

***Java SE 8 Programming Language***

**Lab Guides**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 06/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
|  | 01/Oct/2018 | Create new | Create new | DieuNT1 | VinhNV |
|  | 01/Jun/2019 | Update template | Fsoft template | DieuNT1 | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Unit 5: Basic OOP 4](#_Toc11661583)

[Lab Guide 1: Abstract class, Inheritance, Polymorphism 4](#_Toc11661584)

[Objectives: JPL-9 4](#_Toc11661585)

|  |  |
| --- | --- |
|  | **CODE: JPL.M.L101**  **TYPE: Medium**  **LOC:**  **DURATION: 60 MINUTES** |

# Unit 5: Basic OOP

## Lab Guide 1: Abstract class, Inheritance, Polymorphism

## Objectives: JPL-9

* Able to create Java-based applications that take advantage of Java object-oriented features, including encapsulation, inheritance, and polymorphism.
* Understand the static components in java

## Problem Descriptions:

Create a new package named **fa.training.abstraction** in **JPL.M.L101** project.

* Create an abstract class **Employee** with: *employee name*, *date of birth*, *address, company name*, **calcSalary**() abstract method, **inputData**() method to input employee information, **display**() method to show all information. This class contains constructor and getter/setter if needs.
* There are three employee type:
  + Production staff: amount of product, salary = amount of product \* 20$;
  + Daily staff: number of workdays, salary = number of workdays \* 15$;
  + Manager: basic salary, wage, salary = basic salary \* wage;
* Create class **ProductionStaff**, **DailyStaff** and **Manager** that extends **Employee** class. Override **calcSalary** to calculate salary for each employee type.
* Create another class named **EmployeeManagement** that creates an array contains 3 employees of above type, loop to calls the **display**() method to print the outputs.

## Screen Designs:

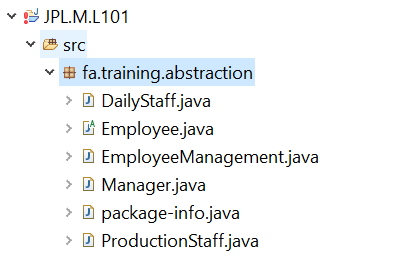
|  |  |
| --- | --- |
| *Screen 1: input data* | *Screen 2: display data* |

## Functional Requirements:

* Run the program and explains result.
* You change the program by adds validation methods to check input data before assign to objects. Demo with n employees.

## Guidelines:

Project struture:



* **ProductStaff** class

1. **package** fa.training.abstraction;
2. **import** java.util.Scanner;
3. **public** **class** ProductionStaff **extends** Employee {
4. **private** **static** **final** **int** ***UNIT\_PRICE*** = 20;
5. **private** **double** amountOfProduct;
6. @Override
7. **public** **double** calcSalary() {
8. **return** amountOfProduct \* ***UNIT\_PRICE***;
9. }
10. @Override
11. **protected** **void** inputData(Scanner scanner) {
12. /\*
13. \* Call inputData method from parent class.
14. \*/
15. **super**.inputData(scanner);
16. System.***out***.println("Enter amount of product: ");
17. amountOfProduct = Double.*parseDouble*(scanner.nextLine());
18. System.***out***.println("---------------------------");
19. }
20. @Override
21. **protected** **void** display() {
22. // Call method of parent class
23. **super**.display();
24. System.***out***.print("\t" + amountOfProduct + "\t" + **this**.calcSalary() + "\n");
25. }
26. }

* **Employee** class

1. **package** fa.training.abstraction;
2. **import** java.util.Scanner;
3. **public** **abstract** **class** Employee {
4. **private** String employeeName;
5. **private** String dateOfBirth;
6. **private** String address;
7. **private** **static** String *companyName*;
8. **public** **static** String getCompanyName() {
9. **return** *companyName*;
10. }
11. **public** **static** **void** setCompanyName(String companyName) {
12. Employee.*companyName* = companyName;
13. }
14. /\*\*
15. \* User enter data.
16. \*
17. \* **@param** scanner
18. \*/
19. **protected** **void** inputData(Scanner scanner) {
20. System.***out***.println("Enter name: ");
21. employeeName = scanner.nextLine();
22. System.***out***.println("Enter birth date: ");
23. dateOfBirth = scanner.nextLine();
24. System.***out***.println("Address: ");
25. address = scanner.nextLine();
26. }
27. /\*\*
28. \* Display data to console.
29. \*/
30. **protected** **void** display() {
31. System.***out***.print(employeeName + "\t" + dateOfBirth + "\t" + address +
32. "\t" + *companyName*);
33. }
34. // getter/setter method
35. **public** String getEmployeeName() {
36. **return** employeeName;
37. }
38. **public** **void** setEmployeeName(String employeeName) {
39. **this**.employeeName = employeeName;
40. }
41. **public** String getDateOfBirth() {
42. **return** dateOfBirth;
43. }
44. **public** **void** setDateOfBirth(String dateOfBirth) {
45. **this**.dateOfBirth = dateOfBirth;
46. }
47. **public** String getAddress() {
48. **return** address;
49. }
50. **public** **void** setAddress(String address) {
51. **this**.address = address;
52. }
53. **public** **abstract** **double** calcSalary();
54. }

* **DailyStaff** class

1. **package** fa.training.abstraction;
2. **import** java.util.Scanner;
3. **public** **class** DailyStaff **extends** Employee {
4. **private** **static** **final** **int** ***WAGE\_DAY*** = 15;
5. **private** **double** numbeOfWorkday;
6. @Override
7. **public** **double** calcSalary() {
8. **return** numbeOfWorkday \* ***WAGE\_DAY***;
9. }
10. @Override
11. **protected** **void** inputData(Scanner scanner) {
12. /\*
13. \* Call inputData method from parent class.
14. \*/
15. **super**.inputData(scanner);
16. System.***out***.println("Enter number of workday: ");
17. numbeOfWorkday = Double.*parseDouble*(scanner.nextLine());
18. System.***out***.println("---------------------------");
19. }
20. @Override
21. **protected** **void** display() {
22. // Call method of parent class
23. **super**.display();
24. System.***out***.print("\t" + numbeOfWorkday + "\t" + **this**.calcSalary() + "\n");
25. }
26. }

* **Manager** class

1. **package** fa.training.abstraction;
2. **import** java.util.Scanner;
3. **public** **class** Manager **extends** Employee {
4. **private** **double** wage;
5. **private** **double** basicSalary;
6. @Override
7. **public** **double** calcSalary() {
8. **return** wage \* basicSalary;
9. }
10. @Override
11. **protected** **void** inputData(Scanner scanner) {
12. /\*
13. \* Call inputData method from parent class.
14. \*/
15. **super**.inputData(scanner);
16. System.***out***.println("Enter wage: ");
17. wage = Double.*parseDouble*(scanner.nextLine());
18. System.***out***.println("Enter basic salary: ");
19. basicSalary = Double.*parseDouble*(scanner.nextLine());
20. System.***out***.println("---------------------------");
21. }
22. @Override
23. **protected** **void** display() {
24. // Call method of parent class
25. **super**.display();
26. System.***out***.print("\t" + wage + "\t" + basicSalary + "\t" +
27. **this**.calcSalary() + "\n");
28. }
29. }

* **EmployeeManagement** class

1. **package** fa.training.abstraction;
2. **import** java.util.Scanner;
3. **public** **class** EmployeeManagement {
4. **public** **static** **void** main(String[] args) {
5. Employee employees[] = **new** Employee[3];
6. // Create 3 objects
7. ProductionStaff productionStaff = **new** ProductionStaff();
8. DailyStaff dailyStaff = **new** DailyStaff();
9. Manager manager = **new** Manager();
10. Scanner scanner = **new** Scanner(System.***in***);
11. // Call inputData method
12. System.***out***.println("Employee 1");
13. productionStaff.inputData(scanner);
14. System.***out***.println("Employee 2");
15. dailyStaff.inputData(scanner);
16. System.***out***.println("Employee 3");
17. manager.inputData(scanner);
18. Employee.*setCompanyName*("FPT");
19. // Push to existed array
20. employees[0] = productionStaff;
21. employees[1] = dailyStaff;
22. employees[2] = manager;
23. // loop
24. **for** (Employee employee : employees) {
25. // An instance of Polymorphims
26. employee.display();
27. }
28. scanner.close();
29. }
30. }

**-- THE END -**