Map, list, iterable iterateur

Exercice 1 - Prendre de la hauteur

Soit la classe `fr.umlv.td.author.Author` :

```
public class Author {
  public Author(String firstName,String lastName) {
    this.firstName=firstName;
    this.lastName=lastName;
  }
  public @Override String toString() {
    return firstName+' '+lastName;
  }
  private final String firstName;
  private final String lastName;
}
```

Expliquer ce que fait le code ci-dessous :

```
Map<Author,String> tels=new HashMap<Author,String>();
Author danBrown=new Author("Dan","Brown");
tels.put(danBrown,"001-745-897");

System.out.println(tels.get(danBrown));
System.out.println(tels.get(new Author("Dan","Brown")));
```

Faites les changements qui s'imposent pour que le code marche de façon plus logique.
Que peut-on faire pour accélérer les accès à la table de hachage ?

Exercice 2 - Interval

On souhaite pouvoir écrire le code suivant :

```
Iterator<Integer> it=rangeIterator(1,5);
for(;it.hasNext();)
  System.out.println(it.next()); // affiche 1 2 3 4 5
```

1 Quel est le profil de la méthode `rangeIterator` ?
2 Implanter de cette méthode en utilisant une classe anonyme.

On souhaite maintenant pouvoir écrire le code suivant :

```
for(int i:range(1,5))
  System.out.println(i); // affiche 1 2 3 4 5
```

1 Quel est le profil de la méthode `range` ?
2 Écrire une implantation utilisant la méthode `rangeIterator` ainsi qu'une version en une seule méthode.
3 Combien le code donné en exemple effectue-t-il d'allocations ?
Et le code ci-dessous ?

```
for(int i:range(190,200))
  System.out.println(i);
```

Exercice 3 - J'en veux plus

On cherche à implanter la méthode `twice` qui prend une liste d'entiers et qui renvoie une nouvelle liste contenant les valeurs de la liste multipliées par deux.

1 Implanter la méthode `twice`.

2 Changer l'implantation pour allouer moins d'éléments en utilisant le concept de vue. Pour l'implantation, regarder du côté de `java.util.AbstractList`.

3 Changer l'implantation de `twice` en fonction du fait que la liste prise en paramètre implante ou non `java.util.RandomAccess`.