COCSIT

IO OPERATIONS

I/O Operations

COCSIT

Agenda

1 Console Operations

2 File Operations

Objectives

At the end of this module, you will be able to:

- Understand Console operations
- Understand File operations

COCSIT

Reading & Printing to Console

Reading Console Input - Stream Wrapping

- The preferred method of reading console input in Java 2 is to use a character stream
- *InputStreamReader* class acts as a bridge between byte and character streams
- Console input is accomplished by reading from System.in
- To get a character-based stream, you wrap ***System.in*** in a BufferedReader object

Reading Console Input - Stream Wrapping

- The **BufferedReader** class supports a buffered input stream. Its most commonly used constructor is shown as follows:

- **BufferedReader(Reader *inputReader*)**

- Here *inputReader* is the stream that is linked to the instance of **BufferedReader** that is being created. **Reader** is an abstract class. One of its concrete subclasses is **InputStreamReader**, which converts bytes to characters. To obtain an **InputStreamReader** object that is linked to **System.in**, use the following constructor:

- **InputStreamReader(InputStream *inputStream*)**

Reading Console Input - Stream Wrapping

Because **System.in** refers to an object of type **InputStream**, it can be used for *inputStream*. Putting it all together, the following line of code creates a **BufferedReader** that is connected to the keyboard, and which in turn enables character input from a byte stream InputStream that is System.in).

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

Reading Characters

```
package m10.io;
import java.io.*;

public class BRRead{

  public static void main (String args[ ]) throws IOException {
        char c;
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        System.out.println("Enter Characters, 'q' to quit");
        do {
                  c = (char) br.read( );
                System.out.println( c );
        }while (c != 'q');
    }
}
```

Refer documentation for BufferedReader and InputStreamReader

COCSIT

Reading Characters

- **int read( ) throws IOException**

- Whenever the **read( ) method** is called, it reads a character from the input stream and returns an integer value. If the end of the stream is encountered, -1 is returned.

Reading Strings

```
package m10.io;
import java.io.*;

public class BRReadLine{

  public static void main (String args[]) throws IOException {
        String str;
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        System.out.println("Enter Characters, 'stop' to quit");
                do {
                    str = br.readLine( );
                  System.out.println ( str );
        }while (!str.equals( "stop"));
  }
}
```

*The above program reads and displays lines of text until you enter the word "**stop**".*

Writing Console Output

- **print( )** and **println( )** are console output methods defined in PrintStream class
- **System.out** is a byte stream used to write bytes

COCSIT

Writing & Reading From File

Reading & Writing to File using FileReader & FileWriter

The **File** class is a convenience class for writing character files. The **File** class deals directly with files and the file system. The **File** class does not specify how information is retrieved from, or stored in files, it describes the properties of a file itself. A **File** object is used to obtain or manipulate information associated with a disk file, such as the permissions, time, date and directory path.

```
public int read() throws IOException (Read a single character)
```

```
public int read(char[] cbuf,int off,int len) throws IOException
```

```
public void write(int c) throws IOException (Write a single character)
```

Reading & Writing to File using FileReader & FileWriter

```
package m10.io;
import java.io.*;


public class Copy {

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

            File inputFile = new File("Source.txt");
            File outputFile = new File("Target.txt");
            FileReader in = new FileReader(inputFile);
            FileWriter out = new FileWriter(outputFile);
            int c;
            while ((c = in.read()) != -1)
               out.write(c);

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

Refer documentation for FileReader and FileWriter

COCSIT

Copy image

```
import java.io.*;

class CopyFile{

      public static void main(String args[]) throws IOException{
      int i;
      FileInputStream fin;
      FileOutputStream fout;

      try{
        fin = new FileInputStream(args[0]);
      }
      catch(FileNotFoundException e){
        System.out.println("File Not Found");
        return;
      }
```

Why can't we use FileReader and FileWriter here?

Copy image (Contd.).

```
      try{
          fout = new FileOutputStream(args[1]);
      }
      catch(IOException e){
             System.out.println("Error Opening Output File");
             return;
      }
      try{
          do {
              i=fin.read();
              if(i!=-1)
              fout.write(i);
          } while (i!=-1);
      }
      catch (IOException e){
          System.out.println("File Error");
      }
      fin.close();
      fout.close();
   }
}
```

COCSIT

Copy image (Contd.).

```
        try{
            fout = new FileOutputStream(args[1]);
        }
        catch(IOException e){
              System.out.println("Error Opening Output File");
              return;
        }
        try{
            do {
                i=fin.read();
                if(i!=-1)
                fout.write(i);
            } while (i!=-1);
        }
        catch (IOException e){
            System.out.println("File Error");
        }
        fin.close();
        fout.close();
    }
}
```

Copy image (Contd.).

- To run this program
- C:\java CopyFile source.bmp dest.bmp
- It will copy image from source.bmp to dest.bmp

COCSIT

Thank You

Sensitivity: Internal & Restricted