id. Response ID	A1	A2	A3	A4	A5	
p1	+3 years	+3 years	Academic knowledge	Theoritical knowledge	Singleton will only be instantiated once, a prototype will be instantiated every time an object is called.	
p2	+3 years	+3 years	Practical knowledge	Practical knowlegde	For the Singleton pattern only one instance is ensured to exist.	
p3	+3 years	1 - 3 years	Academic knowledge	Theoritical knowledge	In Singleton Pattern you create a single instance of a type, and not more than a single instance. While in prototype you create more than one instance, by cloning an existing type.	
p4	+3 years	1 - 3 years	No knowledge	Heard of them	I don't know!	
p5	1 - 3 years	0 - 6 months	Practical knowledge	Practical knowlegde	The difference is that for the Singleten pattern, the "same object" is returned, while for the Prototype pattern a "new object" is created.	
p6	1 - 3 years	6 months - 1 year	No knowledge	Theoritical knowledge	a lock is required to ensure the creation of one instance but prototype no.	
p7	1 - 3 years	1 - 3 years	No knowledge	Theoritical knowledge		
p8	+3 years	+3 years	No knowledge	Theoritical knowledge		
p9	+3 years	+3 years	Practical knowledge	Theoritical knowledge	Single class definition but prototype is the definition of single object referencing multiple types	
p10	6 months - 1 year	6 months - 1 year	No knowledge	Theoritical knowledge	Singleton allow to have a single instance of the class in all the application. Prototype allow you to have multiple objects with the same structure.	
p11	+3 years	+3 years	Practical knowledge	Practical knowlegde	Singleton is a pattern that ensures that one and only one instance of a given class will exist during the execution of a program. It also makes that instance globally available. Prototype is a creational pattern that gives the ability to create new objects by cloning an existing template (the prototype).	
p12	+3 years	+3 years	No knowledge	Practical knowlegde	In Singleton , there must be a unique instance for a class, so there must be a function returning the unique instance. Prototype, the construction of an instance is complicated, so we prefer to copy the first instance for the next instances.	Participants Background (Questions)
						A1. How many years of experience in software development do you have?
						A2. Which of the following describes your experience with Java?
						A3. Which of the following describes your knowledge of the “feature” concept?
						A4. Which of the following describes your knowledge of design patterns?
						A5. What is the difference in terms of implementation between the Singleton and Prototype patterns?

id. Response ID	RHypQ1	RHypQ1Why	RHypQ2	RHypQ2Why	RHypQ3	RHypQ3Why	RHypQ4	RHypQ4Why
P6	Yes		Yes		Yes		reasonable	
P1	Yes		Yes		Yes		reasonable	
P2	Yes		Yes		Yes		too permissive	yes in case of polymorisme or overloading (surcharge)
P7	Yes		Yes		Yes		reasonable	
P4	Yes		Yes		Yes		reasonable	
P3	No	If you've ever observed something happening, you don't need to hypothesize about its occurrence.	Yes		Yes		reasonable	
P5	Yes		Yes		Yes		reasonable	
P8	No	Although I might agree with that, it is (a bit) related to blob antipattern. If you can use references to support this claim I agree. Otherwise, it would be generalizing too much.	Yes		No	I don't quite understand this. OO apps generally use multiple classes to build a functionality. Even OO Design patterns strongly advise that.	reasonable	
P9	Yes		Yes		No	The class members related to the functional aspect may have several responsibilities dedicated to specific classes that need to implement them. Therefore, we can make generalized feature but also assign a designated feature to specific class memebers.	reasonable	
P10	Yes		Yes		Yes		reasonable	
P11	Yes		Yes		Yes		reasonable	
P12	Yes		Yes		Yes		reasonable	

Response ID	US1	US2	US3	US4 - JHotDra	US4 - [JRever	US4 - [PMD]	US4 - [JFreeChart]
p6	Easy	Difficult	Easy	2	1	3	1
p5	Very difficult	Difficult	Very difficult	2	1	2	1
p4	Very easy	Easy	Average				
p2	Very easy	Average	Average	3	4	2	1
p3	Average	Difficult	Average	2	4	3	1
p1	Very easy	Average	Easy	4	2	3	1
p7	Easy	Difficult	Very difficult	3	4	2	1
p8	Easy	Very difficult	Easy	2	3	2	1
p9	Easy	Average	Average	2	2	2	1
p10	Very easy	Easy	Easy	3	4	1	2
p11	Very easy	Easy	Easy	4	2	3	1
p12	Very easy	Average	Average	3	2	4	1

Tool Usability

US1. How would you rate the level of difficulty to use the tool?

US2. How would you rate the level of difficulty to understand the graphs provided by the tool?

US3. Is it easy to navigate within the displayed graph to find features/understand the design of the analysed system?

US4. Looking into the graphs/lattices built from each system, can you assign each system a design quality score (1 for the system having the highest number of refactoring opportunities, and 4 for the system having the smallest number of refactoring opportunities)

PARTICIPANT #	P6	P1	P2	P7	P4	P3	P5	P8	P9	P10	P11	P12	%Yes
JRevNode12Q1	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	60
JRevNode14Q1	Yes	No	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	70
JRevNode15Q1	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	60
JRevNode24Q1	No	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	80
JRevNode20Q1	Yes	No	Yes	No	Yes	No	No	Yes	Yes	No	Yes	Yes	60
JRevNode11Q1	No	Yes	Yes	Yes	No	No	No	No	Yes	Yes	No	Yes	50
JHotNode20Q1	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	90
JHotNode21Q1	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	70
JHotNode110Q1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	90
JHotNode128Q1	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	70
JHotNode135Q1	Yes	Yes	Yes	No	Yes	No	No	No	Yes	Yes	Yes	Yes	60
JHotNode5Q1	No	No	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	60
Correctness - Evaluation													
JRevNode#Q1. Is the multiple occurrence of the same set of methods is meaningful?													

PARTICIPANT #	P6	P1	P2	P7	P4	P3	P5	P8	P9	P10	P11	P12	moyenne
JRevNode12Q2	5	4	4	3	4	4	3	3	2	4	1	4	3,2
JRevNode14Q2	5	5	5	5	2	5	5	5	1	1	5	5	3,9
JRevNode15Q2	5	1	5	5	5	1	4	3	3	4	5	5	4
JRevNode24Q2	3	4	5	2	5	2	4	1	1	4	4	3	3,1
JRevNode20Q2	5	5	5	5	4	1	5	5	1	1	5	4	3,6
JRevNode11Q2	5	1	5	2	1	1	5	1	2	4	5	5	3,1
JHotNode20Q2	5	5	5	3	5	4	5	5	2	4	5	5	4,3
JHotNode21Q2	5	5	5	4	5	4	5	5	1	1	5	5	4
JHotNode110Q2	5	3	3	3	5	5	2	5	1	4	3	5	3,6
JHotNode128Q2	5	2	2	2	2	1	1	1	2	4	1	5	2,1
JHotNode135Q2	5	1	5	4	5	1	5	1	3	4	5	5	3,8
JHotNode5Q2	5	4	3	4	5	2	1	5	2	4	4	3	3,3

Correctness - Evaluation

JRevNode#Q2. Are the elements of the feature functionally cohesive, i.e. do the set of methods that occur together (appear to) contribute to the same functions? Assign a cohesion score from 1 (for low cohesion) to 5 (for high cohesion)

PARTICIPANT #	P6	P1	P2	P7	P4	P3	P5	p8	p9	p10	P11	P12	%Yes
JRevNode12Q3	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	83,33333333
JRevNode14Q3	Yes	Yes	No	Yes	No	Yes	No	Yes	No	No	Yes	Yes	58,33333333
JRevNode15Q3	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	66,66666667
JRevNode24Q3	No	Yes	Yes	No	Yes	No	Yes	No	No	Yes	Yes	Yes	58,33333333
JRevNode20Q3	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes	75
JRevNode11Q3	No	No	No	No	No	No	No	No	Yes	Yes	No	Yes	25
JHotNode20Q3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100
JHotNode21Q3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	83,33333333
JHotNode110Q3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	91,66666667
JHotNode128Q3	No	Yes	Yes	No	No	No	Yes	No	Yes	Yes	No	Yes	50
JHotNode135Q3	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	75
JHotNode5Q3	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	75
Correctness - Evaluation													
JRevNode#Q3. Does the feature correspond to a “useful” or “interesting” domain abstraction”.													

PARTICIPANT #	P6	P1	P2	P7	P4	P3	P5	P8	P9	P10	P11	P12
JRevNode12Q4		some of the methods are the same and could have been defined in a superclass. but also some of them are a bit different (the same name yes, but there is a difference example: initAppState, viewpool)				My answer should be "yes" but I choose "No" to be able to add that the function void main(String[] aArgs) should not be part of this set of methods.						
JRevNode12Q5		Check the content of the methods too instead of just the presence of methods				Exclude some type of methods that are associated to the class such as setters, getters and main() function.						
JRevNode14Q4			the sort is not a domain abstraction, like filter , search, reset...		Lack of generalization		No		This can easily be defined in a single method using different cohesive attributes	The sort() method has no relationship to the feature.		
JRevNode14Q5			Maybe adding some semantic analysis to classify by the titles		Not a rule but a recommendation: it could be like moving the sort() method into an interface and then implementing that interface, and define the class-specific behavior.		No		abstraction	Maybe you can exclude all operations the most used that do not contribute to the feature like sorting, pringting in logs, etc.		
JRevNode15Q4			addComponents is not related to domain			Given the extent, the feature [addComponents()] does not perform the same functionality. However it will be useful if it belongs to the extent [jreversepro.awtui.JCustomListPanel, jreversepro.gui. JCustomListPanel].				It is a placing elements in the layout Grid layout.	This is a void method with no arguments, so it is difficult to grasp what feature this would represent. I think this is just an artifact of the UI framework used that adds this to every Dialog.	
JRevNode15Q5			DO some expectation to term related to development			Mine and compare the content of the methods.				You could exclude the graphic operations.	I think this would require a case-by-case analysis, because not all void methods with no arguments are meaningless (e.g. the previous question in this survey).	
JRevNode24Q4	The constructors have the same name, but can they alone really be considered functional features?			JClassEditPanel represents a Panel which can consist of any element such as different types of text editors etc. which are specific to the panel/window being designed.		The set of methods is not cohesive. Also, JClassEditPanel() and setEditorFont(Font aFont) are implemented differently in the two classes.		C	Defining Edit panel and font can reduce multiple occurrences			
JRevNode24Q5	Exclude constructors from the analysis.			Not sure if it is a rule: but for writetoFile, irrespective of how the functionality is implemented, the end result as output would be the same. However the components included in a panel, differ from one Panel to another. Therefore, the implementation of the method could be different in different occurances.		I don't know. This question should not be mandatory :)		void JClassEditPanel() is a constructor, two other methods return Font and the other is a void.	Polymorphism			
JRevNode20Q4	The constructors have the same name, but can they alone really be considered functional features?			JClassEditPanel represents a Panel which can consist of any element such as different types of text editors etc. which are specific to the panel/window being designed.		The set of methods is not cohesive. Also, JClassEditPanel() and setEditorFont(Font aFont) are implemented differently in the two classes.				It is just getters. Most of the getters have the same structure.		
JRevNode20Q5	Exclude constructors from the analysis.			Not sure if it is a rule: but for writetoFile, irrespective of how the functionality is implemented, the end result as output would be the same. However the components included in a panel, differ from one Panel to another. Therefore, the implementation of the method could be different in different occurances.		I don't know. This question should not be mandatory :)				You can easly ignore the getters and setters.		
JRevNode11Q4	The main method will be present in any runnable class and it is meant to indicate the entry point to a Java program. Usually one would just find one occurrence of the main method, but this application provides different versions of user interfaces that can be run independently, therefore the multiple occurrences of a main method.	each version of the method corresponds to the technology used(awt or swing).	it is related to technical/development purposes not domain	NA	Probably for testing purposes, the developer tried to bootstrap the main function from different classes (AWT, vs GUI).	main function should not be mined for feature functionalities.	No	JAwtFrame does not have a main class			This is a main function, representing the program entry point. I doubt it would make a useful abstraction to extract this.	
JRevNode11Q5	Perhaps the main method should be excluded from the analysis.	studying the code in the method.	delete development-related terms/functions	NA	I am not aware of any, but if the JReversePro can be used in multiple "modes" (AWT, GUI) , then the multiple occurrences of the main function is explainable.	Avoid entry points functions.	the classic methods that can be used in different classes like the get, sets mains etc should not be considered as features	it is a "helper" class			Perhaps main functions should be excluded alltogether from this analysis?	
JHotNode20Q4												
JHotNode20Q5												
JHotNode21Q4			map is not domain functionality							It is a commun method widely used in java.		
JHotNode21Q5			doing some semantic learning							You could ignore the most commun operations in java.		
JHotNode110Q4									The interface that defines meaning full operations and each intent further use or extend this.			
JHotNode110Q5									Inheritance			
JHotNode128Q4	The main method shows up again as having multiple occurrences for the same reason as before.			NA	the main function is used the initialize the application GUI, and one of the main components to be initialized is the Toolbar	createTools() is different in each class. It is irrelevant to add main() to a feature functionality set.		Main method does not make much sense for me here			Same as the previous question with a main method. This time, createTools is also repeated, but that seems to be related to these dialogs all having toolbars, which I don't think is an interesting abstraction.	
JHotNode128Q5	Perhaps the main method should be excluded from the analysis.			NA	The "main" is for technical purposes (to bootstrap the app) should always be isolated from the result of classes, and if we need to call a function (like the createTools in this case), we just need to import and instantiate an object.	.		Maybe excluding main methods?			Same as before with the exclusion of main and perhaps heuristics to detect common UI framework elements.	
JHotNode135Q4						Not the same method.	No	one class returns a DrawingView and the other a Drawing (source)				
JHotNode135Q5						Mine the content of methods.	As previously mentionned, the set and get functions (in my opinion) should not be considered as features	Different return types				
JHotNode5Q4	Some of these methods are already inherited from the same parent class.					I have more time to check all these methods!	No					

JHotNode5Q5	The exclusion depends on the root node considered. Perhaps there should be a lookup for a parent containing the methods.					.	No					
Correctness - Evaluation												
(IF Q3 == NO) - JRevNode#Q4. Do you see a design anomaly that could have explained how/why this case (multiple occurrences of a set of methods) showed up?												
(IF Q3 == NO) - JRevNode#Q5. Can you think of a "rule" that could have excluded this case?												

Participants Background
A1. How many years of experience in software development do you have?
A2. Which of the following describes your experience with Java?
A3. Which of the following describes your knowledge of the “feature” concept?
A4. Which of the following describes your knowledge of design patterns?
A5. What is the difference in terms of implementation between the Singleton and Prototype patterns?
Correctness - Evaluation
RHypQ1. We have observed, and thus hypothesize, that domain classes in legacy OOP applications often implement / centralize several functional aspects, regardless of how they are composed (multiple inheritance, aggregation, aspect weaving, or ad-hoc). Do you agree?
RHypQ1. Why. Why not ?
RHypQ2. We argue that it is important to identify such functional features within legacy applications, for the purposes of understanding such applications, and possibly refactoring, reusing, and independently evolving such features later on. Do you agree?
RHypQ2Why. Why not?
RHypQ3. We argued, with a toy example, that the class members that pertain to a given functional features may be distributed among many classes, and not be limited to a single class. For example, the functionality at hand may itself be specialized into several flavors. Do you agree?
RHypQ3Why. If not, why do you think that it is unlikely that the class members related to a functional aspects be distributed amongst many classes?
RHypQ4. We made the hypothesis to ignore the specific placements of a given set of class members within a subhierarchy when counting occurrences [tu peux inclure l’image du slide 10 pour expliquer la question]. Do you think this is:
RHypQ4Why. Can you imagine a situation where a different topology underlies different semantics?
JRevNode#Q1. Is the multiple occurrence of the same set of methods meaningful?
JRevNode#Q2. Are the elements of the feature functionally cohesive, i.e. do the set of methods that occur together (appear to) contribute to the same functions? Assign a cohesion score from 1 (for low cohesion) to 3 (for high cohesion)
JRevNode#Q3. Does the feature correspond to a “useful” or “interesting” domain abstraction”.
(IF Q3 == NO) - JRevNode#Q4. Do you see a design anomaly that could have explained how/why this case (multiple occurrences of a set of methods) showed up?
(IF Q3 == NO) - JRevNode#Q5. Can you think of a “rule” that could have excluded this case?
Tool Usability
US1. How would you rate the level of difficulty to use the tool?
US2. How would you rate the level of difficulty to understand the graphs provided by the tool?
US3. Is it easy to navigate within the displayed graph to find features/understand the design of the analysed system?
US4. Looking into the graphs/lattices built from each system, can you assign each system a design quality score (1 for the system having the highest number of refactoring opportunities, and 4 for the system having the smallest number of refactoring opportunities)