










# Design Patterns

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

 armahmood786@yahoo.com  
 alphapeeler.sf.net/pubkeys/pkey.htm  
 pk.linkedin.com/in/armahmood  
 www.twitter.com/alphapeeler  
 www.facebook.com/alphapeeler  
 abdulmahmood-sss  alphasecure  
 armahmood786@hotmail.com  
 <http://alphapeeler.sf.net/me>

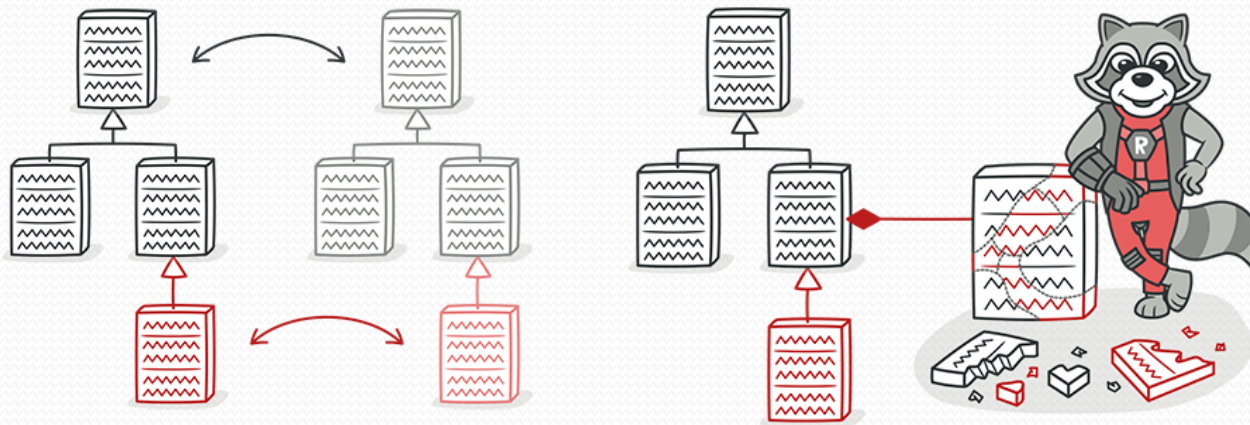
 alphasecure@gmail.com  
 <http://alphapeeler.sourceforge.net>  
 <http://alphapeeler.tumblr.com>  
 armahmood786@jabber.org  
 alphapeeler@aim.com  
 mahmood\_cubix  48660186  
 alphapeeler@icloud.com  
 <http://alphapeeler.sf.net/acms/>

# Refactoring-III

Bad Smells in Code

# Parallel Inheritance Hierarchies

- **Signs and Symptoms**
- Whenever you create a subclass for a class, you find yourself needing to create a subclass for another class.



**Treatment**  
Move Method and  
Move Field.

- **Reasons for the Problem**
- All was well as long as the hierarchy stayed small. But with new classes being added, making changes has become harder and harder.

# Parallel Inheritance Hierarchies

- Think about engineers — just engineers in general. Computer engineers work on computers and deliver projects, whereas civil engineer work on structures. From a design perspective, there are two parallel hierarchies:
  - Engineers
  - Milestones
- The different engineers have different milestones, and each engineer has a specified milestone (special relation).
- The problem is that every time you add a new engineer in the Engineer inheritance, you have to introduce a new Milestone in Milestone hierarchy.

# Parallel Inheritance Hierarchies

```
public interface Engineer {  
    String getType();  
    void setType(String type);  
    int getSalary();  
    void setSalary(int salary);  
    MileStone getMileStone();  
    void setMileStone(MileStone  
mileStone);  
}
```

```
public interface MileStone {  
    public String work();  
    public String target();  
}
```

# Parallel Inheritance Hierarchies

```
public class ComputerEngineer implements Engineer {  
    private String type;  
    private int salary;  
    private MileStone mileStone;  
    public void setType(String type) {  
        this.type = type;  
    }  
    public void setSalary(int salary) {  
        this.salary = salary;  
    }  
    public void setMileStone(MileStone mileStone) {  
        this.mileStone = mileStone;  
    }  
    @Override  
    public String getType() {  
        // TODO Auto-generated method stub  
        return type;  
    }  
    @Override  
    public int getSalary() {  
        // TODO Auto-generated method stub  
        return salary;  
    }  
}
```

# Parallel Inheritance Hierarchies

```
@Override
    public MileStone getMileStone() {
        // TODO Auto-generated method stub
        return mileStone;
    }
@Override
    public String toString() {
        return "ComputerEngineer [type=" + type + ", salary=" + salary
            + ", mileStone=" + mileStone + "]\n";
    }
}
```

# Parallel Inheritance Hierarchies

```
public class ComputerMileStone implements MileStone {
    @Override
    public String work() {
        return "Build a Billing CORBA Client Application";
    }
    @Override
    public String target() {
        return "Has to be finshed in 14 PD";
    }
    @Override
    public String toString() {
        return "ComputerMileStone [work()=" + work() + ", target()="
            + target() + "]";
    }
}
```



# Parallel Inheritance Hierarchies

```
public class CivilEngineer implements Engineer{
    private String type;
    private int salary;
    private MileStone mileStone;
    public void setType(String type) {
        this.type = type;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
    public void setMileStone(MileStone mileStone) {
        this.mileStone = mileStone;
    }
    @Override
    public String getType() {
        // TODO Auto-generated method stub
        return type;
    }
    @Override
    public int getSalary() {
        // TODO Auto-generated method stub
        return salary;
    }
}
```

# Parallel Inheritance Hierarchies

```
@Override
```

```
public MileStone getMileStone() {  
    // TODO Auto-generated method stub  
    return mileStone;  
}
```

```
@Override
```

```
public String toString() {  
    return "CivilEngineer [type=" + type + ", salary=" + salary  
        + ", mileStone=" + mileStone + "];"  
}
```

```
}
```

# Parallel Inheritance Hierarchies

```
public class CivilMileStone implements MileStone {
    @Override
    public String work() {
        // TODO Auto-generated method stub
        return "Creates Habib Bank Plaza";
    }
    @Override
    public String target() {
        // TODO Auto-generated method stub
        return "Has to be completed in 3 years";
    }
    @Override
    public String toString() {
        return "CivilMileStone [work()=" + work() + ",
            target()=" + target() + "];"
    }
}
```

# Parallel Inheritance Hierarchies

```
public class Manager {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Engineer comp = new ComputerEngineer();  
        comp.setType("Computer Engineer");  
        comp.setSalary(50000);  
        comp.setMileStone(new ComputerMileStone());  
        Engineer civil = new CivilEngineer();  
        civil.setType("Civil Engineer");  
        civil.setSalary(60000);  
        civil.setMileStone(new CivilMileStone());  
        System.out.println(comp);  
        System.out.println("*****");  
        System.out.println(civil);  
    }  
}
```

## Output:

```
ComputerEngineer [type=Computer Engineer, salary=50000,  
mileStone=ComputerMileStone [work()=Build a Billing CORBA Client Application,  
target()=Has to be finshed in 14 PD]]
```

\*\*\*\*\*

```
CivilEngineer [type=Civil Engineer, salary=60000, mileStone=CivilMileStone  
[work()=Creates Habib Bank Plaza, target()=Has to be completed in 3 years]]
```

# Parallel Inheritance Hierarchies

- **Sol 1: Technique:** Make a common interface and move methods from another interface.

```
public interface EngineerMileStone {  
    String getType();  
    void setType(String type);  
    int getSalary();  
    void setSalary(int salary);  
    public String work();  
    public String target();  
}
```

# Parallel Inheritance Hierarchies

```
public class ReFactorCivilEngineer implements EngineerMileStone {  
    private String type;  
    private int salary;  
    @Override  
    public String getType() {  
        return type;  
    }  
    @Override  
    public void setType(String type) {  
        this.type=type;  
    }  
    @Override  
    public int getSalary() {  
        return salary;  
    }  
    @Override  
    public void setSalary(int salary) {  
        this.salary=salary;  
    }  
    @Override  
    public String work() {  
        return "Creates Habib Bank Plaza";  
    }  
}
```

# Parallel Inheritance Hierarchies

```
@Override
    public String target() {
        return "Has to be completed in 3 years";
    }
@Override
    public String toString() {
        return "ReFactorCivilEngineer [type=" + type + ", salary=" +
salary
        + ", getType()" + getType() + ", getSalary()" + getSalary()
        + ", work()" + work() + ", target()" + target() + "];
    }
}
```

# Parallel Inheritance Hierarchies

```
public class RefactorComputerEngineer implements EngineerMileStone {
    private String type;
    private int salary;
    @Override
    public String getType() {
        return type;
    }
    @Override
    public void setType(String type) {
        this.type=type;
    }
    @Override
    public int getSalary() {
        return salary;
    }
    @Override
    public void setSalary(int salary) {
        this.salary=salary;
    }
    @Override
    public String work() {
        return"Build a Billing CORBA Client Application";
    }
}
```



# Parallel Inheritance Hierarchies

```
@Override
public String target() {
    return "Has to be finshed in 14 PD";
}
@Override
public String toString() {
    return "RefactorComputerEngineer [type=" + type + ", salary="
+ salary
+ ", getType()=" + getType() + ", getSalary()=" + getSalary()
+ ", work()=" + work() + ", target()=" + target() + "]";
}
}
```

# Parallel Inheritance Hierarchies

```
public class Manager {  
    public static void main(String[] args) throws  
        InstantiationException, IllegalAccessException {  
        EngineerMileStone comp = new RefactorComputerEngineer();  
        comp.setType("Computer Engineer");  
        comp.setSalary(50000);  
        EngineerMileStone civil = new ReFactorCivilEngineer();  
        civil.setType("Civil Engineer");  
        civil.setSalary(60000);  
        System.out.println(comp);  
        System.out.println("*****");  
        System.out.println(civil);  
    }  
}
```

## Output:

RefactorComputerEngineer [type=Computer Engineer, salary=50000,  
getType()=Computer Engineer, getSalary()=50000, work()=Build a Billing CORBA  
Client Application, target()=Has to be finished in 14 PD]

\*\*\*\*\*

ReFactorCivilEngineer [type=Civil Engineer, salary=60000, getType()=Civil  
Engineer, getSalary()=60000, work()=Creates Habib Bank Plaza, target()=Has  
to be completed in 3 years]

# Refactoring Demo

Eclipse

Rename...	Alt+Shift+R
Move...	Alt+Shift+V
Change Method Signature...	Alt+Shift+C
Extract Method...	Alt+Shift+M
Extract Local Variable...	Alt+Shift+L
Extract Constant...	
Inline...	Alt+Shift+I
Convert Local Variable to Field...	
Convert Anonymous Class to Nested...	
Move Type to New File...	
Extract Interface...	
Extract Superclass...	
Use Supertype Where Possible...	
Pull Up...	
Push Down...	
Extract Class...	
Introduce Parameter Object...	
Introduce Indirection...	
Introduce Factory...	
Introduce Parameter...	
Encapsulate Field...	
Generalize Declared Type...	
Infer Generic Type Arguments...	
Migrate JAR File...	
Create Script...	
Apply Script...	
History...	

# Eclipse - Extract Method


```
public class Test3 {  
    public static void main(String[] args) {  
        // corona registration  
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};  
  
        //Print cases *****  
        //Print Header  
        System.out.println("****Corona Cases Registered in Karachi****");  
  
        //Iterate  
        for (int cc:corons_cases_karachi) {  
            System.out.println(cc);  
        }  
  
        //Print Footer  
        System.out.println("*****");  
    }  
}
```

# Eclipse - Extract Method

- Select From Print Header to Print Footer code and Press: Alt+Shift+M

```
public class Test3 {  
    public static void main(String[] args) {  
        // corona registration  
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};  
  
        //Print cases *****  
        //Print Header  
        System.out.println("****Corona Cases Registered in Karachi****");  
  
        //Iterate  
        for (int cc:corons_cases_karachi) {  
            System.out.println(cc);  
        }  
  
        //Print Footer  
        System.out.println("*****");  
    }  
}
```

# Eclipse - Extract Method

 Extract Method

Method name:

Access modifier: ☐ public ☐ protected ☐ package ☒ private

Parameters:

Type	Name
int []	corons_cases_karachi

Edit...

Up

Down

☐ Declare thrown runtime exceptions  
☒ Generate method comment  
☐ Replace additional occurrences of statements with method

Method signature preview:  
**private static void** print\_cases(int []  
corons\_cases\_karachi)

Preview >

OK

Cancel

```
public class Test3 {
    public static void main(String[] args) {
        // corona registration
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};

        //Print cases *****
        print_cases(corons_cases_karachi);
    }

    /**
     * @param corons_cases_karachi
     */
    private static void print_cases(int[] corons_cases_karachi) {
        //Print Header
        System.out.println("****Corona Cases Registered in Karachi****");

        //Iterate
        for (int cc:corons_cases_karachi) {
            System.out.println(cc);
        }

        //Print Footer
        System.out.prinatln("*****")
    }
}
```



# output

```
****Corona Cases Registered in Karachi****
```

```
11  
34  
0  
45  
21  
33  
20  
11  
2  
1  
12
```

```
*****
```

- Output will be same before Refactoring and After Refactoring

# Eclipse - Inline Method

```
public class Test3 {
    public static void main(String[] args) {
        // corona registration
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};
        //Print cases *****
        print_cases(corons_cases_karachi);
    }
    private static void print_cases(int[] corons_cases_karachi) {
        //Print Header
        System.out.println("****Corona Cases Registered in Karachi****");
        //Iterate
        for (int cc:corons_cases_karachi) {
            System.out.println(cc);
        }
        //Print Footer
        System.out.prinatln("*****")
    }
}
```

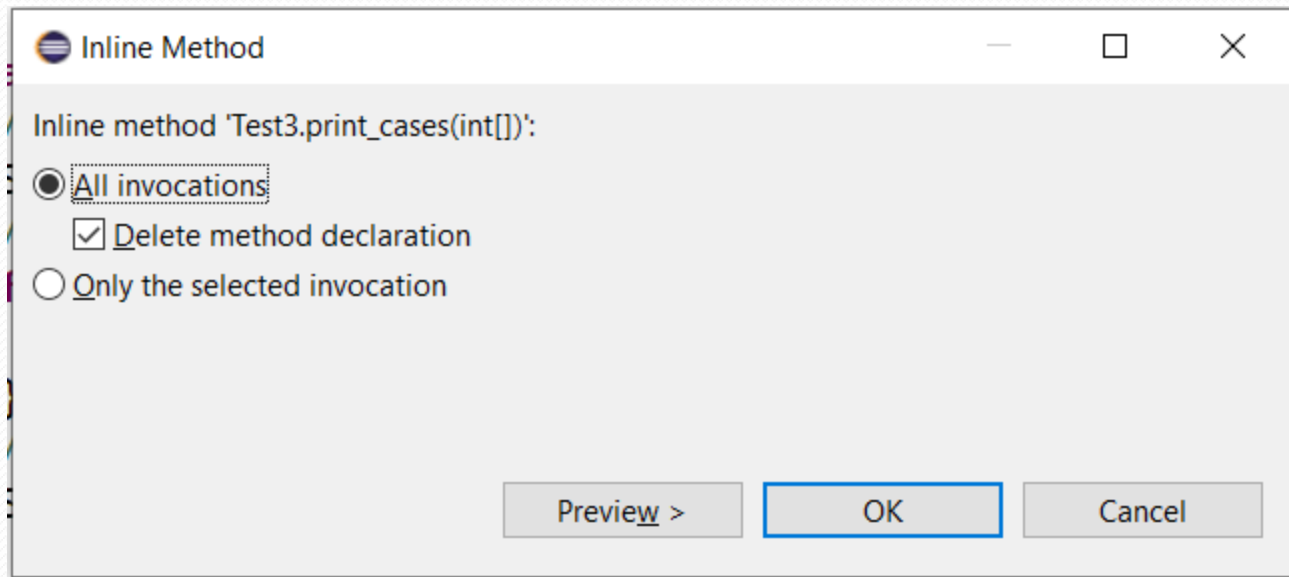
# C

- Select the method to perform Inline method refactoring and press Alt+Shift+I

```
public class Test3 {  
    public static void main(String[] args) {  
        // corona registration  
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};  
  
        //Print cases *****  
        print_cases(corons_cases_karachi);  
    }  
    private static void print_cases(int[] corons_cases_karachi) {  
        //Print Header  
        System.out.println("****Corona Cases Registered in Karachi****");  
        //Iterate  
        for (int cc:corons_cases_karachi) {  
            System.out.println(cc);  
        }  
        //Print Footer  
        System.out.println("*****");  
    }  
}
```

# Eclipse - Inline Method

- After pressing Alt+Shift+I you will see this dialog:



# Eclipse - Inline Method

- Here is the result of refactoring:

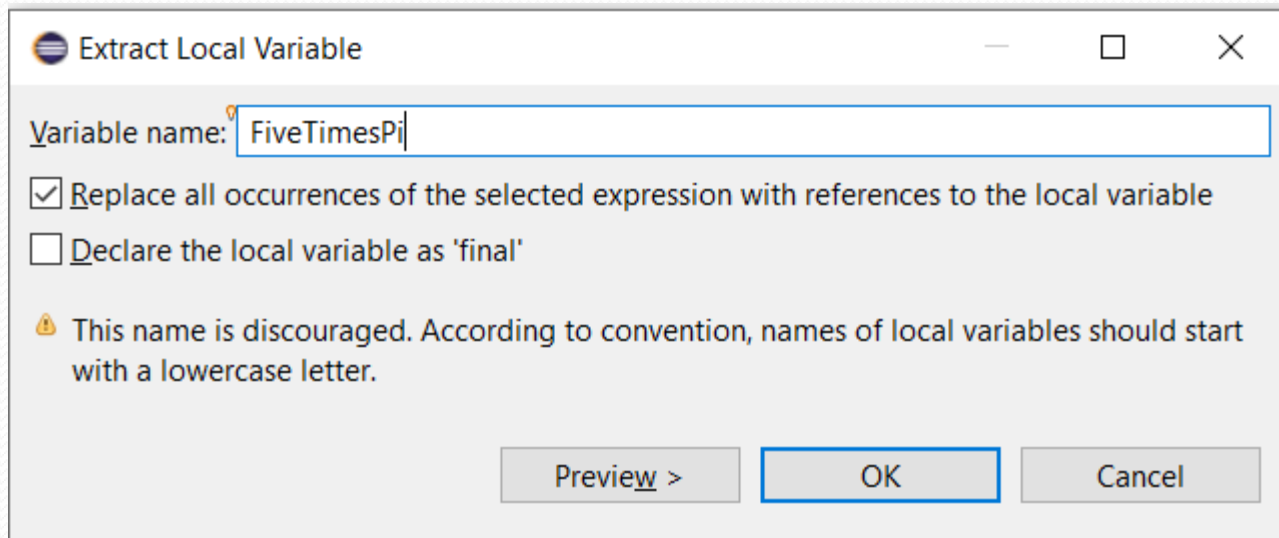
```
public class Test3 {
    public static void main(String[] args) {
        // corona registration
        int[] corons_cases_karachi = {11,34,0,45,21,33,20,11,2,1,12};

        //Print cases *****
        //Print Header
        System.out.println("****Corona Cases Registered in Karachi****");
        //Iterate
        for (int cc:corons_cases_karachi) {
            System.out.println(cc);
        }
        //Print Footer
        System.out.println("*****");
    }
}
```

# Eclipse - Extract Local Variable

```
public class ExtractLocalVar {  
    public static void main(String[] args) {  
        double pi = 3.14159265359;  
        System.out.println(pi*5);  
        System.out.println((pi*5)/7);  
        System.out.println((pi*5)+4);  
    }  
}
```

Select **pi\*5** and press Alt+Shift+L



# Eclipse - Extract Local Variable

```
public class ExtractLocalVar {  
    public static void main(String[] args) {  
        double pi = 3.14159265359;  
        double FiveTimesPi = pi*5;  
        System.out.println(FiveTimesPi);  
        System.out.println(FiveTimesPi/7);  
        System.out.println(FiveTimesPi+4);  
    }  
}
```

Output:

15.70796326795

2.243994752564286

19.70796326795

- Output will be same before Refactoring and After Refactoring

# Change Method signature

```
public class Test5{  
    public Test5() {  
    }  
}
```

```
public void Fun1(int i, double d) {  
    System.out.println("i = " + i);  
    System.out.println("d = " + d);  
}
```

```
public static void main(String[] args) {  
    Test5 t5 = new Test5();  
    t5.Fun1(1, 2.5);  
}
```

**Output:**

i = 1

d = 2.5

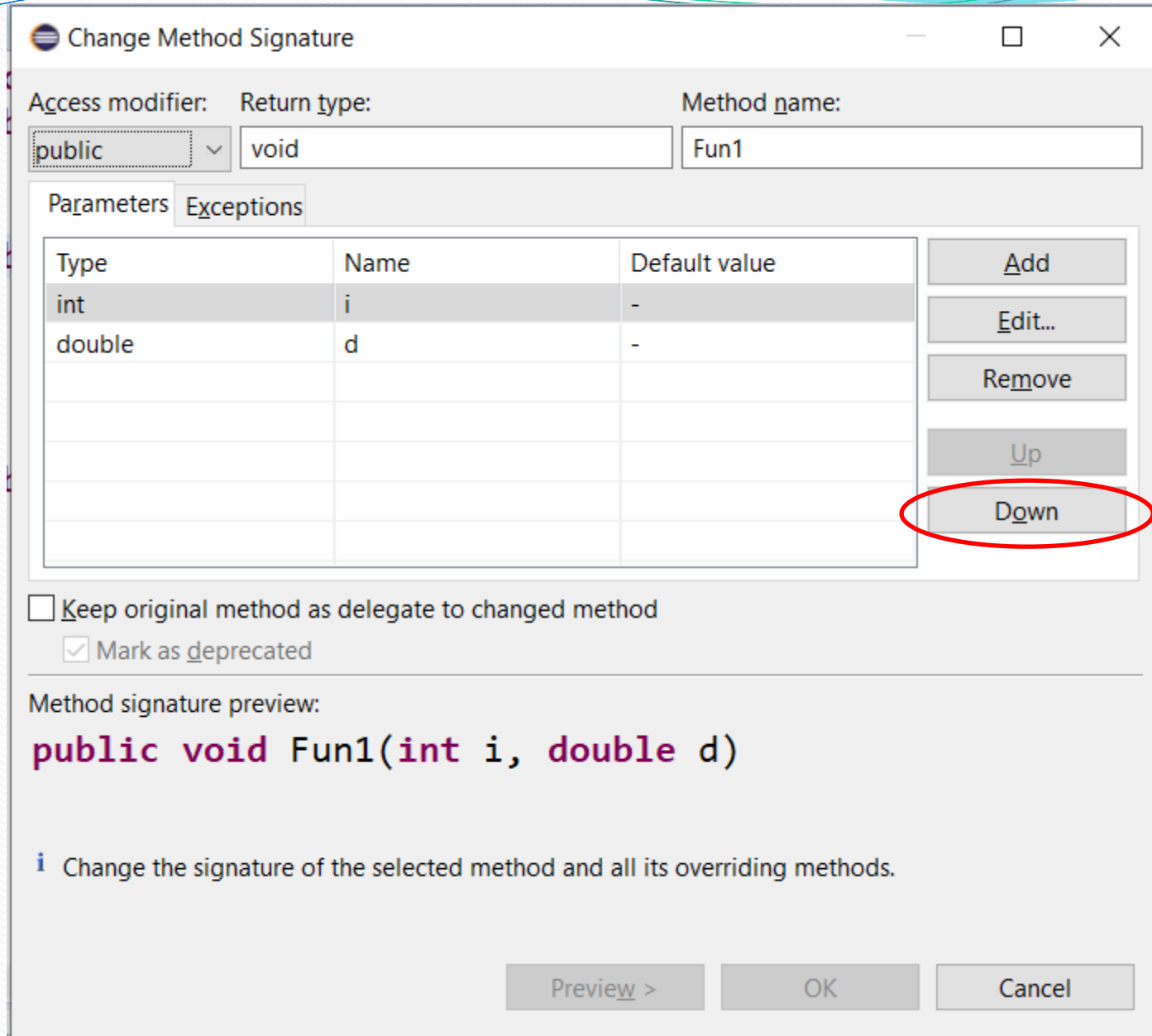


# Change Method signature

- Select function and press Alt+Shift+C

```
public class Test5{  
    public Test5() {  
    }  
  
    public void Fun1(int i, double d) {  
        System.out.println("i = " + i);  
        System.out.println("d = " + d);  
    }  
  
    public static void main(String[] args) {  
        Test5 t5 = new Test5();  
        t5.Fun1(1, 2.5);  
    }  
}
```

# Change Method signature



The dialog box is titled "Change Method Signature". It has three input fields at the top: "Access modifier:" with a dropdown menu showing "public", "Return type:" with a text box containing "void", and "Method name:" with a text box containing "Fun1". Below these are two tabs: "Parameters" (selected) and "Exceptions". The "Parameters" tab contains a table with columns "Type", "Name", and "Default value". The table has two rows: one with "int", "i", and "-" and another with "double", "d", and "-". To the right of the table are five buttons: "Add", "Edit...", "Remove", "Up", and "Down". The "Down" button is circled in red. Below the table are two checkboxes: "Keep original method as delegate to changed method" (unchecked) and "Mark as deprecated" (checked). At the bottom, there is a "Method signature preview:" section showing the code `public void Fun1(int i, double d)`. Below the preview is an information icon and the text "Change the signature of the selected method and all its overriding methods." At the very bottom are three buttons: "Preview >", "OK", and "Cancel".

Change Method Signature

Access modifier: public Return type: void Method name: Fun1


Parameters Exceptions

Type	Name	Default value
int	i	-
double	d	-

Add Edit... Remove Up Down


☐ Keep original method as delegate to changed method  
☒ Mark as deprecated

Method signature preview:  
**public void Fun1(int i, double d)**

 Change the signature of the selected method and all its overriding methods.

Preview > OK Cancel

# Change Method signature

 Change Method Signature — □ ×

Access modifier: public ▼

Return type: void

Method name: Fun1

Parameters

Exceptions

Type	Name	Default value
double	d	-
int	i	-

Add

Edit...

Remove

Up

Down

☐ Keep original method as delegate to changed method  
☒ Mark as deprecated

Method signature preview:  
**public void Fun1(double d, int i)**

Preview >

OK

Cancel

# Change Method signature

```
public class Test5{  
    public Test5() {  
    }  
}
```

```
public void Fun1(double d, int i){  
    System.out.println("i = " + i);  
    System.out.println("d = " + d);  
}
```

```
public static void main(String[] args) {  
    Test5 t5 = new Test5();  
    t5.Fun1(2.5, 1);  
}
```

**Output:**

i = 1

d = 2.5