INTRODUCTION TO SERIALIZATION

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object.

After a serialized object has been written into a file, it can be read from the file and deserialized that is, the type information and bytes that represent the object and its data can be used to recreate the object in memory.

This entire process is JVM independent, meaning an object can be serialized on one platform and deserialized on an entirely different platform.

Classes **ObjectInputStream** and **ObjectOutputStream** are high-level streams that contain the methods for serializing and deserializing an object.

The ObjectOutputStream class contains many write methods for writing various data types

```
public final void writeObject(Object x) throws IOException
```

The above method serializes an Object and sends it to the output stream.

The ObjectInputStream class contains the following method for deserializing an object

```
public final Object readObject() throws IOException,
ClassNotFoundException
```

This method retrieves the next Object out of the stream and deserializes it.

The return value is Object, so you will need to cast it to its appropriate data type.

For a class to be serialized successfully, two conditions must be met:

- The class must implement the java.io.Serializable interface.
- All of the fields in the class must be serializable. If a field is not serializable, it must be marked transient.

ObjectOutputStream can write primitive types and graphs of objects to an OutputStream as a stream of bytes