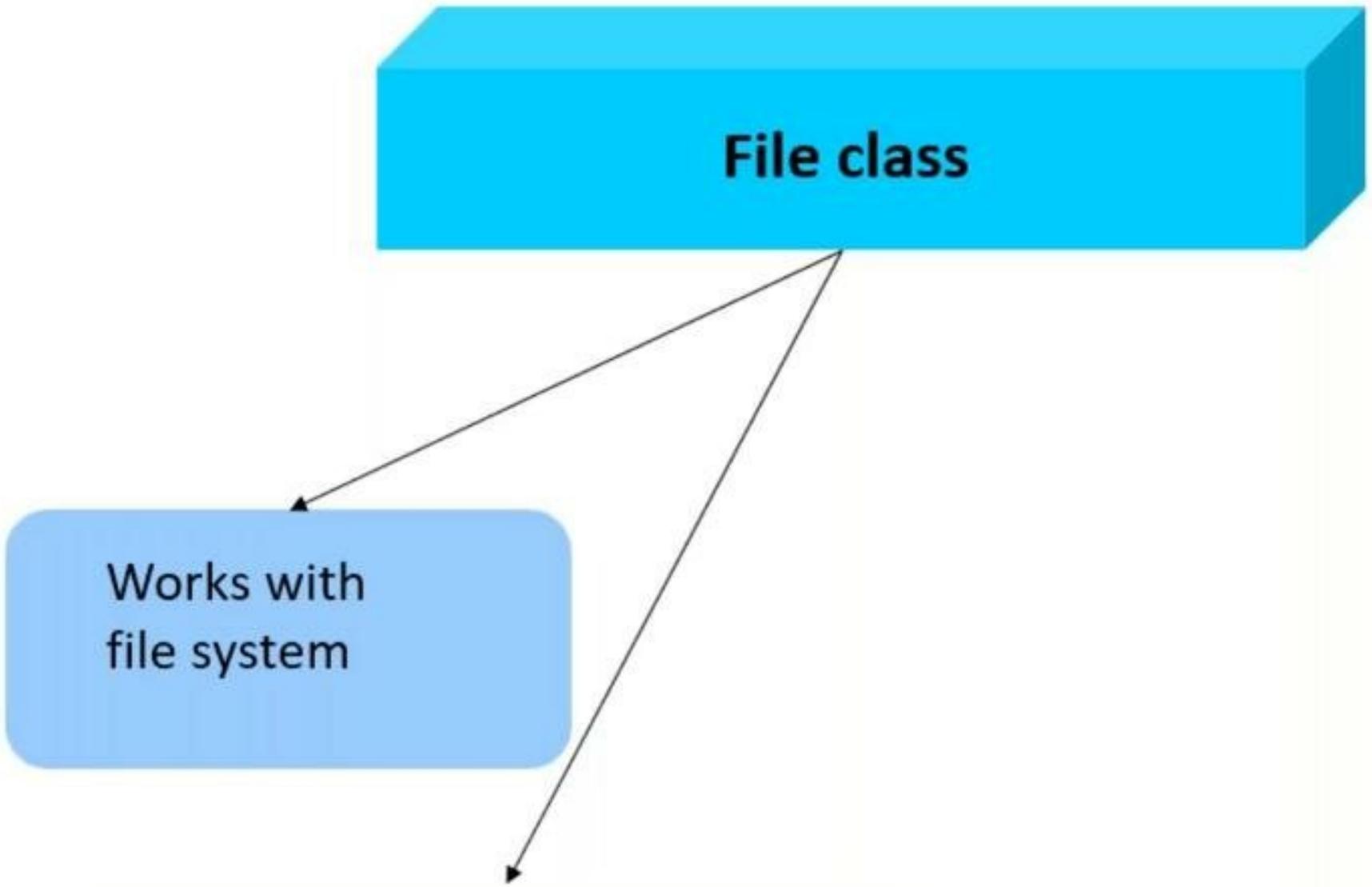
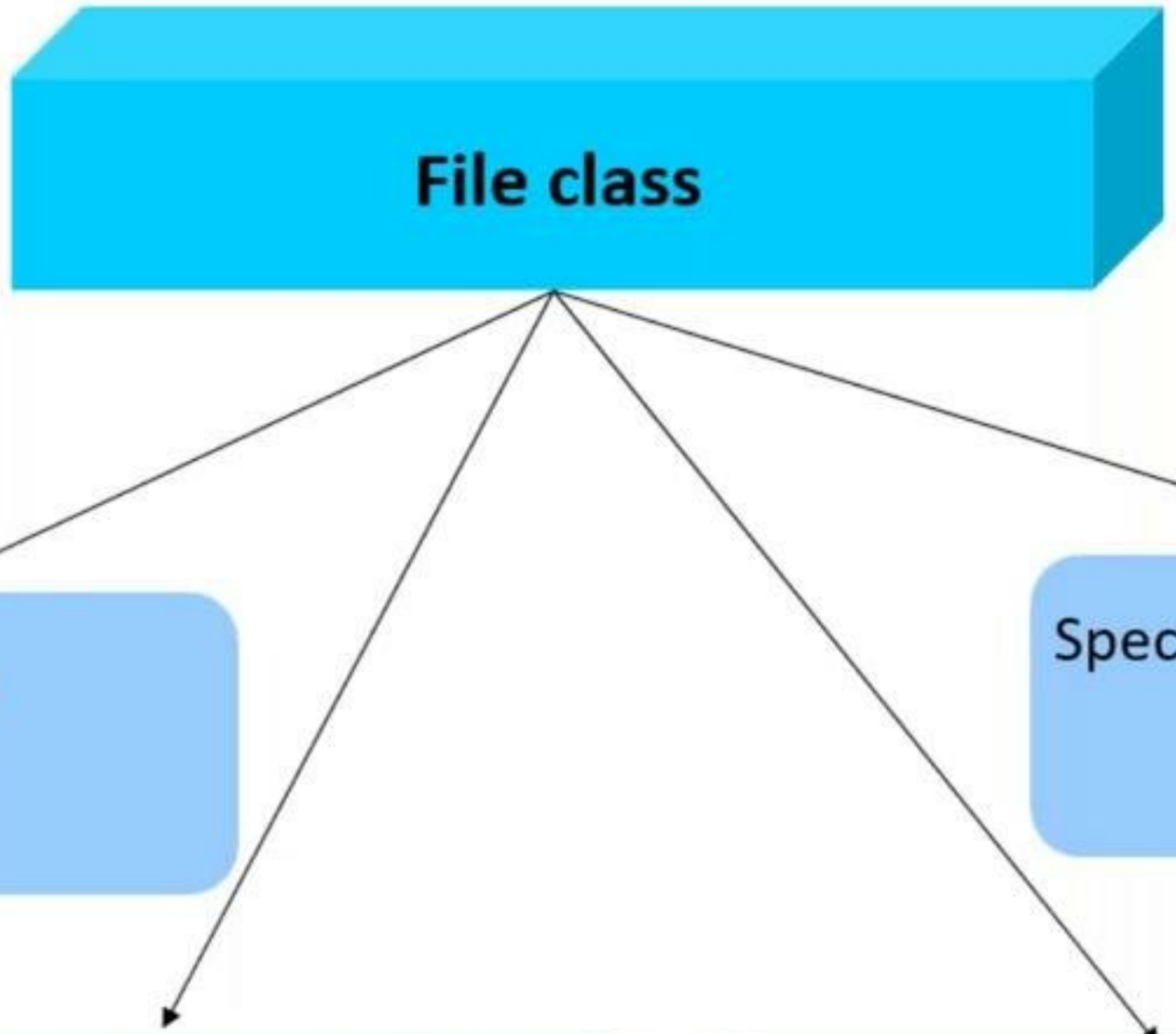
File I/O 3-1


File class
Works with
file system
Does not specify how to retrieve
or store information in the File

File I/O 3-1


File class
Works with file system
Specifies the properties of the File
Does not specify how to retrieve or store information in the File
Common File and directory operations are performed using the access methods

File I/O 3-2

File access methods help in creating, deleting and renaming files as well as checking the read/write access permission.

Directory access methods also help in creating, deleting, renaming and listing of directories.

File I/O 3-3

Constructor	Description
File(String dirpath)	Creates a File object with pathname of the file specified in *dirpath*
File(String dirpath, String filename)	Creates a File object with pathname of the file specified in *dirpath* and filename specified in *filename*
File(File fileObj, String filename)	Creates a File object with another File object specified in *fileObj* and filename specified in *filename*
File(URL urlObj)	Creates a File object by converting the given file:URL into an abstract pathname

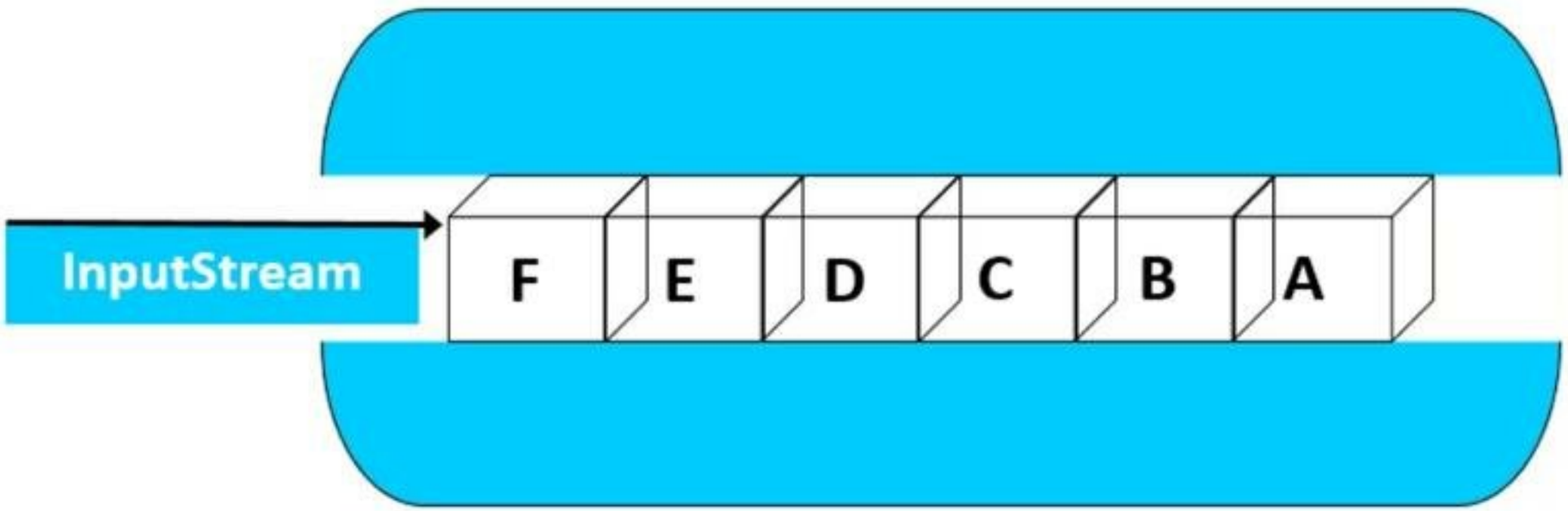
File I/O 3-3


```
void details() {
        File filename = new File("Text.txt");
        System.out.println("Name of the File: " + filename.getName());
        System.out.println("Path of the File: " + filename.getPath());
        System.out.println("File Last Modified: " + filename.lastModified());
        System.out.println("Parent Directory: " + filename.getParent());
        System.out.println("Size of the File: " + filename.length() + " Bytes");
    }
}
```

The path, name, when the file was last modified and its length in bytes is displayed

Streams

- A channel for sending information in first-in, first-out manner.



Stream of data flowing from a source

Streams

Character Stream

Byte Stream

Byte Streams

- Basic unit for this type of stream is a byte.
- Extends InputStream and OutputStream class.
- There are classes to convert byte streams into character streams.

Character Streams

- Basic unit for this type of stream is Unicode character.
- Preferred way of handling strings and text.
- Extend from Reader and Writer classes in the java.io package.
- It creates a BufferedReader and BufferedWriter by passing normal Reader / Writer subclass into the constructors.

InputStream

- It is an abstract base class.

InputStream

- It is an abstract base class.
- Defines methods for reading bytes or array of bytes, marking locations in the stream, skipping bytes of input, finding out number of bytes available for reading and many more.
- Opened automatically when created.

InputStream Hierarchy

- InputStream
  - FileInputStream
  - PipedInputStream
  - FilterInputStream
    - BufferInputStream
    - LinedNumberInputStream
    - PushbackInputStream
  - ByteArrayInputStream
  - StringInputStream

FileInputStream

- Reads input from a file in the form of a stream.
- All methods of InputStream class are overridden except `mark()` and `reset()`.

Constructor	Description
FileInputStream(String filename)	Creates an input stream that reads bytes from a file in the form of a stream. *filename* is the full pathname of the file
FileInputStream(File name)	Creates an input stream that can be used to read bytes from a file where *name* is a File object

FileInputStream


```
void show(final String file) throws IOException {
    int size;
    InputStream fileobject = new FileInputStream(file);
    System.out.println("Bytes available to read: "
    + (size = fileobject.available()));
    char[] text = new char[200];
    for (int count = 0; count < size; count++) {
        text[count] = ((char) fileobject.read());
        System.out.print(text[count]);
    }
    System.out.println("");
    fileobject.close();
}
```

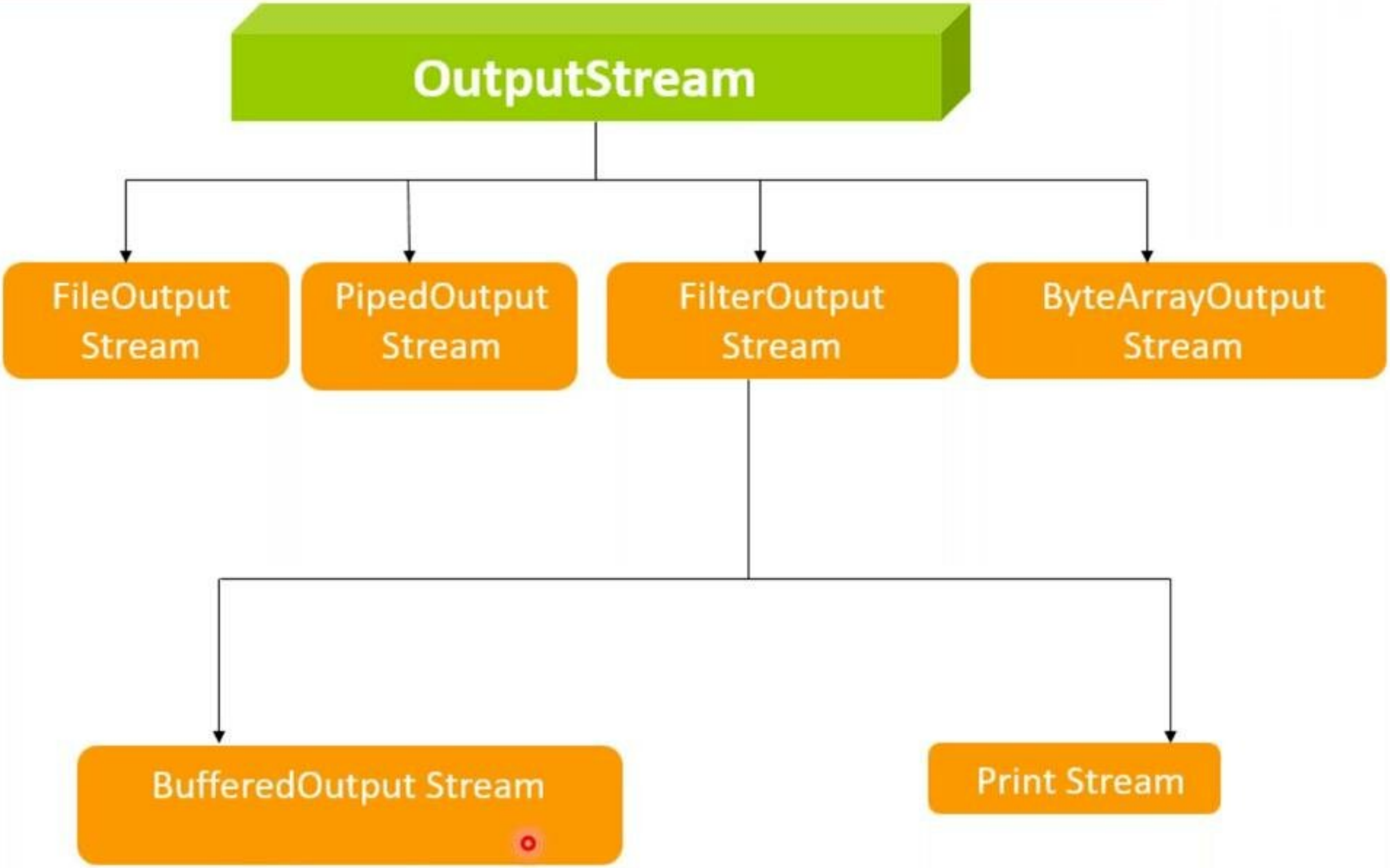
Creates a `FileInputStream` object

Finds out the size of the file by using the `available()` method

OutputStream

- It is an abstract class.
- Defines methods for writing bytes or array of bytes to the stream.
- An OutputStream is automatically opened when it is created.

OutputStream Hierarchy



FileOutputStream

- It is used to write output to a file stream.

FileOutputStream

FileOutputStream

```
void update() {
        byte[] words = new byte[80];
        try {
            System.out.println("Enter a line to be saved into a file");
            int bytes = System.in.read(words);
            FileOutputStream fos = new FileOutputStream("Text");
            fos.write(words, 0, bytes);
            System.out.println("Updated!");
            } catch (IOException obj) {
                System.out.println("Error creating file!");
        }
}
```

Throws an `IOException` if a read only file is opened

FileOutputStream


```
void update() {
        byte[] words = new byte[80];
        try {
            System.out.println("Enter a line to be saved into a file");
            int bytes = System.in.read(words);
            FileOutputStream fos = new FileOutputStream("Text");
            fos.write(words, 0, bytes);
            System.out.println("Updated!");
            } catch (IOException obj) {
                System.out.println("Error creating f
            }
    }
```

Returns the number of bytes read

Uses the `write()` method to write the data into the file

ByteArrayInputStream

- Creates an InputStream using an array of bytes.
- Internal pointer keeps track of the next byte to be read.
- Overrides methods of the base class.

Constructor	Description
ByteArrayInputStream (byte[] b)	Creates a byte array input stream with *b* as the input source
ByteArrayInputStream (byte[] b, int start, int num)	Creates a byte array input stream, *b*, that begins with the character at the index specified by *start* and is *num* bytes long

ByteArrayInputStream


```
void prints() {
        String text = "Jack and Jill went up the hill";
        byte[] data = text.getBytes();
        ByteArrayInputStream bais = new ByteArrayInputStream(data, 0, 4);
        int word;
        while ((word = bais.read()) != -1)
            System.out.print((char) word);
        }
        System.out.println();
        bais.reset();      //using reset ( ) method and again reading
        word = 0;
        while ((word = bais.read()) != -1) {
        System.out.print((char) word);
        }
    }
```

Converts the string into a bytearray by using the `getBytes()` method

ByteArrayInputStream


```
void prints() {
        String text = "Jack and Jill went up the hill";
        byte[] data = text.getBytes();
        ByteArrayInputStream bais = new ByteArrayInputStream(data, 0, 4);
        int word;
        while ((word = bais.read()) != -1) {
            System.out.print((char) word);
        }
        System.out.println();
        bais.reset();      //using reset ( ) method and again reading
        word = 0;
        while ((word = bais.read()) != -1)
        System.out.print((char) word);
        }
}
```

Reads each byte and prints it after casting it into character type

ByteArrayInputStream


```
void prints() {
        String text = "Jack and Jill went up the hill";
        byte[] data = text.getBytes();
        ByteArrayInputStream bais = new ByteArrayInputStream(data, 0, 4);
        int word;
        while ((word = bais.read()) != -1) {
            System.out.print((char) word);
        }
        System.out.println();
        bais.reset();       //using reset ( ) method and again reading
        word = 0;
        while ((word = bais.read()) != -1) {
        System.out.print((char) word);
        }
    }
```

Uses `reset()` method to move the pointer to the beginning of the file and reads it again from the begining

ByteArrayOutputStream

ByteArrayOutputStream

Creates an output stream in which data is written to a byte array

`ByteArrayOutputStream()`

Creates a new byte array output stream

`ByteArrayOutputStream(int size)` Creates a new byte array output stream, with a buffer capacity specified in bytes

Buffered I/O classes

- Buffer means a temporary storage area.
- The time taken to retrieve data is less when data is temporarily stored in a buffer.
- These classes do not create a new stream but provide buffering functionality for existing stream.

BufferedInputStream

Constructor	Description
BufferedInputStream(InputStream in)	Creates a buffered input stream for the specified input stream instance using a default buffer size
BufferedInputStream(InputStream in, int size)	Creates a buffered input stream of given size for the specified input stream instance

BufferedInputStream

```
void print() throws IOException {
    String text = "Jack & Jill, went up the hill";
    System.out.println("Original String: " + text);
    System.out.println("After replacing '&' with 'and': ");
    byte[] buffer = text.getBytes();
    ByteArrayInputStream arrayin = new ByteArrayInputStream(buffer);
    BufferedInputStream bufferin = new BufferedInputStream(arrayin);
    int character;
```

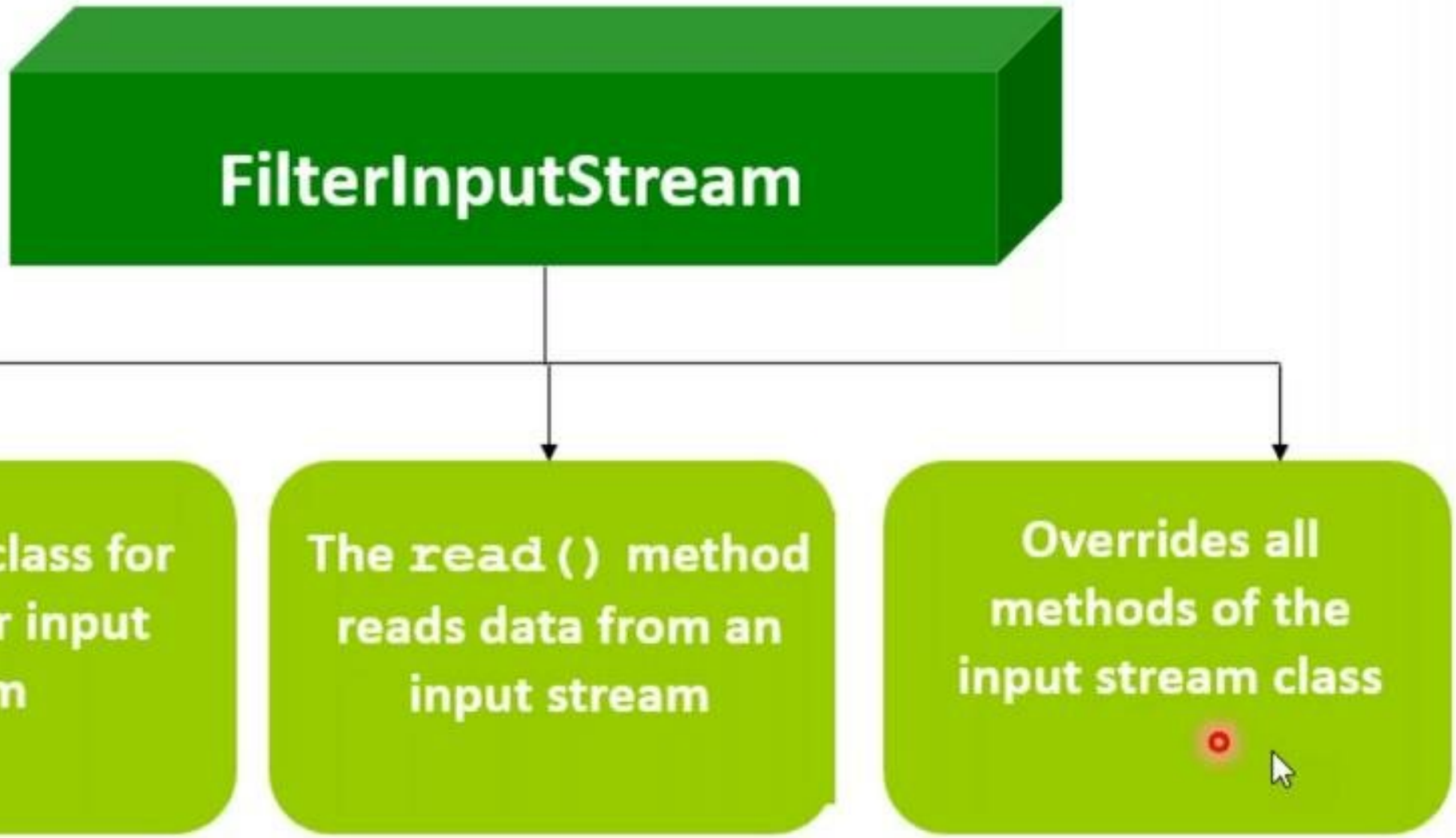
Creates a `BufferedInputStream` object

BufferedOutputStream

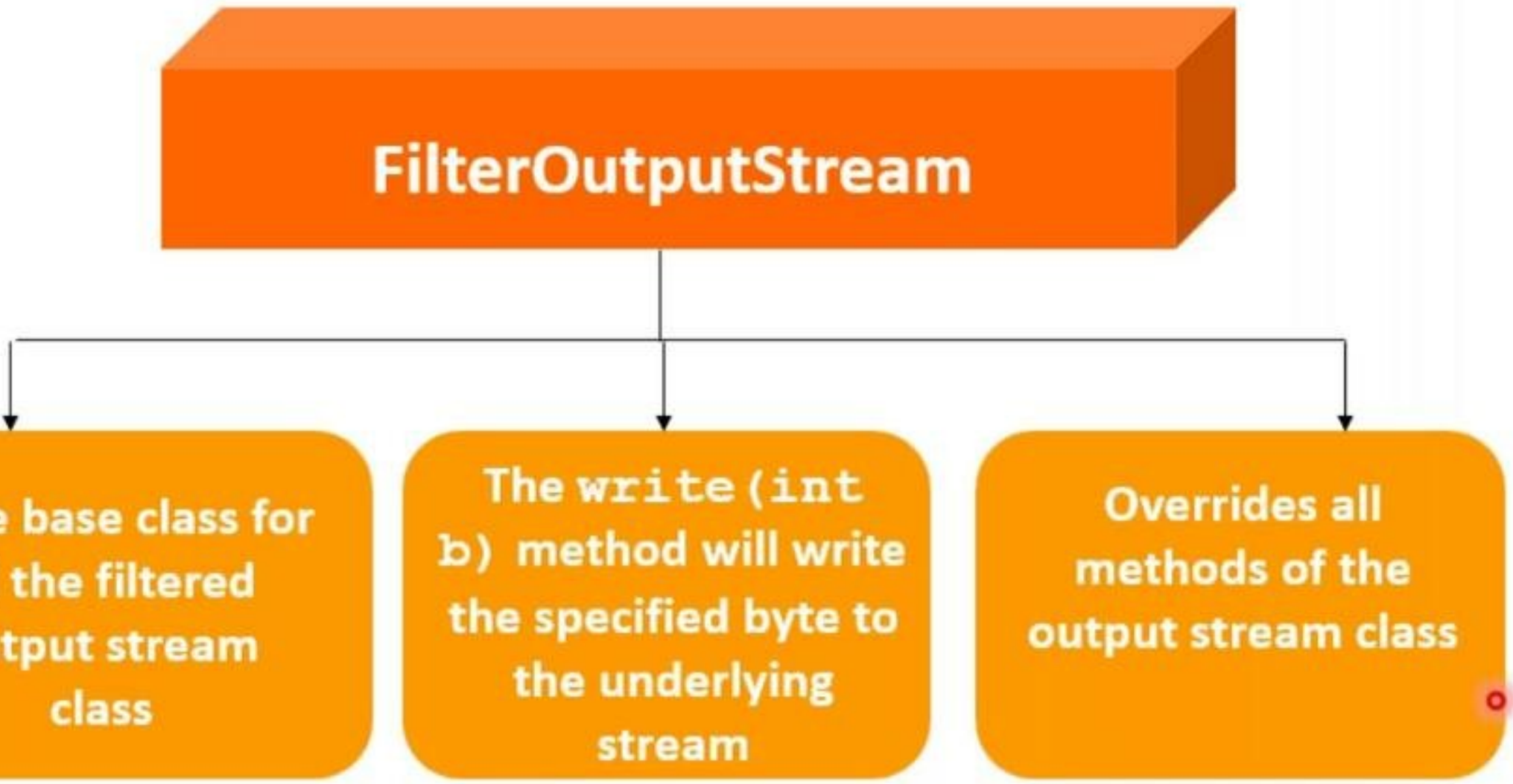
- Provides output buffering.
- The `flush()` method ensures that the data is written to an actual physical output device.

Constructor	Description
BufferedOutputStream(OutputStream os)	Creates a buffered output stream for the specified output stream instance using a default buffer size
BufferedInputStream(OutputStream os, int size)	Creates a buffered output stream of given size for the specified output stream instance

FilterInputStream



FilterOutputStream



PrintStream

- PrintStream extends FilterOutputStream.
- A PrintStream adds functionality to another output stream, namely the ability to print representations of various data values conveniently.
- a PrintStream never throws an IOException.
- All characters printed by a PrintStream are converted into bytes using the platform's default character encoding
- “out ” reference variable is an reference variable of PrintStream.

Demonstration: Example 6

taStream2.java (DataStream2.java.JAV)

DataInput Interface

Reads bytes and reconstructs them into any one of the Java primitive types

dataread.java

Data can be converted to a String format

Has methods for reading Java primitive data types

DATARE~1 - Notepad

File Edit Format View Help

```
import java.io.*;
class DataRead
{
public static void main(String args[])throws IOException
{
    DataInputStream dis=new DataInputStream(new FileInputStream("myfile.txt"));
        String val1=" " ;
    while(val1!=null)
    {
        val1=dis.readLine();
                if(val1!=null)
                System.out.println(val1);
    }
    dis.close();
}
}
```

Windows (CRLF) Ln 1, Col 1 100%

16:00
07-04-2022

DataInputStream can be used to get the input for the user at run time

To perform this task we use “in” reference variable of System class

“in” variable is an reference variable of PrintStream class which internally connected with console to get the input Treated as an unlimited file

DATARE~1 - Notepad

File Edit Format View Help

```
import java.io.*;
class DataRead
{
public static void main(String args[])throws IOException
{
    DataInputStream dis=new DataInputStream(new FileInputStream("myfile.txt"));
        String val1=" " ;
    while(val1!=null)
    {
        val1=dis.readLine();
                if(val1!=null)
                System.out.println(val1);
    }
    dis.close();
}
}
```

Windows (CRLF) Ln 12, Col 28 100%

16:01
07-04-2022

DATAST~1 - Notepad

File Edit Format View Help

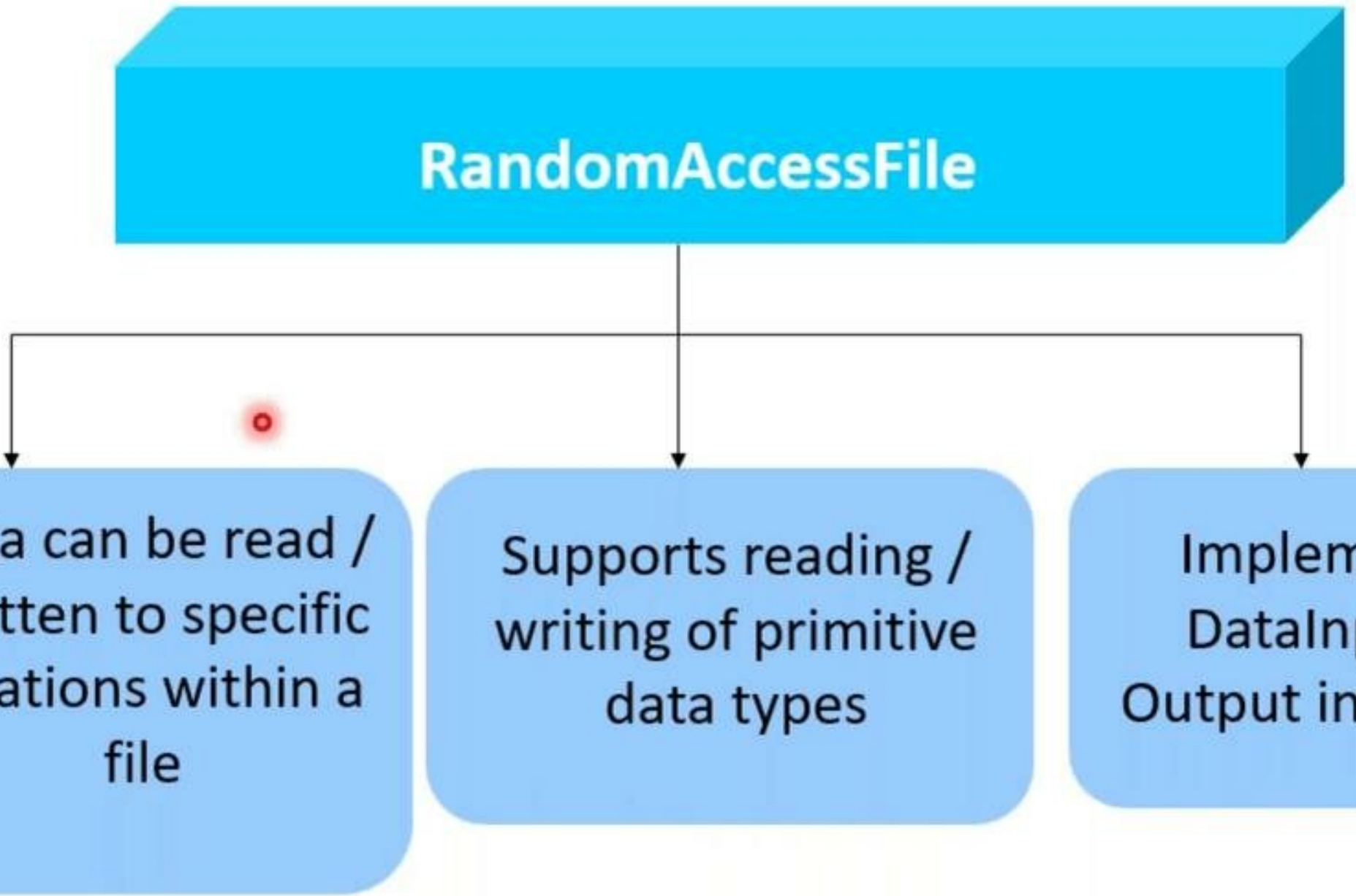
```
import java.io.*;
class DataStream
{
public static void main(String as[])throws IOException
{
    DataInputStream dis=new DataInputStream(System.in);
    FileOutputStream fos=new FileOutputStream("infogain1.txt");

    String var1=" " ;
    while(!var1.equals("stop"))
    {
        var1=dis.readLine();
                System.out.println(var1);
                fos.write(var1.getBytes());
               //dos.flush();
    }
dis.close();
fos.close();
}
}
```

Windows (CRLF) Ln 15, Col 3 100%

16:03
07-04-2022

RandomAccessFile 2-1



RandomAccessFile 2-2

Constructor	Description
RandomAccessFile(File file, String mode)	Creates a random access file stream to read from, and optionally to write to, a File object specified. Mode determines what type of file access is permitted
RandomAccessFile(String name, String mode)	Creates a random access file stream to read from, and optionally to write to, a File with the specified name. Mode determines what type of file access is permitted

RandomAccessFile 2-2


```
void search(final String file) {
    byte b;
    try {
        RandomAccessFile files = new RandomAccessFile(file, "r");
        long size = files.length();
        long data = 0;
        System.out.println("File Exist!!");
        System.out.println("The content of the file is:");
        while (data < size) {
            String s = files.readLine();
            System.out.println(s);
            data = files.getFilePointer();
        }
    } catch (IOException e) {
        System.out.println("File does not exist!");
    }
```

DATAST~1 - Notepad

File Edit Format View Help

```
import java.io.*;
class DataStream
{
public static voi
{
    DataInputStre
    FileOutputStr

    String var1="
    while(!var1.e
    {
        var1=dis.
                S
                f
    }
dis.close();
fos.close();
}
}
```

READAC~1 - Notepad

File Edit Format View Help

```
import java.io.*;

public class ReadAccessFile{
        public static void main(String[] args) throws
IOException{
                BufferedReader in = new BufferedReader
(new InputStreamReader(System.in));
                System.out.print("Enter File name : ");
                String str = in.readLine();
                File file = new File(str);
                if(!file.exists())
                {
                        System.out.println("File does
not exist.");
                        System.exit(0);
                }
                try{
                        //Open the file for both reading
and writing
                        RandomAccessFile rand = new
RandomAccessFile(file,"r");
                        int i=(int)rand.length();
                        System.out.println("Length: " +
i);
```

Windows (CRLF) Ln 1, Col 1 100%

Windows (CRLF) Ln 15, Col 3 100%

16:08
07-04-2022

READAC~1 - Notepad

File Edit Format View Help

```
import java.io.*;

public class ReadAccessFile{
        public static void main(String[] args) throws IOException{
                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                System.out.print("Enter File name : ");
                String str = in.readLine();
                File file = new File(str);
                if(!file.exists())
                {
                        System.out.println("File does not exist.");
                        System.exit(0);
                }
                try{
                        //Open the file for both reading and writing
                        RandomAccessFile rand = new RandomAccessFile(file,"r");
                        int i=(int)rand.length();
                        System.out.println("Length: " + i);
                        rand.seek(0);   //Seek to start point of file
                        for(int ct = 0; ct < i; ct++){
                                byte b = rand.readByte();
                                System.out.print((char)b); //read the character
                        }
                        rand.close();
rand = new RandomAccessFile(file,"rw");
rand.writeBytes("hello");
                }
                        catch(IOException e)
```

Windows (CRLF) Ln 1, Col 1 100%

16:08
07-04-2022

READAC~1 - Notepad

File Edit Format View Help

```
File file = new File(str);
if(!file.exists())
{
        System.out.println("File does not exist.");
        System.exit(0);
}
try{
        //Open the file for both reading and writing
        RandomAccessFile rand = new RandomAccessFile(file,"r");
        int i=(int)rand.length();
        System.out.println("Length: " + i);
        rand.seek(0);   //Seek to start point of file
        for(int ct = 0; ct < i; ct++){
                byte b = rand.readByte();
                System.out.print((char)b); //read the character
                }
                rand.close();
rand = new RandomAccessFile(file,"rw");
rand.writeBytes("hello");
                }
                        catch(IOException e)
                {
                System.out.println(e.getMessage());
                }
        }
}
```

Windows (CRLF) Ln 22, Col 31 100%

16:10
07-04-2022

ration: Example 10

ReadAccessFile.java (ReadAccessFile.java.JAV)

Demonstration: Example 10

ReadAccessFile.java (ReadAccessFile.java.JAV)

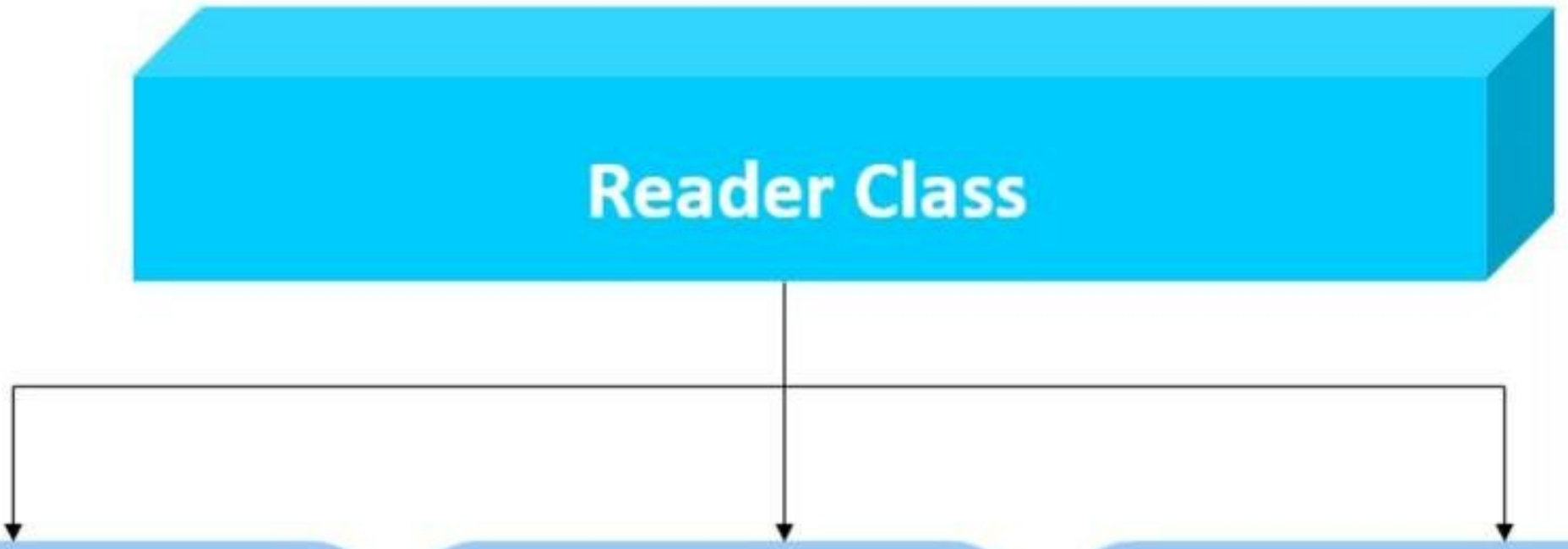
Character Streams

- Handle character oriented input / output operation.

Character Streams

- Handle character oriented input / output operation.
- Reader and Writer class are at the top of the class hierarchy.
- They are abstract classes.

Reader Class

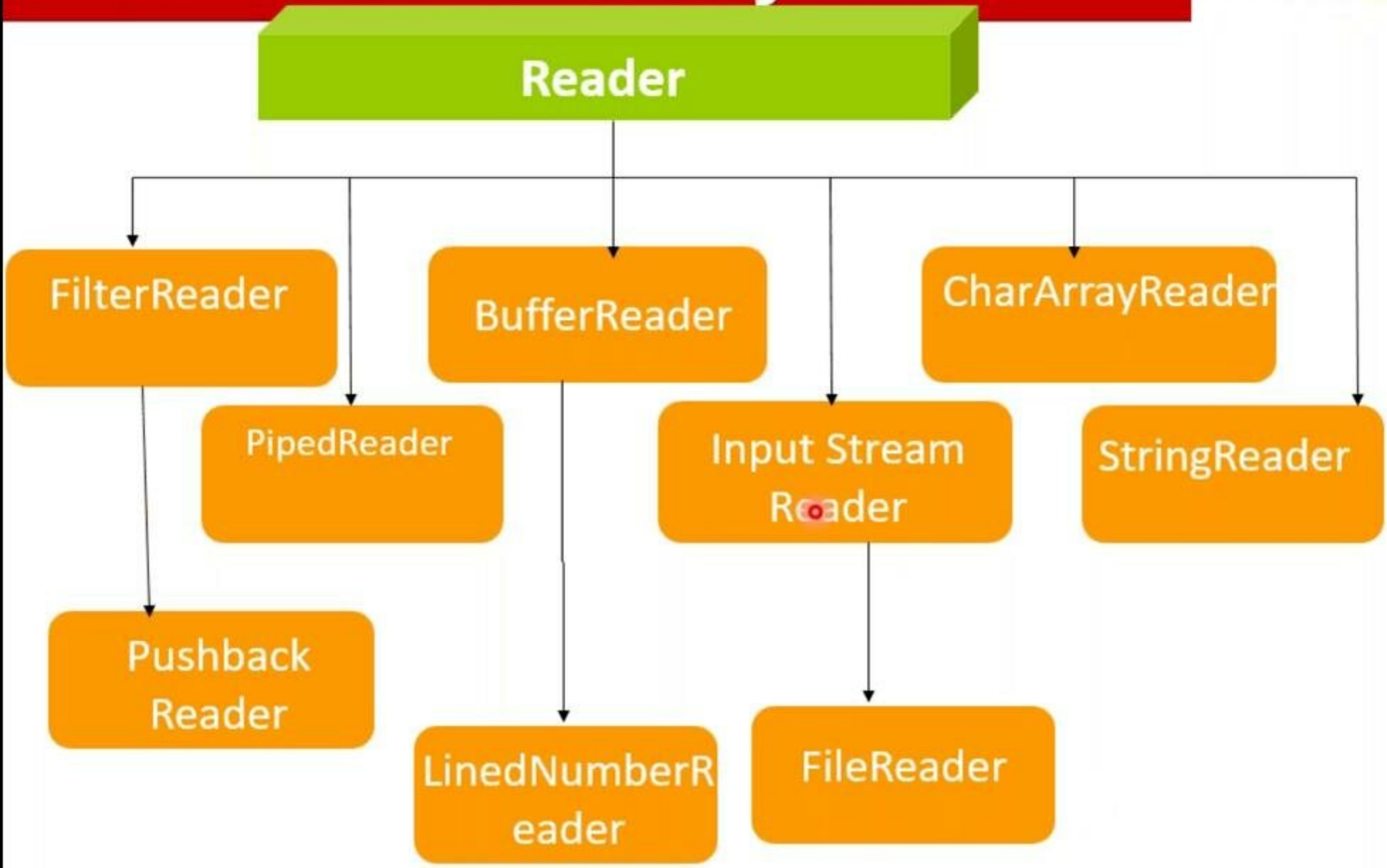


Abstract class for reading character streams

Subclasses override the methods defined in this class to increase efficiency

Subclasses implement `read(char[], int, int)` and `close()` methods

Reader Hierarchy



PrintWriter Class

- Formatted representations of objects are printed to text output stream .
- All the print methods of the PrintStream class are implemented by this class.
- It supports printing of primitive data types, character array, strings and objects.
- `print()` and `println()` methods are used to provide formatted output.

Note :- This class is just same as the PrintStream class

Demonstration: Example 11

CharArrayWriterDemo.java

CharArrayReaderDemo.java

READAC~1 - Notepad

File Edit Format View Help

```
File file = new File(str);
if(!file.exists())
{
        System.out.println("File does not exist.");
        System.exit(0);
}
try{
        //Open the file for both reading and writing
        RandomAccessFile rand = new RandomAccessFile(file,"r");
        int i=(int)rand.length();
        System.out.println("Length: " + i);
        rand.seek(0);   //Seek to start point of file
        for(int ct = 0; ct < i; ct++){
                byte b = rand.readByte();
                System.out.print((char)b); //read the character
                }
                rand.close();
rand = new RandomAccessFile(file,"rw");
rand.writeBytes("hello");
                }
                catch(IOException e)
                {
                System.out.println(e.getMessage());
                }
        }
}
```

Windows (CRLF) Ln 26, Col 26 100%

16:14
07-04-2022

CHARAR~1 - Notepad

File Edit Format View Help

```
import java.io.*;
class CharArrayWriterDemo
{
	public static void main(String args[]) throws IOException
	{
		CharArrayWriter f = new CharArrayWriter();
		String s = "This should end up in the array";
		char buf[] = new char[s.length()];
		s.getChars(0, s.length(), buf, 0);
		f.write(buf);
		System.out.println("Buffer as a string");
		System.out.println(f.toString());
		System.out.println("Into array");
		char c[] = f.toCharArray();
		for (int i=0; i<c.length; i++)
		{
			System.out.print(c[i]);
		}
		System.out.println("\\nTo a FileWriter()");
		FileWriter f2 = new FileWriter("test.txt");
		f.writeTo(f2);
		f2.close();
		System.out.println("Doing a reset");
		f.reset();
		for (int i=0; i<3; i++)
			f.write('X');
		System.out.println(f.toString());
	}
```

Windows (CRLF) Ln 18, Col 4 100%

16:15
07-04-2022

CHARAR~1 - Notepad

File Edit Format View Help

```
import java.io.*;
class CharArrayWr
{
        public st
        {
                C
                S
                c
                s
                f
                S
                S
                S
                c
                f
                {

                }
                S
                F
                f
                f
                S
                f
                f
                f.write( x );
System.out.println(f.toString());
}
```

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class CharArrayReaderDemo
{
        public static void main(String args[]) throws
IOException
        {
                String tmp =
"abcdefghijklmnopqrstuvwxyz";
                int length = tmp.length();
                char c[] = new char[length];
                tmp.getChars(0, length, c, 0);
                CharArrayReader input1 = new
CharArrayReader(c);
                CharArrayReader input2 = new
CharArrayReader(c, 0, 5);
                int i;
                System.out.println("input1 is:");
                while((i = input1.read()) != -1)
                {
                        System.out.print((char)i);
                }
                System.out.println();
                System.out.println("input2 is:");
                while((i = input2.read()) != -1)
```

Windows (CRLF) Ln 1, Col 1 100%

Windows (CRLF) Ln 18, Col 4 100%

16:16
07-04-2022

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class CharArrayReaderDemo
{
	public static void main(String args[]) throws IOException
	{
		String tmp = "abcdefghijklmnopqrstuvwxyz";
		int length = tmp.length();
		char c[] = new char[length];
		tmp.getChars(0, length, c, 0);
		CharArrayReader input1 = new CharArrayReader(c);
		CharArrayReader input2 = new CharArrayReader(c, 0, 5);
		int i;
		System.out.println("input1 is:");
		while((i = input1.read()) != -1)
		{
			System.out.print((char)i);
		}
		System.out.println();
		System.out.println("input2 is:");
		while((i = input2.read()) != -1)
		{
			System.out.print((char)i);
		}
		System.out.println();
	}
}
```

Windows (CRLF) Ln 17, Col 5 100%

16:16
07-04-2022

Serialization

- Serialization is the process of saving an object in a storage medium (such as a file, or a memory buffer) or to transmit it over a network connection in binary form.
- The serialized objects are JVM independent and can be re-serialized by any JVM.
- In this case the "in memory" java objects state are converted into a byte stream.
- This type of the file can not be understood by the user.
- It is a special types of object i.e. reused by the JVM (Java Virtual Machine).
- This process of serializing an object is also called deflating or marshalling an object.

Serialization

- Conceptually, we need to convert every field of the object to bytes
  - Don't need to worry about methods - *why?*

- If an object contains members that are objects, they need to be converted to bytes
  - E.g. an array or Vector
  - It's a recursive process - at some point it reaches some basic types: integers, doubles, characters

Reading and Writing Objects


```
class Student implements Serializable {
                        int rollno = 101;
                        String name = "Scott";
}
public class SerializingObject {
        public static void main(String[] args) throws IOException {
                                try {
                                Student s1 = new Student();
                                FileOutputStream f1 = new
    FileOutputStream("studentdetails.txt");
                                ObjectOutput ObjOut = new ObjectOutputStream(f1);
                                ObjOut.writeObject(s1);
                                ObjOut.close();
                                System.out.println("Serializing completed successfully.");
                                } catch (IOException e) { System.out.println(e);
                                }
        }
}
```

Reading and Writing Objects


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

                File f = new File("studentdetails.txt");
                try {       ObjectInputStream obj = new ObjectInputStream(f);
                            Student temp = (Student) obj.readObject();
                            System.out.println("Student name = " + temp.name);
                            obj.close();
                            System.out.println("Deserializing Completly Successfully.");

                } catch (FileNotFoundException fe) {
                                    System.out.println("File not found ");
                } catch (ClassNotFoundException e) {
                                    e.printStackTrace();
                }
                }
}
```

Demonstration: Example 10

emp.java (emp.java.JAV)

Myserver.java (Myserver.java.JAV)

Myclient1.java (Myclient1.java.JAV)

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class CharArrayReaderDemo
{
	public static void main(String args[]) throws IOException
	{
		String tmp = "abcdefghijklmnopqrstuvwxyz";
		int length = tmp.length();
		char c[] = new char[length];
		tmp.getChars(0, length, c, 0);
		CharArrayReader input1 = new CharArrayReader(c);
		CharArrayReader input2 = new CharArrayReader(c, 0, 5);
		int i;
		System.out.println("input1 is:");
		while((i = input1.read()) != -1)
		{
			System.out.print((char)i);
		}
		System.out.println();
		System.out.println("input2 is:");
		while((i = input2.read()) != -1)
		{
			System.out.print((char)i);
		}
		System.out.println();
	}
}
```

Windows (CRLF) Ln 23, Col 5 100%

16:24
07-04-2022

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Char
{
        public st
        {
                S
                i
                c
                t
                C
                C
                i
                S
                w
                {

                }
                S
                S
                w
                {

                }
                S

        }
}
```

MYCLIE~1 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Myclient1
{
    ObjectOutputStream dout;
    public Myclient1()
    {
         try
         {
            emp e1=new emp("Amit",10,5);
            dout=new ObjectOutputStream(new
FileOutputStream("infogain8.txt"));
         dout.writeObject(e1);
         dout.flush();
         }catch(Exception e){System.out.println(e);}
    }
    public static void main(String s[])
    {
         new Myclient1();
    }
}
```

Windows (CRLF) Ln 1, Col 1 100%

Windows (CRLF) Ln 23, Col 5 100%

16:26
07-04-2022

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Char
{
        public st
        {
                S
                i
                c
                t
                C
                C
                i
                S
                w
                {
                }
                S
                S
                w
                {
                }
                S
        }
}
```

MYSERV~1 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Myserver
{
    ObjectInputStream dis;
    public Myserver()
    {
     try
    {
          dis=new ObjectInputStream(new FileInputStream
("dxc.txt"));
          emp z=(emp)dis.readObject();
          System.out.println(z.name);
          System.out.println(z.age);
          System.out.println(z.a);
    }catch(Exception e){System.out.println(e);}
    }
    public static void main(String s[])
    {
          new Myserver();
    }
}
```

Windows (CRLF) Ln 13, Col 22 100%

Windows (CRLF) Ln 23, Col 5 100%

16:26
07-04-2022

The Externalizable interface

•You can control the process of serialization by implementing the Externalizable interface instead of Serializable.

• This interface extends the original Serializable interface and adds writeExternal() and readExternal().

•These two methods will automatically be called in your object's serialization and deserialization, allowing you to control the whole process.

•There is one major difference between serialization and externalization:

- •When you serialize an Externalizable object, a default constructor will be called automatically; only after that will the readExternal() method be called.

•If you inherit some class from a class implementing the Externalizable interface, you must call writeExternal() and readExternal() methods when you serialize or deserialize this class in order to correctly save and restore the object.

Demonstration: Example 11

Car.java (Car.java.JAV)

ExternExample.java (ExternExample.java.JAV)

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Char
{
        public st
        {
                S
                i
                c
                t
                C
                C
                i
                S
                w
                {
                }
                S
                S
                w
                {
                }
                S
        }
}
```

CAR~1 - Notepad

File Edit Format View Help

```
import java.io.*;

public class Car implements Externalizable
{
        String name;
    int year;
    public Car()
        {
                super();
        }
        Car(String n, int y)
        {
                name = n;
                year = y;
    }

    /**
     * Mandatory writeExernal method.
     */
    public void writeExternal(ObjectOutput out) throws
IOException
        {
                out.writeObject(name);
```

Windows (CRLF) Ln 16, Col 6 100%

Windows (CRLF) Ln 23, Col 5 100%

16:30
07-04-2022

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Char
{
        public st
        {
                S
                i
                c
                t
                C
                C
                i
                S
                w
                {

        }
                S
                S
                w
                {

                }
                S
    }
}
```

CAR~1 - Notepad

File Edit Format View Help

```
        {
                name = n;
                year = y;
    }

    /**
     * Mandatory writeExernal method.
     */
    public void writeExternal(ObjectOutput out) throws
IOException
        {
                out.writeObject(name);
                out.writeInt(year);
    }

    /**
     * Mandatory readExternal method.
     */
    public void readExternal(ObjectInput in) throws
IOException, ClassNotFoundException
        {
                name = (String) in.readObject();
                year = in.readInt();
    }
```

Windows (CRLF) Ln 16, Col 6 100%

Windows (CRLF) Ln 23, Col 5 100%

16:30
07-04-2022

CHARAR~2 - Notepad

File Edit Format View Help

```
import java.io.*;
public class Char
{
        public st
        {
                S
                i
                c
                t
                C
                C
                i
                S
                w
                {

                }
                S
                S
                w
                {

                }
                S
        }
}
```

CAR~1 - Notepad

File Edit Format View Help

```
     * Mandatory writeExernal method.
     */
    public void writeExternal(ObjectOutput out) throws
IOException
        {
                out.writeObject(name);
                out.writeInt(year);
    }

    /**
     * Mandatory readExternal method.
     */
    public void readExternal(ObjectInput in) throws
IOException, ClassNotFoundException
        {
                name = (String) in.readObject();
                year = in.readInt();
    }

    /**
     * Prints out the fields. used for testing!
     */
    public String toString()
        {
```

Windows (CRLF) Ln 33, Col 22 100%

Windows (CRLF) Ln 23, Col 5 100%

16:31
07-04-2022

Demonstration: Example 13

prun.java (prun.java.JAV)