

OODP Tutorial 9

1. Briefly explain the term class. In your explanation include details on the difference between an object and a class.

Ans:

In Java, a **class** is a blueprint or a template that defines the properties and behavior of an object. It is a fundamental concept in Object-Oriented Programming (OOP) and is used to create objects that can be instantiated and manipulated. A class typically includes:

- a) **Modifiers:** Public or default access.
- b) **Class Name:** The name of the class, which should start with a capital letter.
- c) **Superclass (if any):** The name of the parent class, preceded by the keyword `extends`.
- d) **Interfaces (if any):** A comma-separated list of interfaces implemented by the class, preceded by the keyword `implements`.
- e) **Body:** The class body, which includes fields (variables) and methods.

A class can contain:

- **Fields:** Variables that provide the state of the class and its objects.
- **Methods:** Functions that describe the behavior of the class and its objects.

Key differences between a class and an object:

- **Class:** A logical entity that defines the properties and behavior of an object.
- **Object:** A physical entity that is an instance of a class, representing a real-world entity such as a book, car, etc.
- **Memory Allocation:** No memory is allocated when a class is declared, but memory is allocated when an object is created.
- **Instantiation:** A class can be instantiated to create multiple objects, whereas an object is a single instance of a class.
- **Inheritance:** A class can extend another class using the `extends` keyword, but an object cannot inherit from another object.

2. Consider Employee as a part of payroll system. What could be the attributes that will be needed in system?

Ans:

Some potential attributes that an employee system could possess:

Attributes (Member Variables):

1. **name (String):** The name of the employee.
2. **employeeId (String or int):** A unique identifier for the employee.

3. `dateOfBirth` (Date or LocalDate): The date of birth of the employee.
4. `address` (String or an Address object): The address of the employee.
5. `phoneNumber` (String): The phone number of the employee.
6. `email` (String): The email address of the employee.
7. `department` (String or a Department object): The department the employee belongs to.
8. `designation` (String): The job title or designation of the employee.
9. `dateOfJoining` (Date or LocalDate): The date when the employee joined the company.
10. `salary` (double or Big Decimal): The base salary of the employee.
11. `paymentMethod` (String or an enum): The payment method (e.g., bank transfer, check, cash) for the employee's salary.
12. `bankDetails` (BankDetails object): The bank account details for salary transfer.
13. `leaveBalance` (int): The number of remaining leave days for the employee.
14. `employeeType` (String or an enum): The type of employee (e.g., full-time, part-time, contract).
15. `taxRate` (double): The tax rate applicable to the employee's income.

3. How many constructors this class should have?

Ans:

The number of constructors the Employee class should have, it depends on the requirements and design of the system. A good practice to have at least two constructors: a default constructor (no arguments) and a parameterized constructor that takes all the necessary attributes as arguments. We can also have additional constructors that take different combinations of arguments, depending on our needs.

Constructors:

- a) `Employee()`: A default constructor that initializes the object with default values or null values.
- b) `Employee(String name, int employeeId, String department, String designation, double salary)`: A parameterized constructor that initializes the object with essential attributes.
- c) `Employee(String name, int employeeId, String department, String designation, double salary, Date dateOfJoining, Date dateOfBirth, String address, String phoneNumber, String email)`: A parameterized constructor that initializes the object with most of the attributes.

4. What are other methods this class can have?

Ans:

Methods:

- a) `calculateSalary()`: Calculates the salary for the employee based on their base salary, allowances, deductions, and other factors.
- b) `applyLeave(int daysToApply)`: Updates the employee's leave balance by deducting the requested number of days.
- c) `updatePersonalInfo(...)`: Updates the personal information (e.g., address, phone number, email) of the employee.
- d) `changeDepartment(Department newDepartment)`: Changes the department of the employee.

- e) `promoteEmployee(String newDesignation, double newSalary)`: Promotes the employee to a new designation with a new salary.
- f) `terminateEmployee(Date terminationDate)`: Terminates the employee's employment and updates their status accordingly.
- g) `generatePayslip()`: Generates a payslip for the employee with details like salary, deductions, and net pay.
- h) `calculateTaxDeduction()`: Calculates the tax deduction based on the employee's salary and applicable tax rates.
- i) `updateBankDetails(BankDetails newBankDetails)`: Updates the employee's bank account details for salary transfer.
- j) `requestLeave(int daysToRequest)`: Requests a specific number of leave days for the employee.

5. Briefly explain the purpose of the following line of code:

`Employee employee1 = new Employee();`

Ans:

The line of code `Employee employee1 = new Employee();` serves the following purposes:

- a) **Object Declaration**: It declares a variable named `employee1` of the `Employee` class type. This means that `employee1` is a reference variable that can hold an object of the `Employee` class.
- b) **Object Instantiation**: The `new Employee();` part of the statement creates a new instance (object) of the `Employee` class and assigns it to the `employee1` variable. This is known as object instantiation.
- c) **Memory Allocation**: When the `new Employee();` expression is executed, the Java Virtual Machine (JVM) allocates memory for the new `Employee` object and initializes its instance variables with their default values.
- d) **Reference Assignment**: The newly created `Employee` object is then assigned to the `employee1` variable, which acts as a reference to the object. This allows you to access the object's properties and methods using the `employee1` reference.

6. Create new project, package and create this class with all variables, constructors, getter, setter and other methods (to print the details of object and to calculate the pay of employee).

Ans:

`Employee.java`

`package PayrollSystemPack;`

`import java.time.LocalDate;`

`public class Employee {`

`private String name;`

`private int employeeId;`

`private LocalDate dateOfBirth;`

`private String address;`

`private String phoneNumber;`

```
private String email;

private String department;

private String designation;

private LocalDate dateOfJoining;

private double salary;

private PaymentMethod paymentMethod;

private BankDetails bankDetails;

private int leaveBalance;

private EmployeeType employeeType;

private double taxRate;
```

```
// Default constructor
```

```
public Employee() {

}
```

```
// Parameterized constructor with essential attributes
```

```
public Employee(String name, int employeeId, String department, String designation, double salary) {

    this.name = name;

    this.employeeId = employeeId;

    this.department = department;

    this.designation = designation;

    this.salary = salary;

}
```

```
// Parameterized constructor with most attributes
```

```
public Employee(String name, int employeeId, LocalDate dateOfBirth, String address, String
phoneNumber, String email, String department, String designation, LocalDate dateOfJoining, double salary,
PaymentMethod paymentMethod, BankDetails bankDetails, int leaveBalance, EmployeeType employeeType,
double taxRate) {

    this.name = name;

    this.employeeId = employeeId;

    this.dateOfBirth = dateOfBirth;

    this.address = address;
```

```
this.phoneNumber = phoneNumber;

this.email = email;

this.department = department;

this.designation = designation;

this.dateOfJoining = dateOfJoining;

this.salary = salary;

this.paymentMethod = paymentMethod;

this.bankDetails = bankDetails;

this.leaveBalance = leaveBalance;

this.employeeType = employeeType;

this.taxRate = taxRate;
}
```

// Getters and Setters

```
public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }
```

```
public int getEmployeeId() { return employeeId; }
```

```
public void setEmployeeId(int employeeId) { this.employeeId = employeeId; }
```

```
public LocalDate getDateOfBirth() { return dateOfBirth; }
```

```
public void setDateOfBirth(LocalDate dateOfBirth) { this.dateOfBirth = dateOfBirth; }
```

```
public String getAddress() { return address; }
```

```
public void setAddress(String address) { this.address = address; }
```

```
public String getPhoneNumber() { return phoneNumber; }
```

```
public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }
```

```
public String getEmail() { return email; }
```

```
public void setEmail(String email) { this.email = email; }
```

```
public String getDepartment() { return department; }

public void setDepartment(String department) { this.department = department; }


public String getDesignation() { return designation; }

public void setDesignation(String designation) { this.designation = designation; }


public LocalDate getDateOfJoining() { return dateOfJoining; }

public void setDateOfJoining(LocalDate dateOfJoining) { this.dateOfJoining = dateOfJoining; }


public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }


public PaymentMethod getPaymentMethod() { return paymentMethod; }

public void setPaymentMethod(PaymentMethod paymentMethod) { this.paymentMethod =
paymentMethod; }


public BankDetails getBankDetails() { return bankDetails; }

public void setBankDetails(BankDetails bankDetails) { this.bankDetails = bankDetails; }


public int getLeaveBalance() { return leaveBalance; }

public void setLeaveBalance(int leaveBalance) { this.leaveBalance = leaveBalance; }


public EmployeeType getEmployeeType() { return employeeType; }

public void setEmployeeType(EmployeeType employeeType) { this.employeeType = employeeType; }


public double getTaxRate() { return taxRate; }

public void setTaxRate(double taxRate) { this.taxRate = taxRate; }


public double calculateSalary() {

    // Logic to calculate salary based on base salary, allowances, deductions, etc.

    return salary;

}
```

```
public void applyLeave(int daysToApply) {  
    leaveBalance -= daysToApply;  
}
```

```
public void updatePersonalInfo(String address, String phoneNumber, String email) {  
    this.address = address;  
    this.phoneNumber = phoneNumber;  
    this.email = email;  
}
```

```
public void changeDepartment(String newDepartment) {  
    this.department = newDepartment;  
}
```

```
public void promoteEmployee(String newDesignation, double newSalary) {  
    this.designation = newDesignation;  
    this.salary = newSalary;  
}
```

```
public void terminateEmployee(LocalDate terminationDate) {  
    // Logic to terminate the employee and update status  
}
```

```
public void generatePayslip() {  
    // Logic to generate payslip with salary, deductions, and net pay  
    System.out.println("Payslip for Employee: " + name);  
    System.out.println("Employee ID: " + employeeId);  
    System.out.println("Department: " + department);  
    System.out.println("Designation: " + designation);  
    System.out.println("Salary: " + salary);  
    // ... (additional payslip details)
```

```
}
```

```
public double calculateTaxDeduction() {  
    // Logic to calculate tax deduction based on salary and tax rate  
    return salary * taxRate;  
}
```

```
public void updateBankDetails(BankDetails newBankDetails) {  
    this.bankDetails = newBankDetails;  
}
```

```
public void requestLeave(int daysToRequest) {  
    if (daysToRequest <= leaveBalance) {  
        applyLeave(daysToRequest);  
        System.out.println("Leave request approved. Remaining leave balance: " + leaveBalance);  
    } else {  
        System.out.println("Insufficient leave balance. Request denied.");  
    }  
}
```

```
public void printEmployeeDetails() {  
    System.out.println("Employee Details:");  
    System.out.println("Name: " + name);  
    System.out.println("Employee ID: " + employeeId);  
    System.out.println("Date of Birth: " + dateOfBirth);  
    System.out.println("Address: " + address);  
    System.out.println("Phone Number: " + phoneNumber);  
    System.out.println("Email: " + email);  
    System.out.println("Department: " + department);  
    System.out.println("Designation: " + designation);  
    System.out.println("Date of Joining: " + dateOfJoining);  
    System.out.println("Salary: " + salary);  
}
```



```
        System.out.println("Payment Method: " + paymentMethod);

        System.out.println("Bank Details: " + bankDetails);

        System.out.println("Leave Balance: " + leaveBalance);

        System.out.println("Employee Type: " + employeeType);

        System.out.println("Tax Rate: " + taxRate);

    }
}
```

```
    public double calculateNetPay() {

        double grossPay = calculateSalary();

        double taxDeduction = calculateTaxDeduction();

        double netPay = grossPay - taxDeduction;

        return netPay;

    }

}
```

EmployeeType.java

```
package PayrollSystemPack;

public enum EmployeeType {

    FULL_TIME,

    PART_TIME,

    CONTRACT,

    INTERN

}
```

PaymentMethod.java

```
package PayrollSystemPack;

public enum PaymentMethod {

    BANK_TRANSFER,

    DIRECT_DEPOSIT,

    CHECK,

    CASH

}
```

BankDetails.java

```
package PayrollSystemPack;
```

```
public class BankDetails {  
  
    private String bankName;  
  
    private String accountNumber;  
  
    private String accountType;  
  
  
    public BankDetails(String bankName, String accountNumber, String accountType) {  
  
        this.bankName = bankName;  
  
        this.accountNumber = accountNumber;  
  
        this.accountType = accountType;  
  
    }  
  
  
    public String getBankName() {  
  
        return bankName;  
  
    }  
  
  
    public void setBankName(String bankName) {  
  
        this.bankName = bankName;  
  
    }  
  
  
    public String getAccountNumber() {  
  
        return accountNumber;  
  
    }  
  
  
    public void setAccountNumber(String accountNumber) {  
  
        this.accountNumber = accountNumber;  
  
    }  
  
  
    public String getAccountType() {  
  
        return accountType;  
  
    }  
  
}
```

```
public void setAccountType(String accountType) {
```

```
    this.accountType = accountType;
```

```
}
```

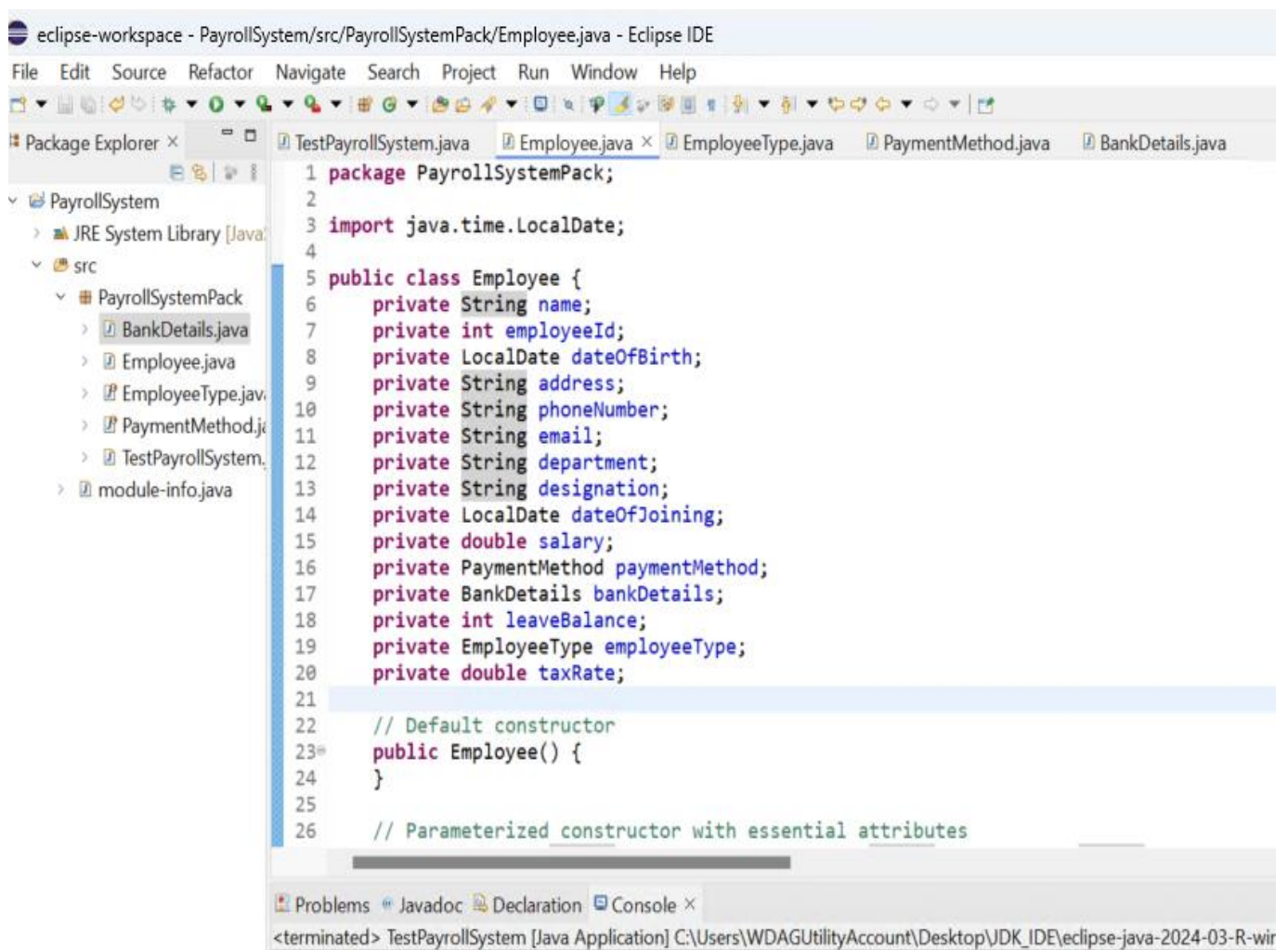
@Override

```
public String toString() {
```

```
    return "Bank: " + bankName + ", Account: " + accountNumber + " (" + accountType + ")";
```

```
}
```

```
}
```



eclipse-workspace - PayrollSystem/src/PayrollSystemPack/Employee.java - Eclipse IDE

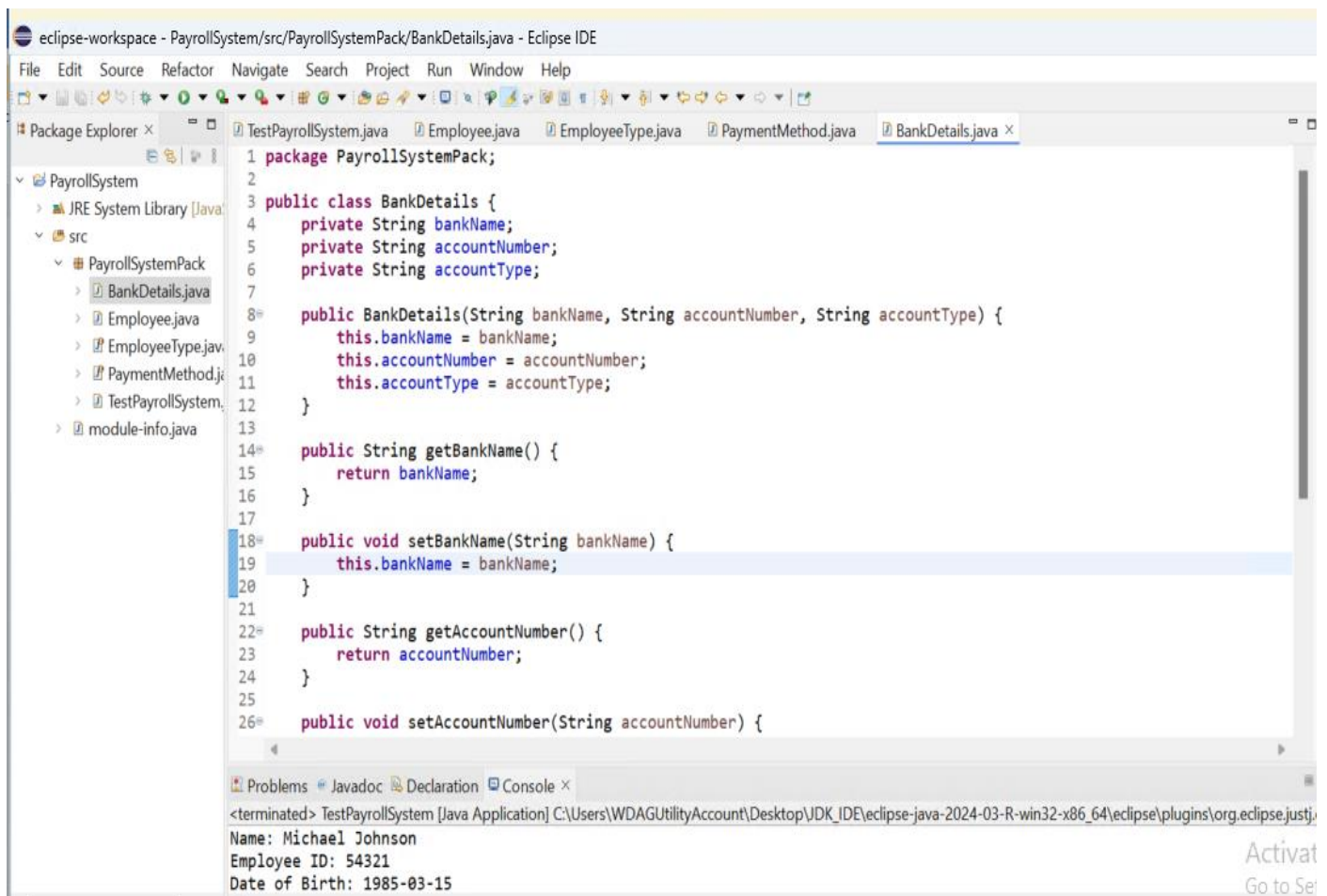
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer × TestPayrollSystem.java Employee.java EmployeeType.java PaymentMethod.java BankDetails.java

```
1 package PayrollSystemPack;
2
3 import java.time.LocalDate;
4
5 public class Employee {
6     private String name;
7     private int employeeId;
8     private LocalDate dateOfBirth;
9     private String address;
10    private String phoneNumber;
11    private String email;
12    private String department;
13    private String designation;
14    private LocalDate dateOfJoining;
15    private double salary;
16    private PaymentMethod paymentMethod;
17    private BankDetails bankDetails;
18    private int leaveBalance;
19    private EmployeeType employeeType;
20    private double taxRate;
21
22    // Default constructor
23    public Employee() {
24    }
25
26    // Parameterized constructor with essential attributes
```

Problems Javadoc Declaration Console ×

<terminated> TestPayrollSystem [Java Application] C:\Users\WDAGUtilityAccount\Desktop\JDK_IDE\eclipse-java-2024-03-R-wir



7. Create a test class which will have main method in it and will do as follows:

- Create five different objects of Employee class using all constructors. One should be created at least using non-parametrised constructor.
- For objects created with non-parametrised constructor, use setter to set the values and display the details.
- For all other objects, display details of all employees using other method.

TestPayrollSystem.java

```

package PayrollSystemPack;

import java.time.LocalDate;

public class TestPayrollSystem {

    public static void main(String[] args) {

        // Create an Employee object using the default constructor

        Employee employee1 = new Employee();

        employee1.setName("John Doe");

        employee1.setEmployeeId(12345);

        // Set other attributes using setters

```

```

employee1.printEmployeeDetails();

// Create an Employee object using the parameterized constructor with essential attributes
Employee employee2 = new Employee("Jane Smith", 67890, "IT", "Software Developer", 6000.0);
employee2.printEmployeeDetails();

// Create an Employee object using the parameterized constructor with most attributes
BankDetails bankDetails = new BankDetails("ABC Bank", "123456789", "Checking");

Employee employee3 = new Employee("Michael Johnson", 54321, LocalDate.of(1985, 3, 15), "123 Main St",
"555-9876", "michael.johnson@email.com", "Marketing", "Marketing Manager", LocalDate.of(2015, 7, 1), 8000.0,
PaymentMethod.BANK_TRANSFER, bankDetails, 20, EmployeeType.FULL_TIME, 0.25);

employee3.printEmployeeDetails();

// Create two more Employee objects using different constructors
Employee employee4 = new Employee("Emily Davis", 24680, "HR", "HR Coordinator", 4500.0);
employee4.printEmployeeDetails();

BankDetails bankDetails2 = new BankDetails("XYZ Bank", "987654321", "Savings");

Employee employee5 = new Employee("David Wilson", 13579, LocalDate.of(1990, 11, 20), "456 Oak St", "555-
2468", "david.wilson@email.com", "Finance", "Accountant", LocalDate.of(2018, 2, 15), 6500.0,
PaymentMethod.DIRECT_DEPOSIT, bankDetails2, 15, EmployeeType.PART_TIME, 0.18);

employee5.printEmployeeDetails();

// Calculate and print net pay for each employee
System.out.println("\nNet Pay for Employees:");

System.out.println("Employee 1 (John Doe): " + employee1.calculateNetPay());
System.out.println("Employee 2 (Jane Smith): " + employee2.calculateNetPay());
System.out.println("Employee 3 (Michael Johnson): " + employee3.calculateNetPay());
System.out.println("Employee 4 (Emily Davis): " + employee4.calculateNetPay());
System.out.println("Employee 5 (David Wilson): " + employee5.calculateNetPay());
}
}

```

eclipse-workspace - PayrollSystem/src/PayrollSystemPack/BankDetails.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

- PayrollSystem
 - JRE System Library [Java]
 - src
 - PayrollSystemPack
 - BankDetails.java
 - Employee.java
 - EmployeeType.java
 - PaymentMethod.java
 - TestPayrollSystem.java
 - module-info.java

```
1 package PayrollSystemPack;
2
3 public class BankDetails {
4     private String bankName;
5     private String accountNumber;
6     private String accountType;
7
8     public BankDetails(String bankName, String accountNumber, String accountType) {
9         this.bankName = bankName;
10        this.accountNumber = accountNumber;
11        this.accountType = accountType;
12    }
13
14    public String getBankName() {
15        return bankName;
16    }
17
18    public void setBankName(String bankName) {
19        this.bankName = bankName;
20    }
21
22    public String getAccountNumber() {
23        return accountNumber;
24    }
25}
```

Problems Javadoc Declaration Console ×

<terminated> TestPayrollSystem [Java Application] C:\Users\WDAGUtilityAccount\Desktop\JDK_IDE\eclipse-java-2024-03-R-win

Employee Details:
Name: John Doe
Employee ID: 12345
Date of Birth: null
Address: null
Phone Number: null

Output:

eclipse-workspace - PayrollSystem/src/PayrollSystemPack/BankDetails.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

- PayrollSystem
 - JRE System Library [Java]
 - src
 - PayrollSystemPack
 - BankDetails.java
 - Employee.java
 - EmployeeType.java
 - PaymentMethod.java
 - TestPayrollSystem.java
 - module-info.java

```
1 package PayrollSystemPack;
2
3 public class BankDetails {
4     private String bankName;
5     private String accountNumber;
6     private String accountType;
7
8     public BankDetails(String bankName, String accountNumber, String accountType) {
9         this.bankName = bankName;
10        this.accountNumber = accountNumber;
11        this.accountType = accountType;
12    }
13
14    public String getBankName() {
15        return bankName;
16    }
17
18    public void setBankName(String bankName) {
19        this.bankName = bankName;
20    }
21
22    public String getAccountNumber() {
23        return accountNumber;
24    }
25}
```

Problems Javadoc Declaration Console ×

<terminated> TestPayrollSystem [Java Application] C:\Users\WDAGUtilityAccount\Desktop\JDK_IDE\eclipse-java-2024-03-R-win

Name: Michael Johnson
Employee ID: 54321
Date of Birth: 1985-03-15
Address: 123 Main St
Phone Number: 555-9876
Email: michael.johnson@email.com
Department: Marketing
Designation: Marketing Manager
Date of Joining: 2015-07-01
Salary: 8000.0
Payment Method: BANK_TRANSFER
Bank Details: Bank: ABC Bank, Account: 123456789 (Checking)
Leave Balance: 20
Employee Type: FULL_TIME
Tax Rate: 0.25
Employee Details:
Name: Emily Davis
Employee ID: 24680
Date of Birth: null
Address: null
Phone Number: null
Email: null
Department: HR
Designation: HR Coordinator
Date of Joining: null
Salary: 4500.0

8. Vaccine Class

Vaccine
<div>- VaccineID: int</div> <div>- VaccineName: String</div> <div>- manufacturer: String</div> <div>- expiryDate: String</div>
<div>+ Vaccine()</div> <div>+ Vaccine(int,String, String, String)</div> <div>+ getVaccinationId(): int</div> <div>+ setVaccinationId(int)</div> <div>+ getVaccineName(): String</div> <div>+ setVaccineName(String)</div> <div>+ getManufacturer(): String</div> <div>+ setManufacturer(String)</div> <div>+ getExpiryDate(): String</div> <div>+ setExpiryDate(String)</div> <div>+ toString(): String</div>

a. How many constructors does the class Vaccine provide?

Ans: The Vaccine class provides two constructors:

- a) A default constructor: Vaccine()
- b) A parameterized constructor: Vaccine(int, String, String, String)

b. How many methods does the class Vaccine provide?

Ans: The Vaccine class provides the following 9 methods:

- 1. Vaccine() (default constructor)
- 2. Vaccine(int, String, String, String) (parameterized constructor)
- 3. getVaccinationId(): int
- 4. setVaccinationId(int)
- 5. getVaccineName(): String
- 6. setVaccineName(String)
- 7. getManufacturer(): String
- 8. setManufacturer(String)
- 9. getExpiryDate(): String
- 10. setExpiryDate(String)

11. toString(): String

c. How many instance variables does the class Vaccine have?

Ans: The Vaccine class has 4 instance variables:

1. VaccineID: int
2. VaccineName: String
3. manufacturer: String
4. expiryDate: String

d. Briefly explain the purpose of the following two lines of code:

```
Vaccine Pfizer;  
Pfizer = new Vaccine();
```

Ans:

The first line `Vaccine Pfizer;` declares a variable named `Pfizer` of type `Vaccine`. This variable can hold an object of the `Vaccine` class.

The second line `Pfizer = new Vaccine();` creates a new instance of the `Vaccine` class using the default constructor and assigns it to the `Pfizer` variable.

e. The two lines of code can be combined into one. Provide this one line of code:

Ans: The two lines of code can be combined into one line: `Vaccine Pfizer = new Vaccine();`

This line declares a variable named `Pfizer` of type `Vaccine` and immediately creates a new instance of the `Vaccine` class using the default constructor and assigns it to the `Pfizer` variable.

f. What method is used to set the VaccineID of an instance of the Vaccine class? What data type does it expect as an argument?

Ans: The method used to set the VaccineID of an instance of the Vaccine class is `setVaccinationId(int)`. It expects an `int` value as an argument.

g. What method is used to set the VaccineName of an instance of the Vaccine class? What data type does it expect as an argument?

Ans: The method used to set the VaccineName of an instance of the Vaccine class is `setVaccineName(String)`. It expects a `String` value as an argument.