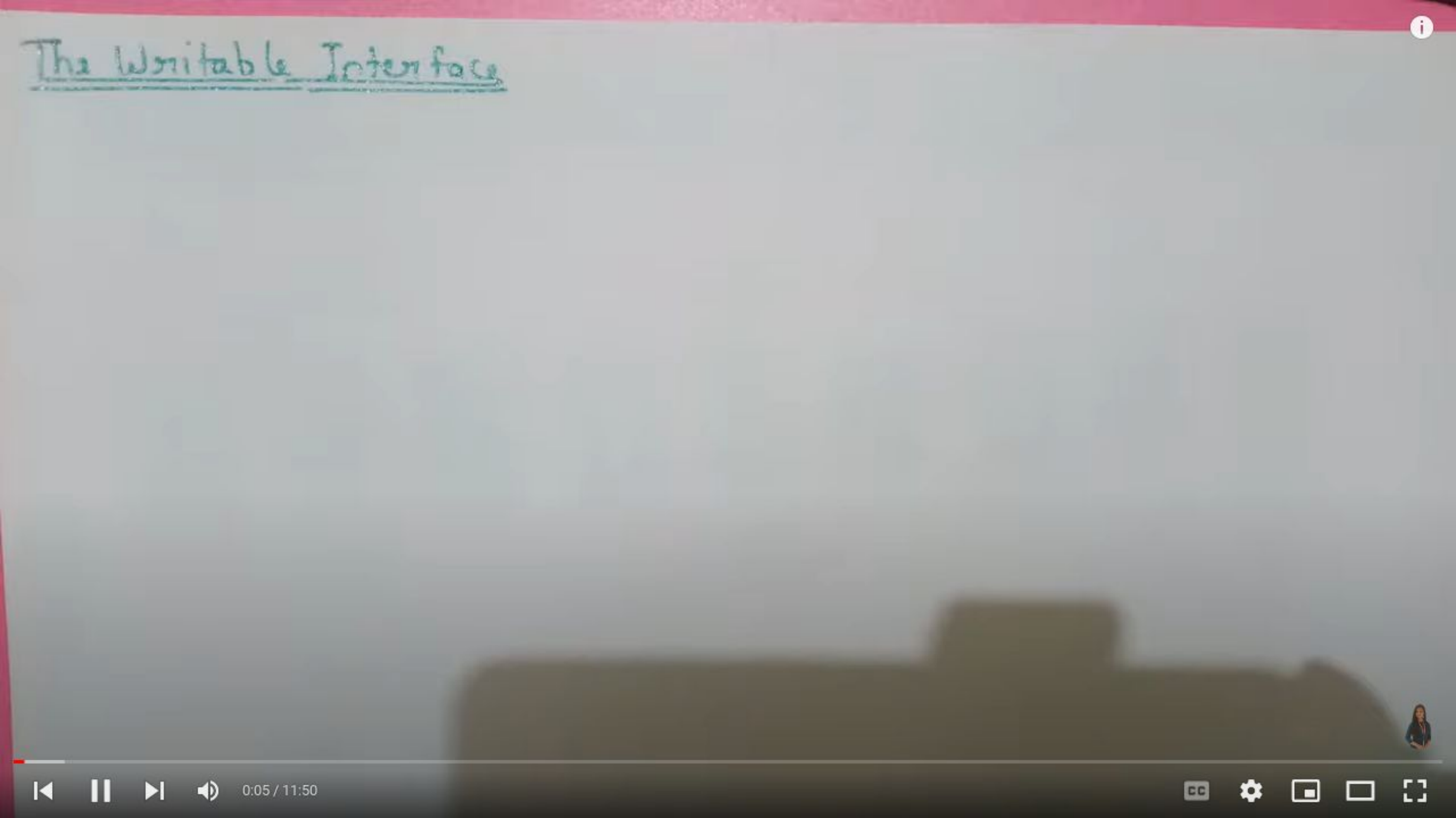
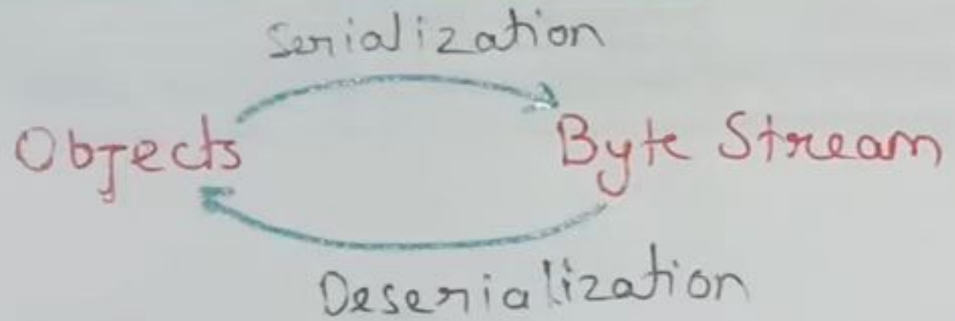
The Writable Interface

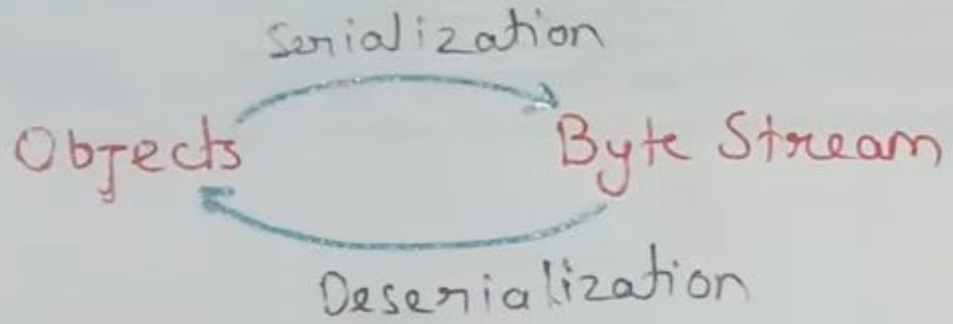


The Writable Interface



* Hadoop uses its own serialization format Writables.

The Writable Interface



* Hadoop uses its own serialization format Writables.

serialization format Writables.

It defines two methods :- DataOutput (write)
DataInput (readfields).

Program:

```
package org.apache.hadoop.io;
import java.io.DataOutput;
import java.io.DataInput;
public interface Writable
{
    void write (DataOutput out);
    void readfields (DataInput in);
}
```

Note: If writable is not present in Hadoop, then it uses the serialization of Java which increases the data over head in network.

serialization format Writables.

It defines two methods :- DataOutput (write)
DataInput (readfields).

Program:

```
package org.apache.hadoop.io;
import java.io.DataOutput;
import java.io.DataInput;
public interface Writable
{
    void write (DataOutput out);
    void readfields (DataInput in);
}
```

Note: If writable is not present in Hadoop, then it uses the serialization of Java which increases the data over head in network.

```
public interface WritableComparable extends Writable, Comparable
{
    void readFields (DataInput in);
    void write (DataOutput out);
    int compareTo (WritableComparable O);
}
```

```
public interface WritableComparable extends Writable, Comparable
{ void readFields (DataInput in);
  void write (DataOutput out);
  int compareTo (WritableComparable O);
}
```

-1, 0, 1

```
public interface WritableComparable extends Writable, Comparable
{ void readFields (DataInput in);
  void write (DataOutput out);
  int compareTo (WritableComparable O);
}
```

-1, 0, 1

public interface WritableComparable extends Writable, Comparab

{ void readfields (DataInput in);
  void write (DataOutput out);
  int compareTo (WritableComparable O);
}

-1, 0, 1

$(K_1, V_1) \rightarrow$ Map $\rightarrow (K_2, V_2)$

$(K_2, \text{List}[V_2]) \rightarrow$ Reduce $\rightarrow (K_3, V_3)$

Program:

void write (DataOutput out);
int compareTo (WritableComparable o);
}

-1, 0, 1

$(K_1, V_1) \rightarrow$ Map $\rightarrow (K_2, V_2)$

$(K_2, \text{List}[V_2]) \rightarrow$ Reduce $\rightarrow (K_3, V_3)$

Program:

```
package org.apache.hadoop.io;
import java.util.Comparator;
Public interface RawComparator extends Comparator
```

void write (DataOutput out);
int compareTo (Writable Comparable O);
}

-1, 0, 1

$(K_1, V_1) \rightarrow$ Map $\rightarrow (K_2, V_2)$

$(K_2, \text{List}[V_2]) \rightarrow$ Reduce $\rightarrow (K_3, V_3)$

Program:

```
package org.apache.hadoop.io;
import java.util.Comparator;
Public interface RawComparator extends Comparator
```

10 seconds

10:26 / 11:50

-1, 0, 1

$(K_1, V_1) \rightarrow$ Map $\rightarrow (K_2, V_2)$

$(K_2, \text{List}[V_2]) \rightarrow$ Reduce $\rightarrow (K_3, V_3)$

Program:

```
package org.apache.hadoop.io;
import java.util.Comparator;
Public interface RawComparator extends Comparator.
{
  public int compare (byte[]b1, int s1, int l1, byte[]b2,
                      int s2, int l2);
}
```

* WritableComparator is a general purpose implementation of RawComparator for WritableComparable classes.