Important Note: In this note, we have just provided the main codes for doing a program by implementing recursion and have not declared the class or written the main() method.
In order to write the complete program, you are required to first declare a class and then write the recursive methods. An example of the code which you need to write before the methods is given below:

```
import java.io.*;
class Sample
   {
static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

Reversing a String

Recursive Method 1	Recursive Method 2	Recursive Method 3
Without Return Type	**With Return Type**	

void rev(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
rev(s,i+1);
System.out.print(ch);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any word : ");
String s=br.readLine();
System.out.print("Reverse of the word = ");
rev(s,0);
}
	String r="";
	String rev(String s, int i)
	{
	if(i<s.length())
	{
	char ch=s.charAt(i);
	rev(s,i+1);
	r=r+ch;
	}
	return r;
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any word : ");
	String s=br.readLine();
	String x=rev(s,0);
	System.out.print("Reverse of the word = "+x);
	}
		String r="";
		String rev(String s, int i)
		{
		if(i>=0)
		{
		char ch=s.charAt(i);
		r=r+ch;
		rev(s,i-1);
		}
		return r;
		}
		
		void display()throws IOException
		{
		System.out.print("Enter a word : ");
		String s=br.readLine();
		int len=s.length();
		String x=rev(s,len-1);
		System.out.print("Reverse of the word = "+x);
		}

Note: In some of the questions related to recursion, you will find mentioned in the question that you don't need to write the main() method.
 	
But, the students are advised to write the main() method after writing the above recursive methods, while they are practicing the programs. The main method is nothing but a few lines calling the above created methods.
 	
An example of the main method for the above program is given below.
 	
public static void main()throws IOException
{
Sample ob=new Sample();
ob.display();
}
		Iterative Method
		
		String r="";
		String rev(String s)
		{
		int i=0;
		while(i<s.length())
		{
		char ch=s.charAt(i);
		r=r+ch;
		i++;
		}
		return r;
		}

Note: In the First example we are using a recursive function without any return type and we are not saving the reverse of the String in any number. We are just printing the characters as we get them in reverse order due to the LIFO property of the stack used in recursion.

In the 2nd and 3rd example we are using a recursive function with a return type and we are saving the reverse of the String in an instance variable 'r' and we are returning the result at the end. The difference in the approach of the 2nd and the 3rd method is that the 2nd method extracts characters from the beginning of the String and uses the LIFO property of the stack used in recursion to reverse the word whereas, the 3rd method extracts characters from the end of the String and adds this to the variable 'r' and does not utilize the LIFO property of the stack used in recursion.

Another Recursive method

```
String r="";
String rev(String s, int i)
 {
  if(i<s.length())
    {
     char ch=s.charAt(i);
     r=ch+r;
     rev(s,i+1);
    }
   return r;
 }
```

The above method is just a slight variation of the 2nd method with the only difference being that we are not using the LIFO property and the change: **r=ch+r;**

Checking for Palindrome Word – ISC 2008

After reversing a String using the 2nd or the 3rd method, you can check whether that String is a palindrome or not by adding the below code in the display () function after the printing of the reverse word:

```
void display()throws IOException
 {
 ……
 ……
 if(x.equals(s))
 System.out.print("The word is a Palindrome");
 else
 System.out.print("The word is Not a Palindrome");
 }
```

Extracting Characters of a String and performing any given operation

Recursive Method	Corresponding Iterative Method

void stringOp(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
Write the operation you want to perform with the character stored in 'ch' over here.
stringOp(s,i+1);
}
}
	void stringOp(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	Write the operation you want to perform with the character stored in 'ch' over here.
	i++;
	}
	}

In the above code, we are extracting the characters from the beginning of the String passed as parameter. After storing it in the variable 'ch', you write the code of the operation you want to perform with that character. You can also send the character to another function which will perform some task with that character.	

Some Programs using the above technique

1. Converting characters of a String from UpperCase to LowerCase and vice versa

Recursive Method 1	Corresponding Iterative Method

void caseConvert(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(Character.isUpperCase(ch))
System.out.print(Character.toLowerCase(ch));
else
System.out.print(Character.toUpperCase(ch));
caseConvert(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
System.out.print("The new String = ");
ob.caseConvert(s,0);
}
	void caseConvert(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(Character.isUpperCase(ch))
	System.out.print(Character.toLowerCase(ch));
	else
	System.out.print(Character.toUpperCase(ch));
	i++;
	} }
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.print("The new String = ");
	ob.caseConvert(s);
	}

Recursive Method 2	Corresponding Iterative Method

String newstr="";
String caseConvert(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(Character.isUpperCase(ch))
ch=Character.toLowerCase(ch);
else
ch=Character.toUpperCase(ch);
newstr=newstr+ch;
caseConvert(s,i+1);
}
return newstr;
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
String x= caseConvert(s,0);
System.out.print("The new String = "+x);
}
	String newstr=""; int i=0;
	String caseConvert(String s)
	{
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(Character.isUpperCase(ch))
	ch=Character.toLowerCase(ch);
	else
	ch=Character.toUpperCase(ch);
	newstr=newstr+ch;
	i++;
	}
	return newstr;
	}
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	String x= caseConvert(s);
	System.out.print("The new String = "+x);
	}

Instead of writing the case conversion code inside the recursive function, you can also send that character to another function which is performing the case conversion. An example of this is given below: (**ISC 2010**)

```
String newstr="";
void recChange(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
newstr=newstr+caseConvert(ch);
recChange(s,i+1);
}
else
System.out.print("The new word = "+newstr);
}
```

```
char caseConvert(char ch)
{
if(Character.isUpperCase(ch))
ch=Character.toLowerCase(ch);
else
ch=Character.toUpperCase(ch);
return ch;
}
void display()throws IOException
{
System.out.print("Enter any word : ");
String s=br.readLine();
recChange(s,0);
}
```

2. Printing the initials of a name

Recursive Method	Corresponding Iterative Method

void initials(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch == ' ')
System.out.print(s.charAt(i+1)+".");
initials(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any name : ");
String s=br.readLine();
s=" "+s;
initials(s,0);
}
	void initials(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch == ' ')
	System.out.print(s.charAt(i+1)+".");
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any name : ");
	String s=br.readLine();
	s=" "+s;
	initials(s);
	}

In the above program, before sending the String to the recursive function, we have added a space before the String and then sent it. This is in accordance with the logic that the initials of a name are the characters which are just after a space.

3. Counting Number of UpperCase and LowerCase characters

Recursive Method	Corresponding Iterative Method

int cap=0,sm=0;
void countCase(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(Character.isUpperCase(ch))
cap++;
if(Character.isLowerCase(ch))
sm++;
countCase(s,i+1);
}
else
{
System.out.println("No. of Capital letters = "+cap);
System.out.println("No. of Small letters = "+sm);
}
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
countCase(s,0);
}
	int cap=0,sm=0;
	void countCase(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(Character.isUpperCase(ch))
	cap++;
	if(Character.isLowerCase(ch))
	sm++;
	i++;
	}
	System.out.println("No. of Capital letters = "+cap);
	System.out.println("No. of Small letters = "+sm);
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	countCase(s);
	}

The printing statements which are inside the **else clause** can also be written inside the display () function if we don't write them in the recursive function. The statements will be written after the statement **countCase(s,0);** and in that case we won't write any else part in the recursive function.	

4. Counting frequency of a character in a String

Recursive Method	Corresponding Iterative Method

int cou=0;
void count(String s, int i, char c)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch == c)
cou++;
count(s,i+1,c);
}
else
System.out.println("Frequency of "+c+" = "+cou);
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
System.out.print("Enter any character : ");
char c=(char)(System.in.read());
count(s,0,c);
}
	int cou=0;
	void count(String s, char c)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch == c)
	cou++;
	i++;
	}
	System.out.println("Frequency of "+c+" = "+cou);
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.print("Enter any character : ");
	char c=(char)(System.in.read());
	count(s,c);
	}

5. Counting Number of spaces, words, vowels, digits and consonants in a sentence

Recursive Method	Corresponding Iterative Method

int sp=0,word=0,vow=0,dig=0,con=0;
void count(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch == ' ')
sp++;
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
vow++;
if(ch>='0' && ch<='9')
dig++;
count(s,i+1);
}
else
{
word=sp+1;
con=s.length()-(sp+vow+dig);
System.out.println("No. of spaces = "+sp);
System.out.println("No. of words = "+word);
System.out.println("No. of vowels = "+vow);
System.out.println("No. of consonants = "+con);
System.out.println("No. of digits = "+dig);
}
}
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
count(s,0);
}
	int sp=0,word=0,vow=0,dig=0,con=0;
	void count(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch == ' ')
	sp++;
	if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
	ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
	vow++;
	if(ch>='0' && ch<='9')
	dig++;
	i++;
	}
	word=sp+1;
	con=s.length()-(sp+vow+dig);
	System.out.println("No. of spaces = "+sp);
	System.out.println("No. of words = "+word);
	System.out.println("No. of vowels = "+vow);
	System.out.println("No. of consonants = "+con);
	System.out.println("No. of digits = "+dig);
	}
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	count(s);
	}

6. Printing the String after removing all the Vowels

Recursive Method	Corresponding Iterative Method

void remVow(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!='a' && ch!='e' && ch!='i' && ch!='o' && ch!='u'
&& ch!='A' && ch!='E' && ch!='I' && ch!='O'
&&ch!='U')
System.out.print(ch);
remVow(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
System.out.print("The String after removing vowels : ");
remVow(s,0);
}
	void remVow(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!='a' && ch!='e' && ch!='i' && ch!='o' && ch!='u'
	&& ch!='A' && ch!='E' && ch!='I' && ch!='O'
	&&ch!='U')
	System.out.print(ch);
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.print("The String after removing vowels : ");
	remVow(s);
	}

7. Printing Pattern

If String is "JAVAFORSCHOOL" then:

```
J
JA
JAV
JAVA
JAVAF
JAVAFO
JAVAFOR
JAVAFORS
JAVAFORSC
JAVAFORSCH
JAVAFORSCHO
JAVAFORSCHOO
JAVAFORSCHOOL
```

Recursive Method 1	Corresponding Iterative Method

void pattern(String s, int i)
{
if(i<s.length())
{
for(int j=0;j<=i;j++)
{
char ch=s.charAt(j);
System.out.print(ch);
}
System.out.println();
pattern(s,i+1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
System.out.println("Output : ");
pattern(s,0);
}
	void pattern(String s)
	{
	int i=0;
	while(i<s.length())
	{
	for(int j=0;j<=i;j++)
	{
	char ch=s.charAt(j);
	System.out.print(ch);
	}
	System.out.println();
	i++;
	}
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.println("Output : ");
	pattern(s);
	}

Recursive Method 2	Corresponding Iterative Method

void pattern(String s, int i)
{
if(i<s.length())
{
System.out.print(s.substring(0,i+1));
System.out.println();
pattern(s,i+1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
System.out.println("Output : ");
pattern(s,0);
}
	void pattern(String s)
	{
	int i=0;
	while(i<s.length())
	{
	System.out.print(s.substring(0,i+1));
	System.out.println();
	i++;
	}
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.println("Output : ");
	pattern(s);
	}

8. Printing Pattern

If String is "JAVAFORSCHOOL" then:

JAVAFORSCHOOL
JAVAFORSCHOO
JAVAFORSCHO
JAVAFORSCH
JAVAFORSC
JAVAFORS
JAVAFOR
JAVAFO
JAVAF
JAVA
JAV
JA
J

Recursive Method 1	Corresponding Iterative Method

void pattern(String s, int i)
{
if(i>=0)
{
for(int j=0;j<=i;j++)
{
char ch=s.charAt(j);
System.out.print(ch);
}
System.out.println();
pattern(s,i-1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
int len=s.length();
System.out.println("Output : ");
pattern(s,len-1);
}
	void pattern(String s)
	{
	int i=s.length()-1;
	while(i>=0)
	{
	for(int j=0;j<=i;j++)
	{
	char ch=s.charAt(j);
	System.out.print(ch);
	}
	System.out.println();
	i=i-1;
	}
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.println("Output : ");
	pattern(s);
	}

Recursive Method 2	Corresponding Iterative Method

void pattern(String s, int i)
{
if(i>=0)
{
System.out.print(s.substring(0,i+1));
System.out.println();
pattern(s,i-1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
int len=s.length();
System.out.println("Output : ");
pattern(s,len-1);
}
	void pattern(String s)
	{
	int i=s.length()-1;
	while(i>=0)
	{
	System.out.print(s.substring(0,i+1));
	System.out.println();
	i=i-1;
	}
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	System.out.println("Output : ");
	pattern(s);
	}

9. Printing the digits of a number in words

Recursive Method	Corresponding Iterative Method

void digit(String n, int i)
{
if(i<n.length())
{
char ch=n.charAt(i);
printWord(ch);
digit(n,i+1);
} }
 	
void printWord(char ch)
{
switch(ch)
{
case '0': System.out.print("Zero "); break;
case '1': System.out.print("One "); break;
case '2': System.out.print("Two "); break;
case '3': System.out.print("Three "); break;
case '4': System.out.print("Four "); break;
case '5': System.out.print("Five "); break;
case '6': System.out.print("Six "); break;
case '7': System.out.print("Seven "); break;
case '8': System.out.print("Eight "); break;
case '9': System.out.print("Nine "); break;
}
}
void display()throws IOException
{
System.out.print("Enter any number : ");
String n=br.readLine();
System.out.print("Output : ");
digit(n,0);
}
	void digit(String n)
	{
	int i=0;
	while(i<n.length())
	{
	char ch=n.charAt(i);
	printWord(ch);
	i++;
	} }
	void printWord(char ch)
	{
	switch(ch)
	{
	case '0': System.out.print("Zero "); break;
	case '1': System.out.print("One "); break;
	case '2': System.out.print("Two "); break;
	case '3': System.out.print("Three "); break;
	case '4': System.out.print("Four "); break;
	case '5': System.out.print("Five "); break;
	case '6': System.out.print("Six "); break;
	case '7': System.out.print("Seven "); break;
	case '8': System.out.print("Eight "); break;
	case '9': System.out.print("Nine "); break;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any number : ");
	String n=br.readLine();
	System.out.print("Output : ");
	digit(n);
	}

In the above program, we have asked the user to enter a number, but have stored it as a String and not an Integer.

10. Arranging a String in Alphabetical order

Recursive Method	Corresponding Iterative Method

String newstr="";
void alpha(String s,int k)
{
if(k<=90)
{
for(int i=0;i<s.length();i++)
{
char ch=s.charAt(i);
if(ch==(char)k||ch==(char)(k+32))
{
newstr=newstr+ch;
}
}
alpha(s,k+1);
}
else
System.out.print("Output : "+newstr);
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
digit(s,65);
}
	String newstr="";
	void alpha(String s)
	{
	int k=65;
	while(k<=90)
	{
	for(int i=0;i<s.length();i++)
	{
	char ch=s.charAt(i);
	if(ch==(char)k||ch==(char)(k+32))
	{
	newstr=newstr+ch;
	}
	}
	k++;
	}
	System.out.print("Output : "+newstr);
	}
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	digit(s);
	}

In the above recursive method **alpha()**, we are passing another variable 'k' which is for comparing the ASCII values of each characters. The ASCII value of 'A' is 65, hence the starting value of variable 'k' is sent as 65 and the value of variable 'k' goes on till 90 which is the ASCII value of 'Z'.

Inside the recursive method **alpha()** we have written a for-loop which will run from the starting of the String till the end for every English alphabets i.e. this for loop will execute 26 times.

11. Encoding every character by 2 positions

Recursive Method	Corresponding Iterative Method

void encode(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if((ch>='a'&&ch<='x') || (ch>='A'&&ch<='X'))
ch=(char)(ch+2);
if(ch=='y'||ch=='Y'||ch=='z'||ch=='Z')
ch=(char)(ch-24);
System.out.print(ch);
encode(s,i+1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any word : ");
String s=br.readLine();
System.out.print("The Encoded word = ");
encode(s,0);
}
	void encode(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if((ch>='a'&&ch<='x') || (ch>='A'&&ch<='X'))
	ch=(char)(ch+2);
	if(ch=='y'||ch=='Y'||ch=='z'||ch=='Z')
	ch=(char)(ch-24);
	System.out.print(ch);
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any word : ");
	String s=br.readLine();
	System.out.print("The Encoded word = ");
	encode(s,0);
	}

12. Finding the position of the First vowel

Recursive Method	Iterative Method

int vowFirst(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
{
return i;
}
return (vowFirst(s,i+1));
}
return -1;
}
 	
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
int pos=vowFirst(s,0);
if(pos!=-1)
System.out.print("Position of first Vowel = "+pos);
else
System.out.println("No Vowel Found ");
}
	int vowFirst(String s)
	{
	int i=0,p=-1;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
	ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
	{
	p=i;
	break;
	}
	i++;
	}
	return p;
	}
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	int pos=vowFirst(s);
	if(pos!=-1)
	System.out.print("Position of first Vowel = "+pos);
	else
	System.out.println("No Vowel Found ");
	}

In the above recursive method **vowFirst()**, we are returning a (-1) when we are not getting any vowel. If we get a vowel then we are returning that position.

Writing a return statement inside the **if-block** when we get a vowel, allows us to stop the execution of the function if we get a vowel. **return** keyword allows us to stop the execution of a function and transfers the control back to the calling function. In the iterative method, we are using the keyword **break** to stop the while loop when we get the first vowel. In the display() method, we are printing the position of the first vowel, if the return value is not -1, else we are printing an error message.

13. Finding the position of the Last vowel

Recursive Method	Iterative Method

int p=-1;
int vowLast(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
p=i;
return (vowLast(s,i+1));
}
return p;
}
void display()throws IOException
{
System.out.print("Enter any String : ");
String s=br.readLine();
int pos=vowLast(s,0);
if(pos!=-1)
System.out.print("Position of last Vowel = "+pos);
else
System.out.println("No Vowel Found ");
}
	int vowFirst(String s)
	{
	int i=0,p=-1;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
	ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
	p=i;
	i++;
	}
	return p;
	}
	void display()throws IOException
	{
	System.out.print("Enter any String : ");
	String s=br.readLine();
	int pos=vowFirst(s);
	if(pos!=-1)
	System.out.print("Position of last Vowel = "+pos);
	else
	System.out.println("No Vowel Found ");
	}

Removing the return keyword from within the if-block allows the recursive function to find the position of the last vowel. Similarly, Removing the break keyword from within the if-block allows the Iterative function to find the position of the last vowel.

14. Printing and Counting the Double letter sequence present in a sentence

Recursive Method	Corresponding Iterative Method

int count=0;
void doubleLetter(String s, int i)
{
if(i<s.length()-1)
{
char ch1=s.charAt(i);
char ch2=s.charAt(i+1);
if(ch1==ch2)
{
System.out.println(ch1+","+ch2);
count++;
}
doubleLetter(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s.toUpperCase();
System.out.println("The Double letter pairs are : ");
doubleLetter(s,0);
System.out.println("No. of pairs = "+count);
}
	int count=0,i=0;
	void doubleLetter(String s)
	{
	while(i<s.length()-1)
	{
	char ch1=s.charAt(i);
	char ch2=s.charAt(i+1);
	if(ch1==ch2)
	{
	System.out.println(ch1+","+ch2);
	count++;
	}
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s.toUpperCase();
	System.out.println("The Double letter pairs are : ");
	doubleLetter(s);
	System.out.println("No. of pairs = "+count);
	}

15. Printing and Counting the Consecutive Character sequence present in a sentence

Recursive Method	Corresponding Iterative Method

int count=0;
void consecLetter(String s, int i)
{
if(i<s.length()-1)
{
char ch1=s.charAt(i);
char ch2=s.charAt(i+1);
if((int)ch2-(int)ch1==1)
{
System.out.println(ch1+","+ch2);
count++;
}
consecLetter(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s.toUpperCase();
System.out.println("Consecutive letter pairs are : ");
consecLetter(s,0);
System.out.println("No. of pairs = "+count);
}
	int count=0,i=0;
	void consecLetter(String s, int i)
	{
	while(i<s.length()-1)
	{
	char ch1=s.charAt(i);
	char ch2=s.charAt(i+1);
	if((int)ch2-(int)ch1==1)
	{
	System.out.println(ch1+","+ch2);
	count++;
	}
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s.toUpperCase();
	System.out.println("Consecutive letter pairs are : ");
	consecLetter(s);
	System.out.println("No. of pairs = "+count);
	}

If the characters are consecutive as in the English Alphabets, then the difference in the ASCII values of the 2nd character and the 1st character will be one. This is what we are checking in the recursive function.

16. Forming a new word by taking first character of every word in a sentence

Recursive Method	Corresponding Iterative Method

String newstr="";
String newWord(String s, int i)
{
if(i<s.length()-1)
{
char ch=s.charAt(i);
if(ch == ' ')
newstr=newstr+s.charAt(i+1);
newWord(s,i+1);
}
return newstr;
}
 	
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=" "+s;
String x= newWord(s,0);
System.out.println("The New Word : "+x);
}
	String newstr="";
	String newWord(String s)
	{
	int i=0;
	while(i<s.length()-1)
	{
	char ch=s.charAt(i);
	if(ch == ' ')
	newstr=newstr+s.charAt(i+1);
	i++;
	}
	return newstr;
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=" "+s;
	String x= newWord(s);
	System.out.println("The New Word : "+x);
	}

As soon as we are encountering a space, we are adding the character just next to it to the variable 'newstr'.

Note: Before sending the String to the recursive function, we have added a space before the String.

17. Finding the Piglatin of a word

In this program you first need to find the position of the first vowel. You can do this by either the recursive method or the iterative method we discussed of finding the first vowel or you can even find the position of the first vowel inside the display() method.

In this example we have used the recursive function for finding the first vowel.

```
String piggy="";
int vowFirst(String s, int i)
 {
  if(i<s.length())
   {
     char ch=s.charAt(i);
     if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
       ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
       {
          return i;
       }
     return (vowFirst(s,i+1));
   }
  return 0;
 }

void piglatin(String s, int i, int q)
 {
  if(i<q)
   {
    char ch=s.charAt(i);
    piggy=piggy+ch;
    piglatin(s,i+1,q);
   }
 }

void display()throws IOException
  {
System.out.print("Enter any word : ");
String s=br.readLine();
int len=s.length();
int p=vowFirst(s,0);
piglatin(s,p,len);
piglatin(s,0,p);
piggy=piggy+"ay";
System.out.println("The Piglatin of the word =
"+piggy);
  }
```

18. Finding the Frequency of each characters present in a word

```
import java.io.*;
class Recursion
 {
static BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
int cap[]=new int[26];
int sm[]=new int[26];
Recursion()
{
 for(int i=0;i<26;i++)
  {
   cap[i]=0;
   sm[i]=0;
  }
}
void freqChar(String s, int i)
 {
  if(i<s.length())
   {
    char ch=s.charAt(i);
    if(ch>='A'&&ch<='Z')
       cap[ch-65]++;
    if(ch>='a'&&ch<='z')
       sm[ch-97]++;
    freqChar(s,i+1);
    }
}
void display()throws IOException
{
System.out.print("Enter any string : ");
String s=br.readLine();
System.out.println("Character\t\tFrequency");
freqChar(s,0);
for(int i=0;i<26;i++)
   {
    if(cap[i]!=0)
    System.out.println((char)(i+65)+"\t\t\t"+cap[i]);
   }
for(int i=0;i<26;i++)
   {
    if(sm[i]!=0)
    System.out.println((char)(i+97)+"\t\t\t"+sm[i]);
   }
}

public static void main()throws IOException
  {
   Recursion ob=new Recursion();
   ob.display();
  }
}
```

In the above example, the array **cap[]** is storing the frequency of the capital letters present in the string, and the array **sm[]** is storing the frequency of the small letters present in the string. In the **display()** method, we are first displaying the frequencies of the capital letters and then the small letters.

Extracting words of a String and performing any given operation

Recursive Method	Iterative Method

String w="";
void word(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!=' ')
{
w=w+ch;
}
else
{
Write the operation you want to perform with the word stored in 'w' over here.
w="";
}
word(s,i+1);
}
}
	String w="";
	void word(String)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!=' ')
	{
	w=w+ch;
	}
	else
	{
	Write the operation you want to perform with the word stored in 'w' over here.
	w="";
	}
	i++;
	} }

In the code given on the left , we are extracting the characters from the beginning of the String passed as parameter and checking whether that character stored in 'ch' is a space or not. If the character is not a space, then we add it to the String variable 'w'. When the character is a space, then we have got a word, and hence you write what you want to do with the word stored in 'w' in the else-block.

Important Note: While using the above code, after entering the String from the user, you need to add a space at its end before passing it on to the recursive method.

Some Programs using the above technique

1. Printing each word of a sentence along with their lengths

Recursive Method	Corresponding Iterative Method

String w="";
int len=0;
void word(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!=' ')
w=w+ch;
else
{
len=w.length();
System.out.println(w+"\t\t"+len);
w="";
}
word(s,i+1);
}
}
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.println("Words\t\tLength");
word(s,0);
}
	String w="";
	int len=0,i=0;
	void word(String s)
	{
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!=' ')
	w=w+ch;
	else
	{
	len=w.length();
	System.out.println(w+"\t\t"+len);
	w="";
	}
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s+" ";
	System.out.println("Words\t\tLength");
	word(s);
	}

Note: We have added a space after the inputted sentence in the display () method.

2. Printing the reverse of each word in a sentence

Recursive Method	Corresponding Iterative Method

String w="";
void word(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!=' ')
w=ch+w;
else
{
System.out.println(w);
w="";
}
word(s,i+1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
word(s,0);
}
	String w="";
	void word(String s)
	{
	int i=0;
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!=' ')
	w=ch+w;
	else
	{
	System.out.println(w);
	w="";
	}
	i++;
	}
	}
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s+" ";
	word(s);
	}

The change in the line of adding the character i.e. **w=ch+w;** reverses the word by adding the character before rather than after.

3. Printing the words which begin with a vowel

Recursive Method	Corresponding Iterative Method

String w="";
void word(String s, int i)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!=' ')
w=w+ch;
else
{
char c=w.charAt(0);
if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||c=='A'||
c=='E'||c=='I'||c=='O'||c=='U')
{
System.out.println(w);
}
w="";
}
word(s,i+1);
}
}
 	
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Words beginning with vowel are :
");
word(s,0);
}
	String w=""; int i=0;
	void word(String s)
	{
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!=' ')
	w=w+ch;
	else
	{
	char c=w.charAt(0);
	if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||c=='A'||
	c=='E'||c=='I'||c=='O'||c=='U')
	{
	System.out.println(w);
	}
	w="";
	}
	i++;
	}
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s+" ";
	System.out.print("Words beginning with vowel are :
	");
	word(s);
	}

4. Searching for a word in a sentence

Recursive Method

```
String w="";
int word(String s, int i, String search)
{
    if(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          if(w.equalsIgnoreCase(search))
               return 1;
          w="";
         }
      return (word(s,i+1,search));
      }
   return 0;
}
void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Enter the word to search : ");
String search=br.readLine();
int res=word(s,0,search);
if(res==1)
System.out.println("Word Found");
else
System.out.print("Word is not present");
  }
```

Corresponding Iterative Method

```
String w=""; int i=0, flag=0;
int word(String s, String search)
{
    while(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          if(w.equalsIgnoreCase(search))
               flag=1;
          w="";
         }
      i++;
      }
   return flag;
}
void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Enter the word to search : ");
String search=br.readLine();
int res=word(s, search);
if(res==1)
System.out.println("Word Found");
else
System.out.print("Word is not present");
  }
```

5. Removing a particular word from a sentence

Recursive Method

```
String w="",newstr="";
void word(String s, int i, String rem)
{
    if(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          if(w.equalsIgnoreCase(rem)==false)
               newstr=newstr+w+" ";
          w="";
         }
      word(s,i+1,rem);
      }
   else
     System.out.println("Output : "+newstr);
}
void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Enter the word to remove : ");
String rem=br.readLine();
word(s,0,rem);
  }
```

Corresponding Iterative Method

```
String w="",newstr=""; int i=0;
void word(String s, String rem)
{
    while(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          if(w.equalsIgnoreCase(rem)==false)
               newstr=newstr+w+" ";
          w="";
         }
      i++;
      }
     System.out.println("Output : "+newstr);
}

void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Enter the word to remove : ");
String rem=br.readLine();
word(s,rem);
  }
```

Note: In both of the codes above for searching for a word or removing a word, we need to send the word we want to search or remove also into the recursive function as a parameter.

In the code for removing a word, the recursive function is extracting one word at a time and adding it to the new string whenever the word is not equal to the word we want to remove.

6. Replacing a particular word with another word

Recursive Method	Corresponding Iterative Method

import java.io.*;
class Recursion
{
static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
 	
String w="",newstr="";
void word(String s, int i, String rep, String n)
{
if(i<s.length())
{
char ch=s.charAt(i);
if(ch!=' ')
w=w+ch;
else
{
if(w.equalsIgnoreCase(rep))
{
w=n;
}
newstr=newstr+w+" ";
w="";
}
word(s,i+1,rep,n);
}
else
System.out.println("Output : "+newstr);
}
 	
void display()throws IOException
{
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Enter the word to replace : ");
String rep=br.readLine();
System.out.print("Enter the new word : ");
String n=br.readLine();
word(s,0,rep,n);
}
 	
public static void main()throws IOException
{
Recursion ob=new Recursion();
ob.display();
}
}
	import java.io.*;
	class Recursion
	{
	static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
	
	String w="",newstr=""; int i=0;
	void word(String s, String rep, String n)
	{
	while(i<s.length())
	{
	char ch=s.charAt(i);
	if(ch!=' ')
	w=w+ch;
	else
	{
	if(w.equalsIgnoreCase(rep))
	{
	w=n;
	}
	newstr=newstr+w+" ";
	w="";
	}
	i++;
	}
	System.out.println("Output : "+newstr);
	}
	
	void display()throws IOException
	{
	System.out.print("Enter any sentence : ");
	String s=br.readLine();
	s=s+" ";
	System.out.print("Enter the word to replace : ");
	String rep=br.readLine();
	System.out.print("Enter the new word : ");
	String n=br.readLine();
	word(s,rep,n);
	}
	
	public static void main()throws IOException
	{
	Recursion ob=new Recursion();
	ob.display();
	}
	}

Note: We need to send the word we want to replace and also the new word into the recursive function as a parameter.

The recursive function is extracting one word at a time and checking whether that word is the same as the word we want to replace or not. If yes, then the word is replaced with the new word and added to the new word.

7. Printing the Piglatin of each word of a sentence

Recursive Method

```
import java.io.*;
class Recursion
   {
static BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
String w="";
int len=0;
void word(String s, int i)
  {
    if(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          piglatin(w);
          w="";
          }
       word(s,i+1);
      }
}

void piglatin(String s)
{
int p=0;
for(int k=0;k<s.length();k++)
 {
  char ch=s.charAt(k);
 if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
   ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
   {
     p=k;
     break;
   }
   }
String a=s.substring(p);
String b=s.substring(0,p);
String piggy=a+b+"ay";
System.out.print(piggy+" ");
}
void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Output: ");
word(s,0);
  }
public static void main()throws IOException
{
Recursion ob=new Recursion();
ob.display();
}
   }
```

Corresponding Iterative Method

```
import java.io.*;
class Recursion
   {
static BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
String w="";
int len=0, i=0;
void word(String s)
  {
    while(i<s.length())
     {
       char ch=s.charAt(i);
       if(ch!=' ')
          w=w+ch;
       else
         {
          piglatin(w);
          w="";
          }
       i++;
      }
}

void piglatin(String s)
{
int p=0;
for(int k=0;k<s.length();k++)
 {
  char ch=s.charAt(k);
 if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||
   ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
   {
     p=k;
     break;
   }
   }
String a=s.substring(p);
String b=s.substring(0,p);
String piggy=a+b+"ay";
System.out.print(piggy+" ");
}
void display()throws IOException
  {
System.out.print("Enter any sentence : ");
String s=br.readLine();
s=s+" ";
System.out.print("Output: ");
word(s);
  }
public static void main()throws IOException
{
Recursion ob=new Recursion();
ob.display();
}
   }
```

The recursive function **word()** is extracting one word at a time and sending it to the function **piglatin()**, which is first finding the position of the first vowel of each word and then doing the needful for converting the word into a piglatin.

Note: Here the function **piglatin()** is **non-recursive** i.e. iterative. Also, we have added a space after the inputted sentence in the **display()** method.