```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

public class EmployeeInformation {

   public static void main(String[] args) {
      String data =
"John:30:HR:50000,Alice:28:Finance:60000,Bob:35:Engineering:75000,Emily:32:HR:
55000";

      // Task 1: Calculate the total number of employees
      int totalEmployees = calculateTotalEmployees(data);
      System.out.println("Total Employees: " + totalEmployees);

      // Task 2: Calculate the average age of all employees
      double averageAge = calculateAverageAge(data);
      System.out.println("Average Age: " + averageAge);

      // Task 3: Find the employee with the highest salary
      String employeeWithHighestSalary = findEmployeeWithHighestSalary(data);
      System.out.println("Employee with Highest Salary: " +
employeeWithHighestSalary);

      // Task 4: Create a new string with names sorted alphabetically
      String sortedNamesString = sortNamesAlphabetically(data);
      System.out.println("Sorted Names: " + sortedNamesString);
   }

   public static int calculateTotalEmployees(String data) {
      return data.split(",").length;
   }

   public static double calculateAverageAge(String data) {
      List<Integer> ages = Arrays.stream(data.split(","))
            .map(employee -> Integer.parseInt(employee.split(":")[3]))
            .collect(Collectors.toList());

      return ages.stream().mapToInt(Integer::intValue).average().orElse(0);
   }

   public static String findEmployeeWithHighestSalary(String data) {
      return Arrays.stream(data.split(","))
            .max(Comparator.comparingDouble(employee ->
Double.parseDouble(employee.split(":")[3])))
            .map(employee -> employee.split(":")[0])
```

```
            .orElse("");
    }

    public static String sortNamesAlphabetically(String data) {
        return Arrays.stream(data.split(","))
                .map(employee -> employee.split(":")[0])
                .sorted()
                .collect(Collectors.joining(","));
    }
}
```

Solution strategy:

1. For Task 1, we split the data by commas and get the length of the resulting array using the stream.
2. For Task 2, we convert the ages into a list of integers using the stream, calculate the average using average(), and handle the optional case if there are no ages.
3. For Task 3, we use the max operation with a custom comparator to find the employee with the highest salary. We then extract the name using the map operation.
4. For Task 4, we extract the employee names, sort them alphabetically, and join them using the joining() collector.

Let's break down what's happening for task 2:

- We start by splitting the input data into individual employee records using Arrays.stream(data.split(",")).
- Then, for each employee record, we extract the age by splitting the record and converting the second element (age) into an Integer. This is achieved with the map operation: .map(employee -> Integer.parseInt(employee.split(":")[1])).
- Now, we have a stream of ages (e.g., [30, 28, 35, 32]).
- The collect operation is used to accumulate the elements of the stream into a collection. In this case, we use Collectors.toList() to collect the ages into a List<Integer> called ages.
- We then convert the List<Integer> back to a stream (ages.stream()) and calculate the average age using mapToInt(Integer::intValue).average().orElse(0).
- Using Collectors.toList() enables us to easily gather the individual ages into a List, which makes it convenient to perform further stream operations or aggregate functions like calculating the average age.