

# OODP– Session 4c

## Session times

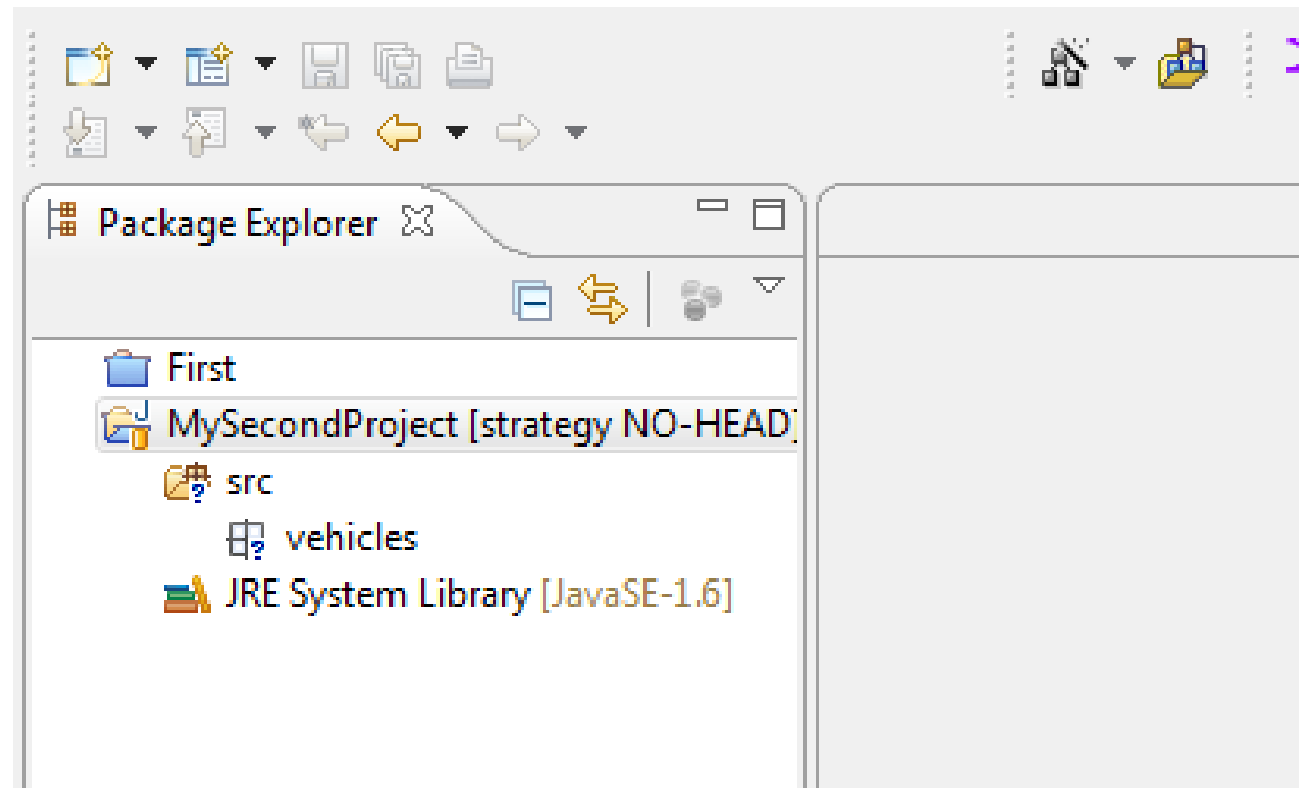
PT group 1 – Monday	18:00-21:00	room: Malet 403
PT group 2 – Thursday	18:00-21:00	room: Malet 407
FT - Tuesday	13:30-17:00	room: Malet 404

Email: [oded@dcs.bbk.ac.uk](mailto:oded@dcs.bbk.ac.uk)

Web Page: <http://www.dcs.bbk.ac.uk/~oded>

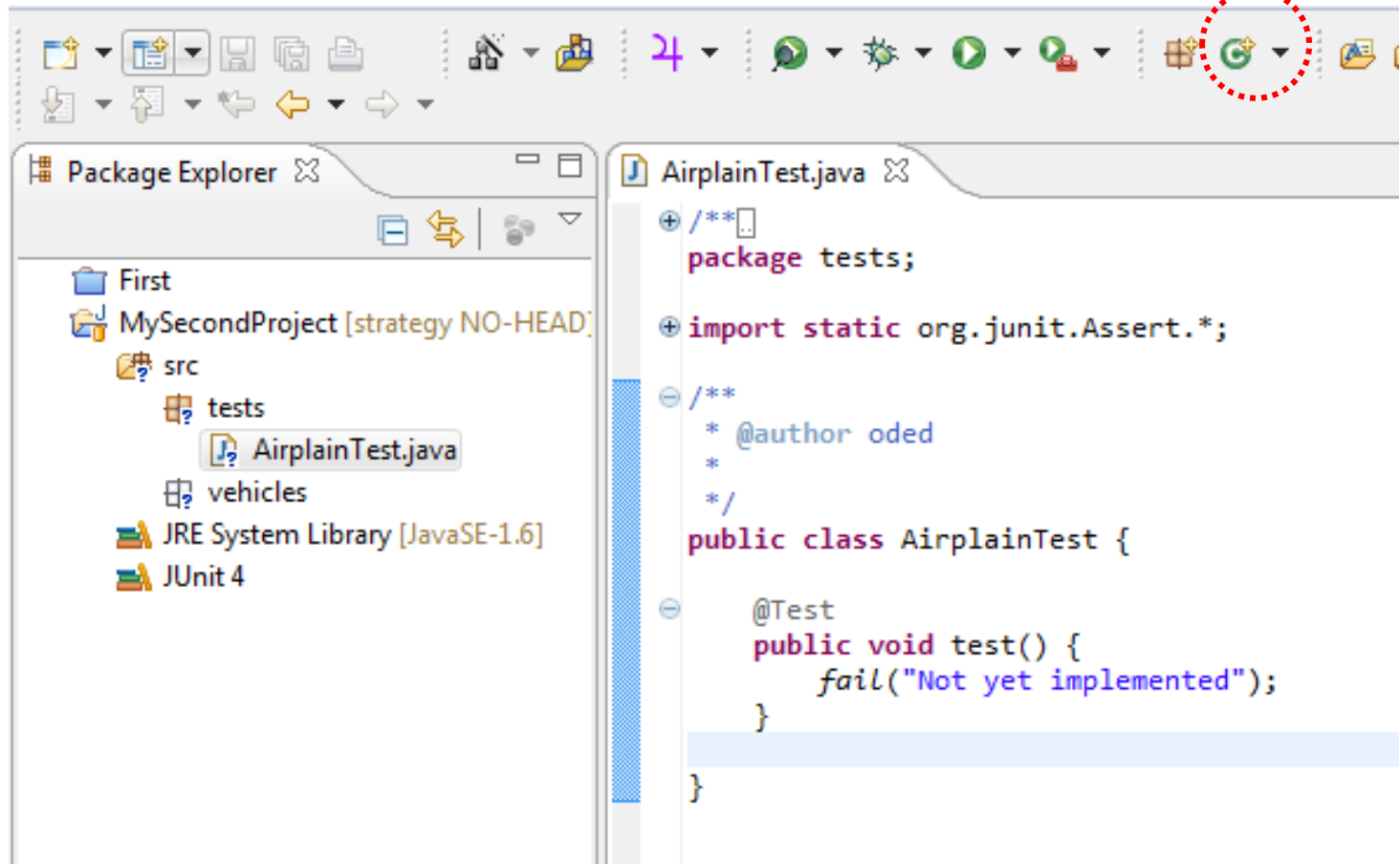
Visiting Hours: [Tuesday 17:00 to 19:00](#)

# Start a new project



# New code

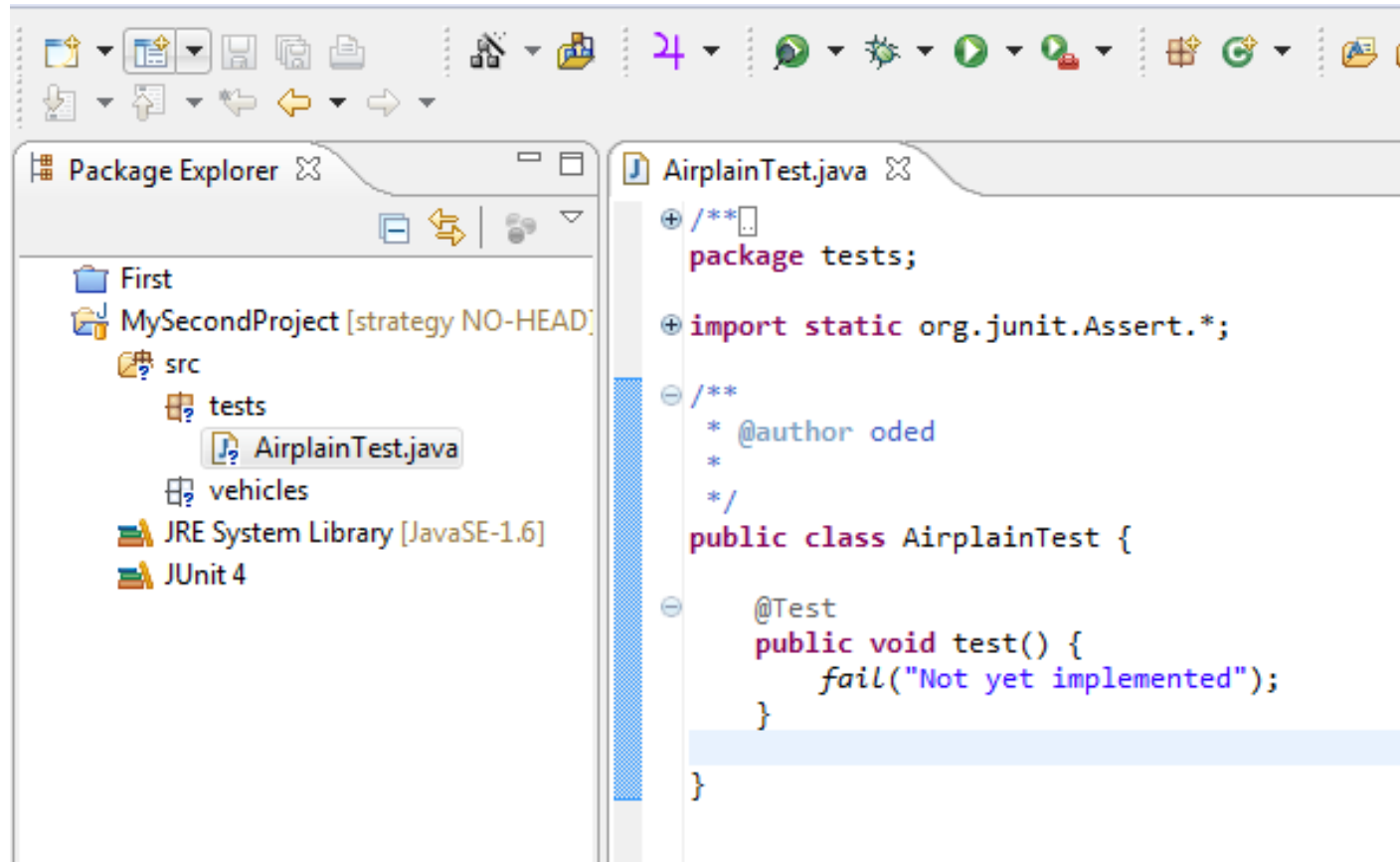
Use this to create test



Remember to check that the test fails

# Start a new project

Don't



# Preparing for a new class

Use this to  
create  
*Airplane*  
class

```
*AirplaneTest.java X
+ /**
  package tests;

+ import static org.junit.Assert.*;

- /**
  * @author oded
  *
  */
  public class AirplaneTest {

-     @Test
     public void test() {
         |
         Airplane classUnderTest = new Airplane(1);

         fail("Not yet implemented");
     }

  }
```

# New class

Make in proper  
Package

The screenshot shows the 'New Java Class' dialog box with the following configuration:

- Source folder: MySecondProject/src
- Package: vehicles (highlighted with a red dashed oval and a red dotted line pointing to the text 'Make in proper Package')
- Enclosing type: tests.AirplainTest
- Name: Airplane
- Modifiers: public (selected), default, private, protected, abstract, final, static
- Superclass: java.lang.Object
- Interfaces: (empty list)
- Which method stubs would you like to create?
  - public static void main(String[] args)
  - Constructors from superclass
  - Inherited abstract methods
- Do you want to add comments? (Configure templates and default value [here](#))
  - Generate comments

Buttons: Finish, Cancel

# Creating the constructor

Use this to  
create  
the  
constru  
ctor for  
*Airplane*  
class

```
*AirplainTest.java X
+ /**
  package tests;

+ import static org.junit.Assert.*;

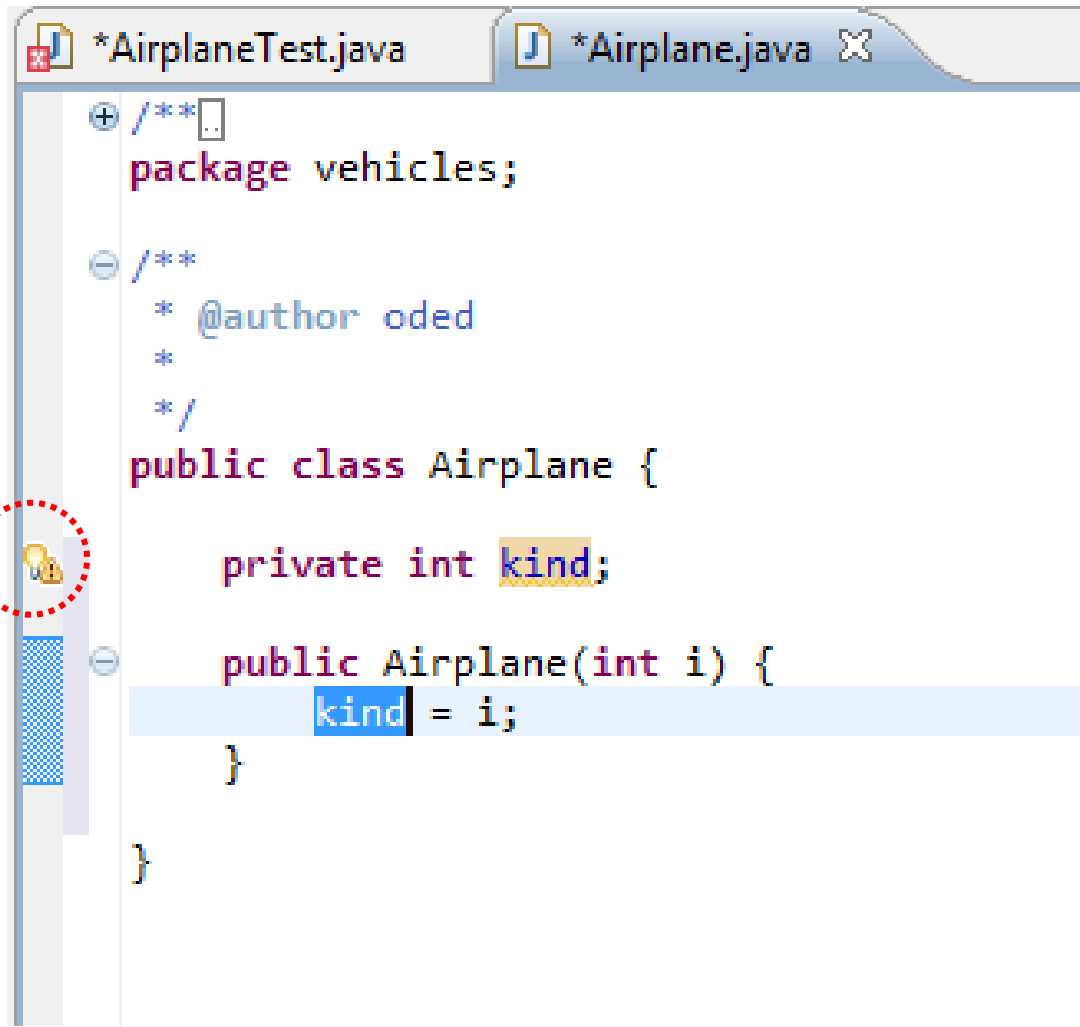
- /**
  * @author oded
  *
  */
  public class AirplainTest {

-   @Test
    public void test() {

      Airplane classUnderTest = new Airplane("Harrier");|
      fail("Not yet implemented");
    }
  }
}
```

# New Airplane code (with addition of a field)

What is this?



```

+ /**
  package vehicles;

- /**
  * @author oded
  *
  */
  public class Airplane {

    private int kind;

    public Airplane(int i) {
      kind = i;
    }

  }

```

Run test and if no problems commit



# New test code

```
public void test() {  
  
    String expectedOutput = "Like a Harrier";  
    String stringReturned = null;  
  
    Airplane classUnderTest = new Airplane(1);  
  
    stringReturned = classUnderTest.howDoYouFly();  
  
    assertEquals("Wrong Answer !", stringReturned, expectedOutput);  
}
```

1. Use IDE to create new method in airplane class

2. Run test – if fails commit

Code may require adjusting if copy pasted

# New how do you fly method

```
public String howDoYouFly() {  
    switch(kind){  
        case 1: return "Like a fighter jet";  
        case 2: return "I don't fly";  
        case 3: return "Like a passenger plane";  
        default: return null;  
    }  
}
```

1. Run test – if passes commit

# New Liftoff test

```
@Test
public void test2() {

    String expectedOutput = "Vertically";
    String stringReturned = null;

    Airplane classUnderTest = new Airplane(1);

    stringReturned = classUnderTest.howDoYouLiftOff();

    assertEquals("Wrong Answer !", stringReturned, expectedOutput);

}
```

1. Use IDE to create new method in airplane class

2. Run test – if fails commit

# howDoYouLiftOff Code (for airplane class)

```
public String howDoYouLiftOff() {  
    switch(kind){  
        case 1: return "Vertically";  
        case 2: return "I Liftoff";  
        case 3: return "Horizontally";  
        default: return null;  
    }  
}
```

2. Run test – if passes commit

**Now  
Apply the  
Strategy Pattern**