**Code:**

**package** com.venkat.mapreduce;

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.ByteWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.ShortWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

**import** org.apache.hadoop.mapreduce.Reducer;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

**public** **class** MaxTemperature {

**public** **static** **class** MaxTempMapper **extends** Mapper<LongWritable, Text, ShortWritable, ByteWritable> {

**private** ShortWritable year = **new** ShortWritable();

**private** ByteWritable temp = **new** ByteWritable();

**public** **void** map(LongWritable key, Text value, Context context) **throws** IOException, InterruptedException

{

String[] tokens = value.toString().split(" ");

**if**(tokens.length == 2)

{

year.set(Short.*parseShort*(tokens[0]));

temp.set(Byte.*parseByte*(tokens[1]));

context.write(year, temp);

}

}

}

**public** **static** **class** MaxTempReducer **extends** Reducer<ShortWritable, ByteWritable, ShortWritable, ByteWritable> {

**private** ByteWritable result = **new** ByteWritable();

**public** **void** reduce(ShortWritable key, Iterable<ByteWritable> values, Context context) **throws** IOException, InterruptedException

{

**byte** maxTemp = -1;

**byte** temp;

**for**(ByteWritable value: values)

{

temp = value.get();

**if**(temp > maxTemp)

{

maxTemp = temp;

}

}

result.set(maxTemp);

context.write(key, result);

}

}

**public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {

Configuration conf = **new** Configuration();

Job job = **new** ~~Job~~(conf, "Max Temperature");

job.setMapperClass(MaxTempMapper.**class**);

job.setCombinerClass(MaxTempReducer.**class**);

job.setReducerClass(MaxTempReducer.**class**);

job.setJarByClass(MaxTemperature.**class**);

job.setMapOutputKeyClass(ShortWritable.**class**);

job.setMapOutputValueClass(ByteWritable.**class**);

job.setOutputKeyClass(ShortWritable.**class**);

job.setOutputValueClass(ByteWritable.**class**);

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);

}

}

**Output File:**



**Output File Content:**

1900 46

1901 48

1902 49

1903 35

1904 46

1905 35

1906 32

1907 49

1908 44

1909 38

1910 47

1911 48

1912 44

1913 43

1914 49

1915 49

1916 18

1917 35

1918 49

1919 42

1920 47

1921 47

1922 45

1923 41

1924 49

1925 48

1926 49

1927 47

1928 48

1929 35

1930 48

1931 37

1932 33

1933 43

1934 47

1935 40

1936 48

1937 44

1938 43

1939 48

1940 49

1941 49

1942 24

1943 45

1944 39

1945 47

1946 48

1947 41

1948 42

1949 48

1950 41

1951 48

1952 49

1953 43

1954 46

1955 47

1956 48

1957 49

1958 34

1959 45

1960 44

1961 35

1962 46

1963 27

1964 42

1965 35

1966 48

1967 41

1968 47

1969 47

1970 49

1971 42

1972 49

1973 49

1974 49

1975 46

1976 40

1977 49

1978 48

1979 31

1980 48

1981 49

1982 45

1983 38

1984 44

1985 48

1986 48

1987 43

1988 49

1989 47

1990 44

1991 49

1992 48

1993 45

1994 38

1995 48

1996 47

1997 49

1998 48

1999 46

2000 49

2001 45

2002 45

2003 30

2004 48

2005 48

2006 47

2007 47

2008 46

2009 43

2010 47

2011 42

2012 45

2013 49