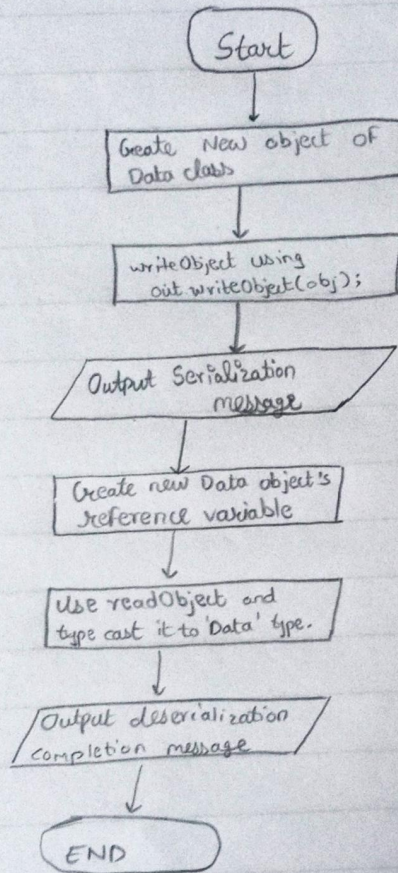


## Practical No. 19

Aim: Create, debug and run java programs based on object Serialization.

flowchart:





Aim: Create, debug and run java programs based on Object Serialization.

Theory:

What is Object Serialization?

Object serialization is a mechanism provided in Java, using which 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 type's of data stored in the object.

After a serialized object has been written to a file, it can be read from a file and be deserialized, i.e. the type of information and bytes used to store the object can be used to recreate the object in memory.

Most impressive is that the entire process of serialization is JVM independent, that is, an object can be serialised on one platform and can be deserialized from a or a completely different platform.

The serialization and deserialization of objects is done through classes 'ObjectInputStream' and 'ObjectOutputStream' which are high level streams which contains the methods.

The method used to Serialize an object:

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

The method used to deserialize an object

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



What is Object Serialization?

Object serialization is a mechanism provided in Java which allows an object to be converted into a sequence of bytes. This sequence of bytes can be stored in a file or transmitted over a network. The object's state is saved as information about the object's type and the values of its attributes.

When an object is serialized, its state is converted into a byte stream. This byte stream can then be saved to a file or sent over a network. The process of converting an object into a byte stream is called serialization. The reverse process, converting a byte stream back into an object, is called deserialization. The `Serializable` interface is used to mark classes that can be serialized. The `ObjectOutputStream` class is used to write objects to a stream, and the `ObjectInputStream` class is used to read objects from a stream.

The `Serializable` interface is a marker interface, meaning it does not contain any methods. It is used to indicate that a class can be serialized. The `ObjectOutputStream` class provides the `writeObject()` method to write objects to a stream. The `ObjectInputStream` class provides the `readObject()` method to read objects from a stream. The `Serializable` interface is implemented by the `ObjectOutputStream` and `ObjectInputStream` classes.

Conclusion:

Hence, I create, debugged and executed java programs based on object serialization.