OODP– Session 7b

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

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

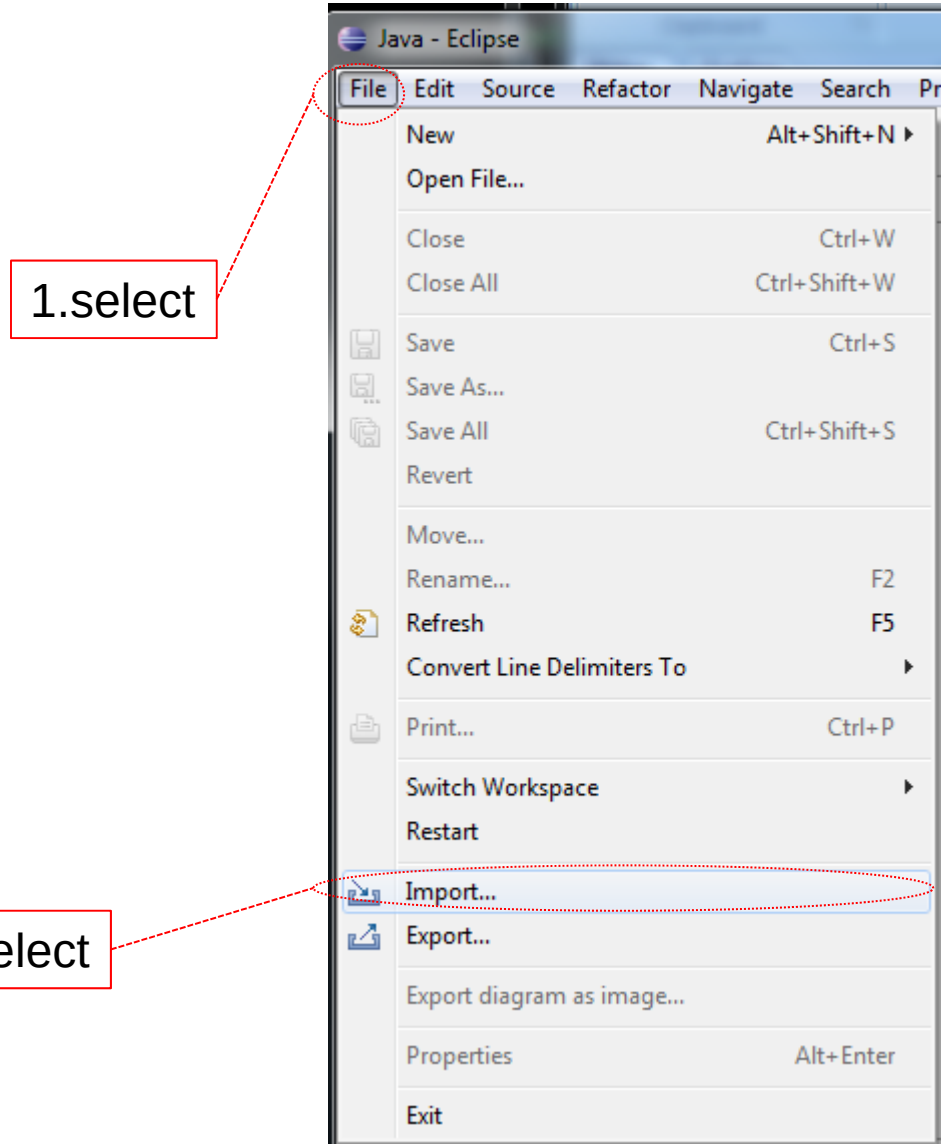
Visiting Hours: Tuesday 17:00 to 19:00

github

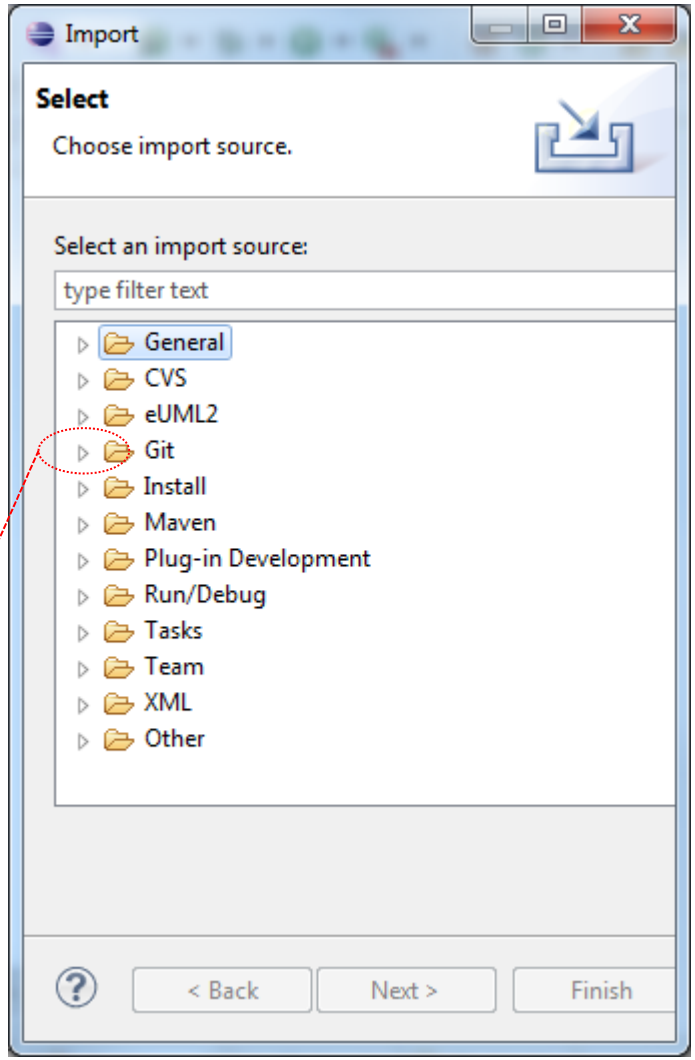
github

1. Start Eclipse
2. Close any open project

- We are next going to clone the Airplane code from Git Hub

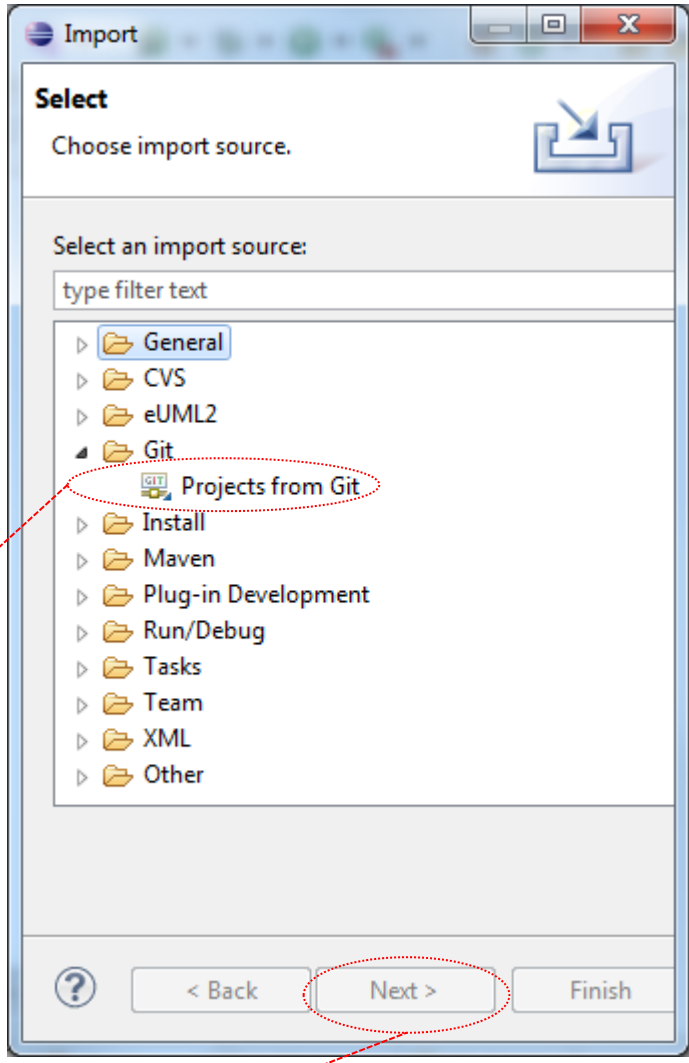


github



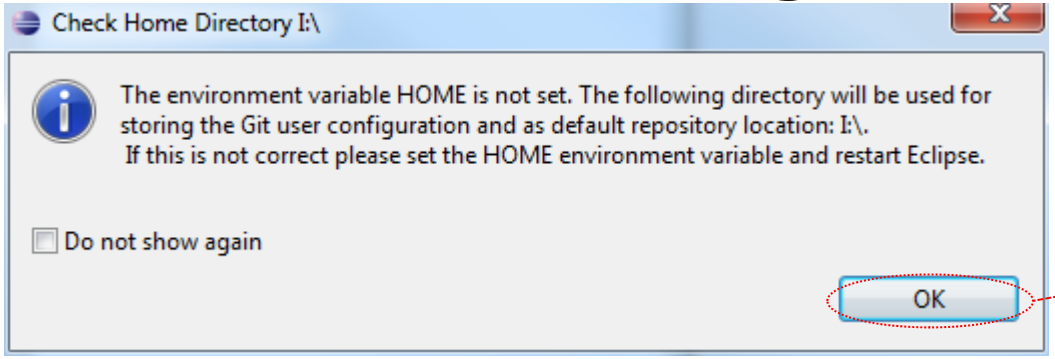
1.select

2.select

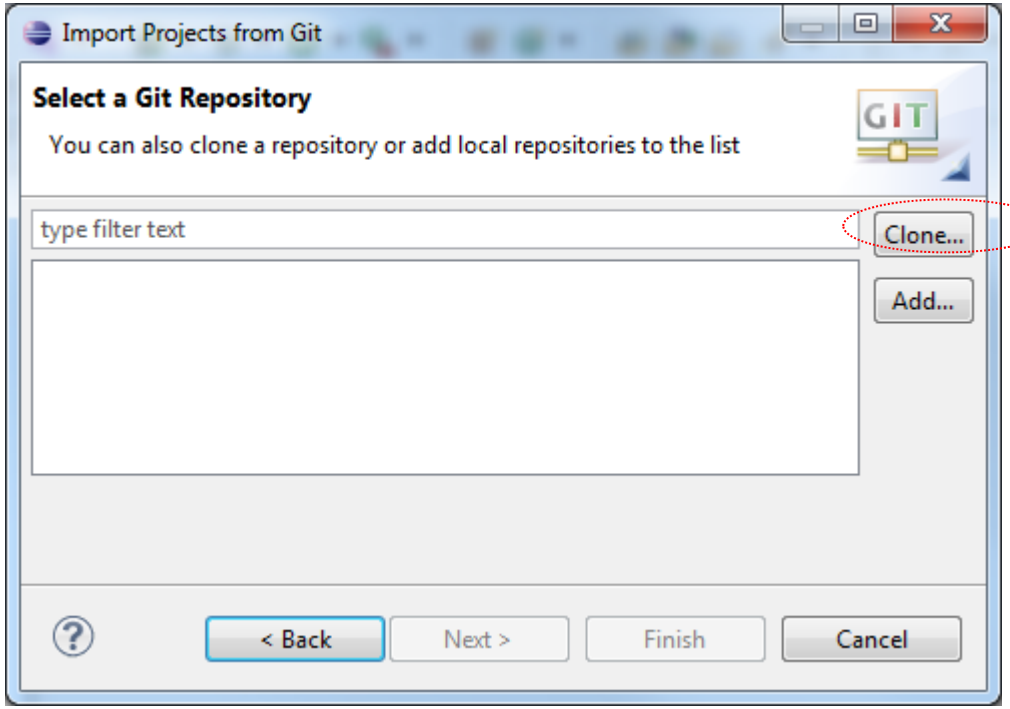


3.select

github



1.select

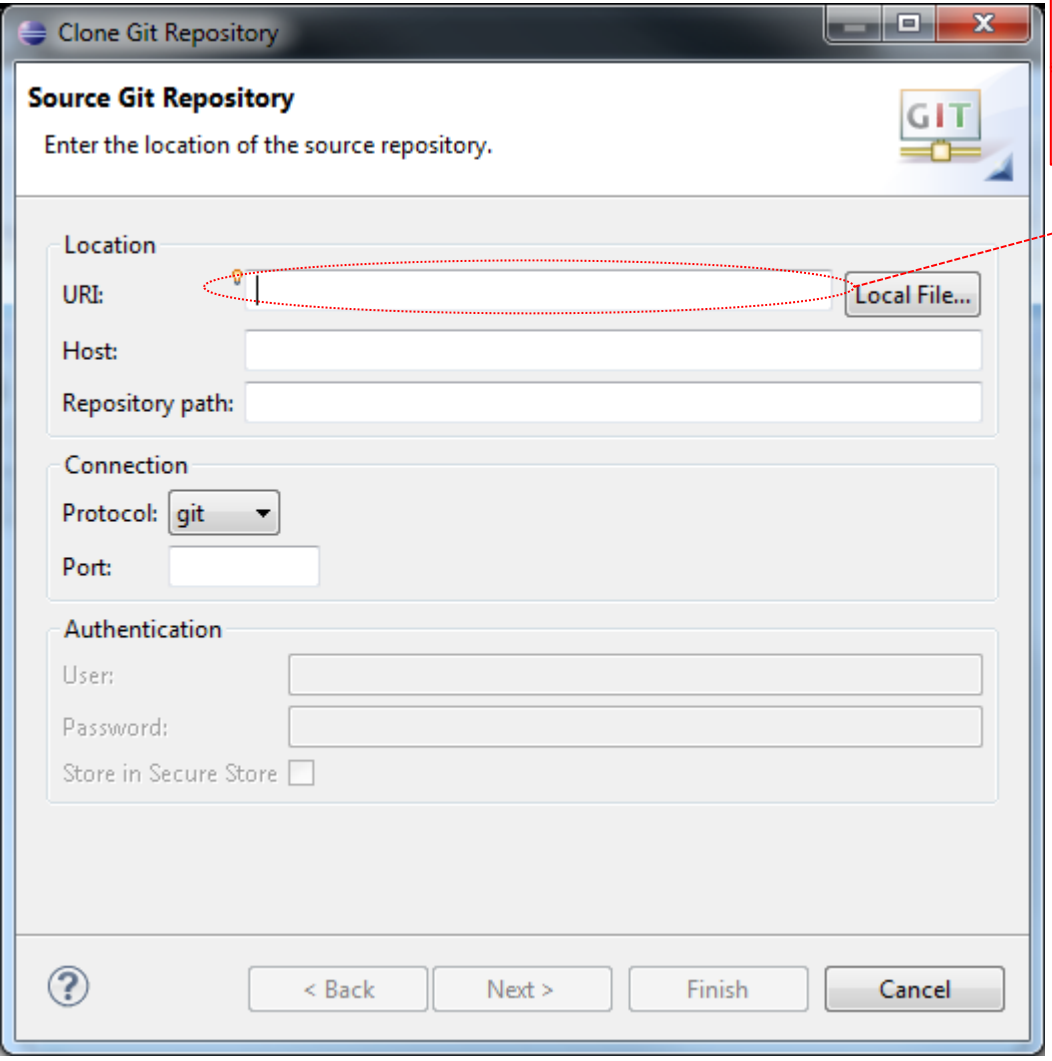


2.select

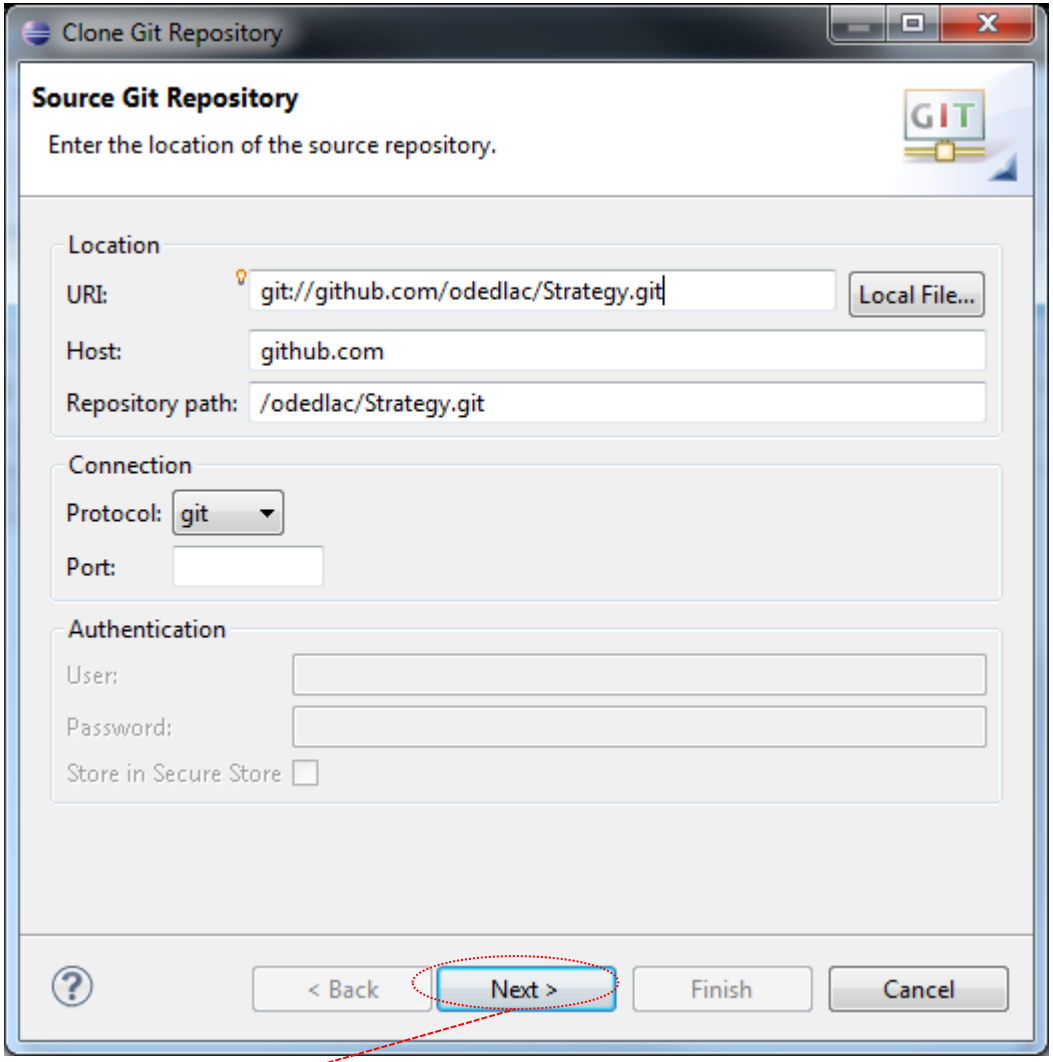
github

1.Fill in

git://github.com/odedlac/Strategy.git

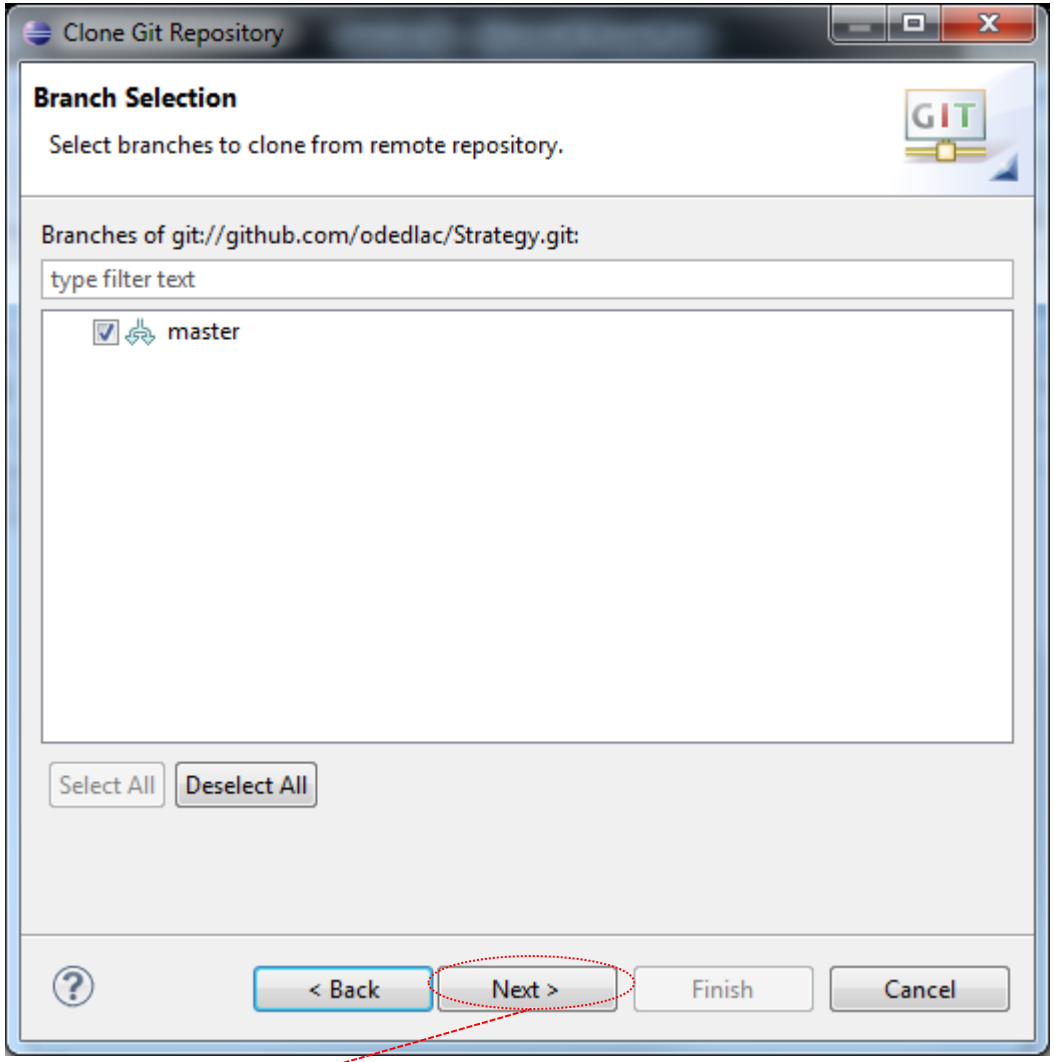


github



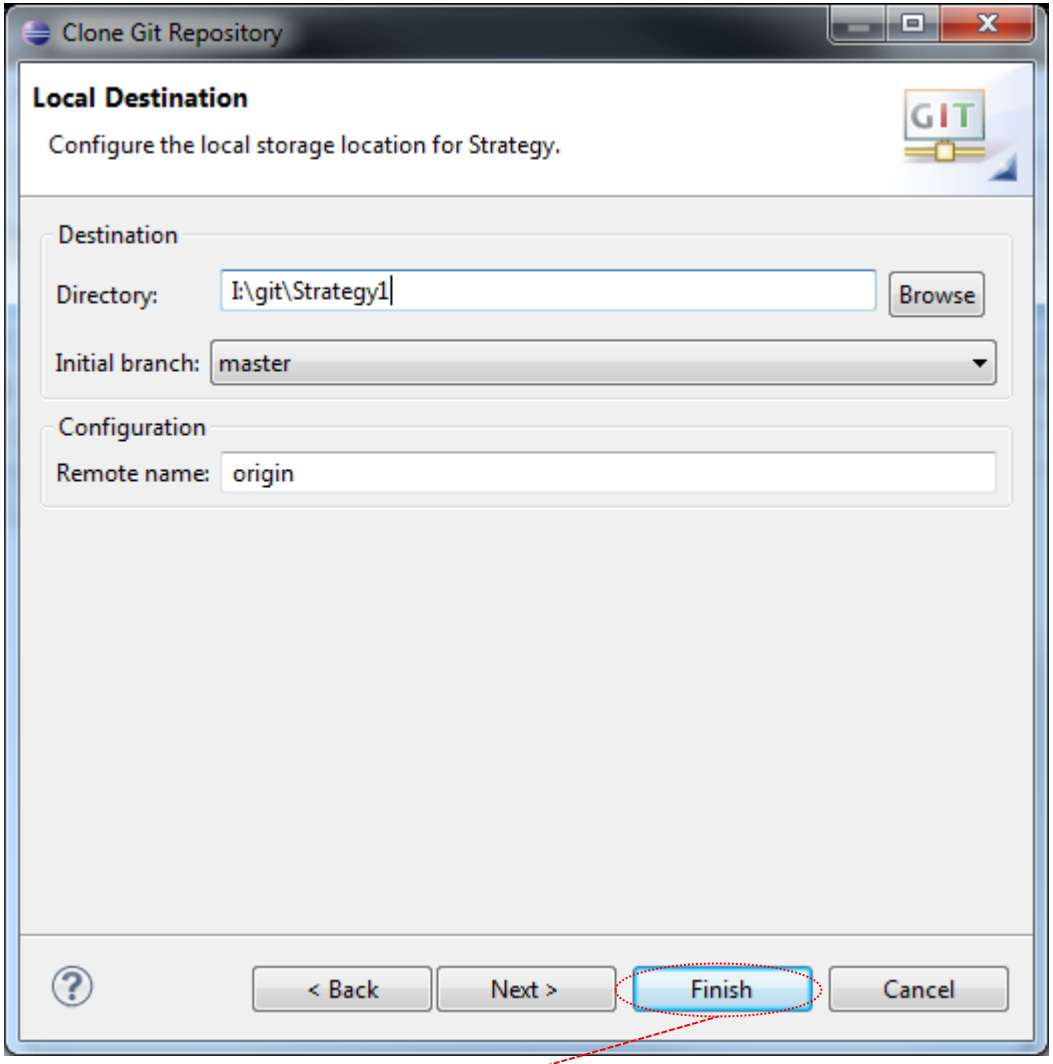
1.select

github

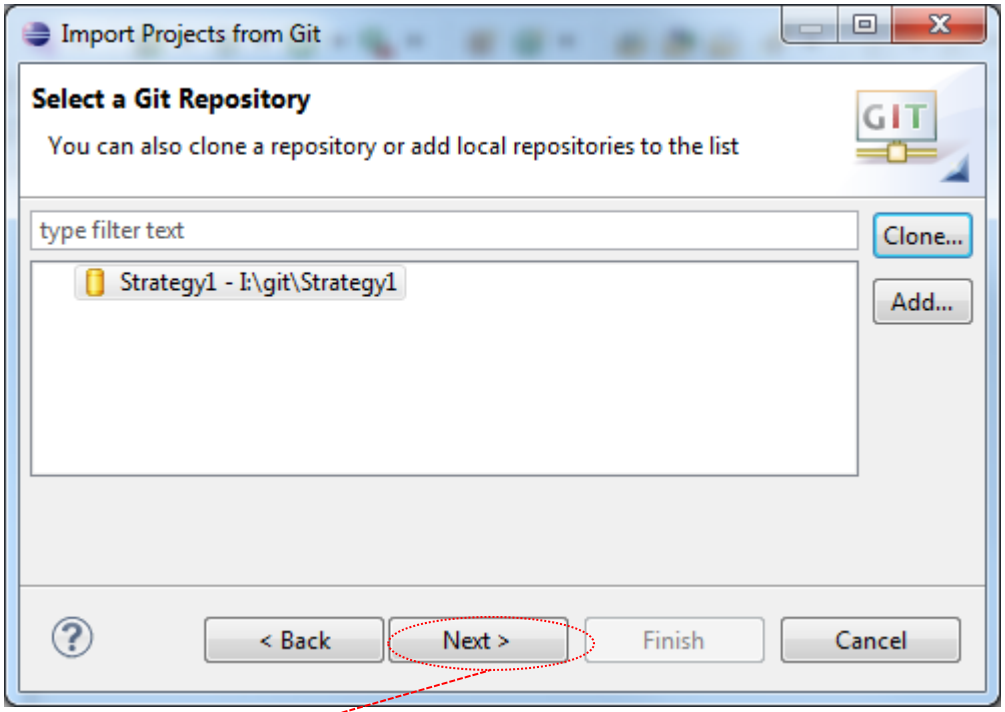


1.select

github

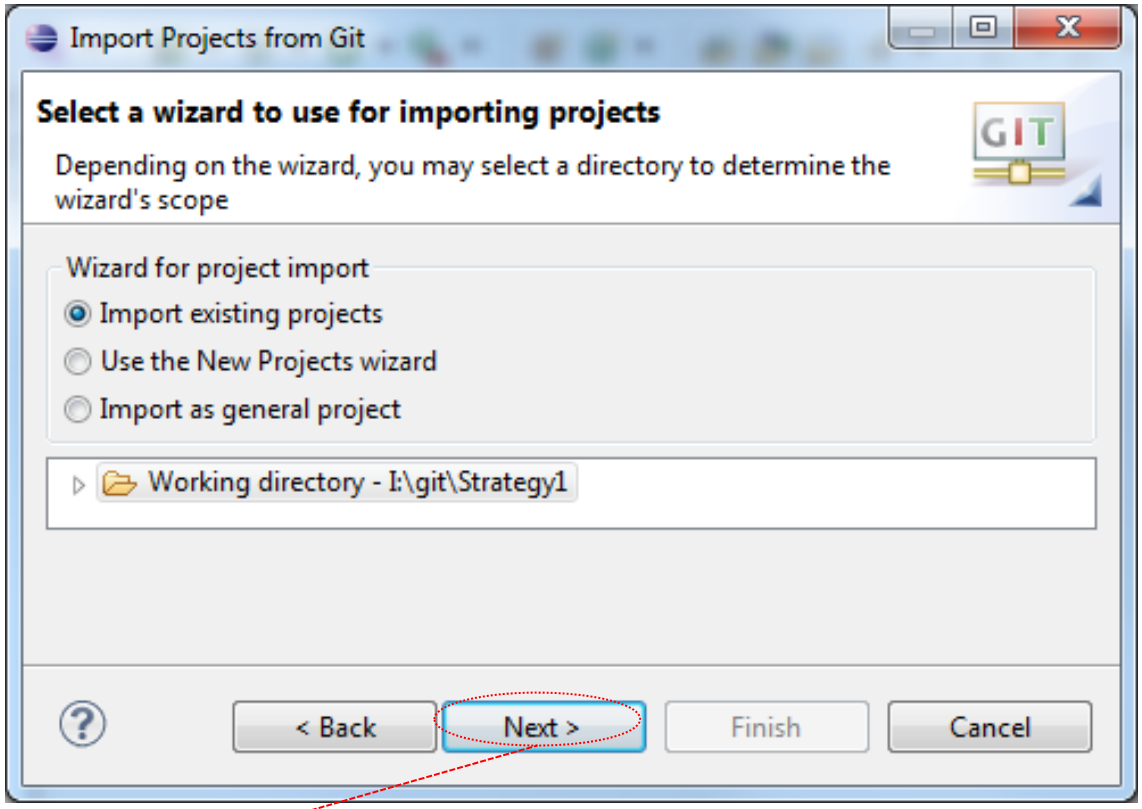

Clone Git Repository
Local Destination
Configure the local storage location for Strategy.
GIT
Destination
Directory:
I:\git\Strategy1
Browse
Initial branch:
master
Configuration
Remote name:
origin
< Back
Next >
Finish
Cancel
1.select

github



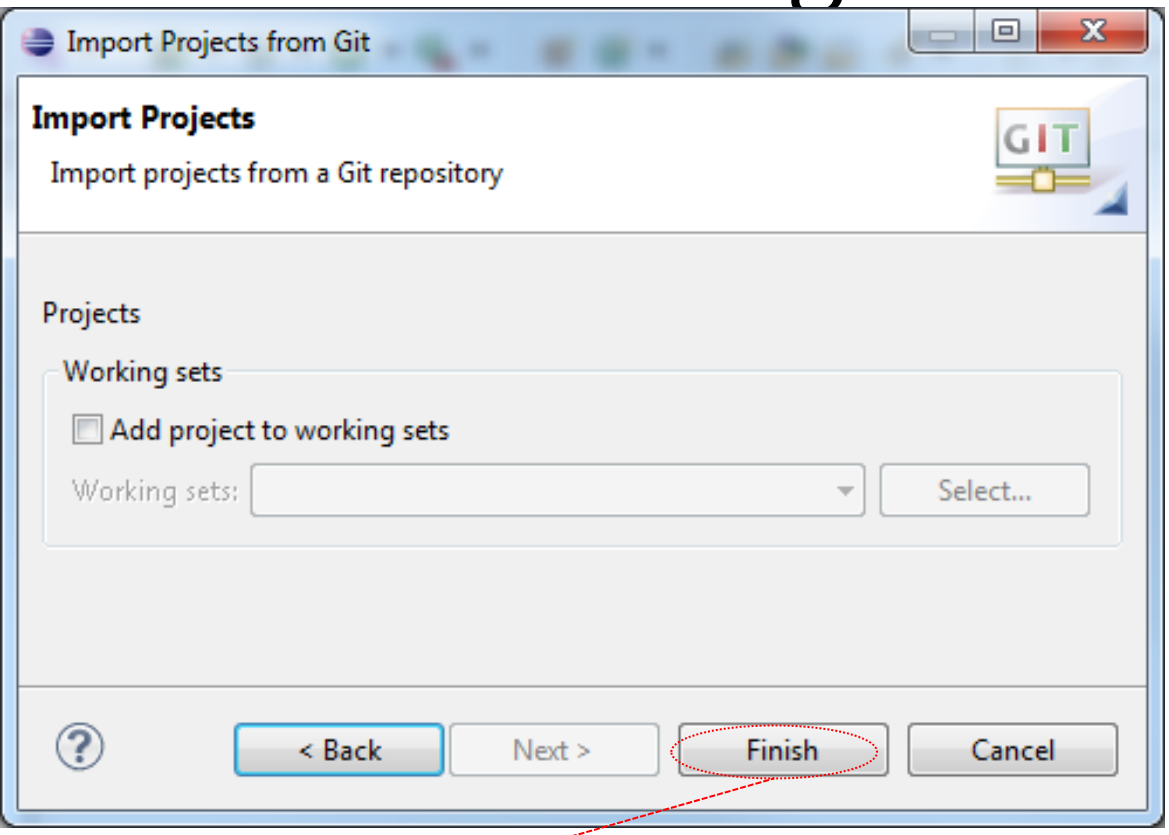
1.select

github

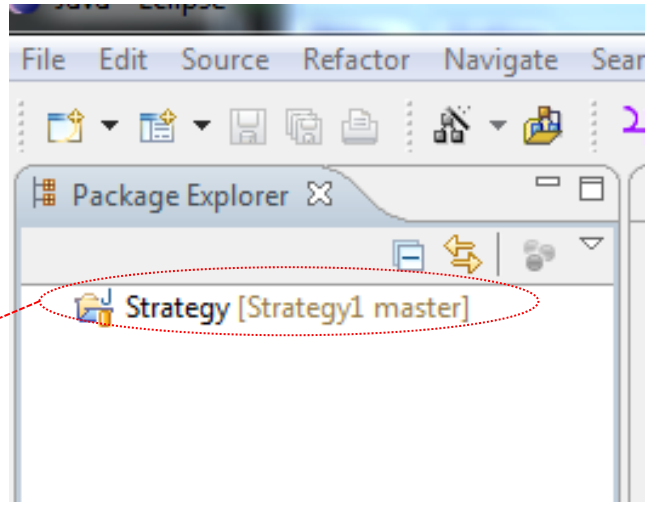


1.select

github



1.select



observe

Factory Method

Factory Method

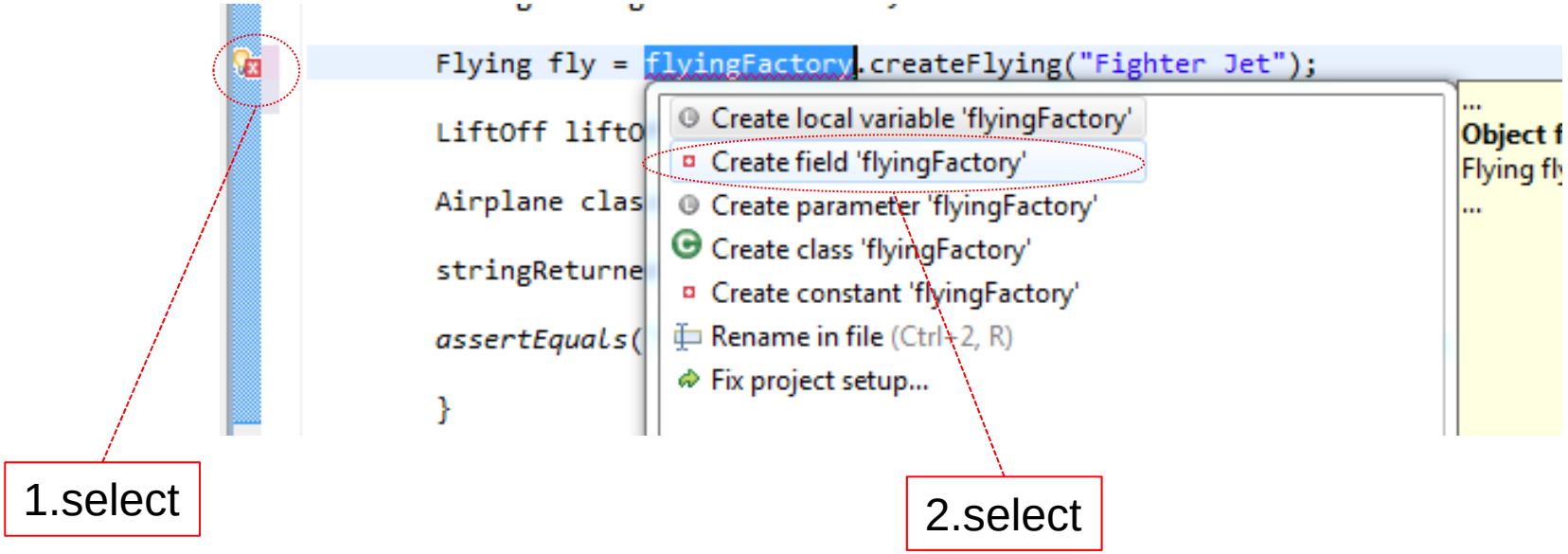
```
*/
public class AirplaneTest {

    @Test
    public void test1() {

        String expectedOutput = "Like a fighter jet";
        String stringReturned = null;

        Flying fly = new IFlyLikeFJ();
        LiftOff liftOff = new ILiftOffV();

        Airplane classUnderTest = new Airplane(liftOff,fly);

        stringReturned = classUnderTest.howDoYouFly();

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

        }
```

1. Replace with this code

flyingFactory.createFlying("Fighter Jet");

Delegating the creation of flying to a dedicated method

Factory Method



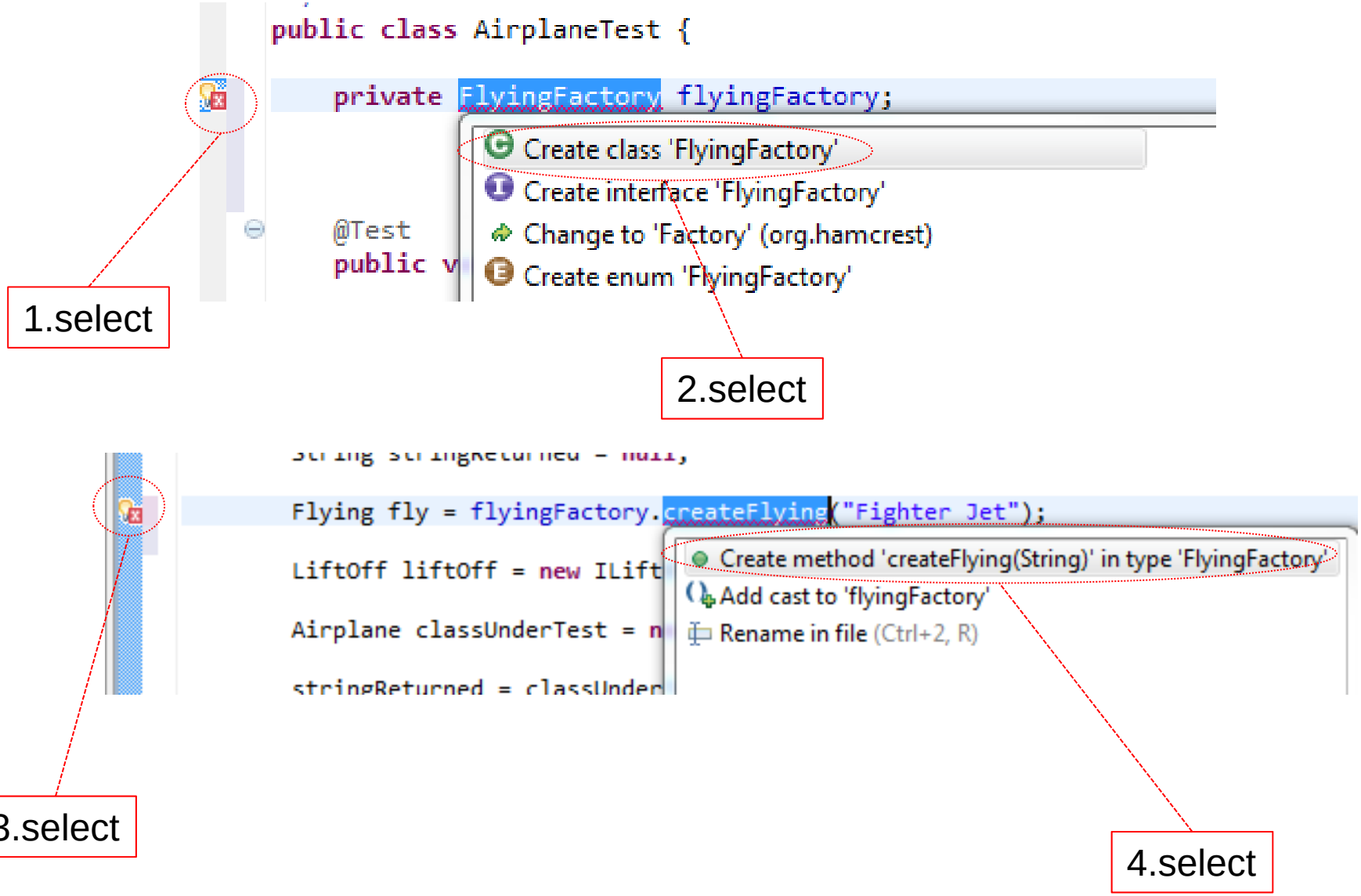
New Code

```
/
public class AirplaneTest {

    private Object flyingFactory;
```

2. Replace with this code

FlyingFactory

Factory Method



Factory Method

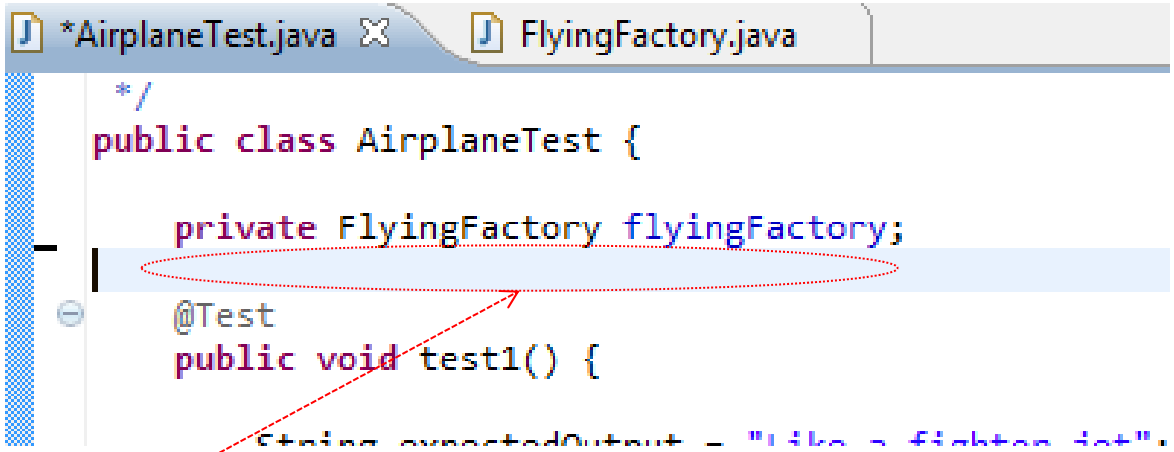
```
/**
 * @author oded
 *
 */
public class FlyingFactory {

    public Flying createFlying(String string) {
        // TODO Auto-generated method stub
        return null;
    }

}
```

1. Replace with this code

```
if(string.equals("Fighter Jet")){
return new IFlyLikeFJ();
}
```

Now it remains to create an instance of FlyingFactory in AirplaneTest

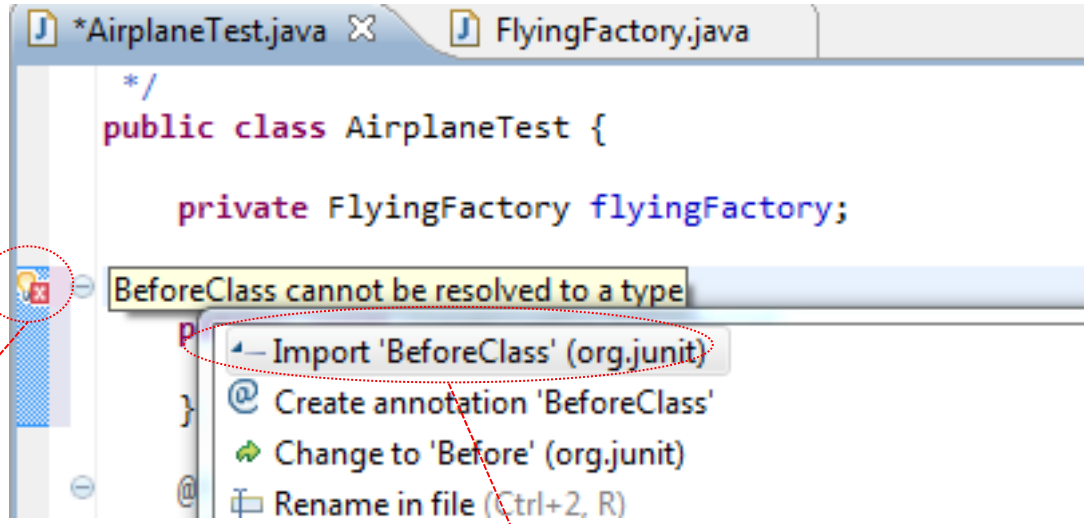
Factory Method



1. Insert this code

```
@BeforeClass
public static void onlyOnce(){
        flyingFactory = new FlyingFactory();
}
```

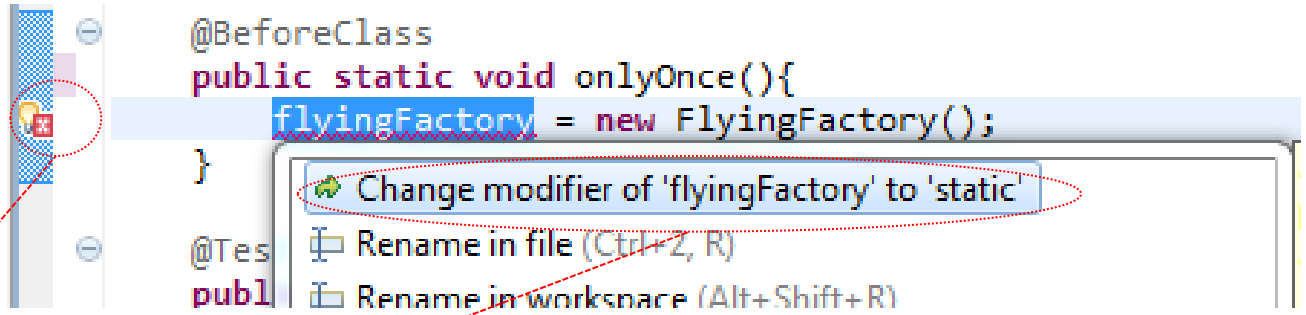
Factory Method



1.select

2.select

Factory Method



1.select

2.select

Static method requires static fields!

3. Run test. If it passes like it should, then commit

Factory Method

1. Change the rest of the test so that they use a factory method
2. Do this also for liftoff

- If you haven't finished you can clone it from

git://github.com/odedlac/StrategyMethodFactory.git

Abstract Factory

Abstract Factory

AirplaneTest.java

```
@Test
public void test1() {

    String expectedOutput = "Like a fighter jet";
    String stringReturned = null;

    Flying fly = flyingFactory.createFlying("Fighter Jet");

    LiftOff liftOff = liftOffFactory.createLiftOff("Fighter Jet");

    Airplane classUnderTest = new Airplane(liftOff,fly);

    stringReturned = classUnderTest.howDoYouFly();

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

    }
```

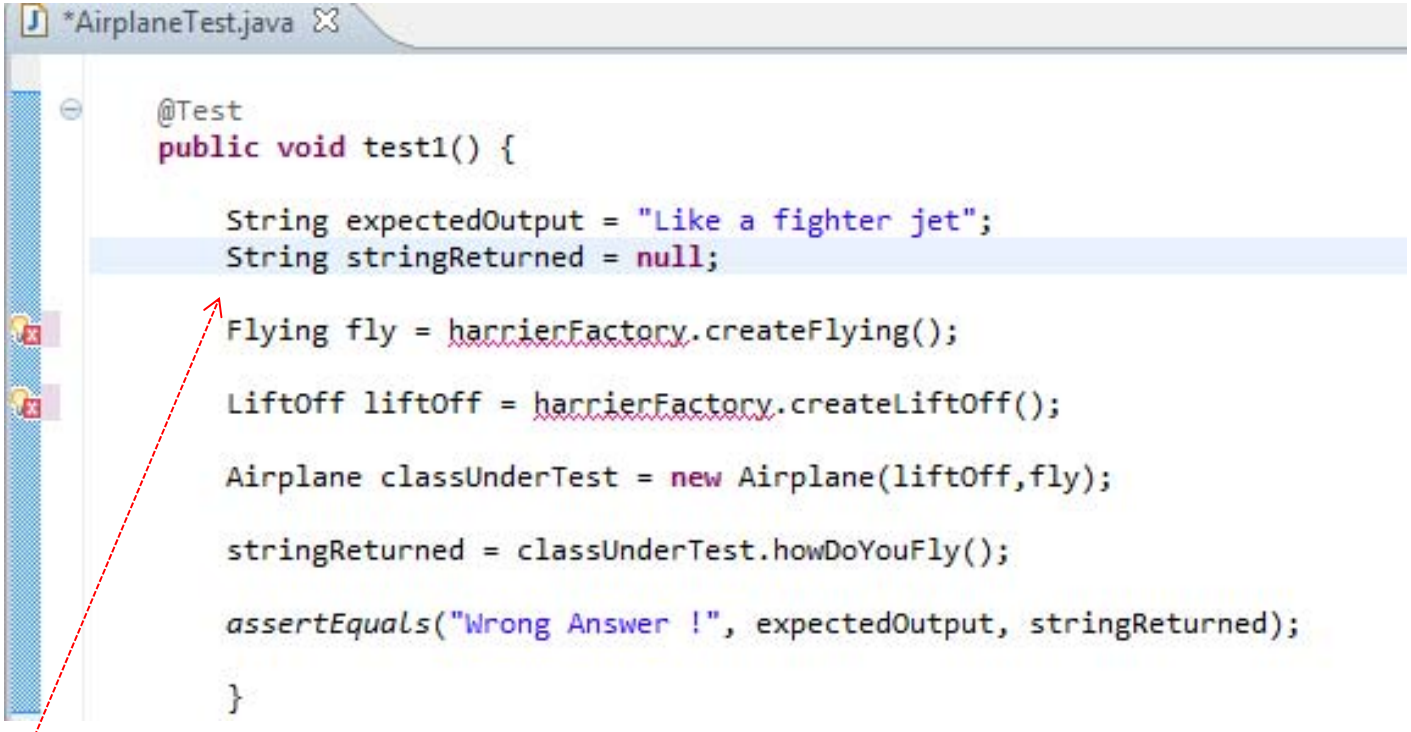
2. Replace with this code

1. Replace with this code

```
harrierFactory.createFlying();
```

```
harrierFactory.createLiftOff();
```

Abstract Factory



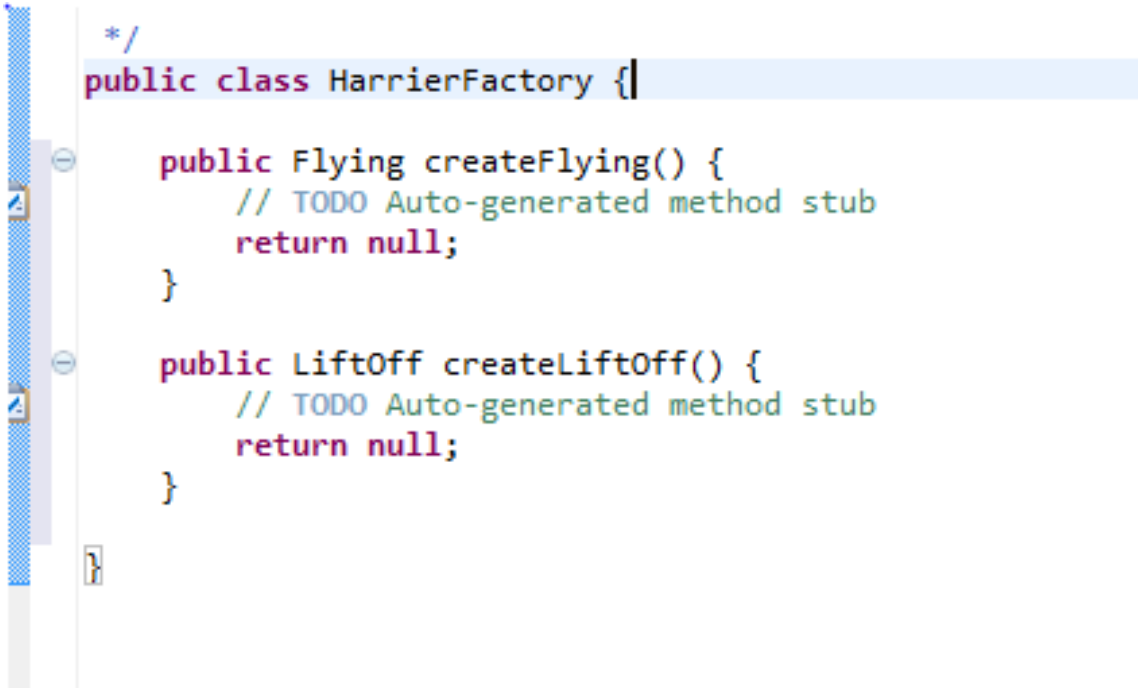
1. Insert code

HarrierFactory harrierFactory = **new HarrierFactory();**

Abstract Factory

1. Use existing code to generate ‘HarrierFactory’ code.
2. Use existing code to generate stubs for missing methods in ‘HarrierFactory’.

You should get



Abstract Factory

1. Code for ‘createFlying’ method in ‘HarrierFactory’

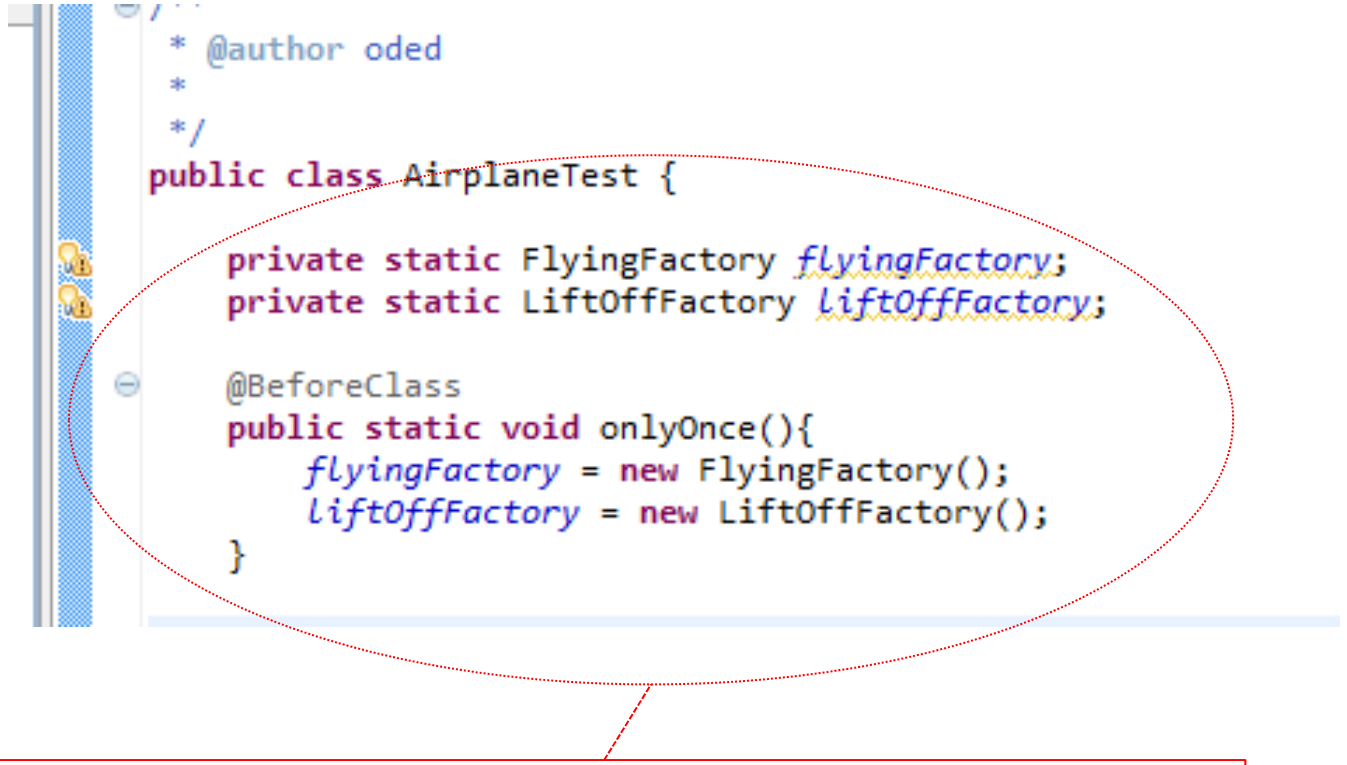
```
public Flying createFlying() {
        FlyingFactory flyingFactory = new FlyingFactory();
        return flyingFactory.createFlying("Fighter Jet");
}
```

2. Create your own code for ‘createTakeOff’ method in ‘HarrierFactory’

3. Exctract interface from ‘HarrierFactory’ and call it ‘AirPlaneFactory’.

4. Write a factory for all the kinds of airplanes

Abstract Factory



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

    private static FlyingFactory flyingFactory;
    private static LiftOffFactory liftOffFactory;

    @BeforeClass
    public static void onlyOnce(){
        flyingFactory = new FlyingFactory();
        liftOffFactory = new LiftOffFactory();
    }
```

1.This code has become redundant and can be deleted

- If you haven't finished you can clone it from

git://github.com/odedlac/StrategyAbstractFactory.git

Builder

Builder

1. If you haven't done so already. Clone code from:

git://github.com/odedlac/StrategyAbstractFactory.git

Builder

```
@Test
public void test1() {

    String expectedOutput = "Like a fighter jet";
    String stringReturned = null;

    AirPlaneFactory harrierFactory = new HarrierFactory();

    Flying fly = harrierFactory.createFlying();

    LiftOff liftOff = harrierFactory.createLiftOff();

    Airplane classUnderTest = new Airplane(liftOff,fly);

    stringReturned = classUnderTest.howDoYouFly();

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

    }
```

1.Replace code

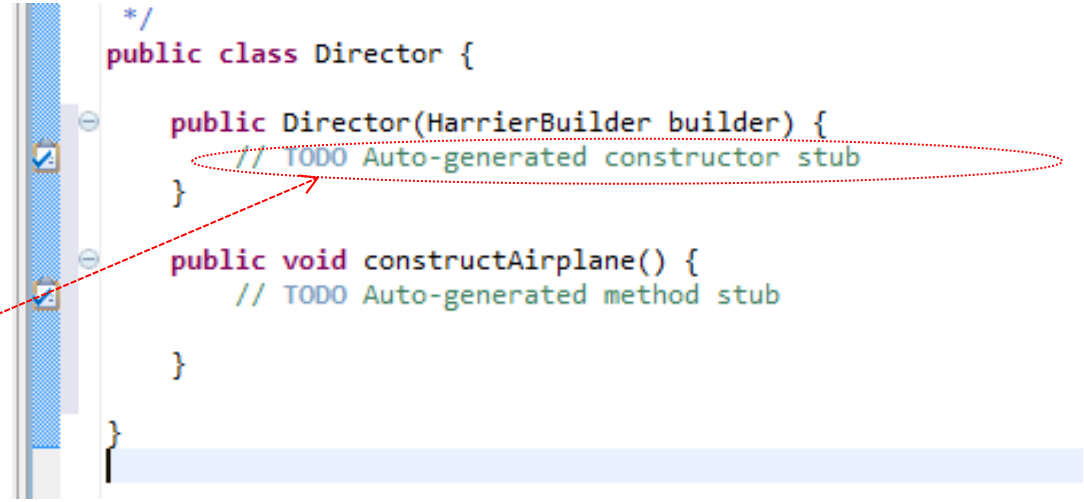
```
HarrierBuilder builder = new HarrierBuilder();

Director director = new Director(builder);

director.constructAirplane();

Airplane classUnderTest = builder.getAirplane();
```

Builder

Use new code to:

1. Generate ‘HarrierBuilder’ class
2. Generate ‘Director’ class
3. Generate constructor for ‘Director’ class
4. Generate ‘constructAirplane’ method in class ‘Director’
5. Generate ‘getAirplane’ method in class ‘HarrierBuilder’

Builder



```
 */
public class Director {

    public Director(HarrierBuilder builder) {
        // TODO Auto-generated constructor stub
    }

    public void constructAirplane() {
        // TODO Auto-generated method stub

    }

}
```

1.Replace code

```
this.builder = builder;
```

2. Use new code to generate field ‘builder’

Builder

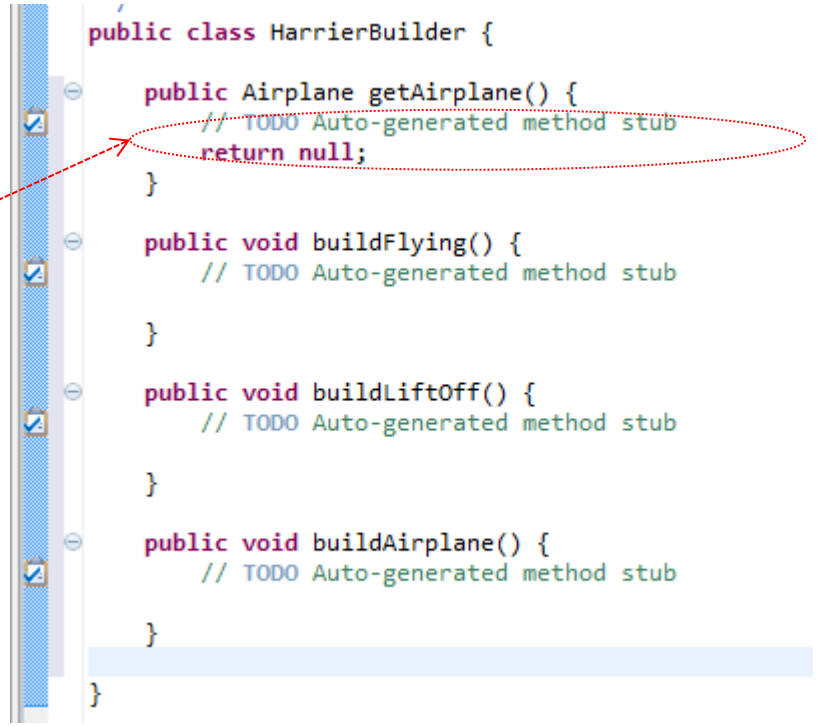
```
 */
public class Director {

    private HarrierBuilder builder;

    public Director(HarrierBuilder builder) {
        this.builder = builder;
    }

    public void constructAirplane() {
        // TODO Auto-generated method stub

    }

}
```

1.Replace code

```
builder.buildFlying();
builder.buildLiftOff();
builder.buildAirplane();
```

2. Use new code to generate new method in ‘builder’

Builder



```
public class HarrierBuilder {

    public Airplane getAirplane() {
        // TODO Auto-generated method stub
        return null;
    }

    public void buildFlying() {
        // TODO Auto-generated method stub

    }

    public void buildLiftOff() {
        // TODO Auto-generated method stub

    }

    public void buildAirplane() {
        // TODO Auto-generated method stub

    }

}
```

1.Replace code

return airplane;

2. Use new code to generate the field ‘airplane’

Builder

```
FlyingFactory flyingFactory = new FlyingFactory();
flying = flyingFactory.createFlying("Fighter Jet");
```

1.Replace code

```
public class HarrierBuilder {

    private Airplane airplane;

    public Airplane getAirplane() {
        return airplane;
    }

    public void buildFlying() {
        // TODO Auto-generated method stub

    }

    public void buildLiftOff() {
        // TODO Auto-generated method stub

    }

    public void buildAirplane() {
        // TODO Auto-generated method stub

    }

}
```

2.Replace code

```
LiftOffFactory liftOffFactory = new LiftOffFactory();
liftOff = liftOffFactory.createLiftOff("Vertically");
```

```
airplane = new Airplane(liftoff,flying);
```

3.Replace code

4. Use new code to generate missing fields

5. Run test if they all pass, then commit

Builder

1. Extract interface ‘Builder’ from ‘HarrierBuilder’
2. Run test if passes, then commit

Static Factory Method

Old

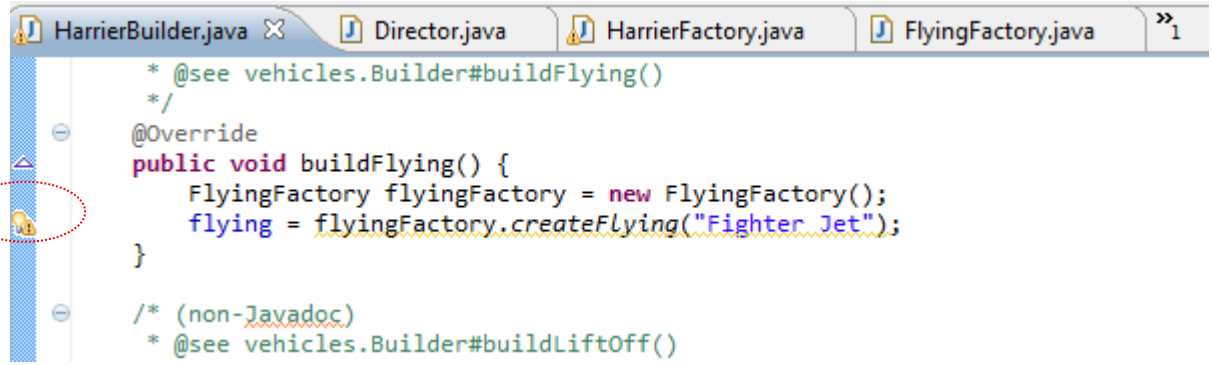
```
public class FlyingFactory {

    public Flying createFlying(String string) {
        if(string.equals("Fighter Jet")){
            return new IFlyLikeFJ();
        } else if (string.equals("Model Airplane")){
            return new IDontFly();
        } else if (string.equals("Passenger Airplane")){
            return new IFlyLikePP();
        } else {
            return null;
        }
    }

}
```

New (add ‘static’)

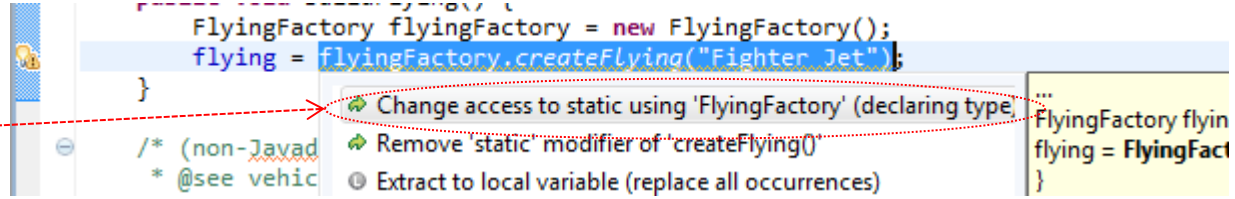
```
*/
public class FlyingFactory {

    public static Flying createFlying(String string) {
        if(string.equals("Fighter Jet")){
            return new IFlyLikeFJ();
        } else if (string.equals("Model Airplane")){
            return new IDontFly();
        } else if (string.equals("Passenger Airplane")){
            return new IFlyLikePP();
        } else {
            return null;
        }
    }

}
```

‘HarrierBuilder’ Code

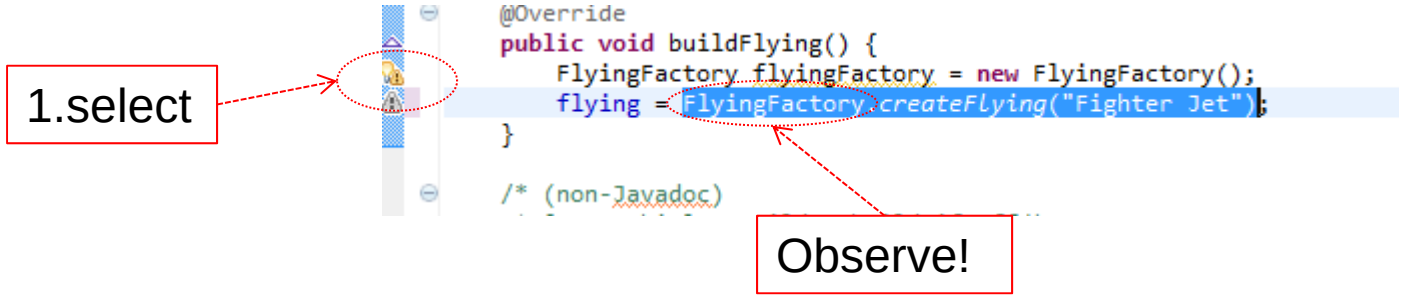
1.select

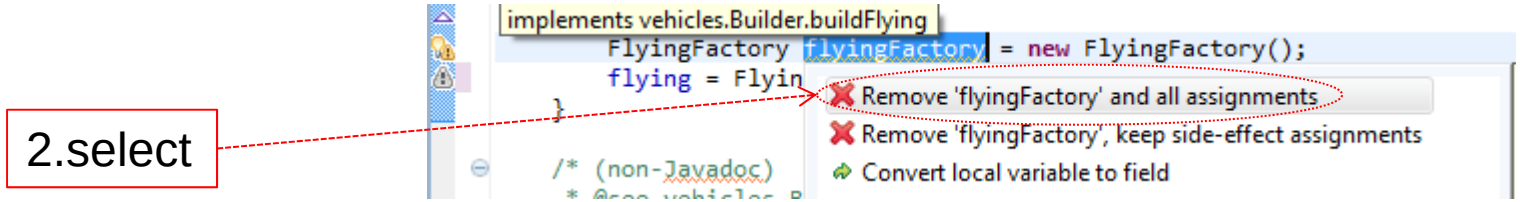


2.select



Builder


1.select
@Override
public void buildFlying() {
FlyingFactory flyingFactory = new FlyingFactory();
flying = FlyingFactory.createFlying("Fighter Jet");
}
/* (non-Javadoc)
Observe!


2.select
implements vehicles.Builder.buildFlying
FlyingFactory flyingFactory = new FlyingFactory();
flying = Flyin
}
/* (non-Javadoc)
Remove 'flyingFactory' and all assignments
Remove 'flyingFactory', keep side-effect assignments
Convert local variable to field

Builder

1. Apply the ‘static’ to ‘liftoffFactory’
2. Write and test a builder for model airplane and passenger airplane
3. Remove redundant code

(don’t forget to commit every time you are happy with the results)

4. Possible result: git://github.com/odedlac/StrategyBuilder.git
5. Why not get the builder from a factory method

(why might it be a problem to have a static director)