

# Refactoring with IntelliJ

A short guide



# Motivation

- Save transformation
- Work is easier
- Use powerfull tooling
- Decide about outcome
- Learn the IDE



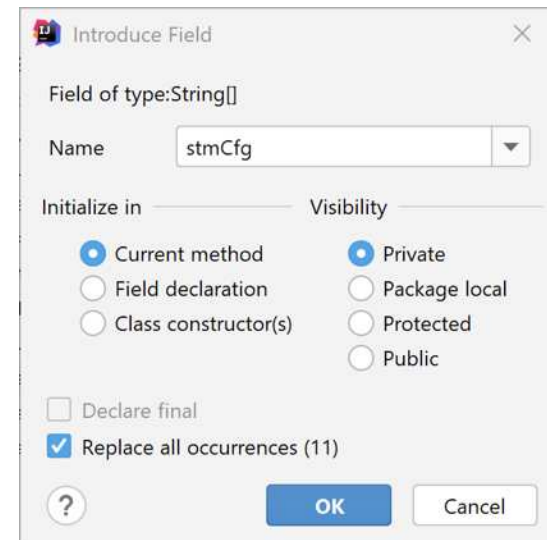
# Extract objects

- Done with Ctrl – Alt – type of extraction
  - M extract Method
  - V extract Variable
  - C extract Constant
  - F extract Field
  - P extract Parameter

# Field extraction

Extract field with **Ctrl-Alt-F**

```
StringWriter sw = new StringWriter();
sw.append(Globals.StmCfg[0]);
sw.append("\n");
for (String key : gt.keySet()) {
    var t : Quintet<String, Date, String, Integer, Integer> = gt.get(key);
    String sn = t.getValue0();
    int nOfShs = t.getValue4();
    DecimalFormat dc = new DecimalFormat(Globals.StmCfg[1]);
    SimpleDateFormat df = new SimpleDateFormat(Globals.StmCfg[2]);
    var cl : HttpClient = HttpClient.newHttpClient();
    var r : HttpRequest = HttpRequest
```



The dialog box 'Introduce Field' is shown with the following settings:

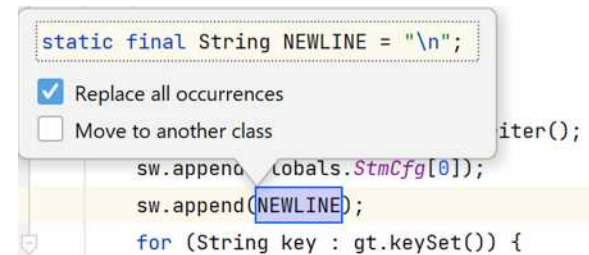
- Field of type: String[]
- Name: stmCfg
- Initialize in: Current method (selected)
- Visibility: Private (selected)
- Other options: Field declaration, Class constructor(s), Package local, Protected, Public (all unselected)
- Declare final:
- Replace all occurrences (11):
- Buttons: ? (help), OK, Cancel

```
String[] stmCfg = Globals.StmCfg;
sw.append(stmCfg[0]);
sw.append("\n");
for (String key : gt.keySet()) {
    var t : Quintet<String, Date, String, Integer, Integer> = gt.get(key);
    String sn = t.getValue0();
    int nOfShs = t.getValue4();
    DecimalFormat dc = new DecimalFormat(stmCfg[1]);
    SimpleDateFormat df = new SimpleDateFormat(stmCfg[2]);
    var cl : HttpClient = HttpClient.newHttpClient();
    var r : HttpRequest = HttpRequest
        .newBuilder(URI.create(stmCfg[3] + URLEncoder.encode(
            stmCfg[4], stmCfg[5]))
```

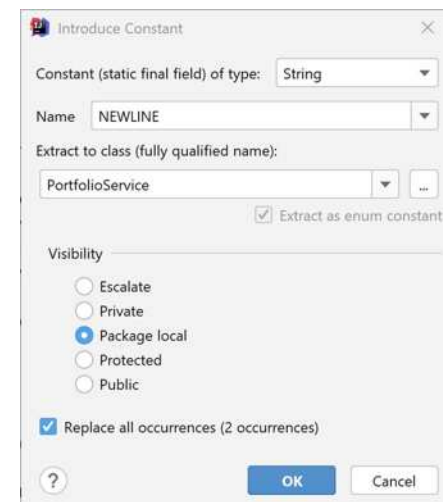
# Extract constant

Extract field with **Ctrl-Alt-C**

```
StringWriter sw = new StringWriter();  
sw.append(Globals.StmCfg[0]);  
sw.append("\n");  
for (String key : gt.keySet()) {  
    var t : Quintet<String, Date, String, Integer, Integer> = gt.get(key);  
    String sn = t.getValue0();
```

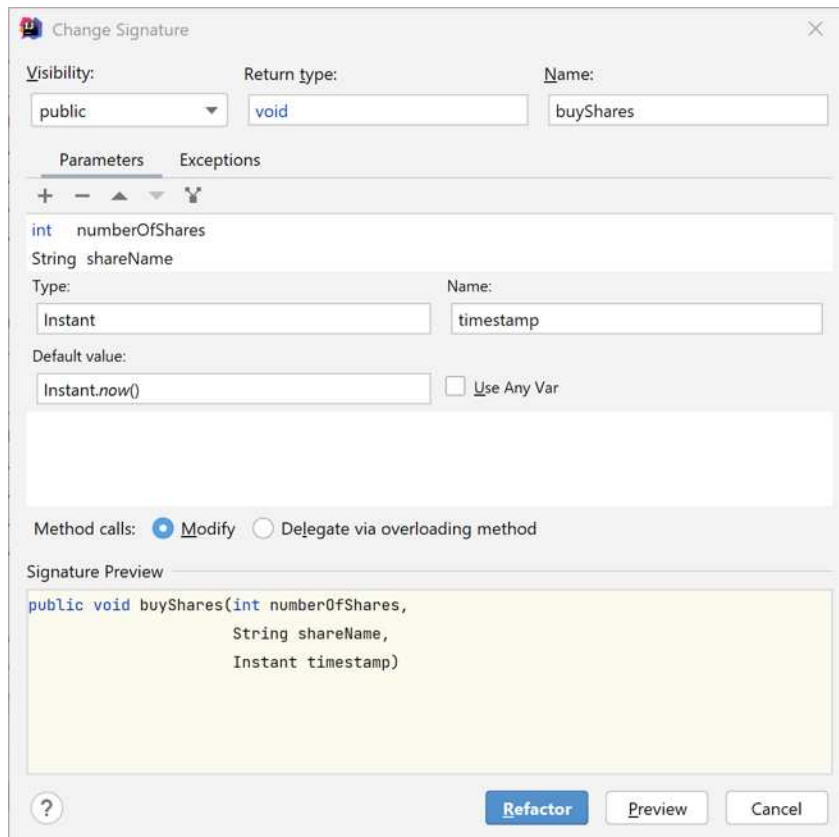


Press Ctrl-Alt-C again



# Change signature

Change of method signature Ctrl-F6



- Change name of method
- add/remove parameters
- give new parameters default
- rename parameters
- change order of parameters
- use preview

# Extract superclass

```
public class MessageForwarder {
    String[] listAddress = new String[]{};
    MessageSender sender;
    SessionHandler sessionHandler;

    public MessageForwarder(SessionHandler sessionHandler, MessageSender sender) {
        this.sessionHandler = sessionHandler;
        this.sender = sender;
    }

    public void forward(Message message) {
        Message messageToForward =
            createForwardMessage(sessionHandler.getSession(), message);
        sender.send(messageToForward);
    }

    private String[] getMailListAddress() { return new String[]{}; }

    private Message createForwardMessage(Session session, Message message) {
        Message forward = new MimeMessage(session);
        forward.setFrom(getFromAddress(message));
    }
}
```

Extract Superclass

Extract superclass from:  
DependencyBreaking.MessageForwarder

Extract superclass  Rename original class and use superclass where possible

Superclass name:  
AbstractMessageForwarder

Package for new superclass:  
DependencyBreaking

Target destination directory:  
...src\main\java\DependencyBreaking

Member	Make Abstract
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> forward(message:Message):void	<input checked="" type="checkbox"/>
<input type="checkbox"/> <input type="checkbox"/> createForwardMessage(session:Session, message:Message):Message	<input type="checkbox"/>
<input checked="" type="checkbox"/> <input type="checkbox"/> getMailListAddress():String[]	<input type="checkbox"/>
<input type="checkbox"/> <input type="checkbox"/> getFromAddress(message:Message):String	<input type="checkbox"/>
<input type="checkbox"/> <input type="checkbox"/> buildForwardContent(message:Message, forward:Message):void	<input type="checkbox"/>
<input checked="" type="checkbox"/> <input type="checkbox"/> listAddress:String[]	<input type="checkbox"/>
<input type="checkbox"/> <input type="checkbox"/> sender:MessageSender	<input type="checkbox"/>
<input type="checkbox"/> <input type="checkbox"/> sessionHandler:SessionHandler	<input type="checkbox"/>

JavaDoc for abstracts:  
 As is  
 Copy  
 Move

Refactor Preview Cancel

```
public class MessageForwarder extends AbstractMessageForwarder {
    MessageSender sender;
    SessionHandler sessionHandler;

    public MessageForwarder(SessionHandler sessionHandler, MessageSender sender) {
        this.sessionHandler = sessionHandler;
        this.sender = sender;
    }

    @Override
    public void forward(Message message) {
        Message messageToForward =
            createForwardMessage(sessionHandler.getSession(), message);
        sender.send(messageToForward);
    }


    private Message createForwardMessage(Session session, Message message) {
        Message forward = new MimeMessage(session);
        forward.setFrom(getFromAddress(message));
    }
}

public abstract class AbstractMessageForwarder {
    String[] listAddress = new String[]{};

    public abstract void forward(Message message);

    protected String[] getMailListAddress() { return new String[0]; }
}
```

# Conclusions

- use IDE when ever possible
- manual work much more error-prone
- easier then in Eclipse
- it's fun with the right tools
- try *analyze* possibilities 
- look for further refactoring possibilities





# References

- Code refactoring on jetbrains:  
<https://www.jetbrains.com/help/idea/refactoring-source-code.html>
- Keyboard shortcuts:  
<https://www.jetbrains.com/help/idea/reference-keymap-win-default.html>
- In IntelliJ: Help → Keymap reference

Thanks for your attention and have fun with IntelliJ

# code refactoring → art style



```
CodeReorderer.java | ordered.txt
1 package DependencyBreaking;
2
3 import java.io.IOException;
4 import java.nio.file.Files;
5 import java.nio.file.Path;
6 import java.util.*;
7 import java.util.stream.Collectors;
8 import java.util.stream.Stream;
9
10 public class CodeReorderer {
11
12     public static void main(String[] args) throws IOException {
13         Path fileName = Path.of( first: "src/main/java/DependencyBreaking/CodeReorderer.java");
14         String actual = Files.readString(fileName);
15
16         CodeReorderer codeReorderer = new CodeReorderer();
17         Map<String, Long> characterCountMap = codeReorderer.getCharacterCountMap(actual);
18
19         Path orderedFile = Path.of( first: "ordered.txt");
20         String result = codeReorderer.extractKeyMap(characterCountMap);
21         Files.writeString(orderedFile, result);
22     }
23
24     public Map<String, Long> getCharacterCountMap(final String input) {
25         return Stream.of(input.split( regex: ""))
26             .filter(this::isPrintable)
27             .collect(Collectors.groupingBy(s -> s, TreeMap::new, Collectors.counting()));
28     }
29
30     public String extractKeyMap(Map<String, Long> characterMap) {
31         return characterMap.entrySet().Set<Map<K, V>.Entry<String, Long>>
32             .stream() Stream<Map<K, V>.Entry<String, Long>>
33             .map(es -> String.join( delimiter: "", Collections.nCopies(es.getValue().intValue(),
34             .collect(Collectors.joining( delimiter: "\n"));
35     }
36
37     private boolean isPrintable(final String character) {
38         return character.charAt(0) > ' ';
39     }
40 }
41
1 *****
2 **
3 (((((((((((((((((((((((((((((((
4 ))))))))))))))))))))))))))))))
5 *
6 .....
7 --
8 .....
9 ////
10 0
11 :::
12 ;;;;;;;;;;;;;;;;;;
13 <<<
14 =====
15 >>>>>
16 A
17 BBB
18 CCCCCCCCCCCCCC
19 DD
20 EE
21 FFFFF
22 II
23 KKK
24 LLL
25 MMMMMMMMMMM
26 NN
27 OO
28 PPPPPPP
29 RRRRRRR
30 SSSSSSSSSSSSSS
31 T
32 VV
33 [
34 \
35 ]
36 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
37 bbbbbbb
38 ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
39 ddddddddddddddddddd
40 eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
41 ffffffff
42 ggggggggggggggggggggggggggggggggg
43 hhhhhhhhhhhhhhh
44 iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
45 jjjjjjjjj
46 kkk
```