Reading characters

The **read() method is used with BufferedReader object to read characters**. As this function returns integer type value has we need to use typecasting to convert it into char type.

Syntax:

int read() throws IOException

Example:

Read character from keyboard

```
import java.io.*;
class Main
{
 public static void main( String args[]) throws IOException
 {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  char c;
  System.out.println("Enter characters, @ to quit");
  do{
    c = (char)br.read();     //Reading character
     System.out.println(c);
  }while(c!='@');
 }
}
```

Sample Output:

Enter characters, @ to quit

abcd23@

a b

c d

2

3

@

Example:

Read string from keyboard

The readLine() function with BufferedReader class's object is used to read string from keyboard.

Syntax:

String readLine() throws IOException

Example :

```
import java.io.*;
public class Main{
public static void main(String args[])throws Exception{
   InputStreamReader r=new InputStreamReader(System.in);
   BufferedReader br=new BufferedReader(r);
   System.out.println("Enter your name");
   String name=br.readLine();
   System.out.println("Welcome "+name);
}
}
```

Sample Output : Enter

your name Priya

Welcome Priya

WRITING CONSOLE OUTPUT

- Console output is most easily accomplished with print( ) and println( ). These methods are defined by the class PrintStream (which is the type of object referenced by System. out).
- Since PrintStream is an output stream derived from OutputStream, it also implements the low-level method write( ).
- So, write( ) can be used to write to the console.

Syntax:

void write(int byteval)

This method writes to the stream the byte specified by byteval.

The following java program uses write( ) to output the character "A" followed by a new-line to the screen:

```
// Demonstrate System.out.write().
class WriteDemo
 {
public static void main(String args[])
{
 int b;
b = 'A';
System.out.write(b);
System.out.write('\n');
```

```
}
}
```

THE PRINT WRITER CLASS

- Although using System.outto write to the console is acceptable, its use is recommended mostly for debugging purposes or for sample programs.
- For real-world programs, the recommended method of writing to the console when using Java is through a PrintWriter stream.
- PrintWriter is one of the character-based classes.
- Using a character-based class for console output makes it easier to internationalize our program.
- PrintWriter defines several constructors.

Syntax:

PrintWriter(OutputStream outputStream, boolean flushOnNewline) Here,

- output Stream is an object of type OutputStream
- flushOnNewline controls whether Java flushes the output stream every time a println( ) method is called.
- If flushOnNewline is true, flushing automatically takes place. If false, flushing isnot automatic.
- PrintWriter supports the print( ) and println( ) methods for all types including Object.
- Thus, we can use these methods in the same way as they have been used with System. out.
- If an argument is not a simple type, the PrintWriter methods call the object's toString( ) method and then print the result.
- To write to the console by using a PrintWriter, specify System.out for the output stream and flush the stream after each newline.

For example, the following code creates a PrintWriter that is connected to console output:

```
PrintWriter pw = new PrintWriter(System.out, true);
```

The following application illustrates using a PrintWriter to handle console output:

```
// Demonstrate PrintWriter import
java.io.*;
public class PrintWriterDemo
{
 public static void main(String args[])
{
PrintWriter pw = new PrintWriter(System.out, true);
pw.println("This is a string");
 int i = -7;
pw.println(i);
```

double

```
d = 4.5e-7;
pw.println(d);
 }
 }
```

Sample Output:

```
This is a string
-7
 4.5E-7
```