

# JNI Environment Functions Reference

---

This document provides an outline of all of the functions used in the **JNIEnv\*** struct used in the “jni.h” header file.

## Table of Contents

<b>Definition of JNIEnv</b> .....	<b>7</b>
<b>GetVersion()</b> .....	<b>7</b>
<b>DefineClass()</b> .....	<b>7</b>
<b>FindClass()</b> .....	<b>7</b>
<b>Reflection Related Functions</b> .....	<b>7</b>
FromReflectedMethod() .....	7
FromReflectedField() .....	7
ToReflectedMethod() .....	8
GetSuperclass() .....	8
IsAssignableFrom() .....	8
ToReflectedField() .....	8
<b>Exceptions</b> .....	<b>8</b>
Throw() .....	8
ThrowNew() .....	8
ExceptionOccurred() .....	8
ExceptionDescribe() .....	9
ExceptionClear() .....	9
FatalError() .....	9
<b>Managing Stack</b> .....	<b>9</b>
pushLocalFrame() .....	9
PopLocalFrame() .....	9
NewGlobalRef() .....	9
DeleteGlobalRef() .....	9
DeleteLocalRef() .....	10
<b>Object Creation &amp; Examination</b> .....	<b>10</b>
IsSameObject() .....	10
NewLocalRef() .....	10
EnsureLocalCapacity() .....	10
AllocObject() .....	10
NewObject() .....	10
NewObjectV() .....	11
NewObjectA() .....	11
GetObjectClass() .....	11
InstanceOf() .....	11
<b>Calling Methods</b> .....	<b>11</b>
GetMethodID() .....	11

CallObjectMethod()	11
CallObjectMethodV()	12
CallObjectMethodA()	12
CallBooleanMethod()	12
CallBooleanMethodV()	12
CallBooleanMethodA()	12
CallByteMethod()	13
CallByteMethodV()	13
CallByteMethodA()	13
CallCharMethod()	13
CallCharMethodV()	13
CallCharMethodA()	13
CallShortMethod()	14
CallShortMethodV()	14
CallShortMethodA()	14
CallIntMethod()	14
CallIntMethodV()	14
CallIntMethodA()	15
CallLongMethod()	15
CallLongMethodV()	15
CallLongMethodA()	15
CallFloatMethod()	15
CallFloatMethodV()	16
CallFloatMethodA()	16
CallDoubleMethod()	16
CallDoubleMethodV()	16
CallDoubleMethodA()	16
CallVoidMethod()	16
CallVoidMethodV()	17
CallVoidMethodA()	17

## Nonvirtual Calling of Methods..... 17

CallNonvirtualObjectMethod()	17
CallNonvirtualObjectMethodV()	17
CallNonvirtualObjectMethodA()	17
CallNonvirtualBooleanMethod()	18
CallNonvirtualBooleanMethodV()	18
CallNonvirtualBooleanMethodA()	18
CallNonvirtualByteMethod()	18
CallNonvirtualByteMethodV()	19
CallNonvirtualByteMethodA()	19
CallNonvirtualCharMethod()	19
CallNonvirtualCharMethodV()	19
CallNonvirtualCharMethodA()	19
CallNonvirtualShortMethod()	20
CallNonvirtualShortMethodV()	20
CallNonvirtualShortMethodA()	20
CallNonvirtualIntMethod()	20
CallNonvirtualIntMethodV()	21
CallNonvirtualIntMethodA()	21

CallNonvirtualLongMethod()	21
CallNonvirtualLongMethodV()	21
CallNonvirtualLongMethodA()	21
CallNonvirtualFloatMethod()	22
CallNonvirtualFloatMethodV()	22
CallNonvirtualFloatMethodA()	22
CallNonvirtualDoubleMethod()	22
CallNonvirtualDoubleMethodV()	23
CallNonvirtualDoubleMethodA()	23
CallNonvirtualVoidMethod()	23
CallNonvirtualVoidMethodV()	23
CallNonvirtualVoidMethodA()	23
<b>Getting Fields</b>	<b>24</b>
GetFieldID()	24
GetObjectField()	24
GetBooleanField()	24
GetByteField()	24
GetCharField()	24
GetShortField()	24
GetIntField()	24
GetLongField()	25
GetFloatField()	25
GetDoubleField()	25
<b>Setting Fields</b>	<b>25</b>
SetObjectField()	25
SetBooleanField()	25
SetByteField()	25
SetCharField()	25
SetShortField()	26
SetIntField()	26
SetLongField()	26
SetFloatField()	26
SetDoubleField()	26
<b>Calling Static Methods</b>	<b>26</b>
GetStaticMethodID()	26
CallStaticObjectMethod()	27
CallStaticObjectMethodV()	27
CallStaticObjectMethodA()	27
CallStaticBooleanMethod()	27
CallStaticBooleanMethodV()	28
CallStaticBooleanMethodA()	28
CallStaticByteMethod()	28
CallStaticByteMethodV()	28
CallStaticByteMethodA()	28
CallStaticCharMethod()	28
CallStaticCharMethodV()	29
CallStaticCharMethodA()	29
CallStaticShortMethod()	29

CallStaticShortMethodV()	29
CallStaticShortMethodA()	29
CallStaticIntMethod()	30
CallStaticIntMethodV()	30
CallStaticIntMethodA()	30
CallStaticLongMethod()	30
CallStaticLongMethodV()	30
CallStaticLongMethodA()	31
CallStaticFloatMethod()	31
CallStaticFloatMethodV()	31
CallStaticFloatMethodA()	31
CallStaticDoubleMethod()	31
CallStaticDoubleMethodV()	32
CallStaticDoubleMethodA()	32
CallStaticVoidMethod()	32
CallStaticVoidMethodV()	32
CallStaticVoidMethodA()	32

## Getting Static Fields ..... 33

GetStaticFieldID()	33
GetStaticObjectField()	33
GetStaticBooleanField()	33
GetStaticByteField()	33
GetStaticCharField()	33
GetStaticShortField()	33
GetStaticIntField()	33
GetStaticLongField()	34
GetStaticFloatField()	34
GetStaticDoubleField()	34

## Setting Static Fields ..... 34

SetStaticObjectField()	34
SetStaticBooleanField()	34
SetStaticByteField()	34
SetStaticCharField()	34
SetStaticShortField()	35
SetStaticIntField()	35
SetStaticLongField()	35
SetStaticFloatField()	35
SetStaticDoubleField()	35

## Working with Strings ..... 35

NewString()	35
GetStringLength()	36
GetStringChars()	36
ReleaseStringChars()	36
NewStringUTF()	36
GetStringUTFLength()	36
GetStringUTFChars()	36
ReleaseStringUTFChars()	36

## Working with Arrays ..... 37

GetArrayLength()	37
NewObjectArray()	37
GetObjectArrayElement()	37
SetObjectArrayElement()	37
NewBooleanArray()	37
NewByteArray()	37
NewCharArray()	37
NewShortArray()	38
NewIntArray()	38
NewLongArray()	38
NewFloatArray()	38
NewDoubleArray()	38
GetBooleanArrayElements()	38
GetByteArrayElements()	38
GetCharArrayElements()	38
GetShortArrayElements()	39
GetIntArrayElements()	39
GetLongArrayElements()	39
GetFloatArrayElements()	39
GetDoubleArrayElements()	39
ReleaseBooleanArrayElements()	39
ReleaseByteArrayElements()	39
ReleaseCharArrayElements()	40
ReleaseShortArrayElements()	40
ReleaseIntArrayElements()	40
ReleaseLongArrayElements()	40
ReleaseFloatArrayElements()	40
ReleaseDoubleArrayElements()	41
GetBooleanArrayRegion()	41
GetByteArrayRegion()	41
GetCharArrayRegion()	41
GetShortArrayRegion()	41
GetIntArrayRegion()	41
GetLongArrayRegion()	41
GetFloatArrayRegion()	42
GetDoubleArrayRegion()	42
SetBooleanArrayRegion()	42
SetByteArrayRegion()	42
SetCharArrayRegion()	42
SetShortArrayRegion()	42
SetIntArrayRegion()	43
SetLongArrayRegion()	43
SetFloatArrayRegion()	43
SetDoubleArrayRegion()	43

## Utility Functions ..... 43

Register Natives()	43
Unregister Natives()	43
MonitorEnter()	44

MonitorExit()	44
GetJavaVM()	44
GetStringRegion()	44
GetStringUTFRegion()	44
GetPrimitiveArrayCritical()	44
ReleasePrimitiveArrayCritical()	44
GetStringCritical()	45
ReleaseStringCritical()	45
NewWeakGlobalRef()	45
DeleteWeakGlobalRef()	45
ExceptionCheck()	45
NewDirectByteBuffer()	45
GetDirectBufferAddress()	45
GetDirectBufferCapacity()	45
GetObjectRefType()	46

## Definition of JNIEnv

```
struct JNIEnv_ {  
    const struct JNINativeInterface_ *functions;  
#ifdef __cplusplus
```

## GetVersion()

```
jint GetVersion() {  
    return functions->GetVersion(this);  
}
```

## DefineClass()

```
jclass DefineClass(const char *name, jobject loader, const jbyte *buf,  
                  jsize len) {  
    return functions->DefineClass(this, name, loader, buf, len);  
}
```

## FindClass()

```
jclass FindClass(const char *name) {  
    return functions->FindClass(this, name);  
}
```

## Reflection Related Functions

---

### FromReflectedMethod()

```
jmethodID FromReflectedMethod(jobject method) {  
    return functions->FromReflectedMethod(this, method);  
}
```

### FromReflectedField()

```
jfieldID FromReflectedField(jobject field) {
```

```
        return functions->FromReflectedField(this,field);
    }
```

### ToReflectedMethod()

```
jobject ToReflectedMethod(jclass cls, jmethodID methodID, jboolean
isStatic) {
    return functions->ToReflectedMethod(this, cls, methodID, isStatic);
}
```

### GetSuperclass()

```
jclass GetSuperclass(jclass sub) {
    return functions->GetSuperclass(this, sub);
}
```

### IsAssignableFrom()

```
jboolean IsAssignableFrom(jclass sub, jclass sup) {
    return functions->IsAssignableFrom(this, sub, sup);
}
```

### ToReflectedField()

```
jobject ToReflectedField(jclass cls, jfieldID fieldID, jboolean
isStatic) {
    return functions->ToReflectedField(this,cls,fieldID,isStatic);
}
```

## Exceptions

---

### Throw()

```
jint Throw(jthrowable obj) {
    return functions->Throw(this, obj);
}
```

### ThrowNew()

```
jint ThrowNew(jclass clazz, const char *msg) {
    return functions->ThrowNew(this, clazz, msg);
}
```

### ExceptionOccured()



```
jthrowable ExceptionOccurred() {
    return functions->ExceptionOccurred(this);
}
```

#### ExceptionDescribe()

```
void ExceptionDescribe() {
    functions->ExceptionDescribe(this);
}
```

#### ExceptionClear()

```
void ExceptionClear() {
    functions->ExceptionClear(this);
}
```

#### FatalError()

```
void FatalError(const char *msg) {
    functions->FatalError(this, msg);
}
```

## Managing Stack

---

#### pushLocalFrame()

```
jint PushLocalFrame(jint capacity) {
    return functions->PushLocalFrame(this, capacity);
}
```

#### PopLocalFrame()

```
jobject PopLocalFrame(jobject result) {
    return functions->PopLocalFrame(this, result);
}
```

#### NewGlobalRef()

```
jobject NewGlobalRef(jobject lobj) {
    return functions->NewGlobalRef(this, lobj);
}
```

#### DeleteGlobalRef()

```
void DeleteGlobalRef(jobject gref) {
```

```

        functions->DeleteGlobalRef(this,gref);
    }

```

### DeleteLocalRef()

```

void DeleteLocalRef(jobject obj) {
    functions->DeleteLocalRef(this, obj);
}

```

## Object Creation & Examination

---

### IsSameObject()

```

jboolean IsSameObject(jobject obj1, jobject obj2) {
    return functions->IsSameObject(this,obj1,obj2);
}

```

### NewLocalRef()

```

jobject NewLocalRef(jobject ref) {
    return functions->NewLocalRef(this,ref);
}

```

### EnsureLocalCapacity()

```

jint EnsureLocalCapacity(jint capacity) {
    return functions->EnsureLocalCapacity(this,capacity);
}

```

### AllocObject()

```

jobject AllocObject(jclass clazz) {
    return functions->AllocObject(this,clazz);
}

```

### NewObject()

```

jobject NewObject(jclass clazz, jmethodID methodID, ...) {
    va_list args;
    jobject result;
    va_start(args, methodID);
    result = functions->NewObjectV(this,clazz,methodID,args);
    va_end(args);
}

```

```

        return result;
    }

```

### NewObjectV()

```

jobject NewObjectV(jclass clazz, jmethodID methodID,
                  va_list args) {
    return functions->NewObjectV(this,clazz,methodID,args);
}

```

### NewObjectA()

```

jobject NewObjectA(jclass clazz, jmethodID methodID,
                  const jvalue *args) {
    return functions->NewObjectA(this,clazz,methodID,args);
}

```

### GetObjectClass()

```

jclass GetObjectClass(jobject obj) {
    return functions->GetObjectClass(this,obj);
}

```

### IsInstanceOf()

```

jboolean IsInstanceOf(jobject obj, jclass clazz) {
    return functions->IsInstanceOf(this,obj,clazz);
}

```

## Calling Methods

---

### GetMethodID()

```

jmethodID GetMethodID(jclass clazz, const char *name,
                     const char *sig) {
    return functions->GetMethodID(this,clazz,name,sig);
}

```

### CallObjectMethod()

```

jobject CallObjectMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jobject result;
    va_start(args,methodID);
}

```

```

    result = functions->CallObjectMethodV(this,obj,methodID,args);
    va_end(args);
    return result;
}

```

#### CallObjectMethodV()

```

jobject CallObjectMethodV(jobject obj, jmethodID methodID,
                          va_list args) {
    return functions->CallObjectMethodV(this,obj,methodID,args);
}

```

#### CallObjectMethodA()

```

jobject CallObjectMethodA(jobject obj, jmethodID methodID,
                          const jvalue * args) {
    return functions->CallObjectMethodA(this,obj,methodID,args);
}

```

#### CallBooleanMethod()

```

jboolean CallBooleanMethod(jobject obj,
                          jmethodID methodID, ...) {
    va_list args;
    jboolean result;
    va_start(args,methodID);
    result = functions->CallBooleanMethodV(this,obj,methodID,args);
    va_end(args);
    return result;
}

```

#### CallBooleanMethodV()

```

jboolean CallBooleanMethodV(jobject obj, jmethodID methodID,
                          va_list args) {
    return functions->CallBooleanMethodV(this,obj,methodID,args);
}

```

#### CallBooleanMethodA()

```

jboolean CallBooleanMethodA(jobject obj, jmethodID methodID,
                          const jvalue * args) {
    return functions->CallBooleanMethodA(this,obj,methodID, args);
}

```

## CallByteMethod()

```
jbyte CallByteMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jbyte result;
    va_start(args, methodID);
    result = functions->CallByteMethodV(this, obj, methodID, args);
    va_end(args);
    return result;
}
```

## CallByteMethodV()

```
jbyte CallByteMethodV(jobject obj, jmethodID methodID,
                      va_list args) {
    return functions->CallByteMethodV(this, obj, methodID, args);
}
```

## CallByteMethodA()

```
jbyte CallByteMethodA(jobject obj, jmethodID methodID,
                      const jvalue * args) {
    return functions->CallByteMethodA(this, obj, methodID, args);
}
```

## CallCharMethod()

```
jchar CallCharMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jchar result;
    va_start(args, methodID);
    result = functions->CallCharMethodV(this, obj, methodID, args);
    va_end(args);
    return result;
}
```

## CallCharMethodV()

```
jchar CallCharMethodV(jobject obj, jmethodID methodID,
                      va_list args) {
    return functions->CallCharMethodV(this, obj, methodID, args);
}
```

## CallCharMethodA()

```

jchar CallCharMethodA(jobject obj, jmethodID methodID,
                      const jvalue * args) {
    return functions->CallCharMethodA(this,obj,methodID,args);
}

```

#### CallShortMethod()

```

jshort CallShortMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jshort result;
    va_start(args,methodID);
    result = functions->CallShortMethodV(this,obj,methodID,args);
    va_end(args);
    return result;
}

```

#### CallShortMethodV()

```

jshort CallShortMethodV(jobject obj, jmethodID methodID,
                        va_list args) {
    return functions->CallShortMethodV(this,obj,methodID,args);
}

```

#### CallShortMethodA()

```

jshort CallShortMethodA(jobject obj, jmethodID methodID,
                        const jvalue * args) {
    return functions->CallShortMethodA(this,obj,methodID,args);
}

```

#### CallIntMethod()

```

jint CallIntMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jint result;
    va_start(args,methodID);
    result = functions->CallIntMethodV(this,obj,methodID,args);
    va_end(args);
    return result;
}

```

#### CallIntMethodV()

```

jint CallIntMethodV(jobject obj, jmethodID methodID,
                    va_list args) {

```

```

        return functions->CallIntMethodV(this, obj, methodID, args);
    }

```

#### CallIntMethodA()

```

jint CallIntMethodA(jobject obj, jmethodID methodID,
                    const jvalue * args) {
    return functions->CallIntMethodA(this, obj, methodID, args);
}

```

#### CallLongMethod()

```

jlong CallLongMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jlong result;
    va_start(args, methodID);
    result = functions->CallLongMethodV(this, obj, methodID, args);
    va_end(args);
    return result;
}

```

#### CallLongMethodV()

```

jlong CallLongMethodV(jobject obj, jmethodID methodID,
                       va_list args) {
    return functions->CallLongMethodV(this, obj, methodID, args);
}

```

#### CallLongMethodA()

```

jlong CallLongMethodA(jobject obj, jmethodID methodID,
                       const jvalue * args) {
    return functions->CallLongMethodA(this, obj, methodID, args);
}

```

#### CallFloatMethod()

```

jfloat CallFloatMethod(jobject obj, jmethodID methodID, ...) {
    va_list args;
    jfloat result;
    va_start(args, methodID);
    result = functions->CallFloatMethodV(this, obj, methodID, args);
    va_end(args);
    return result;
}

```

### CallFloatMethodV()

```
jfloat CallFloatMethodV(jobject obj, jmethodID methodID,  
                        va_list args) {  
    return functions->CallFloatMethodV(this,obj,methodID,args);  
}
```

### CallFloatMethodA()

```
jfloat CallFloatMethodA(jobject obj, jmethodID methodID,  
                        const jvalue * args) {  
    return functions->CallFloatMethodA(this,obj,methodID,args);  
}
```

### CallDoubleMethod()

```
jdouble CallDoubleMethod(jobject obj, jmethodID methodID, ...) {  
    va_list args;  
    jdouble result;  
    va_start(args,methodID);  
    result = functions->CallDoubleMethodV(this,obj,methodID,args);  
    va_end(args);  
    return result;  
}
```

### CallDoubleMethodV()

```
jdouble CallDoubleMethodV(jobject obj, jmethodID methodID,  
                          va_list args) {  
    return functions->CallDoubleMethodV(this,obj,methodID,args);  
}
```

### CallDoubleMethodA()

```
jdouble CallDoubleMethodA(jobject obj, jmethodID methodID,  
                          const jvalue * args) {  
    return functions->CallDoubleMethodA(this,obj,methodID,args);  
}
```

### CallVoidMethod()

```
void CallVoidMethod(jobject obj, jmethodID methodID, ...) {  
    va_list args;  
    va_start(args,methodID);  
    functions->CallVoidMethodV(this,obj,methodID,args);  
}
```



```

        va_end(args);
    }

```

### CallVoidMethodV()

```

void CallVoidMethodV(jobject obj, jmethodID methodID,
                    va_list args) {
    functions->CallVoidMethodV(this, obj, methodID, args);
}

```

### CallVoidMethodA()

```

void CallVoidMethodA(jobject obj, jmethodID methodID,
                    const jvalue * args) {
    functions->CallVoidMethodA(this, obj, methodID, args);
}

```

## Nonvirtual Calling of Methods

---

### CallNonvirtualObjectMethod()

```

jobject CallNonvirtualObjectMethod(jobject obj, jclass clazz,
                                   jmethodID methodID, ...) {
    va_list args;
    jobject result;
    va_start(args, methodID);
    result = functions->CallNonvirtualObjectMethodV(this, obj, clazz,
                                                    methodID, args);
    va_end(args);
    return result;
}

```

### CallNonvirtualObjectMethodV()

```

jobject CallNonvirtualObjectMethodV(jobject obj, jclass clazz,
                                   jmethodID methodID, va_list args) {
    return functions->CallNonvirtualObjectMethodV(this, obj, clazz,
                                                    methodID, args);
}

```

### CallNonvirtualObjectMethodA()

```

jobject CallNonvirtualObjectMethodA(jobject obj, jclass clazz,
                                   jmethodID methodID, const jvalue *

```

```
args) {
    return functions->CallNonvirtualObjectMethodA(this,obj,clazz,
                                                    methodID,args);
}
```

#### CallNonvirtualBooleanMethod()

```
jboolean CallNonvirtualBooleanMethod(jobject obj, jclass clazz,
                                       jmethodID methodID, ...) {
    va_list args;
    jboolean result;
    va_start(args,methodID);
    result = functions->CallNonvirtualBooleanMethodV(this,obj,clazz,
                                                    methodID,args);
    va_end(args);
    return result;
}
```

#### CallNonvirtualBooleanMethodV()

```
jboolean CallNonvirtualBooleanMethodV(jobject obj, jclass clazz,
                                       jmethodID methodID, va_list args) {
    return functions->CallNonvirtualBooleanMethodV(this,obj,clazz,
                                                    methodID,args);
}
```

#### CallNonvirtualBooleanMethodA()

```
jboolean CallNonvirtualBooleanMethodA(jobject obj, jclass clazz,
                                       jmethodID methodID, const jvalue *
args) {
    return functions->CallNonvirtualBooleanMethodA(this,obj,clazz,
                                                    methodID, args);
}
```

#### CallNonvirtualByteMethod()

```
jbyte CallNonvirtualByteMethod(jobject obj, jclass clazz,
                                jmethodID methodID, ...) {
    va_list args;
    jbyte result;
    va_start(args,methodID);
    result = functions->CallNonvirtualByteMethodV(this,obj,clazz,
                                                    methodID,args);
    va_end(args);
}
```

```

        return result;
    }

```

#### CallNonvirtualByteMethodV()

```

jbyte CallNonvirtualByteMethodV(jobject obj, jclass clazz,
                                jmethodID methodID, va_list args) {
    return functions->CallNonvirtualByteMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualByteMethodA()

```

jbyte CallNonvirtualByteMethodA(jobject obj, jclass clazz,
                                jmethodID methodID, const jvalue * args)
{
    return functions->CallNonvirtualByteMethodA(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualCharMethod()

```

jchar CallNonvirtualCharMethod(jobject obj, jclass clazz,
                                jmethodID methodID, ...) {
    va_list args;
    jchar result;
    va_start(args,methodID);
    result = functions->CallNonvirtualCharMethodV(this,obj,clazz,
                                                methodID,args);
    va_end(args);
    return result;
}

```

#### CallNonvirtualCharMethodV()

```

jchar CallNonvirtualCharMethodV(jobject obj, jclass clazz,
                                jmethodID methodID, va_list args) {
    return functions->CallNonvirtualCharMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualCharMethodA()

```

jchar CallNonvirtualCharMethodA(jobject obj, jclass clazz,
                                jmethodID methodID, const jvalue * args)

```

```

{
    return functions->CallNonvirtualCharMethodA(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualShortMethod()

```

jshort CallNonvirtualShortMethod(jobject obj, jclass clazz,
                                jmethodID methodID, ...) {
    va_list args;
    jshort result;
    va_start(args,methodID);
    result = functions->CallNonvirtualShortMethodV(this,obj,clazz,
                                                    methodID,args);
    va_end(args);
    return result;
}

```

#### CallNonvirtualShortMethodV()

```

jshort CallNonvirtualShortMethodV(jobject obj, jclass clazz,
                                jmethodID methodID, va_list args) {
    return functions->CallNonvirtualShortMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualShortMethodA()

```

jshort CallNonvirtualShortMethodA(jobject obj, jclass clazz,
                                jmethodID methodID, const jvalue *
args) {
    return functions->CallNonvirtualShortMethodA(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualIntMethod()

```

jint CallNonvirtualIntMethod(jobject obj, jclass clazz,
                             jmethodID methodID, ...) {
    va_list args;
    jint result;
    va_start(args,methodID);
    result = functions->CallNonvirtualIntMethodV(this,obj,clazz,
                                                  methodID,args);
    va_end(args);
}

```

```

        return result;
    }

```

#### CallNonvirtualIntMethodV()

```

jint CallNonvirtualIntMethodV(jobject obj, jclass clazz,
                              jmethodID methodID, va_list args) {
    return functions->CallNonvirtualIntMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualIntMethodA()

```

jint CallNonvirtualIntMethodA(jobject obj, jclass clazz,
                              jmethodID methodID, const jvalue * args) {
    return functions->CallNonvirtualIntMethodA(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualLongMethod()

```

jlong CallNonvirtualLongMethod(jobject obj, jclass clazz,
                               jmethodID methodID, ...) {
    va_list args;
    jlong result;
    va_start(args,methodID);
    result = functions->CallNonvirtualLongMethodV(this,obj,clazz,
                                                  methodID,args);
    va_end(args);
    return result;
}

```

#### CallNonvirtualLongMethodV()

```

jlong CallNonvirtualLongMethodV(jobject obj, jclass clazz,
                                 jmethodID methodID, va_list args) {
    return functions->CallNonvirtualLongMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualLongMethodA()

```

jlong CallNonvirtualLongMethodA(jobject obj, jclass clazz,
                                 jmethodID methodID, const jvalue * args)
{

```

```

        return functions->CallNonvirtualLongMethodA(this,obj,clazz,
                                                    methodID,args);
    }

```

#### CallNonvirtualFloatMethod()

```

jfloat CallNonvirtualFloatMethod(jobject obj, jclass clazz,
                                jmethodID methodID, ...) {
    va_list args;
    jfloat result;
    va_start(args,methodID);
    result = functions->CallNonvirtualFloatMethodV(this,obj,clazz,
                                                    methodID,args);

    va_end(args);
    return result;
}

```

#### CallNonvirtualFloatMethodV()

```

jfloat CallNonvirtualFloatMethodV(jobject obj, jclass clazz,
                                jmethodID methodID,
                                va_list args) {
    return functions->CallNonvirtualFloatMethodV(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualFloatMethodA()

```

jfloat CallNonvirtualFloatMethodA(jobject obj, jclass clazz,
                                jmethodID methodID,
                                const jvalue * args) {
    return functions->CallNonvirtualFloatMethodA(this,obj,clazz,
                                                methodID,args);
}

```

#### CallNonvirtualDoubleMethod()

```

jdouble CallNonvirtualDoubleMethod(jobject obj, jclass clazz,
                                   jmethodID methodID, ...) {
    va_list args;
    jdouble result;
    va_start(args,methodID);
    result = functions->CallNonvirtualDoubleMethodV(this,obj,clazz,
                                                    methodID,args);

    va_end(args);
}

```

```

        return result;
    }

```

#### CallNonvirtualDoubleMethodV()

```

jdouble CallNonvirtualDoubleMethodV(jobject obj, jclass clazz,
                                     jmethodID methodID,
                                     va_list args) {
    return functions->CallNonvirtualDoubleMethodV(this,obj,clazz,
                                                  methodID,args);
}

```

#### CallNonvirtualDoubleMethodA()

```

jdouble CallNonvirtualDoubleMethodA(jobject obj, jclass clazz,
                                     jmethodID methodID,
                                     const jvalue * args) {
    return functions->CallNonvirtualDoubleMethodA(this,obj,clazz,
                                                  methodID,args);
}

```

#### CallNonvirtualVoidMethod()

```

void CallNonvirtualVoidMethod(jobject obj, jclass clazz,
                              jmethodID methodID, ...) {
    va_list args;
    va_start(args,methodID);
    functions->CallNonvirtualVoidMethodV(this,obj,clazz,methodID,args);
    va_end(args);
}

```

#### CallNonvirtualVoidMethodV()

```

void CallNonvirtualVoidMethodV(jobject obj, jclass clazz,
                              jmethodID methodID,
                              va_list args) {
    functions->CallNonvirtualVoidMethodV(this,obj,clazz,methodID,args);
}

```

#### CallNonvirtualVoidMethodA()

```

void CallNonvirtualVoidMethodA(jobject obj, jclass clazz,
                              jmethodID methodID,
                              const jvalue * args) {
    functions->CallNonvirtualVoidMethodA(this,obj,clazz,methodID,args);
}

```

```
}
```

## Getting Fields

---

### GetFieldID()

```
jfieldID GetFieldID(jclass clazz, const char *name,  
                    const char *sig) {  
    return functions->GetFieldID(this,clazz,name,sig);  
}
```

### GetObjectField()

```
jobject GetObjectField(jobject obj, jfieldID fieldID) {  
    return functions->GetObjectField(this,obj,fieldID);  
}
```

### GetBooleanField()

```
jboolean GetBooleanField(jobject obj, jfieldID fieldID) {  
    return functions->GetBooleanField(this,obj,fieldID);  
}
```

### GetByteField()

```
jbyte GetByteField(jobject obj, jfieldID fieldID) {  
    return functions->GetByteField(this,obj,fieldID);  
}
```

### GetCharField()

```
jchar GetCharField(jobject obj, jfieldID fieldID) {  
    return functions->GetCharField(this,obj,fieldID);  
}
```

### GetShortField()

```
jshort GetShortField(jobject obj, jfieldID fieldID) {  
    return functions->GetShortField(this,obj,fieldID);  
}
```

### GetIntField()

```
jint GetIntField(jobject obj, jfieldID fieldID) {  
    return functions->GetIntField(this,obj,fieldID);  
}
```



```
}
```

### GetLongField()

```
jlong GetLongField(jobject obj, jfieldID fieldID) {  
    return functions->GetLongField(this,obj,fieldID);  
}
```

### GetFloatField()

```
jfloat GetFloatField(jobject obj, jfieldID fieldID) {  
    return functions->GetFloatField(this,obj,fieldID);  
}
```

### GetDoubleField()

```
jdouble GetDoubleField(jobject obj, jfieldID fieldID) {  
    return functions->GetDoubleField(this,obj,fieldID);  
}
```

## Setting Fields

---

### SetObjectField()

```
void SetObjectField(jobject obj, jfieldID fieldID, jobject val) {  
    functions->SetObjectField(this,obj,fieldID,val);  
}
```

### SetBooleanField()

```
void SetBooleanField(jobject obj, jfieldID fieldID,  
                    jboolean val) {  
    functions->SetBooleanField(this,obj,fieldID,val);  
}
```

### SetByteField()

```
void SetByteField(jobject obj, jfieldID fieldID,  
                 jbyte val) {  
    functions->SetByteField(this,obj,fieldID,val);  
}
```

### SetCharField()

```
void SetCharField(jobject obj, jfieldID fieldID,
```

```

        jchar val) {
    functions->SetCharField(this,obj,fieldID,val);
}

```

### SetShortField()

```

void SetShortField(jobject obj, jfieldID fieldID,
                    jshort val) {
    functions->SetShortField(this,obj,fieldID,val);
}

```

### SetIntField()

```

void SetIntField(jobject obj, jfieldID fieldID,
                  jint val) {
    functions->SetIntField(this,obj,fieldID,val);
}

```

### SetLongField()

```

void SetLongField(jobject obj, jfieldID fieldID,
                   jlong val) {
    functions->SetLongField(this,obj,fieldID,val);
}

```

### SetFloatField()

```

void SetFloatField(jobject obj, jfieldID fieldID,
                   jfloat val) {
    functions->SetFloatField(this,obj,fieldID,val);
}

```

### SetDoubleField()

```

void SetDoubleField(jobject obj, jfieldID fieldID,
                    jdouble val) {
    functions->SetDoubleField(this,obj,fieldID,val);
}

```

## Calling Static Methods

---

### GetStaticMethodID()

```

jmethodID GetStaticMethodID(jclass clazz, const char *name,

```

```

        const char *sig) {
    return functions->GetStaticMethodID(this,clazz,name,sig);
}

```

### CallStaticObjectMethod()

```

jobject CallStaticObjectMethod(jclass clazz, jmethodID methodID,
                               ...) {
    va_list args;
    jobject result;
    va_start(args,methodID);
    result = functions-
>CallStaticObjectMethodV(this,clazz,methodID,args);
    va_end(args);
    return result;
}

```

### CallStaticObjectMethodV()

```

jobject CallStaticObjectMethodV(jclass clazz, jmethodID methodID,
                                va_list args) {
    return functions->CallStaticObjectMethodV(this,clazz,methodID,args);
}

```

### CallStaticObjectMethodA()

```

jobject CallStaticObjectMethodA(jclass clazz, jmethodID methodID,
                                const jvalue *args) {
    return functions->CallStaticObjectMethodA(this,clazz,methodID,args);
}

```

### CallStaticBooleanMethod()

```

jboolean CallStaticBooleanMethod(jclass clazz,
                                  jmethodID methodID, ...) {
    va_list args;
    jboolean result;
    va_start(args,methodID);
    result = functions-
>CallStaticBooleanMethodV(this,clazz,methodID,args);
    va_end(args);
    return result;
}

```

### CallStaticBooleanMethodV()

```
jboolean CallStaticBooleanMethodV(jclass clazz,
                                   jmethodID methodID, va_list args) {
    return functions->CallStaticBooleanMethodV(this, clazz, methodID, args);
}
```

### CallStaticBooleanMethodA()

```
jboolean CallStaticBooleanMethodA(jclass clazz,
                                   jmethodID methodID, const jvalue *args)
{
    return functions->CallStaticBooleanMethodA(this, clazz, methodID, args);
}
```

### CallStaticByteMethod()

```
jbyte CallStaticByteMethod(jclass clazz,
                            jmethodID methodID, ...) {
    va_list args;
    jbyte result;
    va_start(args, methodID);
    result = functions->CallStaticByteMethodV(this, clazz, methodID, args);
    va_end(args);
    return result;
}
```

### CallStaticByteMethodV()

```
jbyte CallStaticByteMethodV(jclass clazz,
                             jmethodID methodID, va_list args) {
    return functions->CallStaticByteMethodV(this, clazz, methodID, args);
}
```

### CallStaticByteMethodA()

```
jbyte CallStaticByteMethodA(jclass clazz,
                             jmethodID methodID, const jvalue *args) {
    return functions->CallStaticByteMethodA(this, clazz, methodID, args);
}
```

### CallStaticCharMethod()

```

jchar CallStaticCharMethod(jclass clazz,
                           jmethodID methodID, ...) {
    va_list args;
    jchar result;
    va_start(args, methodID);
    result = functions->CallStaticCharMethodV(this, clazz, methodID, args);
    va_end(args);
    return result;
}

```

#### CallStaticCharMethodV()

```

jchar CallStaticCharMethodV(jclass clazz,
                           jmethodID methodID, va_list args) {
    return functions->CallStaticCharMethodV(this, clazz, methodID, args);
}

```

#### CallStaticCharMethodA()

```

jchar CallStaticCharMethodA(jclass clazz,
                           jmethodID methodID, const jvalue *args) {
    return functions->CallStaticCharMethodA(this, clazz, methodID, args);
}

```

#### CallStaticShortMethod()

```

jshort CallStaticShortMethod(jclass clazz,
                             jmethodID methodID, ...) {
    va_list args;
    jshort result;
    va_start(args, methodID);
    result = functions->CallStaticShortMethodV(this, clazz, methodID, args);
    va_end(args);
    return result;
}

```

#### CallStaticShortMethodV()

```

jshort CallStaticShortMethodV(jclass clazz,
                             jmethodID methodID, va_list args) {
    return functions->CallStaticShortMethodV(this, clazz, methodID, args);
}

```

#### CallStaticShortMethodA()

```

jshort CallStaticShortMethodA(jclass clazz,
                               jmethodID methodID, const jvalue *args) {
    return functions->CallStaticShortMethodA(this,clazz,methodID,args);
}

```

#### CallStaticIntMethod()

```

jint CallStaticIntMethod(jclass clazz,
                         jmethodID methodID, ...) {
    va_list args;
    jint result;
    va_start(args,methodID);
    result = functions->CallStaticIntMethodV(this,clazz,methodID,args);
    va_end(args);
    return result;
}

```

#### CallStaticIntMethodV()

```

jint CallStaticIntMethodV(jclass clazz,
                          jmethodID methodID, va_list args) {
    return functions->CallStaticIntMethodV(this,clazz,methodID,args);
}

```

#### CallStaticIntMethodA()

```

jint CallStaticIntMethodA(jclass clazz,
                          jmethodID methodID, const jvalue *args) {
    return functions->CallStaticIntMethodA(this,clazz,methodID,args);
}

```

#### CallStaticLongMethod()

```

jlong CallStaticLongMethod(jclass clazz,
                           jmethodID methodID, ...) {
    va_list args;
    jlong result;
    va_start(args,methodID);
    result = functions->CallStaticLongMethodV(this,clazz,methodID,args);
    va_end(args);
    return result;
}

```

#### CallStaticLongMethodV()

```

jlong CallStaticLongMethodV(jclass clazz,
                             jmethodID methodID, va_list args) {
    return functions->CallStaticLongMethodV(this,clazz,methodID,args);
}

```

#### CallStaticLongMethodA()

```

jlong CallStaticLongMethodA(jclass clazz,
                             jmethodID methodID, const jvalue *args) {
    return functions->CallStaticLongMethodA(this,clazz,methodID,args);
}

```

#### CallStaticFloatMethod()

```

jfloat CallStaticFloatMethod(jclass clazz,
                              jmethodID methodID, ...) {
    va_list args;
    jfloat result;
    va_start(args,methodID);
    result = functions->CallStaticFloatMethodV(this,clazz,methodID,args);
    va_end(args);
    return result;
}

```

#### CallStaticFloatMethodV()

```

jfloat CallStaticFloatMethodV(jclass clazz,
                              jmethodID methodID, va_list args) {
    return functions->CallStaticFloatMethodV(this,clazz,methodID,args);
}

```

#### CallStaticFloatMethodA()

```

jfloat CallStaticFloatMethodA(jclass clazz,
                              jmethodID methodID, const jvalue *args) {
    return functions->CallStaticFloatMethodA(this,clazz,methodID,args);
}

```

#### CallStaticDoubleMethod()

```

jdouble CallStaticDoubleMethod(jclass clazz,
                                jmethodID methodID, ...) {
    va_list args;
    jdouble result;
    va_start(args,methodID);
}

```

```

        result = functions-
>CallStaticDoubleMethodV(this,clazz,methodID,args);
        va_end(args);
        return result;
    }

```

#### CallStaticDoubleMethodV()

```

jdouble CallStaticDoubleMethodV(jclass clazz,
                                jmethodID methodID, va_list args) {
    return functions->CallStaticDoubleMethodV(this,clazz,methodID,args);
}

```

#### CallStaticDoubleMethodA()

```

jdouble CallStaticDoubleMethodA(jclass clazz,
                                jmethodID methodID, const jvalue *args) {
    return functions->CallStaticDoubleMethodA(this,clazz,methodID,args);
}

```

#### CallStaticVoidMethod()

```

void CallStaticVoidMethod(jclass cls, jmethodID methodID, ...) {
    va_list args;
    va_start(args,methodID);
    functions->CallStaticVoidMethodV(this,cls,methodID,args);
    va_end(args);
}

```

#### CallStaticVoidMethodV()

```

void CallStaticVoidMethodV(jclass cls, jmethodID methodID,
                           va_list args) {
    functions->CallStaticVoidMethodV(this,cls,methodID,args);
}

```

#### CallStaticVoidMethodA()

```

void CallStaticVoidMethodA(jclass cls, jmethodID methodID,
                           const jvalue * args) {
    functions->CallStaticVoidMethodA(this,cls,methodID,args);
}

```



## Getting Static Fields

---

### GetStaticFieldID()

```
jfieldID GetStaticFieldID(jclass clazz, const char *name,
                          const char *sig) {
    return functions->GetStaticFieldID(this, clazz, name, sig);
}
```

### GetStaticObjectField()

```
jobject GetStaticObjectField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticObjectField(this, clazz, fieldID);
}
```

### GetStaticBooleanField()

```
jboolean GetStaticBooleanField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticBooleanField(this, clazz, fieldID);
}
```

### GetStaticByteField()

```
jbyte GetStaticByteField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticByteField(this, clazz, fieldID);
}
```

### GetStaticCharField()

```
jchar GetStaticCharField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticCharField(this, clazz, fieldID);
}
```

### GetStaticShortField()

```
jshort GetStaticShortField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticShortField(this, clazz, fieldID);
}
```

### GetStaticIntField()

```
jint GetStaticIntField(jclass clazz, jfieldID fieldID) {
    return functions->GetStaticIntField(this, clazz, fieldID);
}
```

### GetStaticLongField()

```
jlong GetStaticLongField(jclass clazz, jfieldID fieldID) {  
    return functions->GetStaticLongField(this,clazz,fieldID);  
}
```

### GetStaticFloatField()

```
jfloat GetStaticFloatField(jclass clazz, jfieldID fieldID) {  
    return functions->GetStaticFloatField(this,clazz,fieldID);  
}
```

### GetStaticDoubleField()

```
jdouble GetStaticDoubleField(jclass clazz, jfieldID fieldID) {  
    return functions->GetStaticDoubleField(this,clazz,fieldID);  
}
```

## Setting Static Fields

---

### SetStaticObjectField()

```
void SetStaticObjectField(jclass clazz, jfieldID fieldID,  
                           jobject value) {  
    functions->SetStaticObjectField(this,clazz,fieldID,value);  
}
```

### SetStaticBooleanField()

```
void SetStaticBooleanField(jclass clazz, jfieldID fieldID,  
                           jboolean value) {  
    functions->SetStaticBooleanField(this,clazz,fieldID,value);  
}
```

### SetStaticByteField()

```
void SetStaticByteField(jclass clazz, jfieldID fieldID,  
                        jbyte value) {  
    functions->SetStaticByteField(this,clazz,fieldID,value);  
}
```

### SetStaticCharField()

```
void SetStaticCharField(jclass clazz, jfieldID fieldID,
```

```

        jchar value) {
    functions->SetStaticCharField(this,clazz,fieldID,value);
}

```

#### SetStaticShortField()

```

void SetStaticShortField(jclass clazz, jfieldID fieldID,
        jshort value) {
    functions->SetStaticShortField(this,clazz,fieldID,value);
}

```

#### SetStaticIntField()

```

void SetStaticIntField(jclass clazz, jfieldID fieldID,
        jint value) {
    functions->SetStaticIntField(this,clazz,fieldID,value);
}

```

#### SetStaticLongField()

```

void SetStaticLongField(jclass clazz, jfieldID fieldID,
        jlong value) {
    functions->SetStaticLongField(this,clazz,fieldID,value);
}

```

#### SetStaticFloatField()

```

void SetStaticFloatField(jclass clazz, jfieldID fieldID,
        jfloat value) {
    functions->SetStaticFloatField(this,clazz,fieldID,value);
}

```

#### SetStaticDoubleField()

```

void SetStaticDoubleField(jclass clazz, jfieldID fieldID,
        jdouble value) {
    functions->SetStaticDoubleField(this,clazz,fieldID,value);
}

```

## Working with Strings

---

#### NewString()

```

jstring NewString(const jchar *unicode, jsize len) {

```

```

        return functions->NewString(this,unicode,len);
    }

```

### GetStringLength()

```

jsize GetStringLength(jstring str) {
    return functions->GetStringLength(this,str);
}

```

### GetStringChars()

```

const jchar *GetStringChars(jstring str, jboolean *isCopy) {
    return functions->GetStringChars(this,str,isCopy);
}

```

### ReleaseStringChars()

```

void ReleaseStringChars(jstring str, const jchar *chars) {
    functions->ReleaseStringChars(this,str,chars);
}

```

### NewStringUTF()

```

jstring NewStringUTF(const char *utf) {
    return functions->NewStringUTF(this,utf);
}

```

### GetStringUTFLength()

```

jsize GetStringUTFLength(jstring str) {
    return functions->GetStringUTFLength(this,str);
}

```

### GetStringUTFChars()

```

const char* GetStringUTFChars(jstring str, jboolean *isCopy) {
    return functions->GetStringUTFChars(this,str,isCopy);
}

```

### ReleaseStringUTFChars()

```

void ReleaseStringUTFChars(jstring str, const char* chars) {
    functions->ReleaseStringUTFChars(this,str,chars);
}

```

## Working with Arrays

---

### GetArrayLength()

```
jsize GetArrayLength(jarray array) {  
    return functions->GetArrayLength(this,array);  
}
```

### NewObjectArray()

```
jobjectArray NewObjectArray(jsize len, jclass clazz,  
                             jobject init) {  
    return functions->NewObjectArray(this,len,clazz,init);  
}
```

### GetObjectArrayElement()

```
jobject GetObjectArrayElement(jobjectArray array, jsize index) {  
    return functions->GetObjectArrayElement(this,array,index);  
}
```

### SetObjectArrayElement()

```
void SetObjectArrayElement(jobjectArray array, jsize index,  
                           jobject val) {  
    functions->SetObjectArrayElement(this,array,index,val);  
}
```

### NewBooleanArray()

```
jbooleanArray NewBooleanArray(jsize len) {  
    return functions->NewBooleanArray(this,len);  
}
```

### NewByteArray()

```
jbyteArray NewByteArray(jsize len) {  
    return functions->NewByteArray(this,len);  
}
```

### NewCharArray()

```
jcharArray NewCharArray(jsize len) {  
    return functions->NewCharArray(this,len);  
}
```

### NewShortArray()

```
jshortArray NewShortArray(jsize len) {  
    return functions->NewShortArray(this, len);  
}
```

### NewIntArray()

```
jintArray NewIntArray(jsize len) {  
    return functions->NewIntArray(this, len);  
}
```

### NewLongArray()

```
jlongArray NewLongArray(jsize len) {  
    return functions->NewLongArray(this, len);  
}
```

### NewFloatArray()

```
jfloatArray NewFloatArray(jsize len) {  
    return functions->NewFloatArray(this, len);  
}
```

### NewDoubleArray()

```
jdoubleArray NewDoubleArray(jsize len) {  
    return functions->NewDoubleArray(this, len);  
}
```

### GetBooleanArrayElements()

```
jboolean * GetBooleanArrayElements(jbooleanArray array, jboolean  
*isCopy) {  
    return functions->GetBooleanArrayElements(this, array, isCopy);  
}
```

### GetByteArrayElements()

```
jbyte * GetByteArrayElements(jbyteArray array, jboolean *isCopy) {  
    return functions->GetByteArrayElements(this, array, isCopy);  
}
```

### GetCharArrayElements()

```
jchar * GetCharArrayElements(jcharArray array, jboolean *isCopy) {
```

```

        return functions->GetCharArrayElements(this,array,isCopy);
    }

```

#### GetShortArrayElements()

```

jshort * GetShortArrayElements(jshortArray array, jboolean *isCopy) {
    return functions->GetShortArrayElements(this,array,isCopy);
}

```

#### GetIntArrayElements()

```

jint * GetIntArrayElements(jintArray array, jboolean *isCopy) {
    return functions->GetIntArrayElements(this,array,isCopy);
}

```

#### GetLongArrayElements()

```

jlong * GetLongArrayElements(jlongArray array, jboolean *isCopy) {
    return functions->GetLongArrayElements(this,array,isCopy);
}

```

#### GetFloatArrayElements()

```

jfloat * GetFloatArrayElements(jfloatArray array, jboolean *isCopy) {
    return functions->GetFloatArrayElements(this,array,isCopy);
}

```

#### GetDoubleArrayElements()

```

jdouble * GetDoubleArrayElements(jdoubleArray array, jboolean *isCopy)
{
    return functions->GetDoubleArrayElements(this,array,isCopy);
}

```

#### ReleaseBooleanArrayElements()

```

void ReleaseBooleanArrayElements(jbooleanArray array,
                                jboolean *elems,
                                jint mode) {
    functions->ReleaseBooleanArrayElements(this,array,elems,mode);
}

```

#### ReleaseByteArrayElements()

```

void ReleaseByteArrayElements(jbyteArray array,
                              jbyte *elems,

```

```

        jint mode) {
    functions->ReleaseByteArrayElements(this,array,elems,mode);
}

```

#### ReleaseCharArrayElements()

```

void ReleaseCharArrayElements(jcharArray array,
                               jchar *elems,
                               jint mode) {
    functions->ReleaseCharArrayElements(this,array,elems,mode);
}

```

#### ReleaseShortArrayElements()

```

void ReleaseShortArrayElements(jshortArray array,
                                jshort *elems,
                                jint mode) {
    functions->ReleaseShortArrayElements(this,array,elems,mode);
}

```

#### ReleaseIntArrayElements()

```

void ReleaseIntArrayElements(jintArray array,
                              jint *elems,
                              jint mode) {
    functions->ReleaseIntArrayElements(this,array,elems,mode);
}

```

#### ReleaseLongArrayElements()

```

void ReleaseLongArrayElements(jlongArray array,
                                jlong *elems,
                                jint mode) {
    functions->ReleaseLongArrayElements(this,array,elems,mode);
}

```

#### ReleaseFloatArrayElements()

```

void ReleaseFloatArrayElements(jfloatArray array,
                                jfloat *elems,
                                jint mode) {
    functions->ReleaseFloatArrayElements(this,array,elems,mode);
}

```



### ReleaseDoubleArrayElements()

```
void ReleaseDoubleArrayElements(jdoubleArray array,
                                jdouble *elems,
                                jint mode) {
    functions->ReleaseDoubleArrayElements(this,array,elems,mode);
}
```

### GetBooleanArrayRegion()

```
void GetBooleanArrayRegion(jbooleanArray array,
                            jsize start, jsize len, jboolean *buf) {
    functions->GetBooleanArrayRegion(this,array,start,len,buf);
}
```

### GetByteArrayRegion()

```
void GetByteArrayRegion(jbyteArray array,
                         jsize start, jsize len, jbyte *buf) {
    functions->GetByteArrayRegion(this,array,start,len,buf);
}
```

### GetCharArrayRegion()

```
void GetCharArrayRegion(jcharArray array,
                        jsize start, jsize len, jchar *buf) {
    functions->GetCharArrayRegion(this,array,start,len,buf);
}
```

### GetShortArrayRegion()

```
void GetShortArrayRegion(jshortArray array,
                          jsize start, jsize len, jshort *buf) {
    functions->GetShortArrayRegion(this,array,start,len,buf);
}
```

### GetIntArrayRegion()

```
void GetIntArrayRegion(jintArray array,
                       jsize start, jsize len, jint *buf) {
    functions->GetIntArrayRegion(this,array,start,len,buf);
}
```

### GetLongArrayRegion()

```

void GetLongArrayRegion(jlongArray array,
                        jsize start, jsize len, jlong *buf) {
    functions->GetLongArrayRegion(this,array,start,len,buf);
}

```

#### GetFloatArrayRegion()

```

void GetFloatArrayRegion(jfloatArray array,
                        jsize start, jsize len, jfloat *buf) {
    functions->GetFloatArrayRegion(this,array,start,len,buf);
}

```

#### GetDoubleArrayRegion()

```

void GetDoubleArrayRegion(jdoubleArray array,
                        jsize start, jsize len, jdouble *buf) {
    functions->GetDoubleArrayRegion(this,array,start,len,buf);
}

```

#### SetBooleanArrayRegion()

```

void SetBooleanArrayRegion(jbooleanArray array, jsize start, jsize
len,
                        const jboolean *buf) {
    functions->SetBooleanArrayRegion(this,array,start,len,buf);
}

```

#### SetByteArrayRegion()

```

void SetByteArrayRegion(jbyteArray array, jsize start, jsize len,
                        const jbyte *buf) {
    functions->SetByteArrayRegion(this,array,start,len,buf);
}

```

#### SetCharArrayRegion()

```

void SetCharArrayRegion(jcharArray array, jsize start, jsize len,
                        const jchar *buf) {
    functions->SetCharArrayRegion(this,array,start,len,buf);
}

```

#### SetShortArrayRegion()

```

void SetShortArrayRegion(jshortArray array, jsize start, jsize len,
                        const jshort *buf) {

```

```

        functions->SetShortArrayRegion(this,array,start,len,buf);
    }

```

### SetIntArrayRegion()

```

void SetIntArrayRegion(jintArray array, jsize start, jsize len,
                        const jint *buf) {
    functions->SetIntArrayRegion(this,array,start,len,buf);
}

```

### SetLongArrayRegion()

```

void SetLongArrayRegion(jlongArray array, jsize start, jsize len,
                        const jlong *buf) {
    functions->SetLongArrayRegion(this,array,start,len,buf);
}

```

### SetFloatArrayRegion()

```

void SetFloatArrayRegion(jfloatArray array, jsize start, jsize len,
                        const jfloat *buf) {
    functions->SetFloatArrayRegion(this,array,start,len,buf);
}

```

### SetDoubleArrayRegion()

```

void SetDoubleArrayRegion(jdoubleArray array, jsize start, jsize len,
                        const jdouble *buf) {
    functions->SetDoubleArrayRegion(this,array,start,len,buf);
}

```

## Utility Functions

---

### Register Natives()

```

jint RegisterNatives(jclass clazz, const JNINativeMethod *methods,
                    jint nMethods) {
    return functions->RegisterNatives(this,clazz,methods,nMethods);
}

```

### Unregister Natives()

```

jint UnregisterNatives(jclass clazz) {
    return functions->UnregisterNatives(this,clazz);
}

```

```
}
```

#### MonitorEnter()

```
jint MonitorEnter(jobject obj) {  
    return functions->MonitorEnter(this,obj);  
}
```

#### MonitorExit()

```
jint MonitorExit(jobject obj) {  
    return functions->MonitorExit(this,obj);  
}
```

#### GetJavaVM()

```
jint GetJavaVM(JavaVM **vm) {  
    return functions->GetJavaVM(this,vm);  
}
```

#### GetStringRegion()

```
void GetStringRegion(jstring str, jsize start, jsize len, jchar *buf) {  
    functions->GetStringRegion(this,str,start,len,buf);  
}
```

#### GetStringUTFRegion()

```
void GetStringUTFRegion(jstring str, jsize start, jsize len, char  
*buf) {  
    functions->GetStringUTFRegion(this,str,start,len,buf);  
}
```

#### GetPrimitiveArrayCritical()

```
void * GetPrimitiveArrayCritical(jarray array, jboolean *isCopy) {  
    return functions->GetPrimitiveArrayCritical(this,array,isCopy);  
}
```

#### ReleasePrimitiveArrayCritical()

```
void ReleasePrimitiveArrayCritical(jarray array, void *carray, jint  
mode) {  
    functions->ReleasePrimitiveArrayCritical(this,array,carray,mode);  
}
```

### GetStringCritical()

```
const jchar * GetStringCritical(jstring string, jboolean *isCopy) {  
    return functions->GetStringCritical(this,string,isCopy);  
}
```

### ReleaseStringCritical()

```
void ReleaseStringCritical(jstring string, const jchar *cstring) {  
    functions->ReleaseStringCritical(this,string,cstring);  
}
```

### NewWeakGlobalRef()

```
jweak NewWeakGlobalRef(jobject obj) {  
    return functions->NewWeakGlobalRef(this,obj);  
}
```

### DeleteWeakGlobalRef()

```
void DeleteWeakGlobalRef(jweak ref) {  
    functions->DeleteWeakGlobalRef(this,ref);  
}
```

### ExceptionCheck()

```
jboolean ExceptionCheck() {  
    return functions->ExceptionCheck(this);  
}
```

### NewDirectByteBuffer()

```
jobject NewDirectByteBuffer(void* address, jlong capacity) {  
    return functions->NewDirectByteBuffer(this, address, capacity);  
}
```

### GetDirectBufferAddress()

```
void* GetDirectBufferAddress(jobject buf) {  
    return functions->GetDirectBufferAddress(this, buf);  
}
```

### GetDirectBufferCapacity()

```
jlong GetDirectBufferCapacity(jobject buf) {  
    return functions->GetDirectBufferCapacity(this, buf);  
}
```

## GetObjectRefType()

```
jobjectRefType GetObjectRefType(jobject obj) {  
    return functions->GetObjectRefType(this, obj);  
}  
  
#endif /* __cplusplus */  
};
```