Practices for Lesson 3

Practices Overview

In these practices, you will override methods, including the `toString` method in the Object class. You will also create a method in a class that uses the `instanceof` operator to determine which object was passed to the method.

Practice 3-1: Summary Level: Overriding Methods and Applying Polymorphism

Overview

In this practice, you will override the `toString` method of the `Object` class in the `Employee` class and in the `Manager` class. You will create an `EmployeeStockPlan` class with a `grantStock` method that uses the `instanceof` operator to determine how much stock to grant based on the employee type.

Assumptions

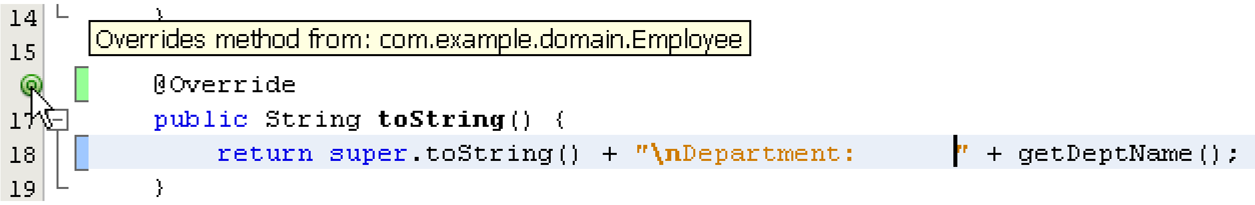
Tasks

1. Open the EmployeePractice project in the `practices` directory.
2. Edit the `Employee` class to override the `toString()` method from the `Object` class. `Object`'s `toString` method returns a `String`.
    a. Add a `return` statement that returns a string that includes the employee ID, name, Social Security number, and a salary as a formatted string, with each line separated with a newline character (`"\n"`).
    b. To format the double salary, use the following:

    `NumberFormat.getCurrencyInstance().format(getSalary())`

    c. Fix any missing import statements.
    d. Save the class.
3. Override the `toString()` method in the `Manager` class to include the `deptName` field value. Separate this string from the Employee string with a newline character.

    Note the Green circle icon with the "o" in the center beside the method signature in the `Manager` class. This indicates that NetBeans is aware that this method overrides the method from the parent class, Employee. Hold the cursor over the icon to read what this icon represents:



Overrides method from: com.example.domain.Employee

```
    }

    @Override
    public String toString() {
        return super.toString() + "\nDepartment:         " + getDeptName();
    }
```

    Click the icon, and NetBeans will open the `Employee` class and position the view to the `toString()` method.

4. (Optional) Override the `toString()` method in the `Director` class as well, to display all the fields of a Director and the available budget.

5. Create a new class called `EmployeeStockPlan` in the package `com.example.business`. This class will include a single method, `grantStock`, which takes an `Employee` object as a parameter and returns an integer number of stock options based on the employee type:

Employee Type	Number of Stock Options
Director	1000
Manager	100
All other Employees	10

   a. Add a `grantStock` method that takes an `Employee` object reference as a parameter and returns an integer

   b. In the method body, determine what employee type was passed in using the `instanceof` keyword and return the appropriate number of stock options based on that type.

   c. Resolve any missing import statements.

   d. Save the `EmployeeStockPlan` class.

6. Modify the `EmployeeTest` class. Replace the four print statements in the `printEmployee` method with a single print statement that uses the `toString` method that you created.

7. Overload the `printEmployee` method to take a second parameter, `EmployeeStockPlan`, and print out the number of stock options that this employee will receive.

   a. Above the `printEmployee` method calls in the main method, create an instance of the `EmployeeStockPlan` and pass that instance to each of the `printEmployee` methods.

   b. The new `printEmployee` method should call the first `printEmployee` method and the number of stocks granted to this employee:

```
printEmployee (emp);
System.out.println("Stock Options:    " + esp.grantStock(emp));
```

8. Save the `EmployeeTest` class and run the application. You should see output for each employee that includes the number of Stock Options, such as:

```
Employee id:          101
Employee name:        Jane Smith
Employee Soc Sec #:   012-34-5678
Employee salary:      $120,345.27
Stock Options:        10
```

9. It would be nice to know what type of employee each employee is. Add the following to your original `printEmployee` method above the print statement that prints the employee data fields:

   ```
   System.out.println("Employee type:       " +
   emp.getClass().getSimpleName());
   ```

   This will print out the simple name of the class (`Manager`, `Engineer`, etc). The output of the first employee record should now look like this:

   ```
   Employee type:      Engineer
   Employee id:        101
   Employee name:      Jane Smith
   Employee Soc Sec #: 012-34-5678
   Employee salary:    $120,345.27
   Stock Options:      10
   ```

Practice 3-1: Detailed Level: Overriding Methods and Applying Polymorphism

Overview

In this practice, you will override the `toString` method of the `Object` class in the `Employee` class and in the `Manager` class. You will create an `EmployeeStockPlan` class with a `grantStock` method that uses the `instanceof` operator to determine how much stock to grant based on the employee type.

Assumptions

Tasks

1. Open the EmployeePractice project in the `practices` directory.
   a. Select File > Open Project.
   b. Browse to `D:\labs\03\practices (or your other directory).`
   c. Select `EmployeePractice` and click Open Project.
2. Edit the `Employee` class to override the `toString()` method from the `Object` class. `Object`'s `toString` method returns a `String`.
   a. Add the `toString` method to the `Employee` class with the following signature:

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

   b. Add a `return` statement that returns a string that includes the employee information: ID, name, Social Security number, and a formatted salary like this:

      ```
      return "Employee ID:      " + getEmpId() + "\n" +
             "Employee Name:    " + getName() + "\n" +
             "Employee SSN:     " + getSsn() + "\n" +
             "Employee Salary: " +
      NumberFormat.getCurrencyInstance().format(getSalary());
      ```

   c. Save the `Employee` class.
3. Override the `toString` method in the `Manager` class to include the `deptName` field value.
   a. Open the `Manager` class.
   b. Add a `toString` method with the same signature as the `Employee` `toString` method:

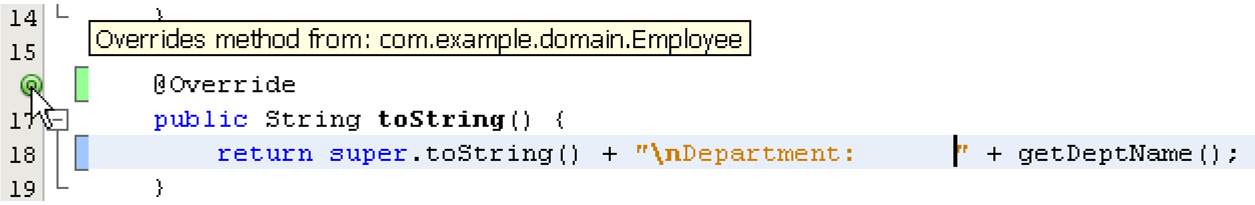
      ```
      public String toString() {
      ```

   The `toString` method in the `Manager` class overrides the `toString` method inherited from the `Employee` class.

c. Call the parent class method by using the `super` keyword and add the department name:

```
return super.toString() + "\nDepartment:        " + getDeptName();
```

Note the Green circle icon with the “o” in the center beside the method signature in the `Manager` class. This indicates that NetBeans is aware that this method overrides the method from the parent class, Employee. Hold the cursor over the icon to read what this icon represents:



Overrides method from: com.example.domain.Employee

```
    }

    @Override
    public String toString() {
        return super.toString() + "\nDepartment:        " + getDeptName();
    }
```

Click the icon, and NetBeans will open the `Employee` class and position the view to the `toString()` method.

d. Save the `Manager` class.

4. (Optional) Override the `toString` method in the `Director` class as well, to display all the fields of a director and the available budget.

5. Create a new class called `EmployeeStockPlan` in the package `com.example.business`. This class will include a single method, `grantStock`, which takes an `Employee` object as a parameter and returns an integer number of stock options based on the employee type:

Employee Type	Number of Stock Options
Director	1000
Manager	100
All other Employees	10

a. Create the new package and class in one step by right-clicking Source Package, and then selecting New > Java Class.

b. Enter `EmployeeStockPlan` as the Class Name and `com.example.business` as the Package and click Finish.

c. In the new class, add fields to the class to define the stock levels, like this:

```
    private final int employeeShares = 10;
    private final int managerShares = 100;
    private final int directorShares = 1000;
```

d. Add a `grantStock` method that takes an `Employee` object reference as a parameter and returns an integer:

```
    public int grantStock(Employee emp) {
```

   e. In the method body, determine what employee type was passed in using the `instanceof` keyword and return the appropriate number of stock options based on that type. Your code might look like this:

```
// Stock is granted based on the employee type
if (emp instanceof Director) {
    return directorShares;
} else {
    if (emp instanceof Manager) {
        return managerShares;
    } else {
        return employeeShares;
    }
}
```

   f. Resolve any missing import statements.

   g. Save the `EmployeeStockPlan` class.

6. Modify the `EmployeeTest` class. Replace the four print statements in the `printEmployee` method with a single print statement that uses the `toString` method that you created.

   a. Replace these lines:

```
System.out.println("Employee id:         " + emp.getEmpId());
System.out.println("Employee name:        " + emp.getName());
System.out.println("Employee Soc Sec #:   " + emp.getSsn());
System.out.println("Employee salary:      " +
NumberFormat.getCurrencyInstance().format((double)
emp.getSalary()));
```

   b. With one line that uses the `toString()` method:

```
System.out.println(emp);
```

7. Overload the `printEmployee` method to take a second parameter, `EmployeeStockPlan`, and print out the number of stock options that this employee will receive.

   a. Create another `printEmployee` method that takes an instance of the `EmployeeStockPlan` class:

```
public static void printEmployee(Employee emp, EmployeeStockPlan
esp) {
```

   b. This method first calls the original `printEmployee` method:

```
printEmployee(emp);
```

   c. Add a print statement to print out the number of stock options that the employee is entitled to:

```
System.out.println("Stock Options:       " +
esp.grantStock(emp));
```

d. Above the `printEmployee` method calls in the main method, create an instance of the `EmployeeStockPlan` and pass that instance to each of the `printEmployee` methods:

```
EmployeeStockPlan esp = new EmployeeStockPlan();
printEmployee(eng, esp);
... modify the remaining printEmployee invocations
```

e. Resolve any missing import statements.

8. Save the `EmployeeTest` class and run the application. You should see output for each employee that includes the number of Stock Options, such as:

```
Employee id:         101
Employee name:       Jane Smith
Employee Soc Sec #:  012-34-5678
Employee salary:     $120,345.27
Stock Options:       10
```

9. It would be nice to know what type of employee each employee is. Add the following to your original `printEmployee` method above the print statement that prints the employee data fields:

```
System.out.println("Employee type:       " +
emp.getClass().getSimpleName());
```

This will print out the simple name of the class (`Manager`, `Engineer`, etc). The output of the first employee record should now look like this:

```
Employee type:       Engineer
Employee id:         101
Employee name:       Jane Smith
Employee Soc Sec #:  012-34-5678
Employee salary:     $120,345.27
Stock Options:       10
```