BufferedWriter:
   By using BufferedWriter object we can write character data to the file.

Constructor
   BufferedWriter bw=new BufferedWriter(writer w);
   BufferedWriter bw=new BufferedWriter(writer w,int buffersize);

Note:BufferedWriter never communicates directly with the file it should communicates via some writer object.

Which of the following declarations are valid?
1. BufferedWriter bw =new BufferedWriter("cricket.txt"); (invalid)
2. BufferedWriter bw =new BufferedWriter (new File("cricket.txt")); (invalid)
3. BufferedWriter bw =new BufferedWriter (new FileWriter("cricket.txt")); (valid)
4. BufferedWriter bw =new BufferedWriter(new BufferedWriter(new FileWriter("crickter.txt")));

Methods
========
1. write(int ch);
2. write(char[] ch);
3. write(String s);
4. flush();
5. close();
6. newline();
      Inserting a new line character to the file.

eg#1.
```
import java.io.*;
import java.util.*;

public class Test
{
   public static void main(String[] args) throws IOException
      {
         //BufferedReader,BufferedWriter
         BufferedWriter bw  = new BufferedWriter(new FileWriter("cricket.txt"));
            bw.write(97);
            bw.newLine();
            char[] arr = {'p','w','s','k','i','l','l','s'};
            bw.write(arr);
            bw.newLine();
            bw.write("PWIOI");
            bw.newLine();

            bw.flush();

            bw.close();
      }
}
```

Note
1.bw.close()// recomended to use

2.fw.close()// not recomended to use
3.bw.close()// not recomended to use
  fw.close()

=> When ever we are closing BufferedWriter automatically underlying writer will be

closed and we are not close explicitly.

Note:
When compared with FileWriter which of the following capability(facility) is available as method in BufferedWriter.
1. Writing data to the file.
2. Closing the writer.
3. Flush the writer.
4. Inserting newline character.

BufferedReader:
This is the most enhanced(better) Reader to read character data from the file.

Constructors:
BufferedReader br=new BufferedReader(Reader r);
BufferedReader br=new BufferedReader(Reader r,int buffersize);

Note
=> BufferedReader can not communicate directly with the File it should communicate via some Reader object.
=> The main advantage of BufferedReader over FileReader is we can read data line by line instead of character by character.

Methods:
1. int read();
2. int read(char[] ch);
3. String readLine();
It attempts to read next line and return it , from the File. if the next line is not available then this method returns null.
4. void close();

eg#1.
```
import java.io.*;
import java.util.*;

//Client Code
public class Test
{
   public static void main(String[] args) throws IOException
      {
         //BufferedReader,BufferedWriter
         BufferedReader br  = new BufferedReader(new FileReader("Sample.txt"));
         String line = br.readLine();

         while (line!=null)
         {
               System.out.println(line);
               line =br.readLine();

         }
            br.close();
      }
}
```

Output
```
Sachin
Saurav
Dhoni
```

```
kohli
raina
dravid
yuvi
zaheer
munaf

Note:
1.br.close()  // recomended to use

2.fw.close() // not recomended to use
3.br.close() // not recomended to use
   fw.close()

=> Whenever we are closing BufferedReader automatically underlying FileReader will
be closed it is not required to close explicitly.
=> Even this rule is applicable for BufferedWriter also.

PrintWriter:
=> This is the most enhanced Writer to write text data to the file.
=> By using FileWriter and BufferedWriter we can write only character data to the
File but  by using  PrintWriter
   we can write any type of data to the File.

Constructors:
PrintWriter pw=new PrintWriter(String name);
PrintWriter pw=new PrintWriter(File f);
PrintWriter pw=new PrintWriter(Writer w);

PrintWriter can communicate either directly to the File or via some Writer object
also.

Methods:
1. write(int ch);
2. write (char[] ch);
3. write(String s);

4. flush();
5. close();

6. print(char ch);
7. print (int i);
8. print (double d);
9. print (boolean b);
10.print (String s);

11.println(char ch);
12.println (int i);
13.println(double d);
14.println(boolean b);
15.println(String s);

eg#1.
import java.io.*;
import java.util.*;

//Client Code
public class Test
{
```

```
   public static void main(String[] args) throws IOException
      {
          //Handled only character type of data
            //FileWriter ---> BufferedWriter---> PrintWriter(println())
            //FileReader ---> BufferdReader(readLine())

            //Handle images,video file,audio files,... -> binary data
            //InputStream   :: BufferedInputStream
            //OutputStream  :: BufferedOutputStream

         //PrintWriter
         PrintWriter out = new PrintWriter(new FileWriter("Cricket.txt"));
         out.write(97);//97 -> unicode value
         out.println(100);//100 -> int value
         out.println(true);
         out.println('c');
         out.println(10.5f);
         out.println("sachintendulkar");

         out.flush();
         out.close();

      }
}
cricket.txt
++++++++++
a100
true
c
10.5
sachintendulkar
```

What is the difference between write(100) and print(100)?
=> In the case of write(100) the corresponding character "d" will be added to the File but
=> In the case of print(100) "100" value will be added directly to the File.

Note 1:
1. The most enhanced Reader to read character data from the File is BufferedReader.
2. The most enhanced Writer to write character data to the File is PrintWriter.

Note 2:
1.In general we can use Readers and Writers to handle character data. Where as we can use InputStreams and OutputStreams to handle
  binary data(like images, audio files, video files etc).

2. We can use OutputStream to write binary data to the File and we can use InputStream to read binary data from the File
            Character Data   => Reader and Writer
            Binary Data   => InputStream and OutputStream

eg#1.Requirement => file1.txt ,file2.txt copy all the contents to output.txt

```
import java.io.*;
import java.util.*;
```

```
public class Test
{
   public static void main(String[] args) throws IOException
      {


        //Setting the writer object to output.txt
        PrintWriter out = new PrintWriter("output.txt");

        //Reader the data from file1.txt
        BufferedReader br1 = new BufferedReader(new FileReader("file1.txt"));
        String line = br1.readLine();
        while (line!=null)
        {
              //write the content to output.txt
            out.println(line);
            line = br1.readLine();
        }

        //Reader the data from file2.txt
        br1 = new BufferedReader(new FileReader("file2.txt"));
        line = br1.readLine();
        while (line!=null)
        {
              //write the content to output.txt
            out.println(line);
            line = br1.readLine();
        }

        out.flush();
        out.close();
        br1.close();

      }
}
```

eg#2.
Requirement => file1.txt file2.txt copy one line from file1.txt and from file2.txt to file3.txt.

```
import java.io.*;
import java.util.*;

public class Test
{
   public static void main(String[] args) throws IOException
      {
        //Requirement => file1.txt ,file2.txt copy all the contents to output.txt

        //Setting the writer object to output.txt
        PrintWriter out = new PrintWriter("output.txt");


        BufferedReader br1 = new BufferedReader(new FileReader("file1.txt"));
        BufferedReader br2 = new BufferedReader(new FileReader("file2.txt"));

        //Reading the data from file1.txt and file2.txt
        String line1 = br1.readLine();
```

```
        String line2 = br2.readLine();

        while (line1!=null || line2!=null)
        {
              //write the content to output.txt
             if (line1!=null)
             {
                   out.println(line1);
                   line1 = br1.readLine();
             }

             if (line2!=null)
             {
                  out.println(line2);
                  line2 = br2.readLine();
             }
        }


        out.flush();
        out.close();
        br1.close();
        br2.close();

      }
}

eg#3.
Requirement => Write a program to perform extraction of mobile no only if there is
no duplicates
import java.io.*;
import java.util.*;

//Client Code
public class Test
{
   public static void main(String[] args) throws IOException
      {
        //Requirement => file1.txt ,file2.txt copy all the contents to output.txt

        //Setting the writer object to output.txt
        PrintWriter out = new PrintWriter("output.txt");



        //Setting the reader object to input.txt
        BufferedReader br = new BufferedReader(new FileReader("input.txt"));
        String target =br.readLine();

            while (target!=null)
            {
                  boolean isAvailable =false;

                  BufferedReader br1 = new BufferedReader(new
FileReader("output.txt"));
                  String line =br1.readLine();


                  //loop and check whether target exists or not in the output.txt
```

```
file
                  while (line!=null)
                  {
                        if (target.equals(line))
                        {
                              isAvailable = true;
                              break;
                        }

                        //read nextLine from output.txt
                        line = br1.readLine();

                  } //end of while loop

                  if (isAvailable==false)
                  {
                        //write to output.txt
                        out.println(target);
                        out.flush();
                  }

                  //read the next line from input.txt
                  target =br.readLine();

            }//end of while loop


        out.close();
        br.close();
      }
}
```

eg#4.
Requirement => Write a program to remove duplicates from the file