

Serializable Objects in Java

In Java terminology, a *serializable object* typically refers to an object that implements the `java.io.Serializable` interface and therefore supports serialization to and from a byte stream. All Java classes generated from Slice definitions implement the `java.io.Serializable` interface.

In addition to serializing Slice types, applications may also need to incorporate foreign types into their Slice definitions. Ice allows you to pass Java [serializable objects](#) directly as operation parameters or as fields of another data type. For example:

Slice

```
[ "java:serializable:SomePackage.JavaClass" ]
sequence<byte> JavaObj;
struct MyStruct {
    int i;
    JavaObj o;
};

interface Example {
    void op(JavaObj inObj, MyStruct s, out JavaObj outObj);
};
```

The generated code for `MyStruct` contains a member `i` of type `int` and a member `o` of type `SomePackage.JavaClass`:

Java

```
public final class MyStruct implements java.lang.Cloneable {
    public int i;
    public SomePackage.JavaClass o;

    // ...
}
```

Similarly, the signature for `op` has parameters of type `JavaClass` and `MyStruct` for the in-parameters, and `Ice.Holder<SomePackage.JavaClass>` for the out-parameter. (Out-parameters are always passed as `Ice.Holder<class>`.)

Java

```
void op(SomePackage.JavaClass inObj,
        MyStruct s,
        Ice.Holder<SomePackage.JavaClass> outObj);
```

Of course, your client and server code must have an implementation of `JavaClass` that derives from `java.io.Serializable`:

Java

```
package SomePackage;

public class JavaClass implements java.io.Serializable {
    // ...
}
```

You can implement this class in any way you see fit — the Ice run time does not place any other requirements on the implementation.

See Also

- [Serializable Objects](#)