**Explain in brief Writable and Writable Comparable in Hadoop with an example.**

**Writable is an interface which is used as wrapper for all the primitive datatypes**

**So that the transfer of the files over the network will become very difficult.**

**This ease of transferring the files over the network is done with the help of the serialization of the data.**

**The writable datatypes we first introduced in Hadoop to address the above problem. This is why we are using the Intwritable, Text, LongWritable etc. as the datatypes.**

**Serialization with the help of the java is also possible but it is generates very large amount of data. Hence it will generate high overhead on the datanode.**

**Eg.**

**smallInt serialized value using Java serializer**

aced0005737200116a6176612e6c616e672e496e74656765

7212e2a0a4f781873802000149000576616c7565787200106a6176612e

6c616e672e4e756d62657286ac951d0b94e08b020000787000000064

**smallInt serialized value using IntWritable**

00000064

----------------------------------------------------------------------

**We can make our own custom Writable type.**

public class add implements Writable{

public int a;

public int b;

public add()

{

this.a=a;

this.b=b;

}

public void write(DataOutput out) throws IOException {

    out.writeInt(a);

    out.writeInt(b);

  }

public void readFields(DataInput in) throws IOException {

    a = in.readInt();

    b = in.readInt();

 }

 public String toString() {

    return Integer.toString(a) + ", " + Integer.toString(b)

}

}

**This above method is similar to that of the creating in java. But two extra method of readfields and write are necessary.**

**---------------------------------------------------------------------------------------------------------------------**

**WritableComparable**

**Suppose that the above custom writable we created needs to be compared with other datatypes then we have to use the writable comparable. Mostly used when are creating it for keys.**

**Suppose we are using writable for a key the shuffling of the keys will not happen in shuffle and sort**

**Mostly shuffling of the keys is to be done. Hence whenever we are dealing with the key that time we are referring writable comparable.**

**Similar to custom writable we can create our own customcomparable here the compare to method is used for sorting.**

public class add implements WritableComparable{

public int a;

public int b;

public add(){

this.a=a;

this.b=b;

}

public void write(DataOutput out) throws IOException {

    out.writeint(a);

    out.writeint(b);

  }

public void readFields(DataInput in) throws IOException {

    a = in.readint();

    b = in.readint();

  }

public int CompareTo(add c){

int presentValue=this.value;

int CompareValue=c.value;

return (presentValue < CompareValue ? -1 : (presentValue==CompareValue ? 0 : 1));

}

public int hashCode() {

    return Integer.IntToIntBits(a)^ Integer.IntToIntBits(b);

  }

}