

MIS 768: Advanced Software Concepts Spring 2024

Variables and Expression

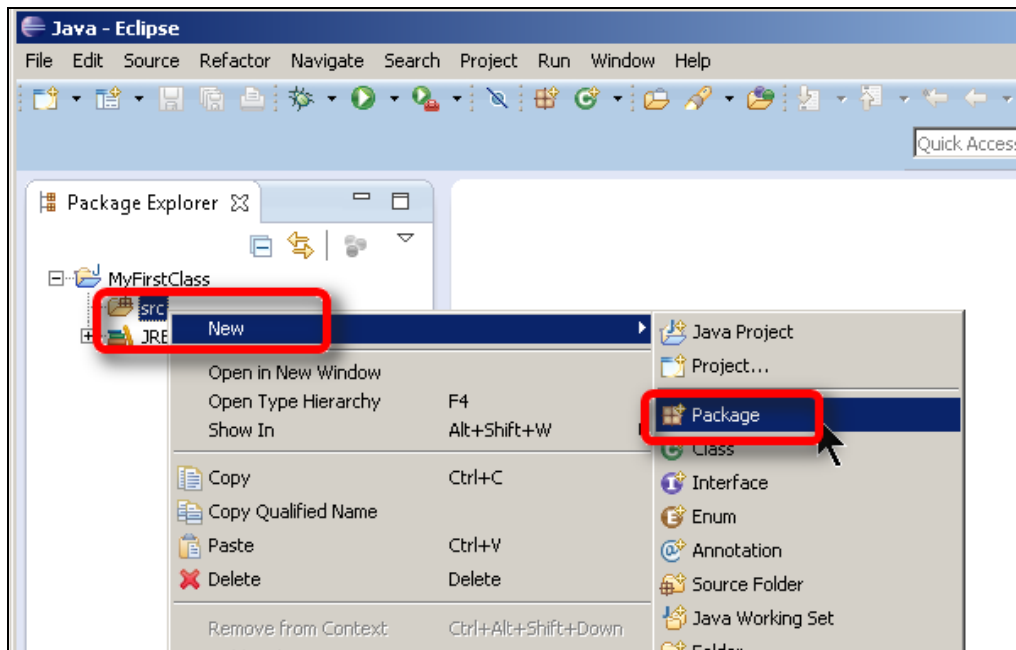
Purpose

- Learn about how to create and run a Java program in Eclipse
- Practice the basic syntax of Java

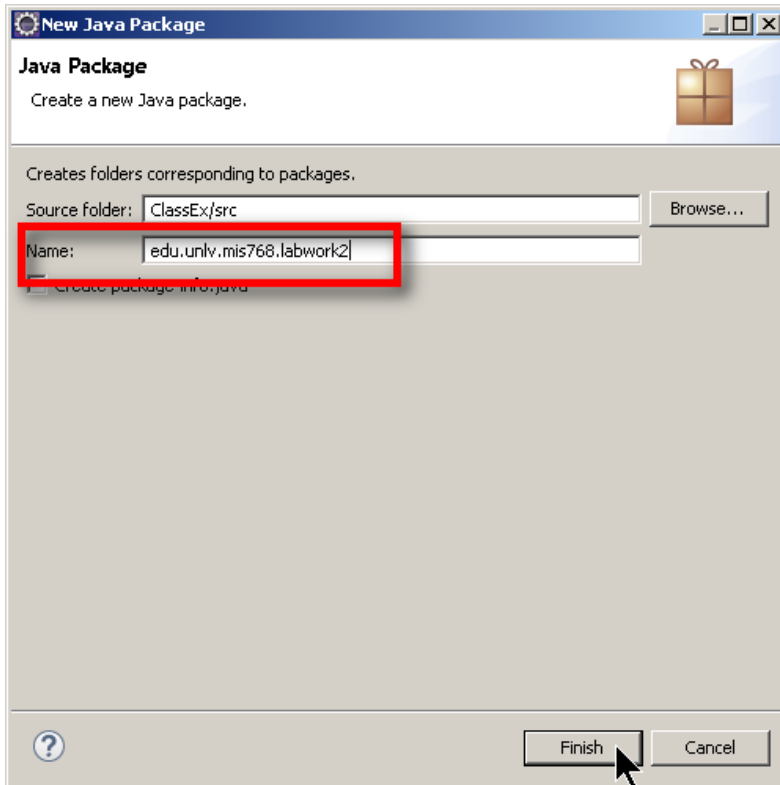
1. Create a Package

- (1) We need to create a **package** to hold our source file. A Java package contains related class files and allows you to use short-cut class names. We will talk more about packages as we go along.

To create a package, we select the folder **src** from the package navigator. Right click on the folder, and then select **New \ Package** from the popup menu.



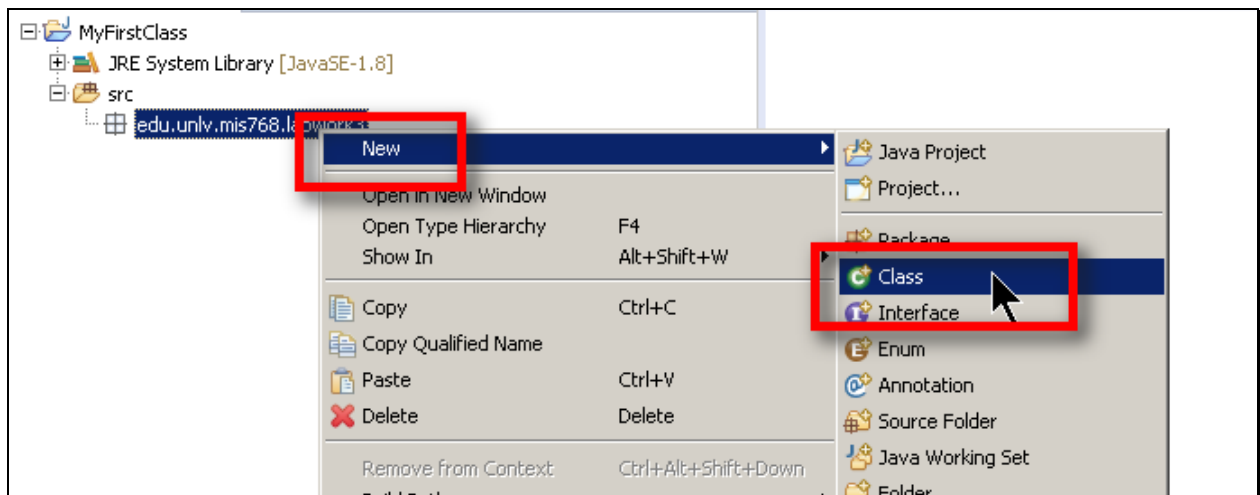
- (2) The packages should be namespaced according to the inverse of your domain name. For this course, we name the package as **edu.unlv.mis768.labwork2**. Click the **Finish** button to continue.



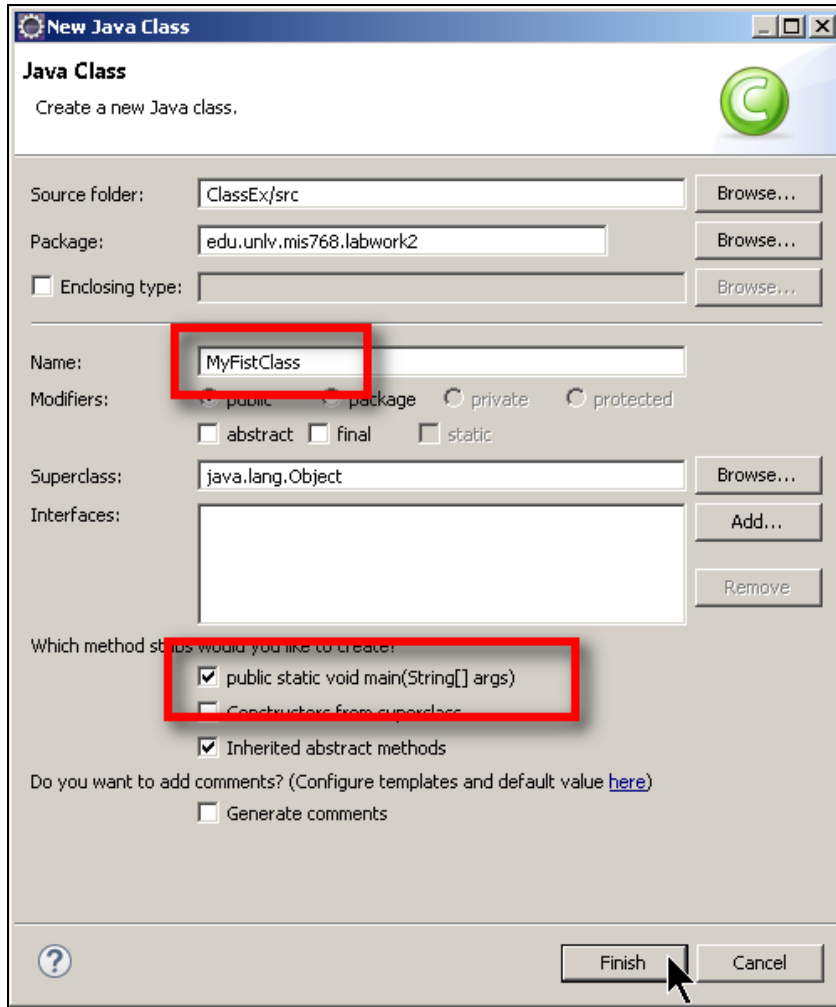
- (3) Now we got the project and the package.

2. Create Java class

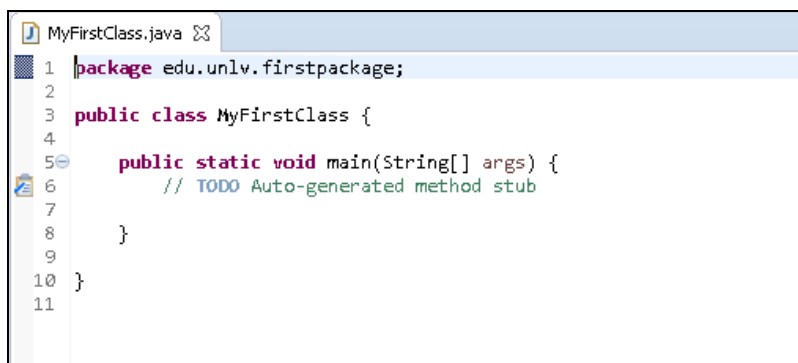
- (4) Right click on your package and select **New \ Class** from the pop-up menu.



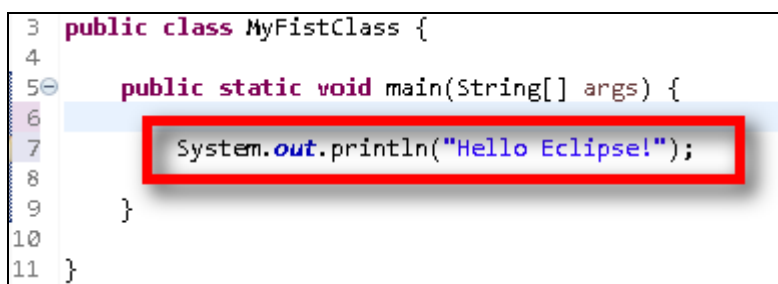
- (5) Enter **MyFirstClass** as the class name and select the **public static void main (String[] args)** option. (This is an application class). Press the **Finish** button.



- (6) This creates a new file and opens the **Editor for Java** source files.

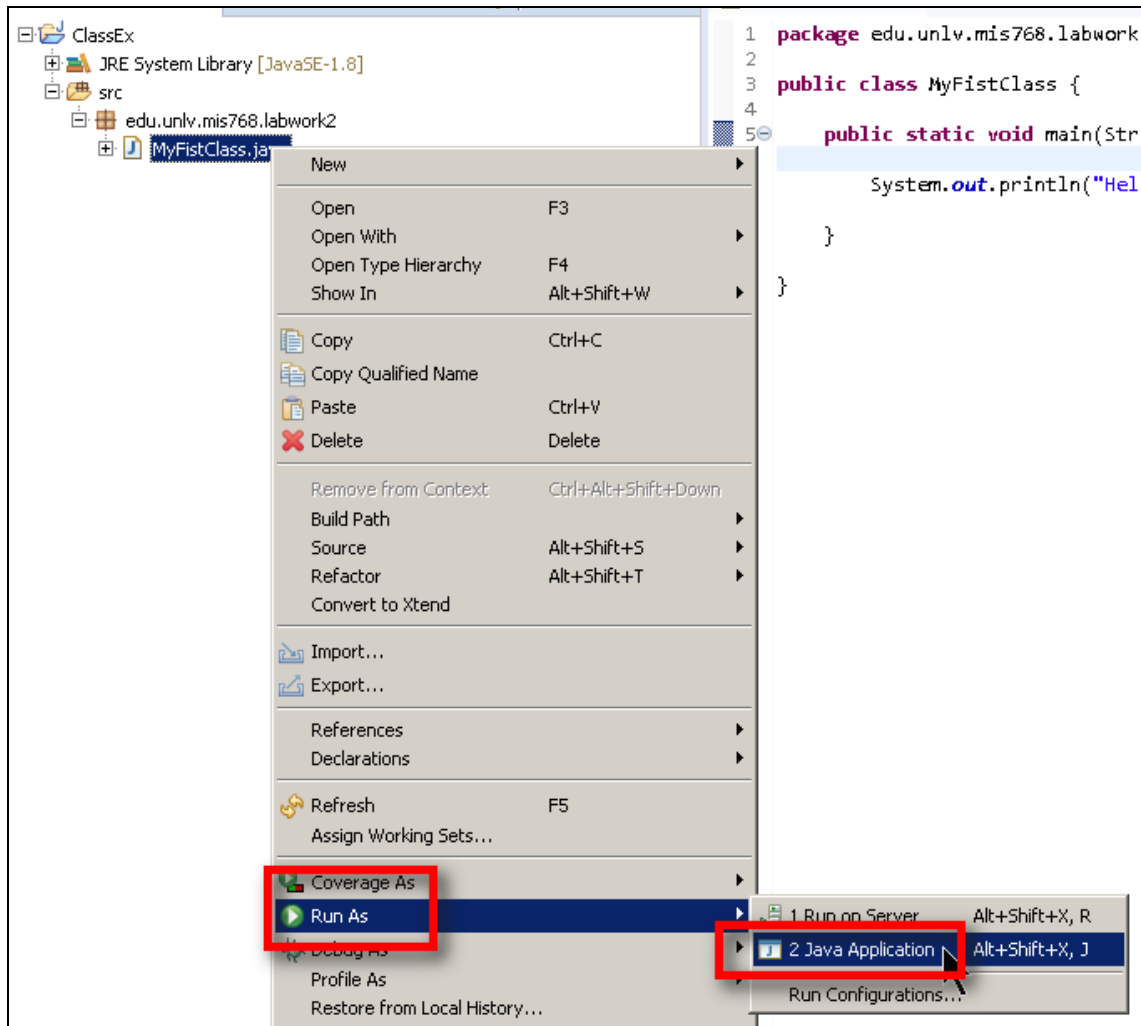


- (7) Enter the following statements for the class.

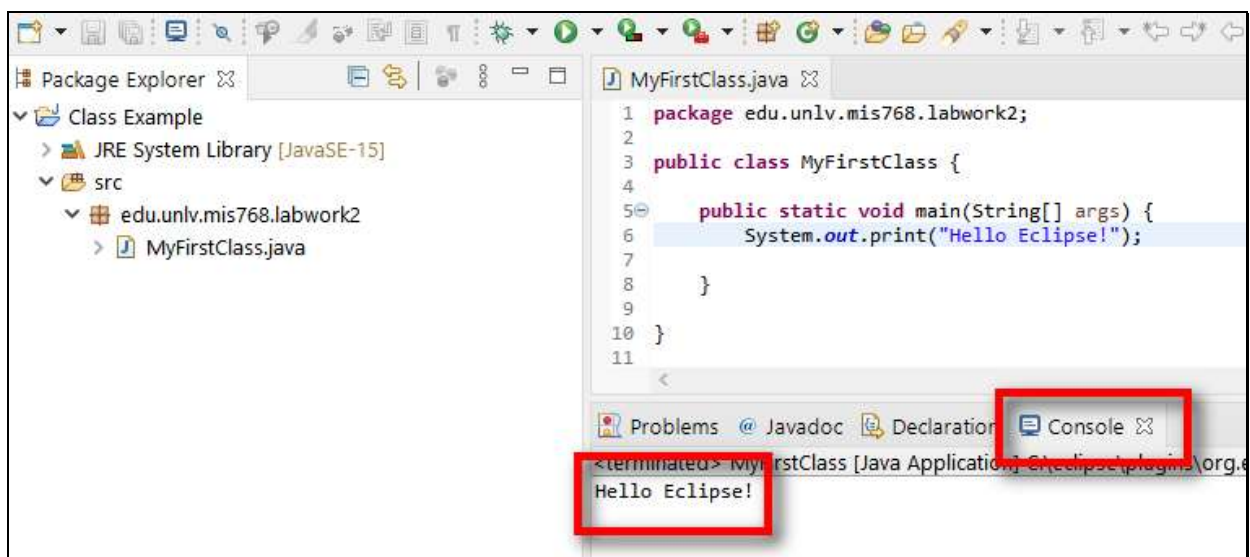


3. Run your project in Eclipse

- (8) Right-click on your Java class in the **Package Explorer** and select **Run-as \ Java application**.



- (9) You can then see the output in the **Console** view.



4. The Wage Application Example: Non-OO Way

We are going to write a program to calculate an employee's gross pay. The base pay is \$25, and the regular hour is 40 hours per week. Employees working more than 40 hours receive time and one-half for the hours over 40.

If an employee worked 57 hours this week, what is the total wage for this employee?

Now, please create another class under the **edu.unlv.mis768.labwork2**. Name it as **WageCalculator**.

- (10) Enter the following statements for the class.

```
3 public class WageCalculator {
4
5     public static void main(String[] args) {
6         // defining constants
7         final double BASE_PAY = 25; // the base pay rate
8         final double REGULAR_HOURS = 40; // the regular working hours
9
10        // defining variables
11        double workingHours = 57; // the number of working hours.
12        double totalWages; // total wage to be calculated
13
14
15
16
17    }
18
19 }
20
```

- (11) Please try to complete the program. Use **System.out.println** to print the result.

```
<terminated> WageCalculator [Java Application] C:\eclipse\plugins
The total wage is $1637.5
```

- (12) In this example, we write a Java program **without** any object-oriented concept.

5. Create Java class: Employee

- (13) Now create a new class, and name it as **Employee**. Please make sure you name this class following the naming convention. Also, this is NOT a main routine, please make sure you do not check the option “**public static void main(String[] args).**”

The screenshot shows the 'New Java Class' dialog box. The 'Name' field contains 'Employee'. The 'Package' field contains 'edu.unlv.mis768.labwork2'. The 'Superclass' field contains 'java.lang.Object'. The 'Which method stubs would you like to create?' section has 'public static void main(String[] args)' selected. The 'Finish' button is highlighted with a mouse cursor.

- (14) The **Employee** class would have the following attributes and operations.

Employee
-name : String -experiencePoint : int = 0 +BASE_PAY : int = 25 +REGULAR_HOURS : int = 40
+Employee() +getName() : String +setName(name : String) : void +getExperiencePoint() : int +earnExperiencePoint(numOfPoints : int) : void +calcSalary(numOfHours : int) : double

- (15) Please enter the following statements for the Employee class. Please make sure you follow the naming convention for the variables.

```
3 public class Employee {
4     // Fields
5     private String name; // name of the employee
6     private int experiencePoint; // experience value of the employee
7
8     // constants
9     final double BASE_PAY = 25; // the base pay rate
10    final double REGULAR_HOURS = 40; // the regular working hours
11
12    /* Constructor
13     * set the default value for experience point*/
14    public Employee() {
15        experiencePoint = 0;
16    }
17
18
19 }
```

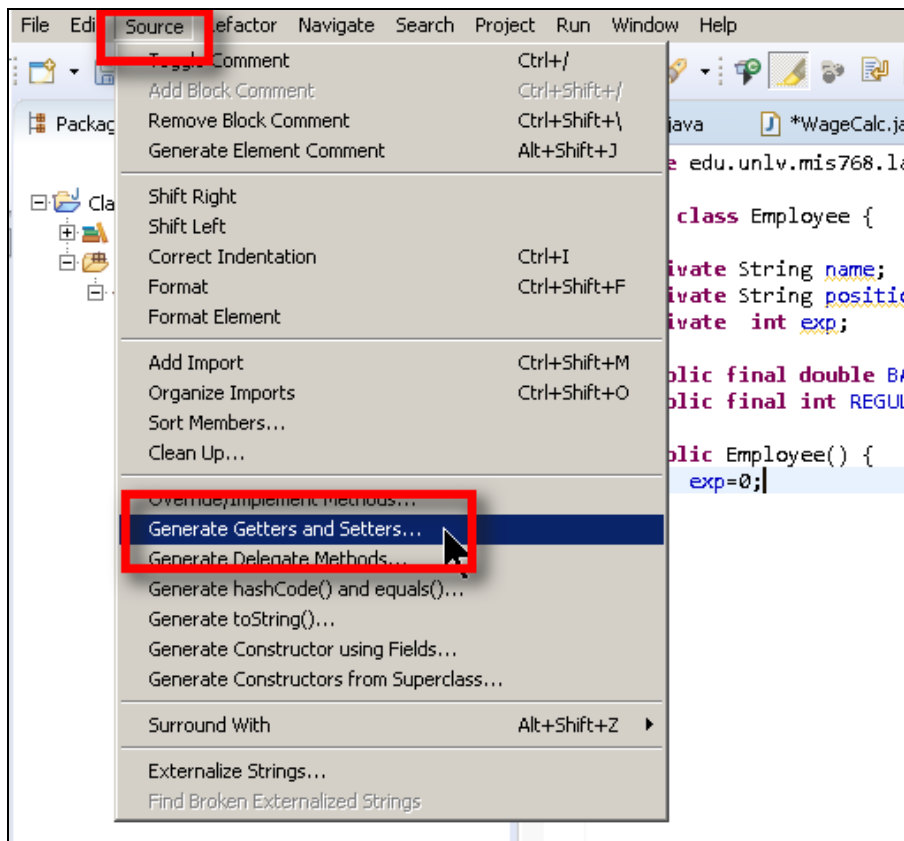
As you enter the code, you can see Eclipse provides real time feedback if you have any typos or syntax errors.

It also use different colors to highlight the reserved keywords.

When you type the open brace { and then press enter, Eclipse will automatically add the close brace } for you.

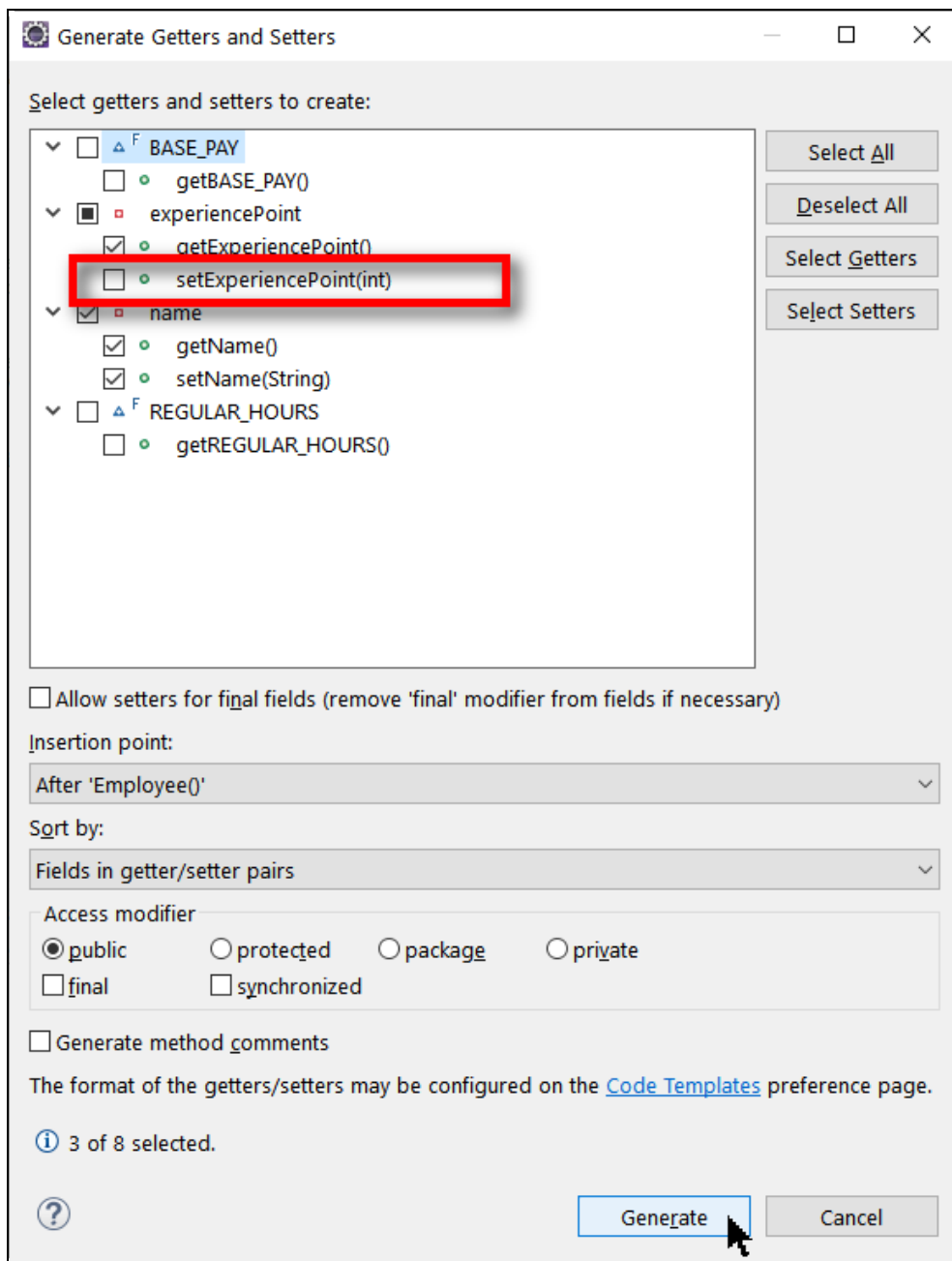
- (16) For the data fields, we need to generate the get and set methods.

Select **Source \ Generate Getters and Setters** from the menu.



- (17) Constants should be public, and therefore the **get method** is not needed for constants. Also, The value for a constant cannot be changed, and therefore **set method** is also not needed.

Select all of the variables, and then deselect **setExp(int)**. Click **OK** button.



(18) The get and set methods are then generated.

```
3 public class Employee {
4     // Fields
5     private String name; // name of the employee
6     private int experiencePoint; // experience value of the employee
7
8     // constants
9     final double BASE_PAY = 25; // the base pay rate
10    final double REGULAR_HOURS = 40; // the regular working hours
11
12    /* Constructor
13     * set the default value for experience point*/
14    public Employee() {
15        experiencePoint = 0;
16    }
17
18    public String getName() {
19        return name;
20    }
21
22    public void setName(String name) {
23        this.name = name;
24    }
25
26    public int getExperiencePoint() {
27        return experiencePoint;
28    }
29
30
31
32 }
```

(19) Now we need to implement three additional methods.

The **earnExp()** method will add the given points to the exp field. We can use the combined assignment operators here.

```
25
26    public int getExperiencePoint() {
27        return experiencePoint;
28    }
29
30    // add a certain number of points to the experience point
31    public void earnExperiencePoint(int numOfPoints) {
32        experiencePoint += numOfPoints;
33    }
34
35 }
36
```

- (20) The `calcSalary()` method will use the given number of hours and retrieve the base pay rate to calculate the salary.

```
30 // add a certain number of points to the experience point
31 public void earnExperiencePoint(int numOfPoints) {
32     experiencePoint+= numOfPoints;
33 }
34
35 // calculate salary based on the number of hours given.
36 public double calcSalary(int numOfHours) {
37     // a simplified formula. We will revise it when we talk about if-else statement
38     double salary = REGULAR_HOURS * BASE_PAY + (numOfHours-REGULAR_HOURS)*BASE_PAY*1.5;
39     return salary;
40 }
41
42 }
43
```

6. Main Routine Class: EmployeeDemo

- (21) In order to demonstrate the usage of the **Employee** class, we need to create a Java application class that starts with a main method.

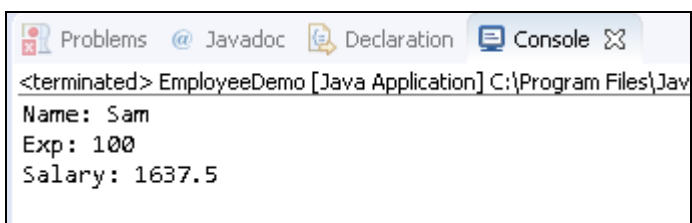
Under the same package as Employee class, create a new class **EmployeeDemo**.

Check the option “**public static void main(Strin[] args).**” Press the **Finish** button.

- (22) Enter the following statements for the class.

```
2
3 public class EmployeeDemo {
4
5     public static void main(String[] args) {
6         Employee newbie = new Employee();
7
8         // assign the name
9         newbie.setName("Sam");
10
11        // increase the exp by 100
12        newbie.earnExp(100);
13
14        // display the name, current status
15        // and the salary when this employee worked for 57 hours.
16        System.out.println("Name: "+ newbie.getName());
17        System.out.println("Exp: "+ newbie.getExp());
18        System.out.println("Salary: "+newbie.calcSalary(57));
19    }
20 }
21
22 }
```

- (23) Run this program and you can then see the output in the **Console** view.



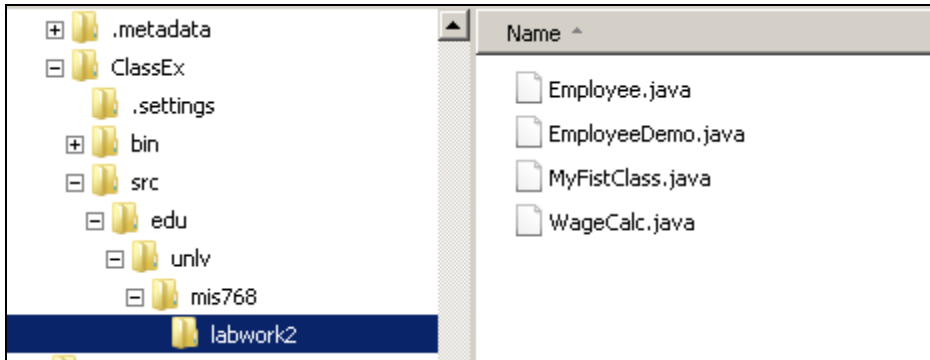
The screenshot shows the IDE's Console view with the following output:

```
<terminated> EmployeeDemo [Java Application] C:\Program Files\Jav
Name: Sam
Exp: 100
Salary: 1637.5
```

This is the OO way to solve this problem.

7. Source Code and .class files

- (24) In your file system, you can find the workspace you designated when you start up eclipse.
Under that workspace, you can find a folder with the name of your project.
The **src** folder contains your source code, with the package hierarchy.
(These are the files you would submit in assignments.)



- (25) On the other hand, your .class files (i.e., the byte code) are saved under the **bin** folder.
(These are the files you would provide to be executed, or to be included/imported in others' code.)

