Experiment

Dear participant:

This type of study can bring you significant gains in your software development activities since, based on the evaluation of tools, we can contribute to its improvement and, with this, make available to you tools with more quality.

You are being invited to participate in this experiment because of your experience in development activities. If you decide to participate, you will be asked to use tools that you may or may not have used previously . During the activities, I will observe how you use this tool and, during its use, I will collect information as we progress in the execution of the tasks so that I understand what (and why) you are doing certain actions.

I herewith make it clear that any information obtained in this study that may have any relationship with you (that is, that may help to identify you) will be kept confidential. Personal data (name of the participant, company data, among others) will be kept confidential since we will not store any information of this nature.

Your participation is completely voluntary. Your decision to participate or not, will not affect your relationship with this researcher or even with the institution. If you decide to pair ticipar the study, you can choose to withdraw - if at any time without any penalties. Please keep a copy of this letter for the purpose of recording the information contained herein.

If you have any questions, problems or suggestions about your participation in this study or your rights as a research participant, please contact this researcher.

Sincere Thanks

Experiment setup

Date: Number of Participant: __________ _______

Material *Checklist*

☐ Clipboard

☐ Evaluation sheets

☐ Questionnaires

☐ *Smell cards* in the envelope

☐ Pens

☐ Mouse

☐ Laptop with charger

☐ clock

Settings

☐ Eclipse version 3.3

☐ Installing the *Stench Blossom Plugin*

☐ Import the project

☐ Leave demo classes previously open

☐ Show lines of code in eclipse

☐ Leave the code tab in full screen

☐ Sitting next to the participant

☐ 3X8 cards in hand

☐ Clipboard with questionnaires and forms

Completion

☐ Conduct the pseudo-interview

☐ Thank the participant and release him

☐ Check date and time on the forms

☐ Close all codes (except for demonstration)

☐ Take notes

Quiz - Pre-Experiment

The following questionnaire aims to provide *background* information as a developer. Your answers will not affect the other activities of this experiment, they will simply provide a context for interpreting the results.

Feel free to write in the margins to explain your answers, if necessary .

1) You work as: _______

2) How long have you been a developer? _ _ _______

3) During the past twelve months, how many hours a week would you say you spend on development (on average)? _ _______

4) How proficient, on a scale of 0 to 4 (where “non-proficient” and 4 “very proficient *” means*), do you consider yourself in relation to the following languages?

a) Java 0 1 2 3 4

b) C ++ 0 1 2 3 4

c) C # 0 1 2 3 4

d) PHP 0 1 2 3 4

e) 0 1 2 3 4_______

5) How often, on a scale of 0 to 4 (where “I never use it” and 4 “I use it very often ”) do you use the following languages?

a) Java 0 1 2 3 4

b) C ++ 0 1 2 3 4

c) C # 0 1 2 3 4

d) PHP 0 1 2 3 4

e) 0 1 2 3 4_______

6) When programming, do you usually use an IDE (Y / N)? If so, which do you use? And for how long? _______

________ ________

If not, why don't you use it?

____

7) If you use any type of IDE, do you usually use some tools that you consider useful for development (search engines, debuggers, visualization of the code structure, metrics, etc.) (Y / N)? ___

If so, which do you use ? And for how long?

________ ________

If not, why don't you use it?

____

8) On a scale of 0 to 4, how familiar are you with the practice of refactoring? (0 = unfamiliar and 4 = very familiar)

a) Refactoring 0 1 2 3 4

If positive, done manually or which (is) tool (s) you use? _

____

9) On a scale of 0 to 4, how familiar are you with analysis / detection of *code smells* ? (0 = unfamiliar, 4 = very familiar)

a) *Code Smells* 0 1 2 3 4

If positive, done manually or which (is) tool (s) you use? _

____

Pick up before the experiment

Experimental procedure

Introduction

What we are going to do in this experiment is to investigate *code smells* that were proposed in Martin Fowler's book (Refactoring: Improving the Design of Existing Code). The idea defined by this author is that *code smells* can assist in the identification of stretches of *code* that are candidates for refactoring. For example, the " *Long Method* " *smell* suggests that you might need to perform the " *Extract Method* " refactoring .

You do not need to be very familiar with these concepts. We are going to make a few comments as we go through this experiment. Feel free to ask questions.

This experiment will consist of 4 parts:

I) In the first part (I), we will conduct a small training on the conceptualization of *code smells* and the forms of code incidence .

II) In the second part (IIA and IIB), we will provide you with a tool to assist in the task of analyzing / detecting *code smells* .

III) In the third part (IIIA and IIIB), we will use the information obtained through the same tool (from the previous step) in order to make judgments for refactoring actions .

IV) In the fourth part (IVA and IVB), we will answer a post-experiment questionnaire and collect the general comments of this experiment.

[I] Training

You are currently viewing 3X8 cards. Each contains the name of a *code smell* and its definition. On its back, we put a small piece of code exemplifying the occurrence of this *code smell* . You will be given 5 minutes to read the cards .

If you have the correct understanding of the concepts and example and want to end this activity ahead of time, just let us know!

After this activity then, you will return them and, when you are ready, please let us know!

After the initial estimated time , let's start looking at some snippets of code. Any questions?

At this point I will hand over some cards for answers and await their conclusion. If it takes longer than the established time (I will inform you before the activity) I will ask if the cards have been properly filled out.

If the completion is not completed, please advise (put a note “Not Done”).

Now let's look at snippets of code. While you are checking these code snippets, please do not modify them, or even browse outside the editor. As a general rule, try not to spend too much time (more than 3-5 minutes) per file.

[IIA] Manual Search

Take the cards (if applicable). Now I am going to ask you to look at some code files in Java and try to detect any of the *code smells* listed on the cards. You will go through this code file (via the scroll bar). If you identify a *code smell* , just fill out the form with the code line and the identified *code smell* .

Exemplify. So, for example, you will go through this file from top to bottom, observing the occurrence of any *code smell* .

Doubts?

Scroll 1		**Scroll 2**	
Code Smell	**Line**	**Code Smell**	**Line**
Large Method		Large Method	
Large class		Large class	
Data Clumps		Data Clumps	
Message Chains		Message Chains	
Switch Statements		Switch Statements	
Typecast		Typecast	
Instanceof		Instanceof	
Feature Envy		Feature Envy	
Scroll 3		**Scroll 4**	
Code Smell	**Line**	**Code Smell**	**Line**
Large Method		Large Method	
Large Class		Large Class	
Data Clumps		Data Clumps	
Message Chains		Message Chains	
Switch Statements		Switch Statements	
Typecast		Typecast	
Instanceof		Instanceof	
Feature Envy		Feature Envy	

[IIB] Search with the aid of the tool

Take the cards (if applicable). I will ask you to identify some of the *code smells* that you saw in the letters. You will browse through a Java code file (through the scroll bar) with the aid of a *Smell Detector* tool .

Tool training . The tool is represented by a line of visualization behind its Java code (Current context and *ambient view*). It is similar to the petals of a flower (the tool's own name suggests - *Stench Blossom*).

Each petal represents a *code smell* , and we can hover over it to see its name (*Active view*). The size of the petal represents the "how strong it can be" the occurrence of this *code smell* in the code you are looking at. We can click on the petal and get more information about the *code smell* (*Explanation view*).

This part of the tool is intended to give you an idea of what *code smells* are present. There are more details that we can quote about the tool, but we will comment on it later.

Accomplishment of the task. So, the objective of our task is to check if the tool helps to identify any *code smell* (and that you believe that, in fact, this occurrence actually exists - remember the tool may have problems in its accuracy (explain the concept of for word), just fill the *form* with the code line and the pattern found.

Ready?

Scroll 1		Scroll 2	
Code Smell	**Line**	**Code Smell**	**Line**
Large Method		*Large Method*	
Large class		*Large class*	
Data Clumps		*Data Clumps*	
Message Chains		*Message Chains*	
Switch Statements		*Switch Statements*	
Typecast		*Typecast*	
Instanceof		*Instanceof*	
Feature Envy		*Feature Envy*	
Scroll 3		**Scroll 4**	
Code Smell	**Line**	**Code Smell**	**Line**
Large Method		*Large Method*	
Large Class		*Large Class*	
Data Clumps		*Data Clumps*	
Message Chains		*Message Chains*	
Switch Statements		*Switch Statements*	
Typecast		*Typecast*	
Instanceof		*Instanceof*	
Feature Envy		*Feature Envy*	

[IIIA] Refactoring - Manual

Now, what we're going to do is look for a *code smell* with particular characteristics (and in greater depth): *Feature Envy* . Suppose you analyze the occurrence of this *code smell* through visual inspection.

Analyzing the context, I could conclude, for example, that the method (NAME OF THE METHOD), or some parts of it, must be changed from the class (NAME OF THE CLASS) to another class (NAME OF THE OTHER CLASS).

So, the task I want you to do is to make some judgments about the need to do refactoring in the code: How dispersed is the *Feature Envy* , what is the probability of removing it, and how can you do this removal. I will ask these questions while you work, and if you have any questions, feel free to ask. When you're done, let us know!

Any question?

Perform these steps in another method.

Feature Envy [1]	
How scattered is it in the code?	
How likely is it to remove it?	

How best to remove it?	
Feature Envy [2]	
How scattered is it in the code?	
How likely is it to remove it?	
How best to remove it?	

[IIIB] Refactoring - Use of the tool

Now, what we're going to do is look for a *code smell* with particular characteristics (and in greater depth): *Feature Envy* . (Open the tool) and check the *ac view* and *explanation view* for more details on the occurrence of this *code smell* .

Opened to the *explanation view wizard,* the color gradation of the classes affected by the *Feature Envy* will be demonstrated . Thus, we can see that many members of the class (CLASS NAME) are referenced, but only one member of this class (CLASS NAME) is referenced. Associate members are highlighted in the source code.

Looking at these details, we can conclude that the method (METHOD NAME), or some parts of it, must be changed to the class (CLASS NAME).

So, the task I want you to do is to make some judgments about the need for refactoring in the code; How scattered is the *Feature Envy* , how likely is it to remove it, and how can you do it. I will ask these questions while you work, and if you have any questions, feel free to ask. When you're done, let me know.

The lguma question? Perform these steps in another method.

Feature Envy [1]	
How scattered is it in the code?	
How likely is it to remove it?	
How best to remove it?	
Feature Envy [2]	
How scattered is it in the code?	

How likely is it to remove it?	
How best to remove it?	