The "Address" class represents a physical address and serves as a blueprint for creating address objects. It has three attributes: "street" to store the street name, "city" to store the city name, and "state" to store the state name. The class includes a constructor that takes in three parameters (street, city, and state) to initialize the address object. This class is designed to provide a simple representation of an address and can be used in various applications that require storing and manipulating address information.

The "Date" class represents a specific date and provides functionality to store and display date information. It has three public attributes: "Day" to store the day value, "Month" to store the month value, and "Year" to store the year value. The class includes a constructor that takes in three parameters (Day, Month, and Year) to initialize the date object. Additionally, it contains a public method called "displayDate" which prints the date in the format "Day/Month/Year". This class enables the manipulation and presentation of date values in various applications that require date-related functionality.

The "Employee" class represents an employee entity and encapsulates their information. It has attributes such as "name" to store the employee's name, "empId" to store their employee ID, "salary" to store their salary, "jobPosition" to store their job position, "contactNumber" to store their contact number, "hireDate" to store their date of hire, and "address" to store their address. The class includes a constructor that takes in parameters to initialize these attributes. It also contains a method called "EmployeeDetails" which prints the employee's details, including their name, employee ID, salary, job position, hire date, contact number, and address.

The "Test" class serves as a driver class to test and demonstrate various functionalities related to the "Employee" class. It includes methods to arrange employees by salary, retrieve employees by job position, filter employees based on their hire date, count the number of foreign employees, and get employees within a specified salary range.

The main method prompts the user to enter the number of employees and their details. It then creates an array of Employee objects to store and manage the employee information. The "arrangeEmployeeBySalary" method arranges the employees in descending order based on their salary and displays their details. The "getEmployeesByJobPosition" method filters and displays employees based on a specified job position. The "getEmployeesByHireDate" method filters and displays employees hired between two specified dates. The "isBetween" method checks if a given date falls between two other dates. The "foreignEmployeeCount" method counts the number of employees with contact numbers not starting with the country code "+91". The "getEmployeesBySalary" method filters and displays employees within a specified salary range.

Overall, the "Test" class provides a comprehensive set of methods to perform operations on employee data, facilitating sorting, filtering, and analysis based on different criteria.