Java Stream Operations Coding Exercise

You are provided with an `ArrayList` of `Person` objects. The `Person` class is defined as follows:

java Kodu kopyala

```java
public class Person {
    private String name;
    private int age;
    private String city;

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

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getCity() {
        return city;
    }

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

You will use the following `ArrayList` for all the questions below:

```java
List<Person> people = Arrays.asList(
    new Person("John", 28, "New York"),
    new Person("Alice", 34, "Los Angeles"),
    new Person("Bob", 23, "Chicago"),
    new Person("David", 28, "New York"),
    new Person("Jane", 29, "Los Angeles"),
    new Person("Emma", 40, "Chicago"),
    new Person("Sophia", 29, "New York")
);
```

Questions:

1. **forEach**: Write a Java stream operation to print the names of all the people in the `people` list.
2. **filter**: Use a stream operation to filter out and print all the people who live in "New York".
3. **map**: Transform the `people` list into a list of their names (List<String>) using a stream operation and print the result.
4. **collect**: Convert the stream of people who are older than 30 into a `List<Person>` and print the result.
5. **reduce**: Calculate the sum of the ages of all people using a stream operation and print the result.
6. **flatMap**: Imagine each `Person` has a list of hobbies (List<String> hobbies). Modify the `Person` class to include this attribute, and then write a stream operation to create a combined list of all hobbies of all people (List<String>) and print it.
7. **sorted**: Sort the `people` list by age in ascending order using a stream operation and print the sorted list.
8. **distinct**: Modify the `people` list to include duplicate entries and use a stream operation to remove duplicates based on the `name` attribute, then print the distinct list.
9. **limit**: Use a stream operation to limit the result to the first 3 people in the `people` list and print the result.
10. **skip**: Use a stream operation to skip the first 2 people in the `people` list and print the remaining people.
11. **anyMatch**: Write a stream operation to check if any person in the list lives in "Chicago". Print `true` if at least one person lives in "Chicago", otherwise print `false`.
12. **allMatch**: Use a stream operation to check if all people in the list are older than 18. Print `true` if all are older than 18, otherwise print `false`.
13. **noneMatch**: Write a stream operation to check if none of the people in the list live in "Boston". Print `true` if none live in "Boston", otherwise print `false`.
14. **findFirst**: Use a stream operation to find the first person who is older than 25 and print the person's details.

15. **findAny**: Use a stream operation to find any person who lives in "Los Angeles" and print the person's details.
16. **count**: Write a stream operation to count the number of people who live in "New York" and print the result.
17. **min**: Use a stream operation to find the person with the minimum age and print the person's details.
18. **max**: Write a stream operation to find the person with the maximum age and print the person's details.
19. **toArray**: Convert the stream of people to an array of `Person` objects and print the array.
20. **sum**: Modify the `Person` class to include a salary attribute (int salary) and then write a stream operation to sum the salaries of all people and print the result.
21. **average**: Write a stream operation to calculate the average age of all people in the list and print the result.