### Survey on Multi-language Design Smells

### Survey on Multi-language Design Smells

Thank you for agreeing to participate, it will take around 30 minutes to complete.

### **Study Policy:**

- Participation in this study is completely voluntary. If you decide not to participate there will not be any negative consequences. If you
  decide to participate, you may stop participating at any time and withdraw entirely your participation or you may decide not to answer
  any specific question.
- Your identity and the data collected thanks to your participation will remain anonymous and will never be released to the public. Only
  anonymous data (aggregated or not) will be published in scientific articles, ensuring that the data cannot be linked back to a particular
  participant. The data will be kept by the principal investigator for five years before being destroyed.
- By submitting this survey, you are indicating that you have read the description of the study, are over the age of 18, and that you agree
  to the terms and consent as described in <a href="https://drive.google.com/file/d/1aZfHRCr0bEX0i331\_oQHIS9ui9h6rlC5/view?usp=sharing">https://drive.google.com/file/d/1aZfHRCr0bEX0i331\_oQHIS9ui9h6rlC5/view?usp=sharing</a>

If you have any questions, please contact us at mouna.abidi@polymtl.ca

<u>Study Design:</u> The purpose of this study is to investigate the prevalence of design smells related to multi-language systems. These systems are developed using more than one programming language. We aim to investigate the perceived prevalence and impact of the design smells detailed below. Our main goal is to improve the quality of those systems.

### **Definition of terminologies:**

Not Handling Exceptions	The exceptions are not handled, developers generally rely on the exceptions provided by the other language
Assuming Safe Return	A value is returned to the other language without being checked. Thus, the interaction between both languages may
Value	not be correctly performed
Excessive Inter-language	eA wrong partitioning in both languages leads to many calls in a way or the other. It adds complexity takes more time
Communication	to run and may indicate a bad separation of concerns
Too Much Clustering	The multi-language code is concentrated in a few classes, regardless of their concerns and responsibilities.
Too Much Scattering	Many classes are scarcely used in multi-language communication
	When different libraries are needed depending on the operating system, they are not loaded with conditions on the
Hard Coding Libraries	operating system, but for instance, with a try-catch mechanism, making it hard to know which library has really been loaded
Local References Abuse	The developer does not manage the memory in the native space properly and does not release local and global references
Memory Management	Reference types passed from one language to another are not released in a language that does not handle the
Mismatch	management of memory causing memory leaks
Not Caching Objects	A method is called to retrieve a field every time this field is needed, although the field's ID or value could have been cached.
Not Securing Libraries	The code loads a foreign library without any security check or restriction privilege
Not Using Relative Path	A library is loaded using only the name not the path. It cannot be accessed in the same way from everywhere
Excessive Objects	A whole object is passed as an argument, although only some of the fields were needed, and it would have been better for the system performance to pass only these fields
Unused Method	better for the system performance to pass only these neites
Declaration	A method is declared in the host language but not implemented in the foreign language
Unused Method	A method is declared in the host language and implemented in the foreign language, but never called from the host
Implementation	language
Unused Parameters	Some arguments of a function are used neither in its body nor in the other language.

(Khomh, F., & Gueheneuce, Y. G. (2008, April). Do design patterns impact software quality positively?. In Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on (pp. 274-278).

IEEE.)							
– Exp	andability: The degree to which the design	of a system can be extended.					
- Sin	- Simplicity: The degree to which the design of a system can be understood easily.						
– Rei	sability: The degree to which a piece of des	sign can be reused in another design.					
– Lea	rnability: The degree to which the code sou	rce of a system is easy to learn.					
	derstandability: The degree to which the cod						
	formance: The degree to which the code me						
	-						
- Mod	dularity: The degree to which the implement	ation of the functions of a system is independent of one another.					
Thanl	< you.						
Best i	regards,						
* 1.	What is your role within your organi (Yamashita, A., & Moonen, L. (2013, October). Do o Conference on (pp. 242-251). IEEE.)	<b>ization?</b> developers care about code smells? an exploratory survey. In Reverse Engineering (WCRE), 2013 20th Working					
	O Software Engineer	O Developer					
	O Team Lead	○ Tester					
	<ul> <li>Architect</li> </ul>	QA Manager					
	Project Manager	○ Self-employed					
	Other, please specify						
<b>*</b> 2.	How many years of experience do y	ou have in software engineering?					
	Less than 1 year	1 year - 5 Years					
	5-10 Years	○ More than 10 years					
* 3.	What is the domain of activity of you	ur organization?					
	Research and development	○ Networks					
	Healthcare	Analytics (Business,IT services, BigData)					
	Banking and insurance	Robotics and Embeeded systems					

### \* 4. What is your level of skill in the following languages? Please specify which other languages if relevant:

Other, please specify

(https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages)

Games

Other, please specify

	1 Novice	Little Knowledge	3 Practical	4 Comfortable	5 Expert
Python				0	0
C++					
Java					
С					
C#					
PHP					
R					
JavaScript					
Go					
Assembly					
Other, please specify					

### \* 5. How often do you encounter the following design smells in your project(s)?

Please check the definitions provided above before answering this questions

	1 Very Often	2 Often	3 Rarely	N/A
Not Handling Exceptions	0			0
Assuming Safe Return Value			0	
Excessive Inter-language Communication				
Too Much Clustering				
Too Much Scattering				
Hard Coding Libraries				
Local References Abuse			0	
Memory Management Mismatch				
Not Caching Objects			0	
Not Securing Libraries			0	
Not Using Relative Path			0	
Excessive Objects			0	
Unused Method Declaration			0	
Unused Method Implementation			0	

Unused Parameters		

### \* 6. How do you evaluate the impact of the following design smells in those software quality attributes?

Please carefully read the definition of the smells provided bellow and the reference provided.

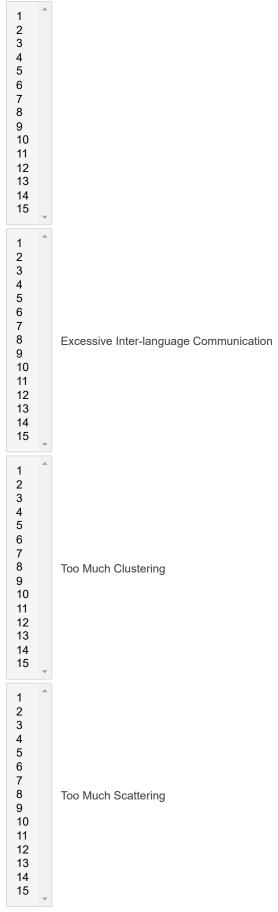
(VN: Very Negative, N: Negative, NS: Not significant/Neutral, P: Positive, and VP: Very Positive)

	Expandability	Simplicity	Reusability I	Learnability	Understandabilit	y Performance	Modularity	N/A
Not Handling Exceptions								
Assuming Safe Return Value								
Excessive Inter-language Communication								
Too Much Clustering								
Too Much Scattering								
Hard Coding Libraries								
Local References Abuse								
Memory Management Mismatch								
Not Caching Objects								
Not Securing Libraries								
Not Using Relative Path								
Excessive Objects								
Unused Method Declaration								
Unused Method Implementation								
Unused Parameters								

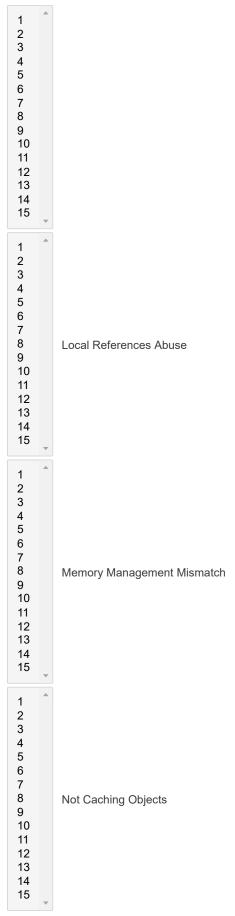
### \* 7. Please rank the following design smells from the most harmful to the less harmful

(Most harmful to the less harmful: 15 -> 1)

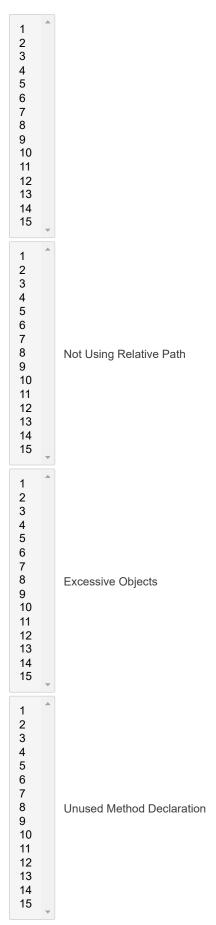
Assuming Safe Return Value



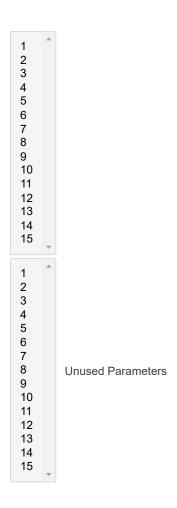
Hard Coding Libraries



Not Securing Libraries



Unused Method Implementation



### \* 8. <u>Task:</u>

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

O No

```
void JNICALL Java_i2jrt_TAOObject__1release(JNIEnv *jni, jobject jThis)
{
jclass clazz = findClass(jni, "i2jrt/TAOObject");
jfieldID fid = jni->GetFieldID(clazz, "_jni_ptr", "J");
jlong _jni_ptr = jni->GetLongField(jThis, fid);
CORBA::Object_ptr o = reinterpret_cast(_jni_ptr);
CORBA::release(o);
jni->SetLongField(jThis, fid, reinterpret_cast(CORBA::Object::_nil()));
}

Yes
```

- 9. b) If YES, please provide an explanation or specify the design smell(s) involved?
- 10. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?
- \* 11. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)



\* 12. e) If YES, would you apply this refactored solution?

```
void JNICALL Java_i2jrt_TAOObject__1release(JNIEnv *jni, jobject jThis)
{
jclass clazz = findClass(jni, "i2jrt/TAOObject");
if (clazz == NULL) {
jclass newExc = env->FindClass("java/lang/NullPointerException");
env->ThrowNew(newExc, "The native object does not exist.");
return 0;}
jfieldID fid = jni->GetFieldID(clazz, "_jni_ptr", "J");
jlong _jni_ptr = jni->GetLongField(jThis, fid);
CORBA::Object_ptr o = reinterpret_cast(_jni_ptr);
CORBA::release(o);
jni->SetLongField(jThis, fid, reinterpret_cast(CORBA::Object::_nil()));
}
Yes (Refactor with this solution)
                                                                      Yes (Refactor with an alternative solution)
No (No refactoring)
```

### \* 13. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
public static char convertToChar(StreamItem item) throws MessageFormatException {
   if (item == null) throw new NullPointerException();
   final ItemKind itemKind = item.discriminator();
   if (compare(itemKind, ItemKind.CHAR_KIND)) {
     return item.charValue();
   } else {
     throw new MessageFormatException("Cannot convert stream item to char");
   }
}
Yes
No
```

14. b) If YES, please provide an explanation or specify the design smell(s) involved?

15. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

<b>*</b> 16.	d) Please rate the s	severity of the imple	mentation problem	(if any), from 1 (Ver	ry Low) to 5 (Very Hi	gh)
	1 Very Low	2 Low	3 Medium	4 High	5 Very High	N/A
* 17.	if (item == null) throw new I final ItemKind itemKind = it if (ItemKind !== null)){   if (compare(itemKind, ItemI return item.charValue();   }} else {	oChar(StreamItem item) thro NullPointerException(); em.discriminator();	ws MessageFormatException	on {		
	Yes (Refactor with No (No refactoring)	·	○ Yes (	Refactor with an altern	native solution)	
* 18.	Task:  a) In your opinion, design problem)? static { try {   System.loadLibrary("JMPla} } catch (Throwable err) {   err.printStackTrace();   } }		code(s) contain any	occurrence of des	ign smell(implement	ation and-or
	O Yes			○ No		
19. I	b) If YES, please pro	vide an explanation	or specify the desi	gn smell(s) involve	d?	
20. (	c) If YES, (In your op	oinion,) What is the	motivation behind u	sing this specific v	vay of implementation	on?
* 21.	d) Please rate the s	severity of the imple	mentation problem	(if any), from 1 (Ver	ry Low) to 5 (Very Hi	gh)

	Very Low	Low	Medium	High	Very High	N/A
* 22.	e) If YES, would yo public static void loadLibrar static {     AccessController.doPriviled     public Void run() {     try {         System.loadLibrary("JMPla     } catch (Throwable err) {         err.printStackTrace();     }} });     }      Yes (Refactor with     No (No refactoring)	ged( new PrivilegedAction() yer"); this solution)	{	(Refactor with an alterr	native solution)	
* 23.	Task:  a) In your opinion, design problem)? std::vector certBufferRefs(r std::vector certBuffers(num for (size_t i = 0; i < numCer ScopedLocalRef certArray(certBufferRefs[i] = ByteArra if (!certBufferRefs[i]) { return; } certBuffers[i] = certBufferRef }	numCerts); Certs); ts; ++i) { env, reinterpret_cast(env-> ayToCryptoBuffer(env, certA	GetObjectArrayElement(end		ign smell(implemei	ntation and-or
	Yes			O No		
24. I	b) If YES, please pro	vide an explanation	n or specify the des	ign smell(s) involve	ed?	
25. (	c) If YES, (In your op	oinion,) What is the	motivation behind	using this specific v	way of implementat	ion?
<b>*</b> 26.	d) Please rate the s	everity of the imple	ementation problem	(if any), from 1 (Ve	ry Low) to 5 (Very F	ligh)

1	2	3	4	5	N/A
Very Low	Low	Medium	High	Very High	

\* 27. e) If YES, would you apply this refactored solution?

```
std::vector<bssl::uniqueptr> certBufferRefs(numCerts);
std::vector<crypto_buffer*> certBuffers(numCerts);
for (size_t i = 0; i < numCerts; ++i) {
    ScopedLocalRef certArray(
    env, reinterpret_cast(
    env->GetObjectArrayElement(encodedCertificatesJava, i)));
    certBufferRefs[i] = ByteArrayToCryptoBuffer(env, certArray.get(), nullptr);
    if (!certBufferRefs[i]) {
        return;
    }
    certBuffers[i] = certBufferRefs[i].get();
    (*env)->DeleteLocalRef(env, certBufferRefs);
}

    Yes (Refactor with this solution)
    No (No refactoring)
```

### \* 28. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
bool setApplicationProtocols(JNIEnv* e, jbyteArray applicationProtocolsJava) {
clearApplicationProtocols();
if (applicationProtocolsJava != nullptr) {
jbyte* applicationProtocols =
e->GetByteArrayElements(applicationProtocolsJava, nullptr);
if (applicationProtocols == nullptr) {
clearCallbackState();
JNI_TRACE("appData=%p setApplicationCallbackState => applicationProtocols == null", this);
return false;
applicationProtocolsLength = static_cast(e->GetArrayLength(applicationProtocolsJava));
applicationProtocolsData = new char[applicationProtocolsLength];
memcpy(applicationProtocolsData, applicationProtocols, applicationProtocolsLength);
return true;
}
Yes
                                                                                        O No
```

29. b) If YES, please provide an explanation or specify the design smell(s) involved?

30. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

\* 31. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)



### \* 32. e) If YES, would you apply this refactored solution?

```
bool setApplicationProtocols(JNIEnv* e, jbyteArray applicationProtocolsJava) {
clearApplicationProtocols();
if (applicationProtocolsJava != nullptr) {
jbyte* applicationProtocols =
e->GetByteArrayElements(applicationProtocolsJava, nullptr);
if (applicationProtocols == nullptr) {
clearCallbackState();
JNI_TRACE("appData=%p setApplicationCallbackState => applicationProtocols == null", this);
applicationProtocolsLength =
static_cast(e->GetArrayLength(applicationProtocolsJava));
applicationProtocolsData = new char[applicationProtocolsLength];
memcpy(applicationProtocolsData, applicationProtocols, applicationProtocolsLength);
e->ReleaseByteArrayElements(applicationProtocolsJava, applicationProtocols, JNI_ABORT);
}
return true;
}
Yes (Refactor with this solution)
                                                                     Yes (Refactor with an alternative solution)
No (No refactoring)
```

### \* 33. Task:

# a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
bool composite_mapping::gen_struct(UTL_ScopedName *name, const vector &fields, const char *repoid) {
    for (vector::iterator it(components_.begin());
    it != components_.end(); ++it) {
        if (!(*it)->gen_struct(name, fields, repoid))
        return false;
    }
    return true;
}

Yes
```

35.	c) If YES, (In your op	pinion,) What is the	motivation behind	using this specific v	vay of implementat	ion?
* 36.	d) Please rate the s	severity of the imple	ementation problem	(if any), from 1 (Ve	ry Low) to 5 (Very H	ligh)
	1 Very Low	2 Low	3 Medium	4 High	5 Very High	N/A
* 37.	e) If YES, would yo bool composite_mapping::{     for (vector::iterator it(composite!= componentsend(); ++     if (!(*it)->gen_struct(name, return false;     }     return true; }  Yes (Refactor with  No (No refactoring)	gen_struct(UTL_ScopedNationentsbegin()); +it) { fields, repoid))  this solution)	me *name, const vector &fie	elds, const char *repoid)	native solution)	
* 38.		v.jobject obj.jobject allVal) ntField(env,allVal,a); tField(env,allVal,b); tField(env,allVal,c);	code(s) contain any	occurrence of des	ign smell(implemer	ntation and-or
	○ Yes			O No		
39.	b) If YES, please pro	ovide an explanation	n or specify the des	ign smell(s) involve	d?	
40.	c) If YES, (In your op	oinion,) What is the	motivation behind	using this specific v	vay of implementat	ion?

### \* 41. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1	2	3	4	5	N/A
Very Low	Low	Medium	High	Very High	

### \* 42. e) If YES, would you apply this refactored solution?

int sumNative (JNIEnv\* env, jobject obj, jint a, jint b, jint c){ return a + b + c};

Yes (Refactor with this solution)

No (No refactoring)

Yes (Refactor with an alternative solution)

### \* 43. Task:

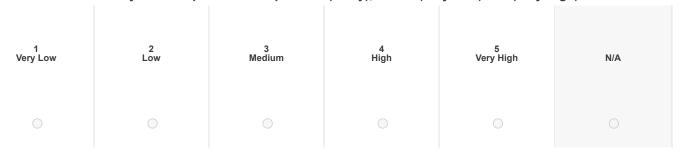
# a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

design problem)?
public class GodotLib {
public static GodotlO io;
static { System.loadLibrary("godot_android");}
$public\ static\ native\ void\ initialize (Godot\ p\_instance,\ Object\ p\_asset\_manager,\ boolean\ use\_apk\_expansion);$
public static native void setup(String[] p_cmdline);
public static native void resize(int width, int height);
public static native void newcontext(boolean p_32_bits);
public static native void back();
public static native void step();
public static native void touch(int what, int pointer, int howmany, int[] arr);
public static native void accelerometer(float x, float y, float z);
public static native void gravity(float x, float y, float z);
public static native void magnetometer(float x, float y, float z);
public static native void gyroscope(float x, float y, float z);
public static native void key(int p_scancode, int p_unicode_char, boolean p_pressed);
public static native void compress(int p_device, int p_but, boolean p_pressed);
public static native void decompress(int p_device, int p_axis, float p_value);
<pre>public static native void securehat(int p_device, int p_hat_x, int p_hat_y);</pre>
public static native void joyconnectionchanged(int p_device, boolean p_connected, String p_name);
public static native void image();
public static native void video();
public static native void audio();
public static native void setmedia(String p_name, Object p_object);
public static native void secure(String p_sname, String p_name, String p_ret, String[] p_params);
public static native String getGlobal(String p_key);
<pre>public static native void callobject(int p_ID, String p_method, Object[] p_params);</pre>
public static native void calldeferred(int p_ID, String p_method, Object[] p_params);
$public\ static\ native\ void\ request Permission Conenction String\ p\_permission,\ boolean\ p\_result);$
public static native void setVirtualKeyboardHeight(int p_height);
}
○ Yes

44.	b) If YES, please provide an explanation or specify the design smell(s) involved?				

45. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

### \* 46. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)



e) if YES, would you apply this refactored solution	1.
public class GodotLib {	
public static GodotlO io;	
static { System.loadLibrary("godot_android");}	
public static native void initialize(Godot p_instance, Object p_asset_man	nager, boolean use_apk_expansion);
<pre>public static native void setup(String[] p_cmdline);</pre>	
public static native void newcontext(boolean p_32_bits);	
public static native void resize(int width, int height);	
public static native void back();	
public static native void step();	
public static native void touch(int what, int pointer, int howmany, int[] arr)	);
public static native void accelerometer(float x, float y, float z);	
public static native void gravity(float x, float y, float z);	
public static native void magnetometer(float x, float y, float z);	
public static native void gyroscope(float x, float y, float z);	
<pre>public static native String getGlobal(String p_key);}</pre>	
public class media{	
public static native void image();	
public static native void video();	
public static native void audio();	
<pre>public static native void setmedia(String p_name, Object p_object);}</pre>	
public class compression{	
public static native void compress(int p_device, int p_but, boolean p_pre	essed);
public static native void decompress(int p_device, int p_axis, float p_value)	ue);}
public class connection{	
public static native void key(int p_scancode, int p_unicode_char, boolea	n p_pressed);
public static native void securehat(int p_device, int p_hat_x, int p_hat_y)	);
public static native void joyconnectionchanged(int p_device, boolean p_	connected, String p_name);
public static native void secure(String p_sname, String p_name, String p	o_ret, String[] p_params);
public static native void callobject(int p_ID, String p_method, Object[] p_	params);
public static native void calldeferred(int p_ID, String p_method, Object[]	p_params);
public static native void requestPermissionConenctionString p_permissi	on, boolean p_result);
public static native void setVirtualKeyboardHeight(int p_height);}	
Yes (Refactor with this solution)	Yes (Refactor with an alternative solution)

	O No (No refactoring)					
	Task: a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?					
	static {     java.security.AccessController.doPrivileged(					
	(PrivilegedAction) () -> {					
	System.loadLibrary("dispatch");					
	return null; });					
	<pre>} private LibDispatchNative() {</pre>					
	static native boolean nativeIsDispatchSupported();					
	static native long nativeCreateConcurrentQueue(int priority); }					
	public final class Dispatch {					
	public static Dispatch getInstance() {					
	checkSecurity();					
	if (!LibDispatchNative.nativeIsDispatchSupported()) return null;					
	return instance; } JNIEXPORT jboolean JNICALL Java_com_apple_concurrent_LibDispatchNative_nativelsDispatchSupported (JNIEnv *env, jclass clazz) { return JNI_TRUE; } JNIEXPORT jlong JNICALL Java_com_apple_concurrent_LibDispatchNative_nativeCreateConcurrentQueue (JNIEnv *env, jclass clazz, jint priority) {					
	dispatch_queue_t queue = dispatch_get_global_queue((long)priority, 0);					
	return pt					
	○ Yes ○ No					
h	b) If YES, please provide an explanation or specify the design smell(s) involved?					
~	) in 123, please provide an explanation of specify the design smell(s) involved:					
С	c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?					

1	2	3	4	5	N/A
Very Low	Low	Medium	High	Very High	

### \* 52. e) If YES, would you apply this refactored solution?

```
static {
java.security.AccessController.doPrivileged(
(PrivilegedAction) () -> {
System.loadLibrary("dispatch");
return null;
});
private LibDispatchNative() {
static native boolean nativeIsDispatchSupported();
public final class Dispatch {
public static Dispatch getInstance() {
checkSecurity();
if \ (!LibDispatchNative.nativelsDispatchSupported()) \ return \ null; \\
return instance;
JNIEXPORT jboolean JNICALL Java_com_apple_concurrent_LibDispatchNative_nativelsDispatchSupported (JNIEnv *env, jclass clazz)
return JNI TRUE;
                                                                      Yes (Refactor with an alternative solution)
Yes (Refactor with this solution)
No (No refactoring)
```

### \* 53. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
public static void setSearchField(JTextField txt, boolean isSearchField) {
if (isSearchField == isSearchField(txt)) {
txt.putClientProperty(MAC_TEXT_FIELD_VARIANT_PROPERTY, "_triggerevent_");
} else if (isSearchField) {
uiChangeHandler.install(txt);
} else {
uiChangeHandler.uninstall(txt);
}
if (isSearchField) {
txt.putClientProperty(MAC TEXT FIELD VARIANT PROPERTY, MAC SEARCH VARIANT);
txt.putClientProperty("Quaqua.TextField.style", MAC_SEARCH_VARIANT);
} else {
txt.putClientProperty(MAC\_TEXT\_FIELD\_VARIANT\_PROPERTY, "default");\\
txt.putClientProperty("Quaqua.TextField.style", "default");
}}
                                                                                    O No
Yes
```

54. b) If YES, please provide an explanation or specify the design smell(s) involved?

\* 56. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)



### \* 57. e) If YES, would you apply this refactored solution?

```
public static void setSearchField(JTextField txt, boolean isSearchField) {
  if (isSearchField == isSearchField(txt)) {
    txt.putClientProperty(MAC_TEXT_FIELD_VARIANT_PROPERTY, "_triggerevent_");
} else if (isSearchField) {
    uiChangeHandler.install(txt);
} else {
    uiChangeHandler.uninstall(txt);
}
txt.putClientProperty(MAC_TEXT_FIELD_VARIANT_PROPERTY, "default");
txt.putClientProperty("Quaqua.TextField.style", "default");
}

    Yes (Refactor with this solution)
    No (No refactoring)
Yes (Refactor with an alternative solution)
```

## Your responses have been registered!

Thank you for taking the time to complete the survey, your input is valuable to us.