeriksa apakah objek yang dibandingkan adalah objek yang sama menggunakan this == o. Jika benar, maka equals() mengembalikan true,

karena dua referensi menunjuk ke objek yang sama. Jika objek yang dibandingkan (o) adalah null atau bukan tipe kelas yang sama (Uncle), metode ini akan mengembalikan false. Selanjutnya, jika tipe objek cocok, ia melakukan perbandingan nilai name dari objek this dan objek uncle (hasil cast dari o). Jika name pada kedua objek sama, maka equals() akan mengembalikan true, menunjukkan bahwa kedua objek dianggap sama.

Sedangkan method hascode() akan menggunakan Objects.hash(name), yang mengembalikan nilai hash unik berdasarkan nilai name. Method ini memastikan bahwa dua objek Uncle dengan nilai name yang sama akan memiliki nilai hash yang sama. Sehingga nantinya dalam set uncle dua nama uncle dengan nama yang sama akan memiliki nilai hash yang sama sehingga set akan secara otomatis menolah memasukan nama uncle yang sama pada percobaan insert kedua.

Sebagai contoh pada main program saya mencoba menginsertkan uncle dengan list nama sebagai berikut

```
family.addUncle(name:"John");
family.addUncle(name:"Paul");
family.addUncle(name:"Paul");
```

Disini kita bisa lihat bahwa saya mencoba menambahkan daftar uncle dengan nama yang sama yaitu paul. Meskpun begitu Ketika program dijalankan dan fungsi listUncle dijalankan akan mendapatkan hasil sebagai berikut

```
List Uncle Ordered alphabetical by name:
John
Paul
```

Selanjutnya terdapat method addNiece yang berfungsi unuk menambahkan daftar keponakan pada suatu keluarga dengan implementasi method sebagai berikut

```
public boolean addNiece(String name, int day, int month) {
    Niece niece = new Niece(name,day,month,this);
    nieces.add(niece);
    return true;
}
```

Method ini bertujuan untuk menambahkan daftar keponakan pada suatu keluarga beserta data tanggal ulang tahun untuk masing masing keponakan. Untuk mencapai hal ini atribut daftar keponakan dalam suatu keluarga saya simpan kedalam treeMap dengan implementasi sebagai berikut

```java
public class Family {
    private Set<Uncle> uncles;
    private Set<Niece> nieces;
```

```java
public Family() {
    this.uncles = new TreeSet<>();
    this.nieces = new TreeSet<>();
}
```

Mengapa saya memilih atiribut set untuk manampung daftar keponakan? Alasannya identic dengan untuk menampung daftar Uncle, karena keunikan juga menjadi hal penting disini dimana keponakan dengan nama yang sama tidak boleh ada pada daftar keponakan dalam suatu keluarga.

2. A list of uncles can be generated – in alphabetical order by name.
   Untuk mencapai ini cukup mudah, karena kita telah menggunakan TreeSet yang memungkinkan pengurutan berdasarkan suatu ketentuan saat memasukan data pada daftar Uncle di class family, maka pada class Uncle kita hanya perlu mengimplementasikan interface generic Comparable sebagai berikut

```java
public class Uncle implements Comparable<Uncle> {
    private String name;
    private Map<Niece, Set<Present>> presents;
    private Family family;
```

Kemudian kita perlu mengimplementasikan method CompareTo sesuai dengan ketentuan pengurutan yang kita inginkan. Karena disini kita diminta untuk melakukan pengurutan berdasarkan nama, maka kita hanya perlu mengimplementasikan method compareTo() sebagai berikut

```java
@Override
public int compareTo(Uncle other) {
    return this.name.compareTo(other.name);
}
```

Kemudian kita hanya perlu mendefinisikan method untuk menampilkan daftar paman karena paman telah secara otomatis di simpan pada set uncle berdasarkan atribut nama dengan urutan alfabet dengan menggunakan method sebagai berikut.

```java
public void listUncle() {
    System.out.println(x:"List Uncle Ordered alphabetical by name: ");
    for (Uncle uncle : uncles) {
        System.out.println(uncle.getName());
    }
}
```

Dimana method ini hanya akan melooping masing masing element pada set uncle dan menampilkan nama masing masing objek dalam set uncles dengan menggunakan method getName().

3. A list of nieces can be generated – in order of birthday.
   Untuk mencapai hal ini kita hanya perlu melakukan hal yang sama seperti yang dilakukan pada daftar Uncle yaitu mengimplementasikan interface generic Comparable pada class Niece sebagai berikut

```java
public class Niece implements Comparable<Niece> {
    private String name;
    private int day;
    private int month;
    private Family family;
```

Kemudian kita perlu mengimplementasikan method CompareTo() sesuai dengan ketentuan pengurutan yang kita inginkan. Karena disini kita diminta untuk melakukan pengurutan berdasarkan tanggal ulang tahun, maka kita hanya perlu mengimplementasikan method compareTo() sebagai berikut

```java
@Override
public int compareTo(Niece other) {
    int monthComparison = Integer.compare(this.month, other.month);

    if (monthComparison != 0) {
        return monthComparison;
    }
    return Integer.compare(this.day, other.day);
}
```

Method compareTo() pada kelas Niece ini berfungsi untuk membandingkan dan mengurutkan objek Niece berdasarkan tanggal lahirnya, yaitu menggunakan atribut month (bulan) terlebih dahulu, kemudian day (hari). Cara kerjanya dimulai dengan membandingkan month milik this (instansi Niece saat ini) dengan month milik other (instansi Niece yang diberikan sebagai parameter). Metode Integer.compare(this.month, other.month) digunakan untuk perbandingan ini, yang akan menghasilkan nilai negatif jika this.month lebih kecil dari other.month, nilai nol jika bulannya sama, dan nilai positif jika this.month lebih besar. Jika

perbandingan month tidak menghasilkan kesamaan (nilai nol), maka hasil perbandingan tersebut langsung dikembalikan, sehingga proses perbandingan berakhir di sini. Namun, jika bulan dari kedua objek sama, perbandingan dilanjutkan ke atribut day (hari) menggunakan Integer.compare(this.day, other.day).

Kemudian kita hanya perlu mendefinisikan method untuk menampilkan daftar paman karena paman telah secara otomatis di simpan pada set uncle berdasarkan atribut day dan month dengan urutan dari yang terawal hingga yang terakhir dengan menggunakan method berikut.

```
public void listNiece() {
    System.out.println(x:"List Niece Ordered by birtday: ");
    for (Niece niece : nieces) {
        System.out.println(niece.getName() + " - Birthdate: " + niece.getMonth() + "/" + niece.getDay());
    }
}
```

Dimana method ini hanya akan melooping masing masing element pada set nieces dan menampilkan nama, bulan lahir, dan tangggal lahir masing masing objek Niece dalam set nieces dengan menggunakan method getName(), getMonth(), dan getDay().

4. The system holds a list of the presents selected by each uncle for the next birthday of one of his nieces. Each present is described in a few words.

```
public class Uncle implements Comparable<Uncle> {
    private String name;
    private Map<Niece, Set<Present>> presents;
    private Family family;
```

```
public class Present {
    private Uncle giver;
    private Niece recipient;
    private String description;
```

Disini class Uncle memiliki atribut bernama presents, yang bertipe Map dengan Niece sebagai key dan Set<Present> sebagai nilai. Ini berarti setiap paman (Uncle) dapat menyimpan daftar hadiah yang dipilih untuk masing-masing keponakannya (Niece). Setiap hadiah diwakili oleh objek Present, yang memiliki atribut giver bertipe Uncle (pemberi hadiah), recipient bertipe Niece (penerima hadiah), dan description bertipe String yang berisi deskripsi singkat mengenai hadiah tersebut.

Dengan struktur ini, sistem dapat mengatur dan menyimpan informasi hadiah yang telah dipilih oleh setiap Uncle untuk ulang tahun keponakannya. Setiap Uncle memiliki daftar hadiah tersendiri yang terkait dengan keponakannya, sehingga setiap keponakan dapat menerima hadiah-hadiah spesifik yang disesuaikan untuk mereka. Penggunaan set Presents pada map presents dimaksudkan bahwa suatu

paman dapat memberikan hadiah lebih dari satu untuk satu niece dan hal ini juga memastikan paman tersebut tidak akan memberikan hadiah yang sama lagi untuk suatu keponakan tersebut karena set menjaga uniqueness berdasarkan atribut desctription pada class Presents.

5. An uncle can enter the present he intends to give to one of his nieces. The program ensures that: i. each niece receives something different from each uncle ii. each uncles gives something different to each niece.

```
public boolean addPresent(Niece recipient, String description) {
    if (!isUniquePresent(recipient, description)) {
        return false;
    }

    Present newPresent = new Present(this, recipient, description);
    Set<Present> recipientPresents = presents.get(recipient);

    if (recipientPresents == null) {
        recipientPresents = new HashSet<>();
        presents.put(recipient, recipientPresents);
    }

    recipientPresents.add(newPresent);
    return true;
}
```

Method addPresent disini bertugas untuk menambahkan hadiah baru ke daftar hadiah Niece. Sebelum menambahkan hadiah, method ini memanggil isUniquePresent untuk memastikan bahwa deskripsi hadiah tersebut unik dan belum pernah diberikan kepada Niece oleh Uncle lain. Jika isUniquePresent mengembalikan false, hadiah tidak akan ditambahkan, dan method mengembalikan nilai false. Jika isUniquePresent mengembalikan true, addPresent akan membuat objek Present baru dengan deskripsi hadiah dan menambahkannya ke dalam daftar hadiah Niece di presents. Dengan demikian, addPresent memastikan bahwa setiap Niece hanya menerima hadiah unik dari masing-masing Uncle dan bahwa setiap Uncle memberikan hadiah yang berbeda kepada setiap Niece.

```
boolean isUniquePresent(Niece recipient, String description) {
    for (Uncle uncle : family.getUncles()) {
        Set<Present> presents = uncle.getPresents().get(recipient);
        if (presents != null) {
            Iterator<Present> iterator  = presents.iterator();
            while(iterator.hasNext()){
                Present present = iterator.next();
                if (present.getDescription().equals(description)) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

Method isUniquePresent berfungsi untuk memeriksa apakah hadiah dengan deskripsi yang sama sudah pernah diberikan kepada Niece yang sama oleh Uncle lain dalam keluarga. Pertama, method ini akan mencari semua Uncle yang ada di dalam family dan mengambil daftar hadiah yang telah diterima oleh Niece tersebut dari setiap Uncle. Jika ditemukan hadiah dengan deskripsi yang sama, method akan mengembalikan nilai false, yang berarti hadiah tersebut tidak unik dan sudah pernah diterima oleh Niece tersebut. Jika deskripsi hadiah tidak ada dalam list hadiah uncle pada niece yang akan diberikan hadiah, maka method akan mengembalikan nilai true, yang berarti hadiah tersebut belum pernah diterima oleh niece yang akan diberikan hadiah dan dapat diberikan.

6. A list of the presents given by one of the uncles can be generated, showing the niece who is to receive it. The nieces for whom no present has been chosen should also be listed.

```java
public void listPresent() {
    System.out.println(name + " has given the following presents:");

    Iterator<Niece> iterator1 = presents.keySet().iterator();
    while (iterator1.hasNext()) {
        Niece niece = iterator1.next();
        Set<Present> recipientPresents = presents.get(niece);
        for (Present present : recipientPresents) {
            System.out.println(present.getRecipient().getName() + " received: " + present.getDescription());
        }
    }


    System.out.println("Nieces who have not received presents from " + name + ":");
    Iterator<Niece> itertor = family.getNieces().iterator();
    while(itertor.hasNext()){
        Niece niece = itertor.next();
        if (!presents.containsKey(niece)) {
            System.out.println(niece.getName());
        }
    }

}
```

Unutk memungkinkan sistem menampilkan daftar hadiah yang pernah diberikan oleh suatu paman serta menampilkan keponakan yang tidak diberi hadiah oleh suatu paman, disini saya membuat suatu method yang memiliki 2 tahapan yaitu tahapan memprint daftar hadiah yang diberikan suatu paman dan daftar niece yang tidak diberikan oleh suatu paman.

Pada tahapan pertama disini saya melakukan mapping menggunakan iterator untuk key Niece pada map, dimana value setiap map saya lakukan looping masing masing element setnya yang berisi object Presents kemudian saya print nama keponakan dengan menggunakan method getReciptent yang mengembalikan objek Niece kemudian menggunakan method getName untuk mendapatkan nama keponakan yang mendapatkan hadiah tersebut. Kemudian saya juga menggunakan method getDesctiption untuk mendapatkan deskripsi atau nama hadiah yang diberikan untuk keponkana tersebut.

Kemudian pada tahapan kedua disini memiliki proses yang identic hanya saja disini saya melakukan looping menggunakan iterator untuk set niece yang ada pada class family menggunakan method family.getNiece. kemudian setiap niece akan dilakukan pemerikasaan apakah keponakan tersebut ada pada daftar atribut map presents pada class uncle dengan menggunakan method containsKey. Jika keponakan tidak terdaftar, maka sistem akan menampilkan nama keponakan tersebut pada console atau terminal.

7. A list of presents to be received by one of the nieces can be generated, showing the uncle giving it. The uncles who have no present for the niece should also be listed.

```java
public void listPresent() {
    System.out.println(name + " has got the following presents:");

    Iterator<Uncle> iterator1 = family.getUncles().iterator();
    while (iterator1.hasNext()) {
        Uncle uncle = iterator1.next();
        if (uncle.getPresents().containsKey(this)) {
            Set<Present> presentsGiven = uncle.getPresents().get(this);
            for (Present present : presentsGiven) {
                System.out.println(uncle.getName() + " gave: " + present.getDescription());
            }
        }
    }

    System.out.println("Uncle who have not given presents for " + name + ":");
    Iterator<Uncle> iterator2 = family.getUncles().iterator();
    while(iterator2.hasNext()){
        Uncle uncle = iterator2.next();
        if (!uncle.getPresents().containsKey(this)) {
            System.out.println(uncle.getName());
        }
    }
}
```

Untuk memungkinkan sistem menampilkan daftar paman yang memberikan hadiah beserta paman yang tidak memberi hadiah juga memiliki Solusi yang sama seperti sebelumnya. Dimana disini saya membaginya menjadi 2 tahapan yaitu menampilkan daftar paman yang memberi hadiah untuk suatu keponkan dan menampilkan daftar paman yang tidak memberi hadiah untuk suatu keponakan.

Untuk tahapan pertama disini sistem akan melooping set uncle dari suaut family menggunakan iterator. Kemudian object dari masing masing element set akan memanggil method getPresents yang ada pada class Uncle dan memanggil method containsKey karena getPresents mengembalikan presents dengan tipe map untuk memvalidasi apakah keponakan ini ada pada daftar hadiah yang diberikan objek paman tersebut. Jika ada maka sistem akan mentumpan daftar hadiah tersebut pada set presentGiven dan melooping set tersebut denga nisi memprint nama paman menggunakan method getName yang memberikan beserta hadiah yang diberikan menggunakan method getDesctiption.

Untuk tahapan kedua disini memiliki implementasi yang serupa dengan requirement sebelumnya untuk menampilkan daftar keponakan yang tidak diberikan hadiah oleh suatu paman hanya saja kini dibalik yaitu yang sebelumnya keponakan kini menjadi paman.

8. The list of presents for a niece can be deleted (that’s done when her birthday is past).

```java
public int clearPresent() {
    int removedCount = 0;

    Iterator<Uncle> iterator1 = family.getUncles().iterator();
    while (iterator1.hasNext()) {
        Uncle uncle = iterator1.next();
        if (uncle.getPresents().containsKey(this)) {
            Set<Present> presentsGiven = uncle.getPresents().get(this);
            removedCount += presentsGiven.size();
            presentsGiven.clear();
            uncle.getPresents().remove(this);
        }
    }

    return removedCount;
}
```

Untuk menghapus daftar hadiah yang diberikan oleh suatu keponakan oleh paman, disini saya akan memanfaatkan yang Namanya method remove yang dimiliki oleh attribute collection map. Disini proses akan dimulai dengan melakukan looping set uncle menggunakan iterator untuk mendapatkan daftar hadiah yang pernah diberiakan oleh masing-masing paman. Kemudian method akan memeriksan apakah pada daftar hadiah yang diberikan oleh paman pernah memberikan hadiah untuk objek keponakan? Jika iya sistem akan menyimpan daftar hadiah tersebut pada atribut presentsGiven dengan tipe collection set Presents, kemudian sistem akan mengambil jumlah hadiah dan menyimpannya pada variabel int removedCount. Kemudian collection presentsGiven ini akan dihapus isi elemennya karena akan digunakan untuk menampung daftar hadiah oleh paman pada iterasi selanjutnya. Kemudian terakhiir method akan menghapus daftar hadiah untuk objek keponakan pada collection map presents pada class Uncle.

Untuk method sisanya adalah method fineUncle dan findNiece dengan implementasi sebagai berikut

```java
public Uncle findUncle(String name){
    Iterator<Uncle> iterator = uncles.iterator();
    while (iterator.hasNext()) {
        Uncle uncle = iterator.next();
        if(uncle.getName().equals(name)){
            return uncle;
        }
    }

    return null;

}
```

```java
public Niece findNiece(String name){
    Iterator<Niece> iterator = nieces.iterator();
    while (iterator.hasNext()) {
        Niece niece = iterator.next();
        if(niece.getName().equals(name)){
            return niece;
        }
    }

    return null;
```

Dimana masing masing method memiliki cara kerja yang sama yaitu method akan melooping elemetn masing masing set uncle dan nieces menggunakan iterator. Kemudian method akan membandingkan nama masing masing elemetnt uncle atau niece dalam set dengan nama yang dikirimkan melalui parameter. Apabila nama sama maka method akan mereterun objek niece atau uncle dalam element set tersebut. Jika tidak ada nama tersebut pada masing masing objek dalam colleciton set uncle atau niece maka method akan mengembalikan nilai null.

Link Github: Tugas-PBO/week-11 at main · RaditZX/Tugas-PBO