class Employee {

String name;

double baseSalary;

String employmentType;

public Employee(String name, double baseSalary, String employmentType) {

this.name = name;

this.baseSalary = baseSalary;

this.employmentType = employmentType;

}

}

class SalaryCalculator implements Runnable {

private Employee employee;

public SalaryCalculator(Employee employee) {

this.employee = employee;

}

@Override

public void run() {

calculateSalary();

}

private synchronized void calculateSalary() {

int priority = calculateThreadPriority(employee.employmentType);

Thread.currentThread().setPriority(priority);

double bonus = calculateBonus(employee.employmentType);

double tax = calculateTax(employee.baseSalary);

double totalSalary = employee.baseSalary + bonus - tax;

System.out.println("Employee: " + employee.name + ", Base Salary: " + employee.baseSalary +

", Bonus: " + bonus + ", Tax: " + tax + ", Total Salary: " + totalSalary +

", Thread Priority: " + Thread.currentThread().getPriority());

}

private double calculateBonus(String employmentType) {

if ("full-time".equals(employmentType)) {

return Math.random() \* 1500;

} else if ("part-time".equals(employmentType)) {

return Math.random() \* 800;

} else {

return 0;

}

}

private double calculateTax(double baseSalary) {

return baseSalary \* 0.15;

}

private int calculateThreadPriority(String employmentType) {

return "full-time".equals(employmentType) ? Thread.MIN\_PRIORITY : Thread.NORM\_PRIORITY;

}

}

public class Lab5 {

public static void main(String[] args) {

Employee[] employees = new Employee[5];

employees[0] = new Employee("John", 50000, "full-time");

employees[1] = new Employee("Alice", 60000, "part-time");

employees[2] = new Employee("Bob", 55000, "full-time");

employees[3] = new Employee("Eve", 70000, "part-time");

employees[4] = new Employee("Charlie", 80000, "full-time");

Thread[] threads = new Thread[employees.length];

for (int i = 0; i < employees.length; i++) {

Runnable task = new SalaryCalculator(employees[i]);

threads[i] = new Thread(task);

}

for (int i = 0; i < threads.length; i++) {

threads[i].start();

}

}

}

