

Java Coding Exercise: Stream Operations with `ArrayList`

Problem Description:

You are required to implement a small program that processes a list of `Employee` objects using various Java Stream operations. The `Employee` class is defined as follows:

```
java Kodu kopyala

public class Employee {
    private int id;
    private String name;
    private int age;
    private double salary;
    private String department;

    // Constructor, Getters, and Setters

    public Employee(int id, String name, int age, double salary, String department) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.salary = salary;
        this.department = department;
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public int getAge() { return age; }
    public double getSalary() { return salary; }
    public String getDepartment() { return department; }

    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + '\'' + ", age=" + age + ", salary=" + salary + ", department='" + department + "'}";
    }
}
```

You will be given an `ArrayList<Employee>` containing a list of employees. Your task is to implement the following stream operations to perform various tasks on this list:

Tasks:

1. **forEach**: Print the details of all employees.
2. **filter**: Find all employees who work in the "HR" department and are over 30 years old.
3. **map**: Extract and print a list of all employee names in uppercase.
4. **collect**: Collect the names of all employees who earn more than 50,000 into a `List<String>`.
5. **reduce**: Calculate the total sum of all employees' salaries.
6. **flatMap**: Assuming each department has multiple teams, and you have a list of teams for each department, flatten the list to print all unique team names across all departments.
7. **sorted**: Sort the employees by their salary in ascending order and print the sorted list.


8. **distinct**: Given a list of employee names with possible duplicates, print out the distinct names.
9. **limit**: Print the first 5 employees from the list after sorting them by age in descending order.
10. **skip**: Skip the first 3 employees and print the remaining ones after sorting by their IDs.
11. **anyMatch**: Check if there is any employee who earns more than 100,000.
12. **allMatch**: Check if all employees are at least 18 years old.
13. **noneMatch**: Check if no employee is below 18 years old.
14. **findFirst**: Find and print the first employee who works in the "Engineering" department.
15. **findAny**: Find any employee who is older than 40 years.
16. **count**: Count the number of employees in the "Marketing" department.
17. **min**: Find the employee with the minimum salary.
18. **max**: Find the employee with the maximum salary.
19. **toArray**: Convert the list of employee names into an array of `String`.
20. **sum**: Calculate the sum of ages of all employees.
21. **average**: Calculate the average salary of all employees.

Implementation Guidelines:

- Create an `ArrayList<Employee>` and populate it with sample data.
- Implement each task as a separate method in your program.
- Use appropriate stream operations to complete the tasks.
- Ensure your program is well-structured and includes comments for clarity.

Sample Data:

java

 Kodu kopyala

```
List<Employee> employees = Arrays.asList(  
    new Employee(1, "Alice", 30, 55000, "Engineering"),  
    new Employee(2, "Bob", 35, 60000, "HR"),  
    new Employee(3, "Charlie", 40, 70000, "Marketing"),  
    new Employee(4, "David", 25, 50000, "Engineering"),  
    new Employee(5, "Eve", 28, 40000, "HR"),  
    new Employee(6, "Frank", 50, 90000, "Marketing"),  
    new Employee(7, "Grace", 45, 80000, "Engineering")  
);
```