Web Technology 13
Java I/O

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

I/O Basics

Stream

Wrapper Class

Reading Console Input

Reading with java.util.Scanner class

Writing Console Output

PrintWriter Class

Reading & Writing Files

Character-based Reading & Writing

I/O Basics

I/O Basics

- Java I/O is used to process the input and produce the output based on the input
- Java uses the concept of stream to make I/O operations
- *java.io* package contains all the classes required for input and output operations

- Two important methods are *read() and write()*

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Stream

Stream

- A stream is an abstraction that either produces or consumes information. It is linked to a physical device by the Java IO system
- *A stream is a sequence of data. It is composed of bytes*
- java.lang package defines a class called **System**, which encapsulates several aspects of the run-time environment
- **System.out**:
  - It refers to the standard output stream
- **System.in**:
  - It refers to the standard input stream
- **System.err**:
  - It refers to the standard error stream

System.in is an object of type InputStream. System.out and System.err are objects of type PrintStream

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Stream...

Byte Stream Classes

- These are defined by using two class hierarchies at the top: **InputStream** and **OutputStream**
- To use stream classes, you must import java.io

- **OutputStream**:
  - It is an abstract class. It is the superclass of all classes representing an output stream of bytes
  - It is used to write data to a destination

- **InputStream**:
  - It is an abstract class. It is the superclass of all classes representing an input stream of bytes
  - It is used to read data from a source

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Stream...

Character Stream Classes

- These are defined by using two class hierarchies at the top: **Reader** and **Writer**
- These classes handle Unicode character streams

- **Reader**:
  - It is the abstract class that describes character stream input
  - Reader class contains methods that are identical to those available in InputStream class, except Reader is designed to handle characters
- **Writer**:
  - It is the abstract class that describes character stream output
  - It provides support for all output operations by defining methods that are identical to those in OutputStream class

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

I/O Basics

Stream

Wrapper Class

Reading Console Input

Reading with java.util.Scanner class

Writing Console Output

PrintWriter Class

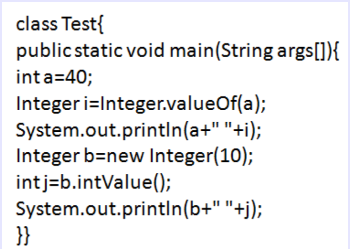
Reading & Writing Files

Character-based Reading & Writing

Wrapper Class

Wrapper Class

- It is a class whose object wraps or contains a primitive data types. We can wrap a primitive value into a wrapper class object
- Wrapper classes convert primitive data types into objects
- Data structures in the Collection framework store only objects and not primitive types
- An object is needed to support synchronization in multi-threading
- Boolean, Character, Byte, Short, Integer, Long, Float, Double



```
class Test{
public static void main(String args[]){
int a=40;
Integer i=Integer.valueOf(a);
System.out.println(a+" "+i);
Integer b=new Integer(10);
int j=b.intValue();
System.out.println(b+" "+j);
}}
```

Reading Console Input

Reading Console Input

- Console input is accomplished by reading from *System.in*. To obtain a character based stream that is attached to console, wrap System.in in a BufferedReader object
- **BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

Reading Characters

- **int read() throws IOException**
- *char ch=(char)br.read();*

Reading Strings

- **String readLine() throws IOException**
- *String str=br.readLine();*

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Reading Console Input...

Reading Integers

- **int Integer.parseInt(br.readLine());**
- *int item=Integer.parseInt(br.readLine());*

Reading Other Types of Values

- **Float.parseFloat(br.readLine())**
- **Double.parseDouble(br.readLine())**
- **Byte.parseByte(br.readLine())**
- **Short.parseShort(br.readLine())**
- **Long.parseLong(br.readLine())**
- **Boolean.parseBoolean(br.readLine())**

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Reading with java.util.Scanner class

Reading with java.util.Scanner class

- **Scanner sc=new Scanner(System.in);**
- When the Scanner class receives input, it breaks it into several pieces, called *tokens*
- *String str=sc.next();*
  String str=sc.nextLine();
  char ch=sc.next.charAt(0);
  int item=sc.nextInt();
  float bal=sc.nextFloat();
  long a=sc.nextLong();
  long b=sc.nextDouble();

Writing Console Output

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Writing Console Output

- Console output is accomplished by *print() and println()*. These methods are defined by *PrintStream* class
- PrintStream is an output stream derived from OutputStream. It also implements *write()*
- *int b='P';*
  System.out.write(b);

PrintWriter Class

PrintWriter Class

- PrintWriter is one of the character based classes
- **PrintWriter(OutputStream outputStream, boolean flushOnNewline)**
- PrintWriter supports print() and println() methods for all types including Object
- To write to the console by using PrintWriter, specify System.out for the output stream and flush the stream after each newline
- *PrintWriter pw=new PrintWriter(System.out, true); pw.println("Bye");*

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Reading & Writing Files

Reading & Writing Files

- Java files are byte-oriented
- Java allows wrapping of a byte-oriented file stream within a character-based object
- Most used stream classes are: *FileInputStream* and *FileOutputStream*
- **FileInputStream(String fname) throws FileNotFoundException**

  FileOutputStream(String fname) throws FileNotFoundException
- When an output file is opened, any preexisting file by the same name is destroyed

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Reading & Writing Files...

Reading & Writing Files...

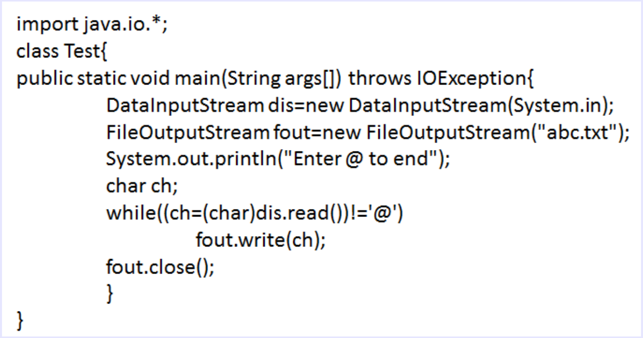
- After the work is over, the file should be closed by using *close()* method
  void close() throws IOException
- To read from a file, use
  int read() throws IOException
  Each time the read() is called, it reads a single byte from the file and returns the byte as an integer value
- To write to a file, use
  void write(int byteval) throws IOException
  Though byteval is declared as an integer, only the low-order 8 bits are written to the file

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Reading & Writing Files...

Writing Files

DataInputStream class is used to read data from keyboard
DataInputStream dis = new DataInputStream(System.in);



```
import java.io.*;
class Test{
public static void main(String args[]) throws IOException{
            DataInputStream dis=new DataInputStream(System.in);
            FileOutputStream fout=new FileOutputStream("abc.txt");
            System.out.println("Enter @ to end");
            char ch;
            while((ch=(char)dis.read())!='@')
                        fout.write(ch);
            fout.close();
            }
}
```

I/O Basics

Stream

Wrapper Class

Reading Console Input

Reading with java.util.Scanner class

Writing Console Output

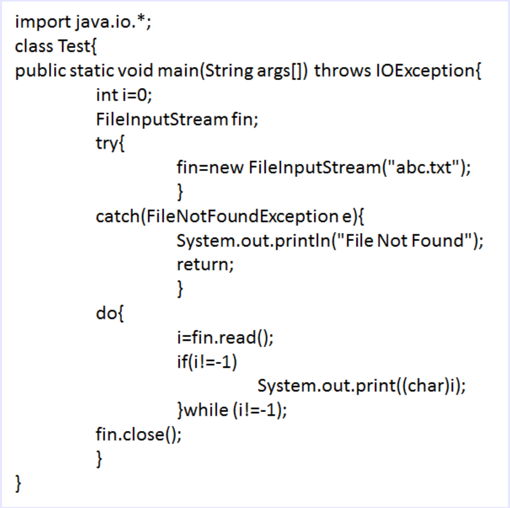
PrintWriter Class

Reading & Writing Files

Character-based Reading & Writing

Reading & Writing Files...

Reading Files



```
import java.io.*;
class Test{
public static void main(String args[]) throws IOException{
            int i=0;
            FileInputStream fin;
            try{
                        fin=new FileInputStream("abc.txt");
                        }
            catch(FileNotFoundException e){
                        System.out.println("File Not Found");
                        return;
                        }
            do{
                        i=fin.read();
                        if(i!=-1)
                                    System.out.print((char)i);
                        }while (i!=-1);
            fin.close();
            }
}
```

I/O Basics

Stream

Wrapper Class

Reading Console Input

Reading with java.util.Scanner class

Writing Console Output

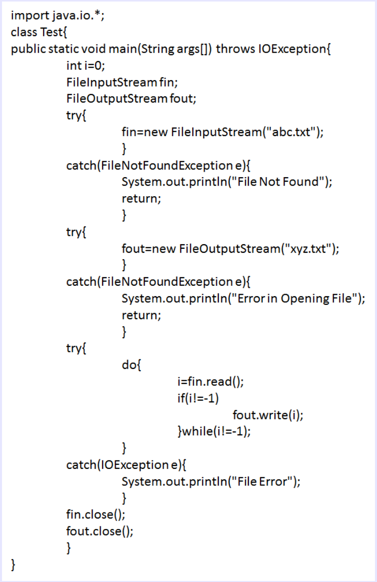
PrintWriter Class

Reading & Writing Files

Character-based Reading & Writing

Reading & Writing Files...

Copying Files

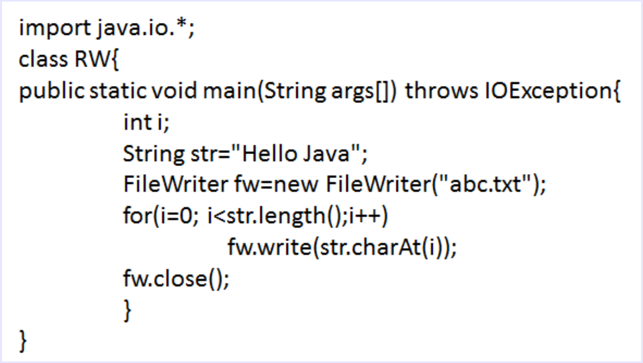


```
import java.io.*;
class Test{
public static void main(String args[]) throws IOException{
        int i=0;
        FileInputStream fin;
        FileOutputStream fout;
        try{
                fin=new FileInputStream("abc.txt");
                }
        catch(FileNotFoundException e){
                System.out.println("File Not Found");
                return;
                }
        try{
                fout=new FileOutputStream("xyz.txt");
                }
        catch(FileNotFoundException e){
                System.out.println("Error in Opening 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();
        }
}
```

Character-based Reading & Writing

Writing Files using FileWriter

FileWriter is useful to create file by writing characters into it



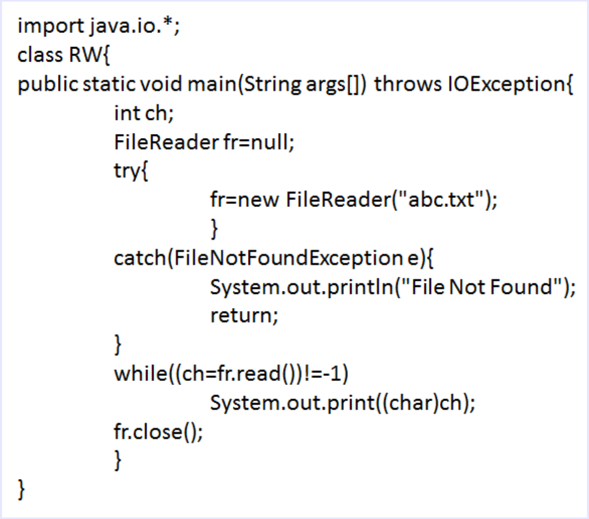
```
import java.io.*;
class RW{
public static void main(String args[]) throws IOException{
            int i;
            String str="Hello Java";
            FileWriter fw=new FileWriter("abc.txt");
            for(i=0; i<str.length();i++)
                        fw.write(str.charAt(i));
            fw.close();
            }
}
```

I/O Basics
Stream
Wrapper Class
Reading Console Input
Reading with java.util.Scanner class
Writing Console Output
PrintWriter Class
Reading & Writing Files
Character-based Reading & Writing

Character-based Reading & Writing...

Reading Files using FileReader

FileReader is useful to read data in the form of characters



```
import java.io.*;
class RW{
public static void main(String args[]) throws IOException{
          int ch;
          FileReader fr=null;
          try{
                    fr=new FileReader("abc.txt");
                    }
          catch(FileNotFoundException e){
                    System.out.println("File Not Found");
                    return;
          }
          while((ch=fr.read())!=-1)
                    System.out.print((char)ch);
          fr.close();
          }
}
```

I/O Basics

Stream

Wrapper Class

Reading Console Input

Reading with java.util.Scanner class

Writing Console Output

PrintWriter Class

Reading & Writing Files

Character-based Reading & Writing