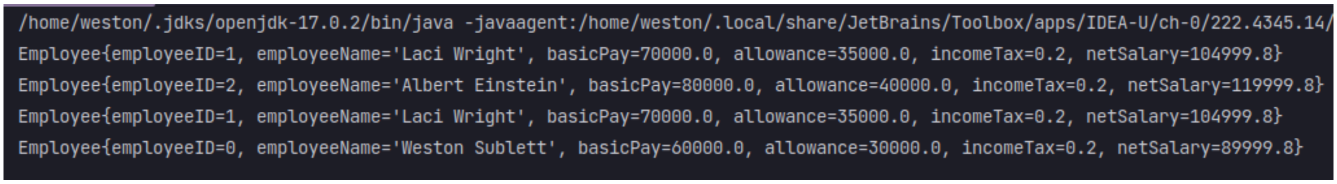
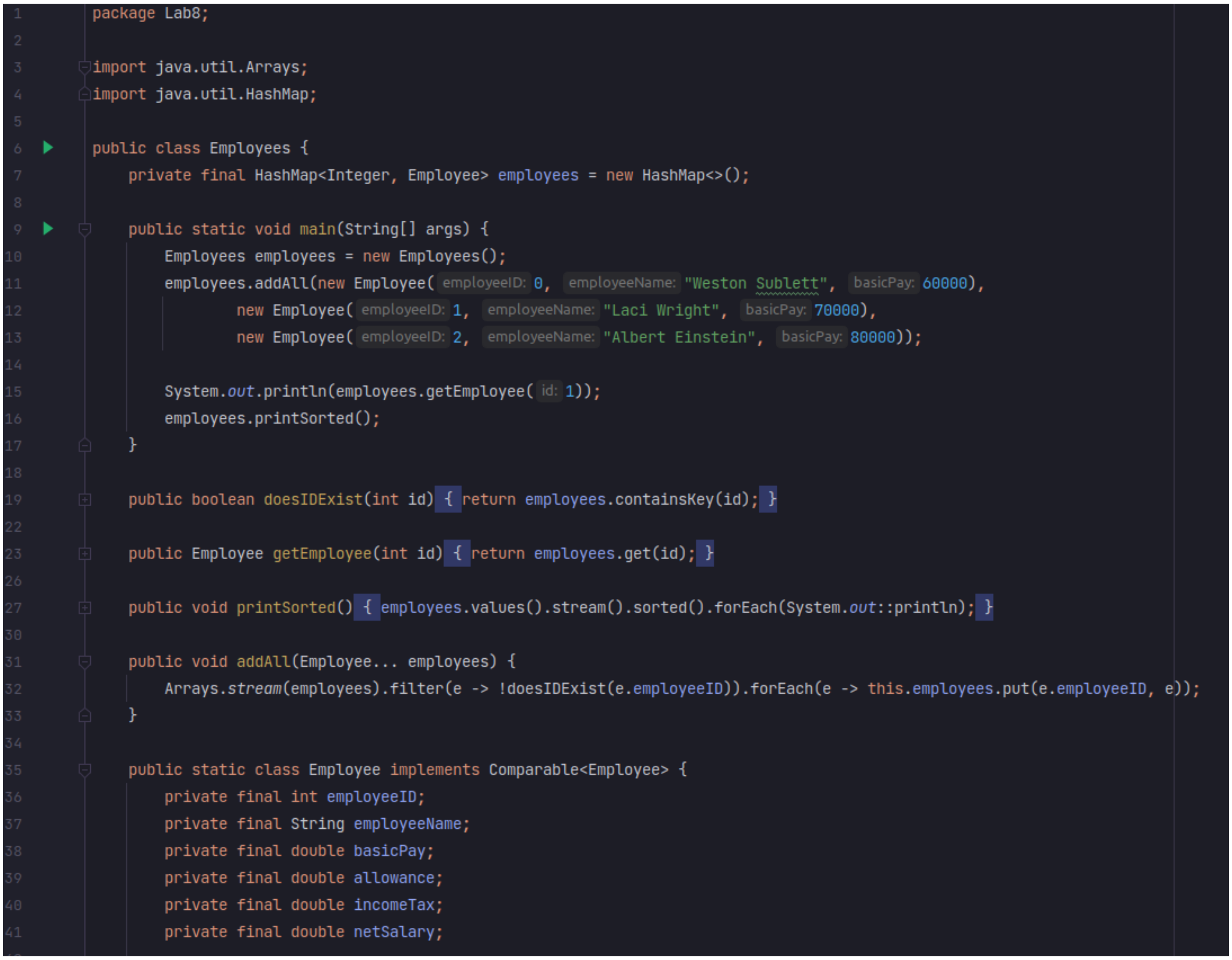
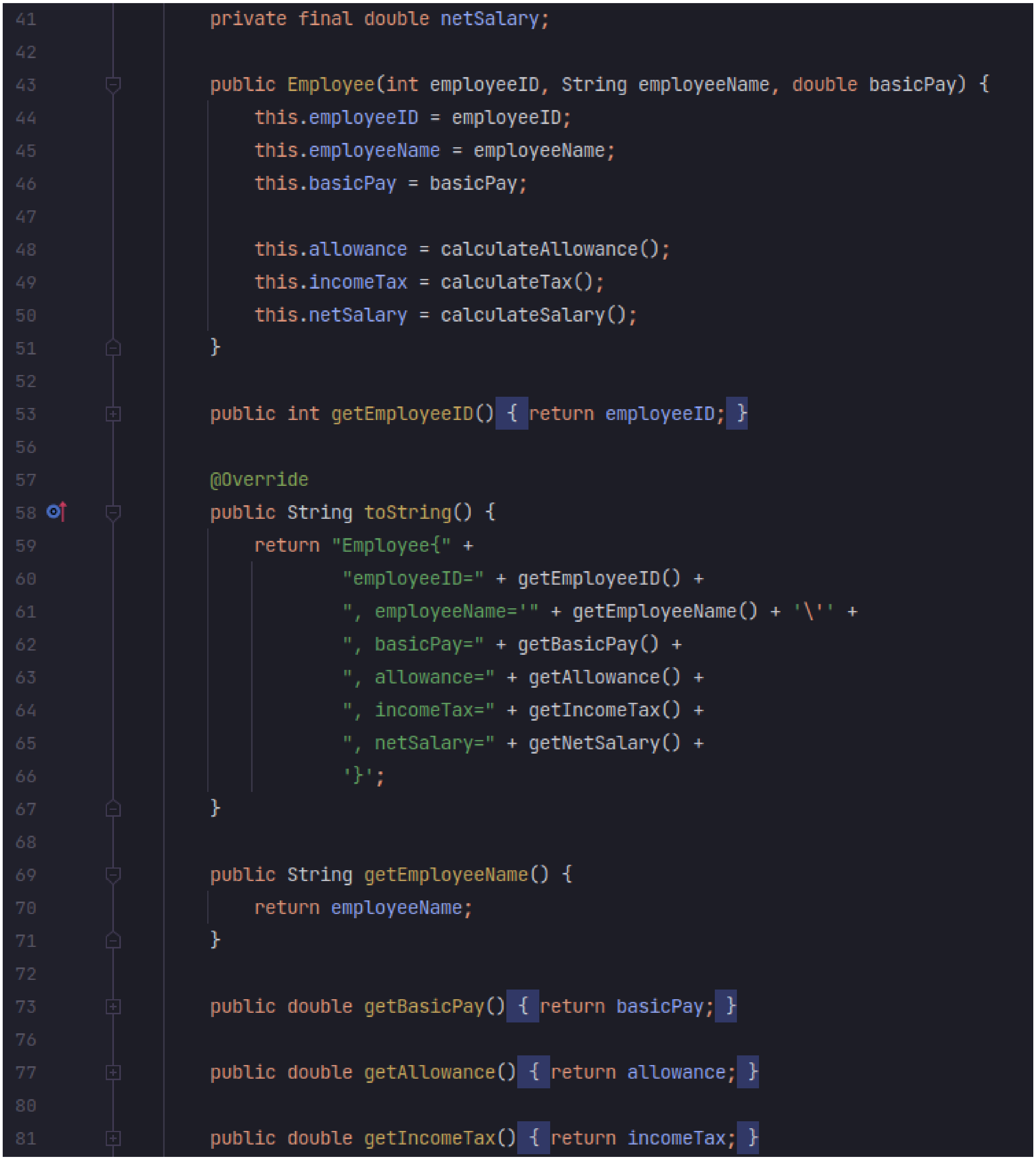


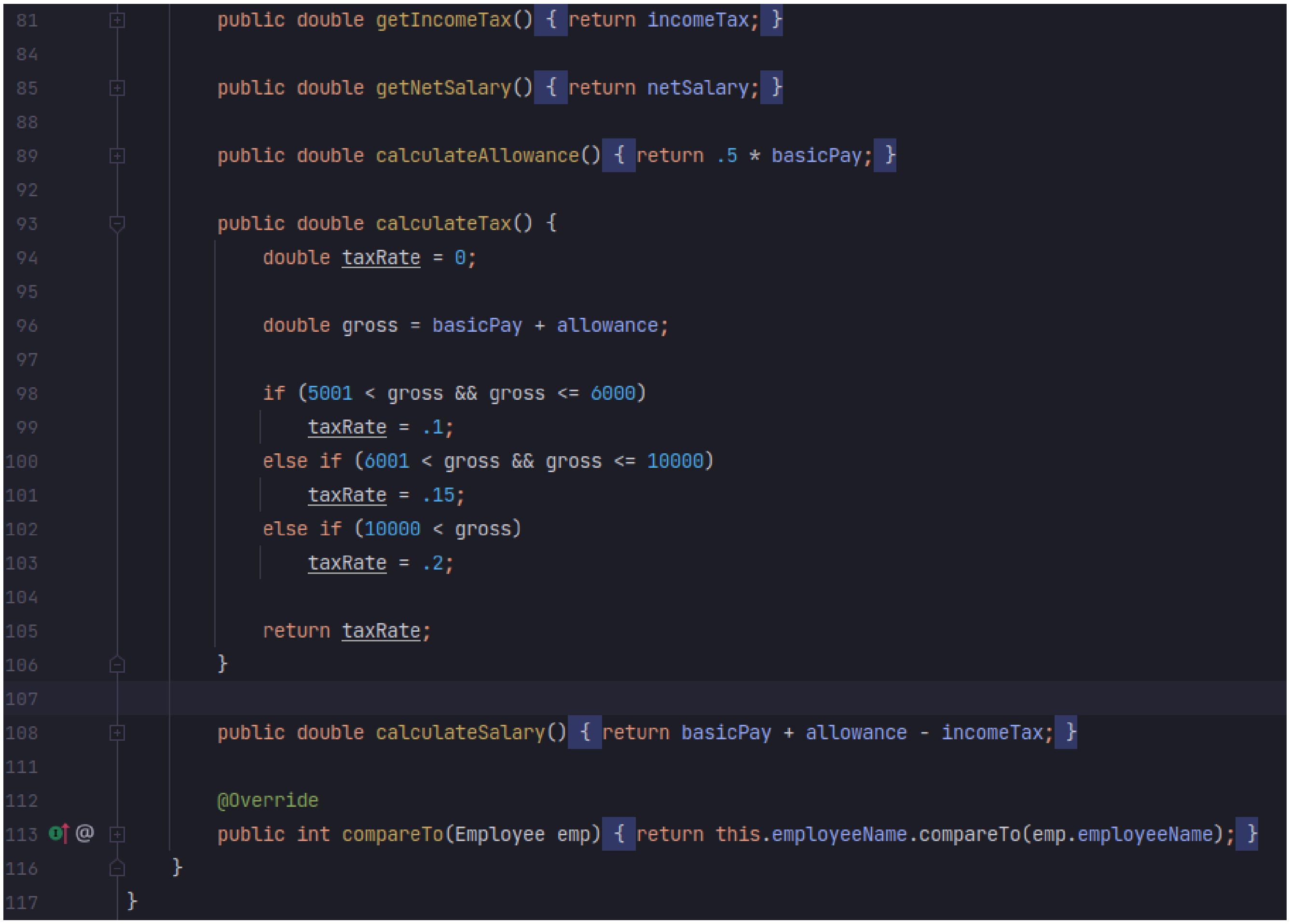
```
/home/weston/.jdks/openjdk-17.0.2/bin/java -javaagent:/home/weston/.local/share/JetBrains/Toolbox/apps/IDEA-U/ch-0/222.4345.14/
Employee{employeeID=1, employeeName='Laci Wright', basicPay=70000.0, allowance=35000.0, incomeTax=0.2, netSalary=104999.8}
Employee{employeeID=2, employeeName='Albert Einstein', basicPay=80000.0, allowance=40000.0, incomeTax=0.2, netSalary=119999.8}
Employee{employeeID=1, employeeName='Laci Wright', basicPay=70000.0, allowance=35000.0, incomeTax=0.2, netSalary=104999.8}
Employee{employeeID=0, employeeName='Weston Sublett', basicPay=60000.0, allowance=30000.0, incomeTax=0.2, netSalary=89999.8}
```



```java
package Lab8;

import java.util.Arrays;
import java.util.HashMap;

public class Employees {
    private final HashMap<Integer, Employee> employees = new HashMap<>();

    public static void main(String[] args) {
        Employees employees = new Employees();
        employees.addAll(new Employee( employeeID: 0,  employeeName: "Weston Sublett",  basicPay: 60000),
                new Employee( employeeID: 1,  employeeName: "Laci Wright",  basicPay: 70000),
                new Employee( employeeID: 2,  employeeName: "Albert Einstein",  basicPay: 80000));

        System.out.println(employees.getEmployee( id: 1));
        employees.printSorted();
    }

    public boolean doesIDExist(int id) { return employees.containsKey(id); }

    public Employee getEmployee(int id) { return employees.get(id); }

    public void printSorted() { employees.values().stream().sorted().forEach(System.out::println); }

    public void addAll(Employee... employees) {
        Arrays.stream(employees).filter(e -> !doesIDExist(e.employeeID)).forEach(e -> this.employees.put(e.employeeID, e));
    }

    public static class Employee implements Comparable<Employee> {
        private final int employeeID;
        private final String employeeName;
        private final double basicPay;
        private final double allowance;
        private final double incomeTax;
        private final double netSalary;
```



```java
    private final double netSalary;

    public Employee(int employeeID, String employeeName, double basicPay) {
        this.employeeID = employeeID;
        this.employeeName = employeeName;
        this.basicPay = basicPay;

        this.allowance = calculateAllowance();
        this.incomeTax = calculateTax();
        this.netSalary = calculateSalary();
    }

    public int getEmployeeID() { return employeeID; }

    @Override
    public String toString() {
        return "Employee{" +
                "employeeID=" + getEmployeeID() +
                ", employeeName='" + getEmployeeName() + '\'' +
                ", basicPay=" + getBasicPay() +
                ", allowance=" + getAllowance() +
                ", incomeTax=" + getIncomeTax() +
                ", netSalary=" + getNetSalary() +
                '}';
    }

    public String getEmployeeName() {
        return employeeName;
    }

    public double getBasicPay() { return basicPay; }

    public double getAllowance() { return allowance; }

    public double getIncomeTax() { return incomeTax; }
```



```java
        public double getIncomeTax() { return incomeTax; }

        public double getNetSalary() { return netSalary; }

        public double calculateAllowance() { return .5 * basicPay; }

        public double calculateTax() {
            double taxRate = 0;

            double gross = basicPay + allowance;

            if (5001 < gross && gross <= 6000)
                taxRate = .1;
            else if (6001 < gross && gross <= 10000)
                taxRate = .15;
            else if (10000 < gross)
                taxRate = .2;

            return taxRate;
        }

        public double calculateSalary() { return basicPay + allowance - incomeTax; }

        @Override
        public int compareTo(Employee emp) { return this.employeeName.compareTo(emp.employeeName); }
    }
}
```