Serialization (or Externalization?)

- Serialization uses certain default behaviors to store and later recreate the object. You may specify in what order or how to handle references and complex data structures, but eventually it comes down to using the default behavior for each primitive data field.

  Externalization is used in the rare cases that you really want to store and rebuild your object in a completely different way and without using the default serialization mechanisms for data fields. For example, imagine that you had your own unique encoding and compression scheme.

- By implementating `java.io.Serializable`, you get "automatic" serialization capability for objects of your class. No need to implement any other logic, it'll just work. The Java runtime will use reflection to figure out how to marshal and unmarshal your objects.

  In earlier version of Java, reflection was very slow, and so serializaing large object graphs (e.g. in client-server RMI applications) was a bit of a performance problem. To handle this situation, the `java.io.Externalizable` interface was provided, which is like `java.io.Serializable` but with custom-written mechanisms to perform the marshalling and unmarshalling functions (you need to implement `readExternal` and `writeExternal` methods on your class). This gives you the means to get around the reflection performance bottleneck.

  In recent versions of Java (1.3 onwards, certainly) the performance of reflection is vastly better than it used to be, and so this is much less of a problem. I suspect you'd be hard-pressed to get a meaningful benefit from `Externalizable` with a modern JVM.

  Also, the built-in Java serialization mechanism isn't the only one, you can get third-party replacements, such as JBoss Serialization, which is considerably quicker, and is a drop-in replacement for the default.

  A big downside of `Externalizable` is that you have to maintain this logic yourself - if you add, remove or change a field in your class, you have to change your `writeExternal`/`readExternal` methods to account for it.

  In summary, `Externalizable` is a relic of the Java 1.1 days. There's really no need for it any more.