**实验五**

**1.题目：**

|  |
| --- |
| 1、某公司的雇员分为以下若干类：  Employee：属性，员工的姓名，员工的生日月份；方法，getSalary(intmonth)根据参数月份来确定工资，如果该月员工过生日，则公司会额外奖励100 元。SalariedEmployee：拿固定工资的员工。属性：月薪HourlyEmployee：按小时拿工资的员工，每月工作超出160 小时的部分按照1.5 倍工资发放。属性：每小时的工资、每月工作的小时数  SalesEmployee：销售人员，工资由月销售额和提成率决定。属性：月销售额、提成率  BasePlusSalesEmployee：有固定底薪的销售人员，工资 由底薪加上销售提成部分。属性：底薪。  根据要求创建以上四个类对象各一个，并计算某个月这四个对象的工资。注意：要求把每个类都做成完全封装，不允许非私有化属性。 |

**完整代码：**

Main：

|  |
| --- |
| Java public class Main {  public static void main(String[] args) {  // 创建各类型员工  SalariedEmployee salariedEmployee = new SalariedEmployee("aaa", 5, 5000);  HourlyEmployee hourlyEmployee = new HourlyEmployee("bbb", 6, 30, 180);  SalesEmployee salesEmployee = new SalesEmployee("ccc", 7, 100000, 0.1);  BasePlusSalesEmployee basePlusSalesEmployee =  new BasePlusSalesEmployee("ddd", 8, 80000, 0.08, 3000);   // 测试计算5月份的工资  int month = 5;   System.*out*.println(salariedEmployee.getName() + "的" + month + "月工资为："  + salariedEmployee.getSalary(month));  System.*out*.println(hourlyEmployee.getName() + "的" + month + "月工资为："  + hourlyEmployee.getSalary(month));  System.*out*.println(salesEmployee.getName() + "的" + month + "月工资为："  + salesEmployee.getSalary(month));  System.*out*.println(basePlusSalesEmployee.getName() + "的" + month + "月工资为："  + basePlusSalesEmployee.getSalary(month));  } } |

BasePlusSalesEmployee：

|  |
| --- |
| Java public class BasePlusSalesEmployee extends SalesEmployee {  private double baseSalary;   public BasePlusSalesEmployee(String name, int birthMonth, double monthlySales,  double commissionRate, double baseSalary) {  super(name, birthMonth, monthlySales, commissionRate);  this.baseSalary = baseSalary;  }   public double getBaseSalary() {  return baseSalary;  }   public void setBaseSalary(double baseSalary) {  this.baseSalary = baseSalary;  }   @Override  protected double calculateBaseSalary() {  return baseSalary + super.calculateBaseSalary();  } } |

Employee：

|  |
| --- |
| Java public abstract class Employee {  private String name;  private int birthMonth;   public Employee(String name, int birthMonth) {  this.name = name;  this.birthMonth = birthMonth;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public int getBirthMonth() {  return birthMonth;  }   public void setBirthMonth(int birthMonth) {  this.birthMonth = birthMonth;  }   public double getSalary(int month) {  double salary = calculateBaseSalary();  if (month == birthMonth) {  salary += 100; // 生日奖励  }  return salary;  }   protected abstract double calculateBaseSalary(); } |

HourlyEmployee：

|  |
| --- |
| Java public class HourlyEmployee extends Employee {  private double hourlyWage;  private int hoursWorked;   public HourlyEmployee(String name, int birthMonth, double hourlyWage, int hoursWorked) {  super(name, birthMonth);  this.hourlyWage = hourlyWage;  this.hoursWorked = hoursWorked;  }   public double getHourlyWage() {  return hourlyWage;  }   public void setHourlyWage(double hourlyWage) {  this.hourlyWage = hourlyWage;  }   public int getHoursWorked() {  return hoursWorked;  }   public void setHoursWorked(int hoursWorked) {  this.hoursWorked = hoursWorked;  }   @Override  protected double calculateBaseSalary() {  if (hoursWorked <= 160) {  return hourlyWage \* hoursWorked;  } else {  return (160 \* hourlyWage) + ((hoursWorked - 160) \* hourlyWage \* 1.5);  }  } } |

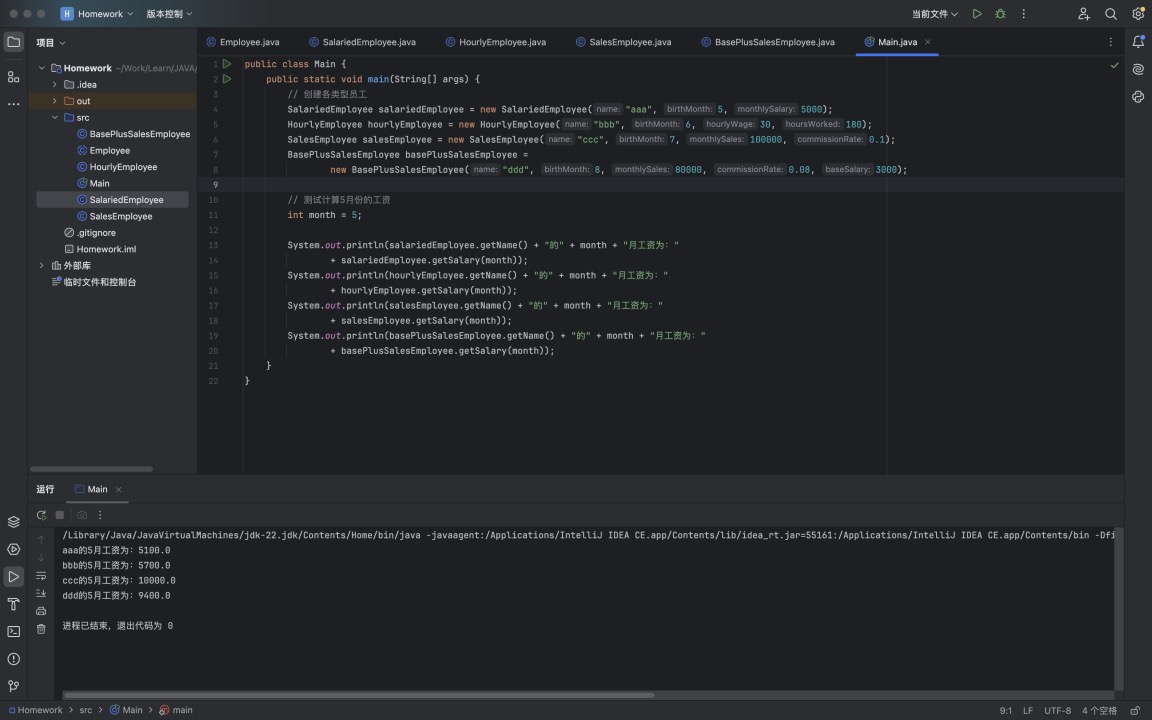
SalariedEmployee：

|  |
| --- |
| Java public class SalariedEmployee extends Employee {  private double monthlySalary;   public SalariedEmployee(String name, int birthMonth, double monthlySalary) {  super(name, birthMonth);  this.monthlySalary = monthlySalary;  }   public double getMonthlySalary() {  return monthlySalary;  }   public void setMonthlySalary(double monthlySalary) {  this.monthlySalary = monthlySalary;  }   @Override  protected double calculateBaseSalary() {  return monthlySalary;  } } |

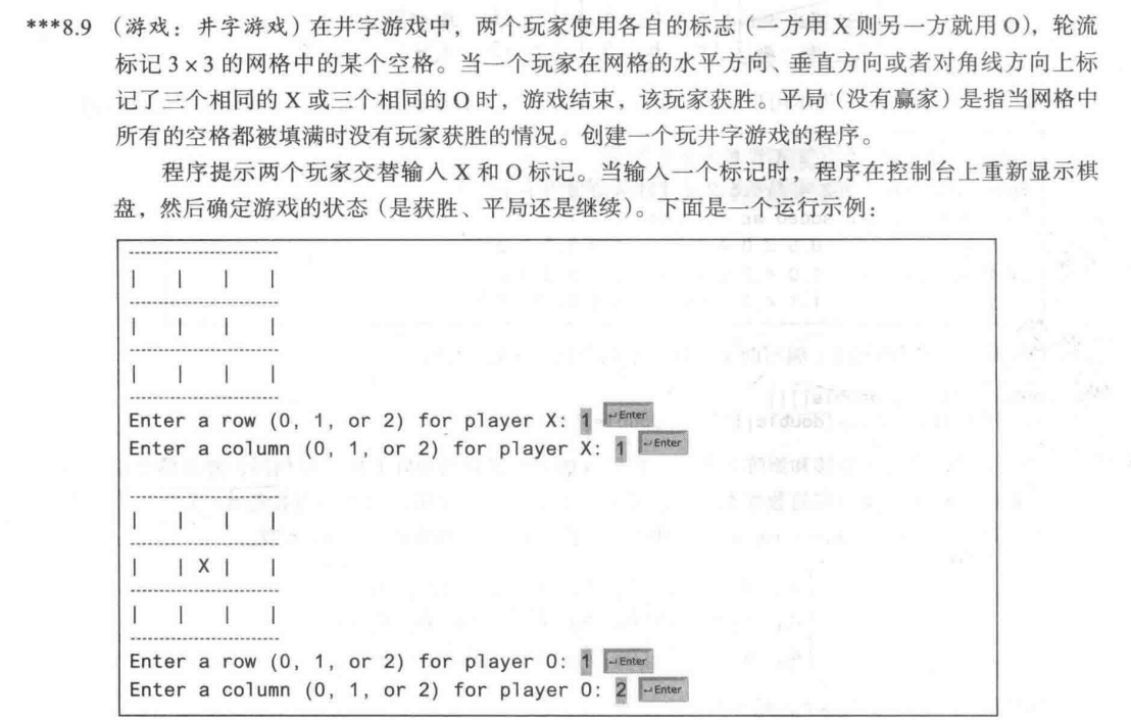
SalesEmployee：

|  |
| --- |
| Java public class SalesEmployee extends Employee {  private double monthlySales;  private double commissionRate;   public SalesEmployee(String name, int birthMonth, double monthlySales, double commissionRate) {  super(name, birthMonth);  this.monthlySales = monthlySales;  this.commissionRate = commissionRate;  }   public double getMonthlySales() {  return monthlySales;  }   public void setMonthlySales(double monthlySales) {  this.monthlySales = monthlySales;  }   public double getCommissionRate() {  return commissionRate;  }   public void setCommissionRate(double commissionRate) {  this.commissionRate = commissionRate;  }   @Override  protected double calculateBaseSalary() {  return monthlySales \* commissionRate;  } } |

**代码运行截图：**



**2.题目：**



**完整代码：**

TicTacToeGame：

|  |
| --- |
| Java import java.util.Scanner;  public class TicTacToeGame {  private char[][] board;  private char currentPlayer;  private boolean gameEnded;   public TicTacToeGame() {  board = new char[3][3];  currentPlayer = 'X';  gameEnded = false;  initializeBoard();  }   private void initializeBoard() {  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 3; j++) {  board[i][j] = ' ';  }  }  }   public void play() {  Scanner scanner = new Scanner(System.*in*);  System.*out*.println("欢迎来到井字游戏！");  System.*out*.println("玩家1使用X，玩家2使用O");   while (!gameEnded) {  displayBoard();  makeMove(scanner);   if (checkWin()) {  displayBoard();  System.*out*.println("玩家 " + currentPlayer + " 获胜！");  gameEnded = true;  } else if (isBoardFull()) {  displayBoard();  System.*out*.println("游戏平局！");  gameEnded = true;  } else {  currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';  }  }  scanner.close();  }   private void displayBoard() {  System.*out*.println("-------------");  for (int i = 0; i < 3; i++) {  System.*out*.print("| ");  for (int j = 0; j < 3; j++) {  System.*out*.print(board[i][j] + " | ");  }  System.*out*.println();  System.*out*.println("-------------");  }  }   private void makeMove(Scanner scanner) {  int row, col;  do {  System.*out*.println("玩家 " + currentPlayer + " 的回合");  System.*out*.print("请输入行号 (1-3): ");  row = scanner.nextInt() - 1;  System.*out*.print("请输入列号 (1-3): ");  col = scanner.nextInt() - 1;  } while (!isValidMove(row, col));   board[row][col] = currentPlayer;  }   private boolean isValidMove(int row, int col) {  if (row < 0 || row >= 3 || col < 0 || col >= 3) {  System.*out*.println("无效的位置！请重新输入。");  return false;  }  if (board[row][col] != ' ') {  System.*out*.println("该位置已被占用！请重新输入。");  return false;  }  return true;  }   private boolean checkWin() {  // 检查行  for (int i = 0; i < 3; i++) {  if (board[i][0] == currentPlayer &&  board[i][1] == currentPlayer &&  board[i][2] == currentPlayer) {  return true;  }  }   // 检查列  for (int j = 0; j < 3; j++) {  if (board[0][j] == currentPlayer &&  board[1][j] == currentPlayer &&  board[2][j] == currentPlayer) {  return true;  }  }   // 检查对角线  if (board[0][0] == currentPlayer &&  board[1][1] == currentPlayer &&  board[2][2] == currentPlayer) {  return true;  }  if (board[0][2] == currentPlayer &&  board[1][1] == currentPlayer &&  board[2][0] == currentPlayer) {  return true;  }   return false;  }   private boolean isBoardFull() {  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 3; j++) {  if (board[i][j] == ' ') {  return false;  }  }  }  return true;  } } |

MainGame：

|  |
| --- |
| Java public class MainGame {  public static void main(String[] args) {  TicTacToeGame game = new TicTacToeGame();  game.play();  } } |

**代码运行截图：**

