/\*\*

\* Read the non-static and non-transient fields of the current class from

\* this stream. This may only be called from the readObject method of the

\* class being deserialized. It will throw the NotActiveException if it is

\* called otherwise.

\*

\* **@throws** ClassNotFoundException if the class of a serialized object

\* could not be found.

\* **@throws** IOException if an I/O error occurs.

\* **@throws** NotActiveException if the stream is not currently reading

\* objects.

\*/

**public** **void** defaultReadObject()

**throws** IOException, ClassNotFoundException

{

SerialCallbackContext ctx = curContext;

**if** (ctx == **null**) {

**throw** **new** NotActiveException("not in call to readObject");

}

Object curObj = ctx.getObj();

ObjectStreamClass curDesc = ctx.getDesc();

bin.setBlockDataMode(**false**);

defaultReadFields(curObj, curDesc);

bin.setBlockDataMode(**true**);

**if** (!curDesc.hasWriteObjectData()) {

defaultReadObject🡪defaultReadFields

/\*\*

\* Reads in values of serializable fields declared by given class

\* descriptor. If obj is non-null, sets field values in obj. Expects that

\* passHandle is set to obj's handle before this method is called.

\*/

**private** **void** defaultReadFields(Object obj, ObjectStreamClass desc)

**throws** IOException

{

Class<?> cl = desc.forClass();

**if** (cl != **null** && obj != **null** && !cl.isInstance(obj)) {

**throw** **new** ClassCastException();

}

**int** primDataSize = desc.getPrimDataSize();

**if** (primVals == **null** || primVals.length < primDataSize) {

primVals = **new** **byte**[primDataSize];

}

bin.readFully(primVals, 0, primDataSize, **false**);

**if** (obj != **null**) {

desc.setPrimFieldValues(obj, primVals);

}

**int** objHandle = passHandle;

ObjectStreamField[] fields = desc.getFields(**false**);

Object[] objVals = **new** Object[desc.getNumObjFields()];

**int** numPrimFields = fields.length - objVals.length;

**for** (**int** i = 0; i < objVals.length; i++) {

ObjectStreamField f = fields[numPrimFields + i];

objVals[i] = readObject0(f.isUnshared());

**if** (f.getField() != **null**) {

handles.markDependency(objHandle, passHandle);

}

}

**if** (obj != **null**) {

desc.setObjFieldValues(obj, objVals);

}

passHandle = objHandle;

}