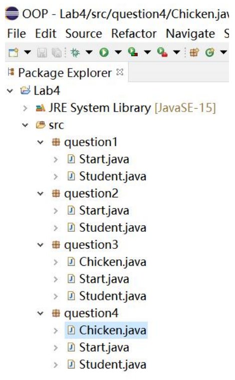
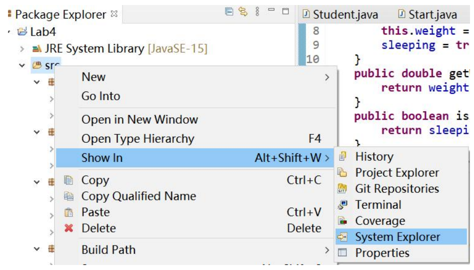
# Lab 4 Requirements

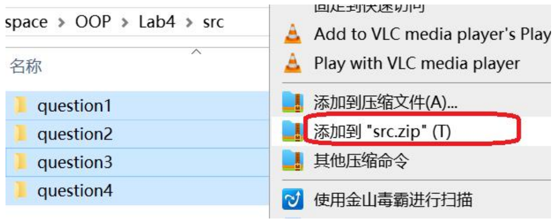
Create a new Eclipse workspace named "**OOP**" on the desktop of your computer. Create a Java Project named "Lab4". Download the sample code **lab4-src-for-student.zip** from iSpace and unzip it. Next to import the sample code( right click the src -> Import… -> File System -> locate to the unzip folder -> import all the questions to your project) . More details, refer to file *How to Use Eclipse and submit to Autolab for student.pdf* on iSpace. Note: you **must** use the sample code, so AutoLab can work correctly.

Answer all questions below. Note: a question may need answers of the previous question(s), just copy paste the previous code to your new package. At the end of the lab, create a ZIP archive of all the src. In Eclipse, right click src -> Show In -> System Explorer, then zip up all folders inside the src folder. In Lab4 case, zip up question1, question2, question3, and question4 folders as below.





```
        this.weight =
        sleeping = tr
    }
    public double get
        return weight
    }
    public boolean is
        return sleepi
```



Extract your zip file to confirm that your zip file contains folders named question1, question2, question3, question4. Rename your src.zip to "**Lab4_1234567890.zip**" (replace **1234567890** with your student ID number). Upload the ZIP file onto AutoLab.

## question 1(Reviewed)

Create a class **Student** with the following UML diagram (remember that **+** means public and **−** means private):

```
+-------------------------+
|         Student         |
+-------------------------+
| - ID: int               |
| - sleeping: boolean     |
+-------------------------+
| + Student(int ID)       |
| + getID(): int          |
| + isSleeping(): boolean |
| + fallAsleep(): void    |
| + wakeUp(): void        |
| + testStudent(): void   |
+-------------------------+
```

**ID** is the student's ID number. The **sleeping** instance variable indicates whether the student is currently sleeping or not. When a student is created, it is initially awake (not sleeping). The **isSleeping** method indicates as a result whether the student is currently sleeping or not. The **fallAsleep** method makes the student fall asleep (or does nothing if the student is already sleeping). The **wakeUp** method wakes the student up (or does nothing if the student is already awake).

The **testStudent** method is static and is used for testing the **Student** class. Here is the code for this **testStudent** method:

```
      public static void testStudent() {
            Student s = new Student(1234567890);

            System.out.println(s.getID() == 1234567890);
            System.out.println(s.isSleeping() == false);
            s.fallAsleep();
            System.out.println(s.isSleeping() == true);
            s.fallAsleep(); // should do nothing because the student is already sleeping
            System.out.println(s.isSleeping() == true);
            s.wakeUp();
            System.out.println(s.isSleeping() == false);
            s.wakeUp(); // should do nothing because the student is already awake
            System.out.println(s.isSleeping() == false);
      }
```

And here is the **Start** class to test the **Student** class:

```
public class Start {
      public static void main(String[] args) {
            Student.testStudent();
      }
}
```

## question 2 (Mandatory)

Add to the **Start** class a new method called **check** that takes a **Student** object as argument and returns a string as a result. This **check** method should return the string "**sweet dreams**" if the student is currently sleeping, and should return the string "**need coffee**" if the student is currently awake.

Should the **check** method be static or not? Why?

Add tests to the **main** method of the **Start** class to test your new **check** method. You can use the **==** operator to compare strings.

**Test cases (Study and learn before using them!)**

```
public static void testStudent() {
      Student s = new Student(1234567890);

      System.out.println(s.getID() == 1234567890);
      System.out.println(s.isSleeping() == false);
      s.fallAsleep();
      System.out.println(s.isSleeping() == true);
      s.fallAsleep(); // should do nothing because the student is already sleeping
      System.out.println(s.isSleeping() == true);
      s.wakeUp();
      System.out.println(s.isSleeping() == false);
      s.wakeUp(); // should do nothing because the student is already awake
      System.out.println(s.isSleeping() == false);
}
```

## question 3 (Mandatory)

Add to your program a class **Chicken** with the following UML diagram:

```
+--------------------------+
|         Chicken          |
+--------------------------+
| - weight: double         |
| - sleeping: boolean      |
+--------------------------+
| + Chicken(double weight) |
| + getWeight(): double    |
| + isSleeping(): boolean  |
| + fallAsleep(): void     |
| + wakeUp(): void         |
| + testChicken(): void    |
+--------------------------+
```

The **weight** instance variable indicates the current weight of the chicken. The **sleeping** instance variable indicates whether the chicken is currently sleeping or not. When a chicken is created, it is initially sleeping (not awake). The **isSleeping** method indicates as a result whether the chicken is currently sleeping or not. The **fallAsleep** method makes the chicken fall asleep (or does nothing if the chicken is already sleeping). The **wakeUp** method wakes the chicken up (or does nothing if the chicken is already awake).

**Test cases (Study and learn before using them!)**

```java
public static void testStudent() {
      Student s = new Student(1234567890);

      System.out.println(s.getID() == 1234567890);
      System.out.println(s.isSleeping() == false);
      s.fallAsleep();
      System.out.println(s.isSleeping() == true);
      s.fallAsleep(); // should do nothing because the student is already sleeping
      System.out.println(s.isSleeping() == true);
      s.wakeUp();
      System.out.println(s.isSleeping() == false);
      s.wakeUp(); // should do nothing because the student is already awake
      System.out.println(s.isSleeping() == false);
}

public static void testChicken() {
      Chicken c = new Chicken(2.3);

      System.out.println(c.getWeight() == 2.3);
      System.out.println(c.isSleeping() == true);
      c.wakeUp();
      System.out.println(c.isSleeping() == false);
      c.wakeUp(); // should do nothing because the chicken is already awake
      System.out.println(c.isSleeping() == false);
      c.fallAsleep();
      System.out.println(c.isSleeping() == true);
      c.fallAsleep(); // should do nothing because the chicken is already sleeping
      System.out.println(c.isSleeping() == true);
}
```

## question 4 ((Mandatory)

Add to the **`Start`** class a new method called **`check`** that takes a **`Chicken`** object as argument and returns a string as a result. This **`check`** method should return the string "**`sweet dreams`**" if the chicken is currently sleeping, and should return the string "**`need coffee`**" if the chicken is currently awake.

Should the **`check`** method be static or not? Why?

Add tests to the **`main`** method of the **`Start`** class to test your new check method. You can use the **==** operator to compare strings.

**Test cases:**

```
public static void testStudent() {
      Student s = new Student(1234567890);

      System.out.println(s.getID() == 1234567890);
      System.out.println(s.isSleeping() == false);
      s.fallAsleep();
      System.out.println(s.isSleeping() == true);
      s.fallAsleep(); // should do nothing because the student is already sleeping
      System.out.println(s.isSleeping() == true);
      s.wakeUp();
      System.out.println(s.isSleeping() == false);
      s.wakeUp(); // should do nothing because the student is already awake
      System.out.println(s.isSleeping() == false);
}

public static void testChicken() {
      Chicken c = new Chicken(2.3);

      System.out.println(c.getWeight() == 2.3);
      System.out.println(c.isSleeping() == true);
      c.wakeUp();
      System.out.println(c.isSleeping() == false);
      c.wakeUp(); // should do nothing because the chicken is already awake
      System.out.println(c.isSleeping() == false);
      c.fallAsleep();
      System.out.println(c.isSleeping() == true);
      c.fallAsleep(); // should do nothing because the chicken is already sleeping
      System.out.println(c.isSleeping() == true);
}
```