

# OODP– Session 4c

## Session times

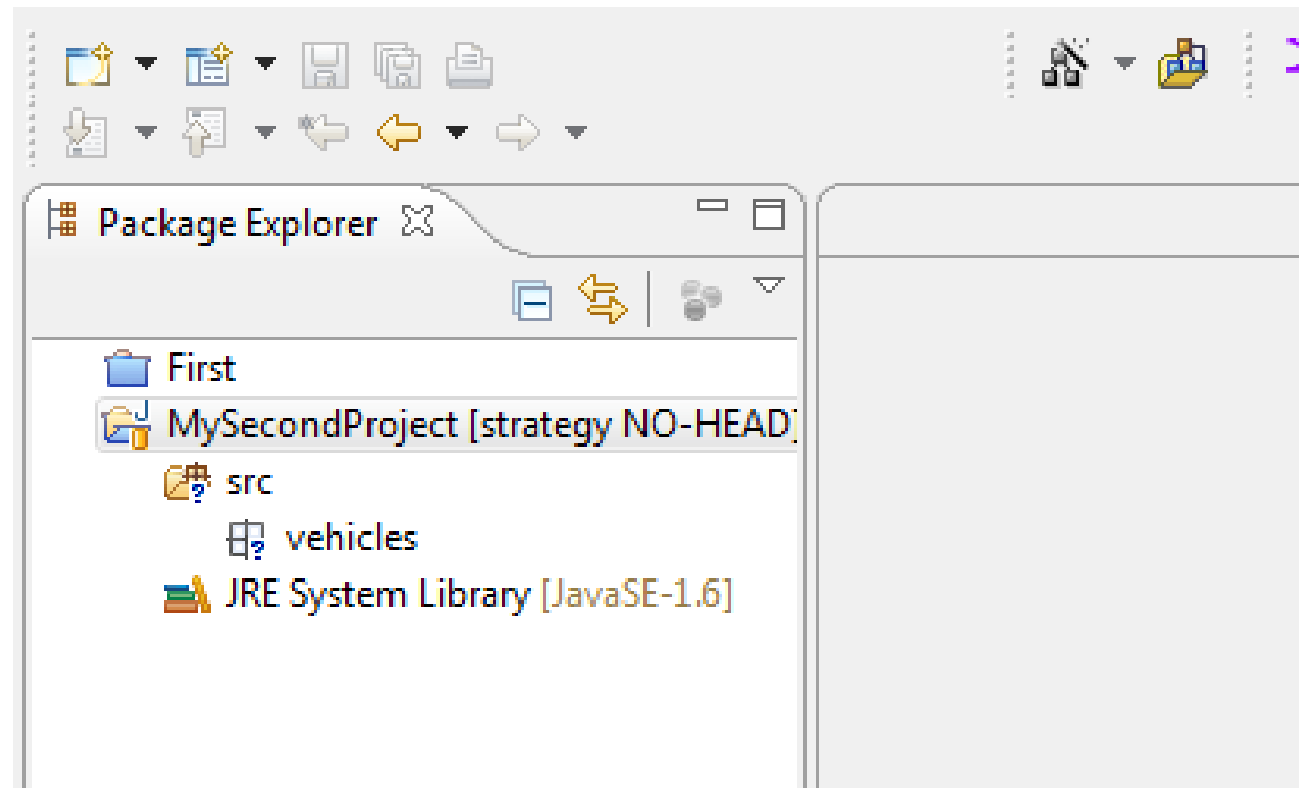
PT group 1 – Monday	18:00-21:00	room: Malet 403
PT group 2 – Thursday	18:00-21:00	room: Malet 407
FT - Tuesday	13:30-17:00	room: Malet 404

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

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

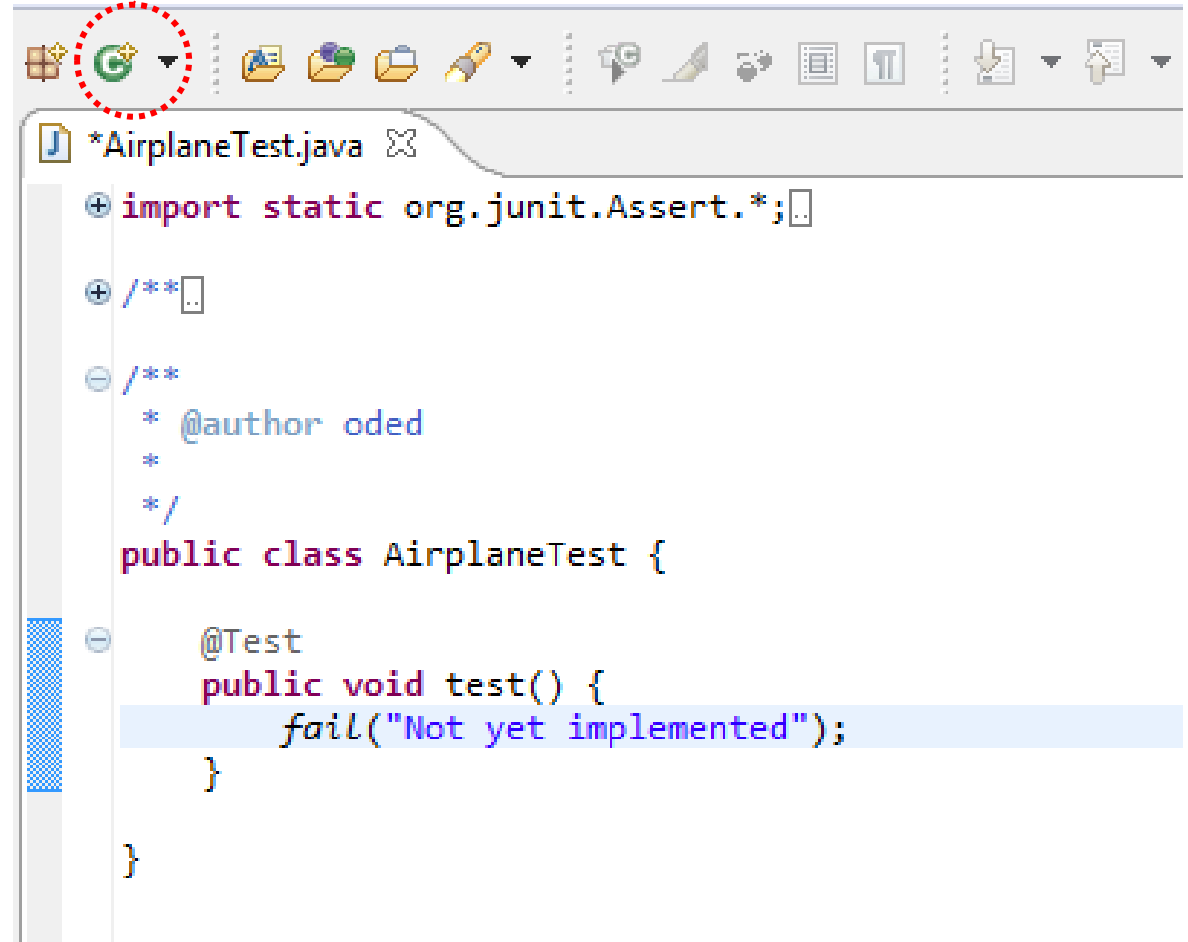
Visiting Hours: [Tuesday 17:00 to 19:00](#)

# Start a new project



# New test

Use this to create test



```
import static org.junit.Assert.*;

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

    @Test
    public void test() {
        fail("Not yet implemented");
    }
}
```

1. check that the test fails if it does, then commit

# Preparing for a new class

New Code

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)
- 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  
constructor  
for *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);

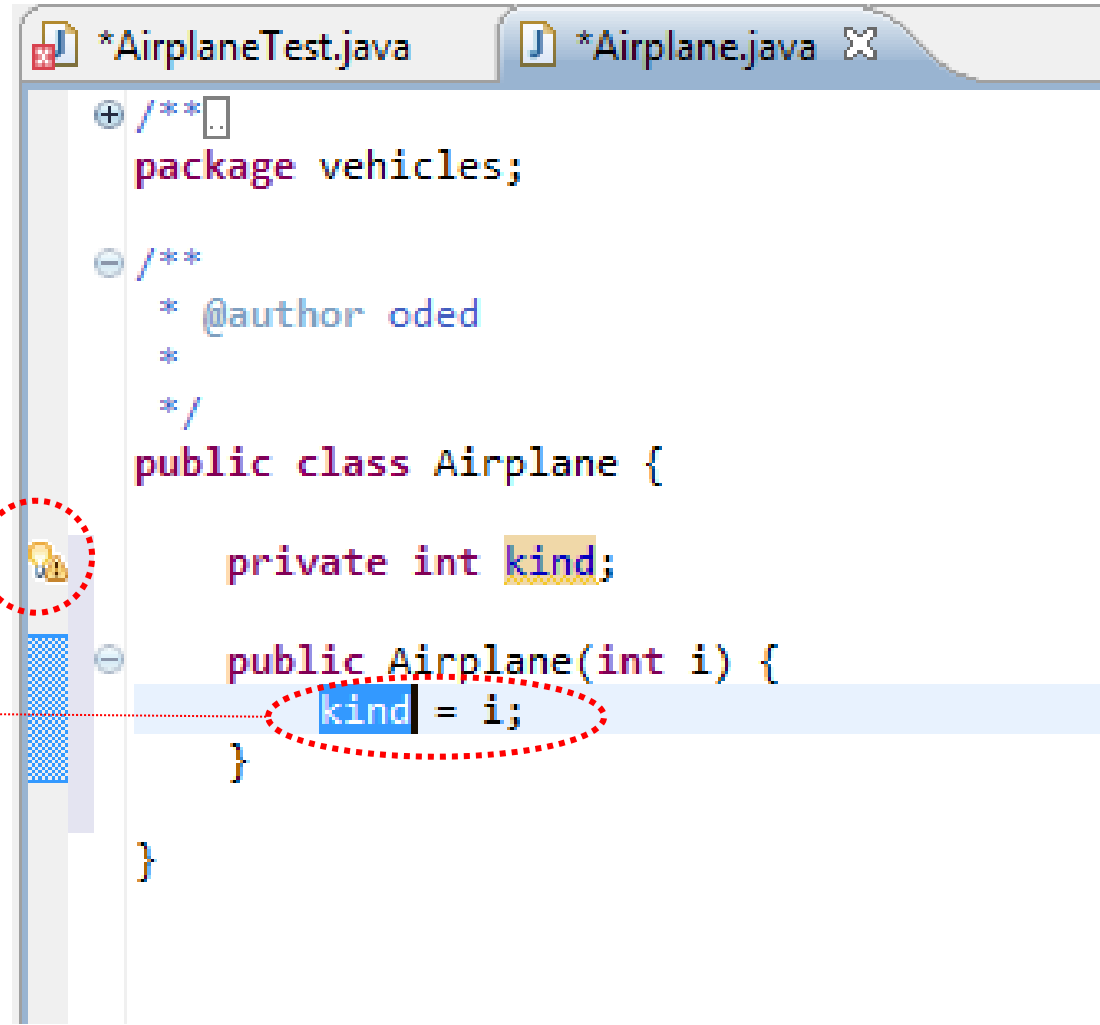
        fail("Not yet implemented");
    }

  }
```

# New Airplane code (with addition of a field)

What is this?

Start by adding this and then use IDE to create the field



```
/**  
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 fighter jet";  
    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 (in airplane class)

```
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 (in airplane class)

```
@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;  
    }  
}
```

1. Run test – if passes commit

**Now  
Apply the  
Strategy Pattern**

# Refactoring Flying

```
public String howDoYouFly() {  
    return hIF.howIFLly();  
}
```

1. New Code for `howDoYouFly()` in airplane class use IDE to create `hIF` field

Type object was created, because the type was not explicit (in Java all classes inherit Object by default)

```
public class Airplane {  
    private int kind;  
    private Object hIF;  
    public Airplane(int i) {  
        kind = i;  
    }  
    public String howDoYouFly() {  
        return hIF.howIFLly();  
    }  
}
```

# Refactoring Flying

1. Change Object to Flying

```
public class Airplane {  
    private int kind;  
    private Object hIF;  
  
    public Airplane(int i) {  
        kind = i;  
    }  
  
    public String howDoYouFly() {  
        return hIF.howIFly();  
    }  
}
```

2. Use this to create Flying interface

```
public class Airplane {  
    private int kind;  
    private Flying hIF;  
  
    public Airplane(int i) {  
        kind = i;  
    }  
}
```

# Java interface window

Make sure correct package

**Java Interface**  
Create a new Java interface.

Source folder: MySecondProject/src

Package: vehicles

Enclosing type: vehicles.Airplane

Name: Flying

Modifiers:  public  default  private  protected

Extended interfaces:

Do you want to add comments? (Configure templates and default value [here](#))  
 Generate comments

Generate comments

# Java interface code

```
*Airplane.java | AirplaneTest.java | Flying.java X
+ /**
package vehicles;

- /**
 * @author oded
 *
 */
public interface Flying {
}
```

```
~/
public class Airplane {

    private int kind;
    private Flying hIF;

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

    public String howDoYouFly() {
        return hIF.howIFly();
    }

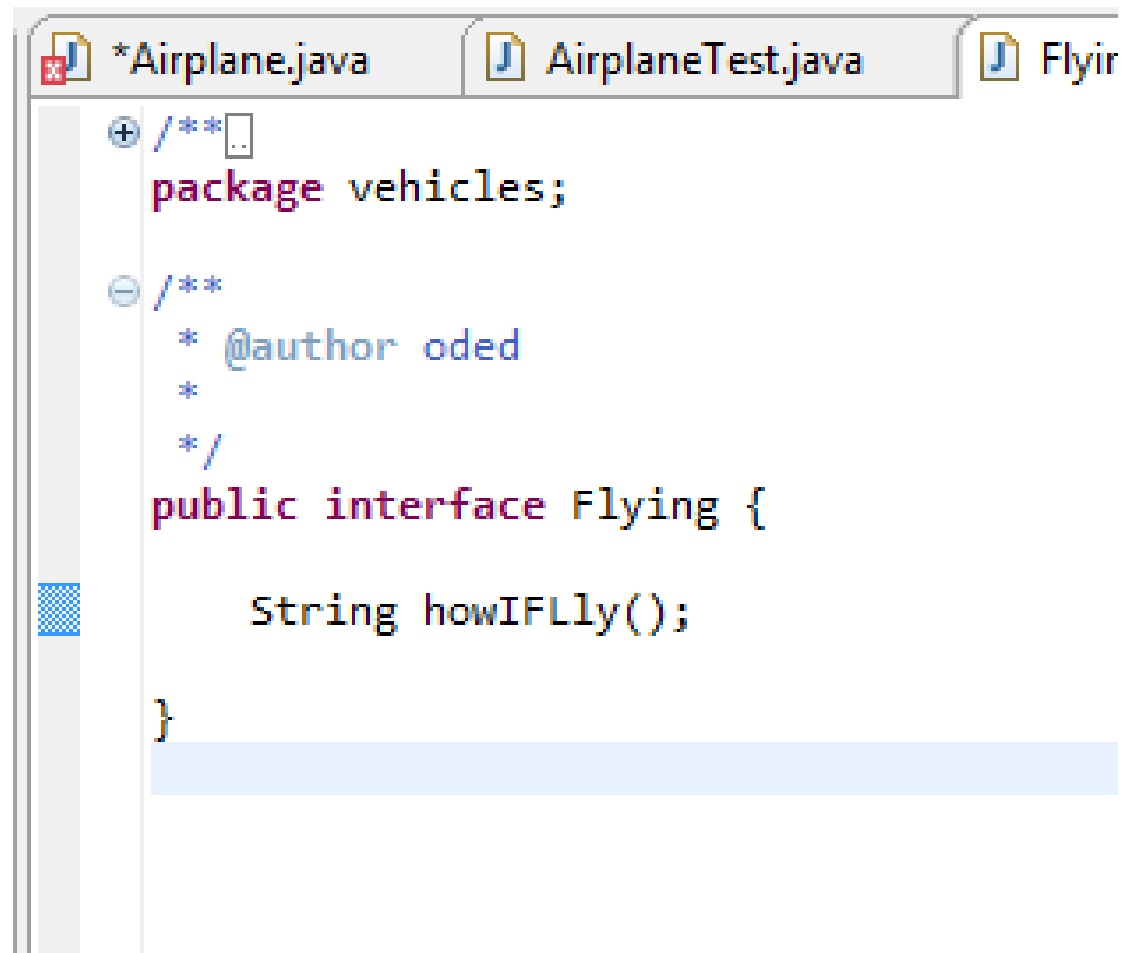
    public String howDoYouLiftOff() {
```

Use this to create 'howIFly' method





# New 'HowIFly' Method



```
*Airplane.java | AirplaneTest.java | Flyir
+ /**
package vehicles;

- /**
 * @author oded
 *
 */
public interface Flying {

    String howIFLly();

}
```