Serialization

I can have my class implement the Serializable interface, and then use readObject and writeObject with Object streams, and it all works.

Well, that's only part of the truth.

serialVersionUID

The serialVersionUID field is a runtime field, that the compiler will implicitly create, if it's not explicitly declared, for classes that are serializable.

It's based on class details such as the number of fields, their types, and declarations.

When we read an object from a stream, the runtime checks the stored serialVersionUID.

If they don't match, then there's a compatibility problem and the runtime will throw this invalid class exception.

Versions may be incompatible between JVMs

In addition, it's possible that different compilers will generate this implied field differently.

To ensure this doesn't happen, it is strongly recommended that you include this as a private static field, as shown on this slide.

```
private final static long serialVersionUID = 1L;
```

What constitutes an Incompatible Change?

- Changing the declared type of a primitive field.
- Deleting fields.
- Changing a non-static field to static, or a non-transient field to transient.

There are other more complicated changes, such as moving a class within its hierarchy, changing the writeObject and readObject methods after you've used them to serialize previously, and a few others.

For the full documentation on incompatible changes, you can visit the link shown here.

https://docs.oracle.com/en/java/javase/17/docs/specs/serialization/version.html#incompatible-changes

What constitutes a Compatible Change?

The good news is that not all changes you make to your class are going to invalidate the serialization process.

Some changes are compatible changes:

- Adding fields.
- Adding writeObject and readObject methods.
- Changing the access to a field.
- Changing a field from static to nonstatic, or transient.

For the full documentation on compatible changes, you can visit the link shown here.

https://docs.oracle.com/en/java/javase/17/docs/specs/serialization/version.html#compatible-changes

What constitutes an Incompatible Change?

Notice that the first bullet point, says the incompatible change is when you change the declared type of a primitive field.

- Changing the declared type of a **primitive** field.
- Deleting fields.
- Changing a non-static field to static or a non-transient field to transient.

The serialization process, for an object field, includes information about the object's type, and object's super type.

For the ArrayList and LinkedList, they share a super type, so the deserialization went smoothly here.