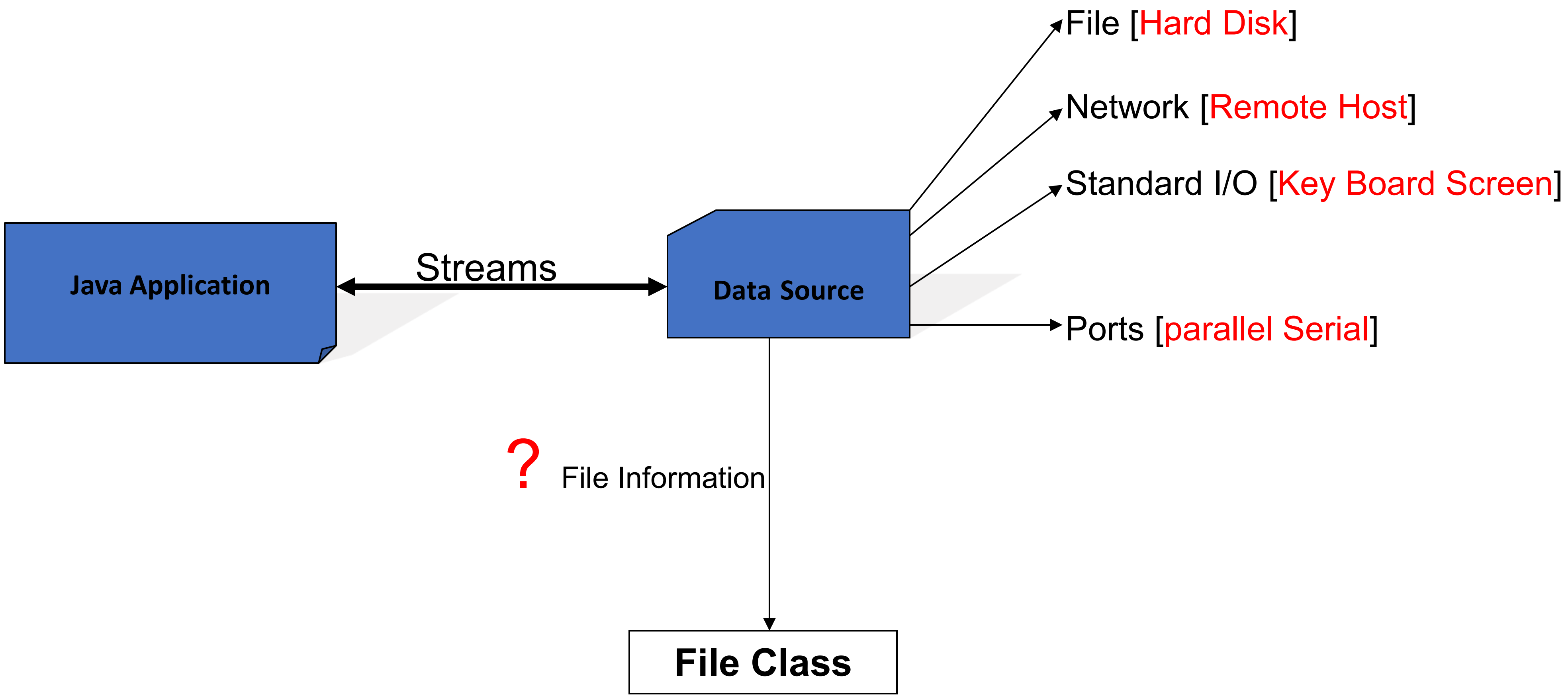
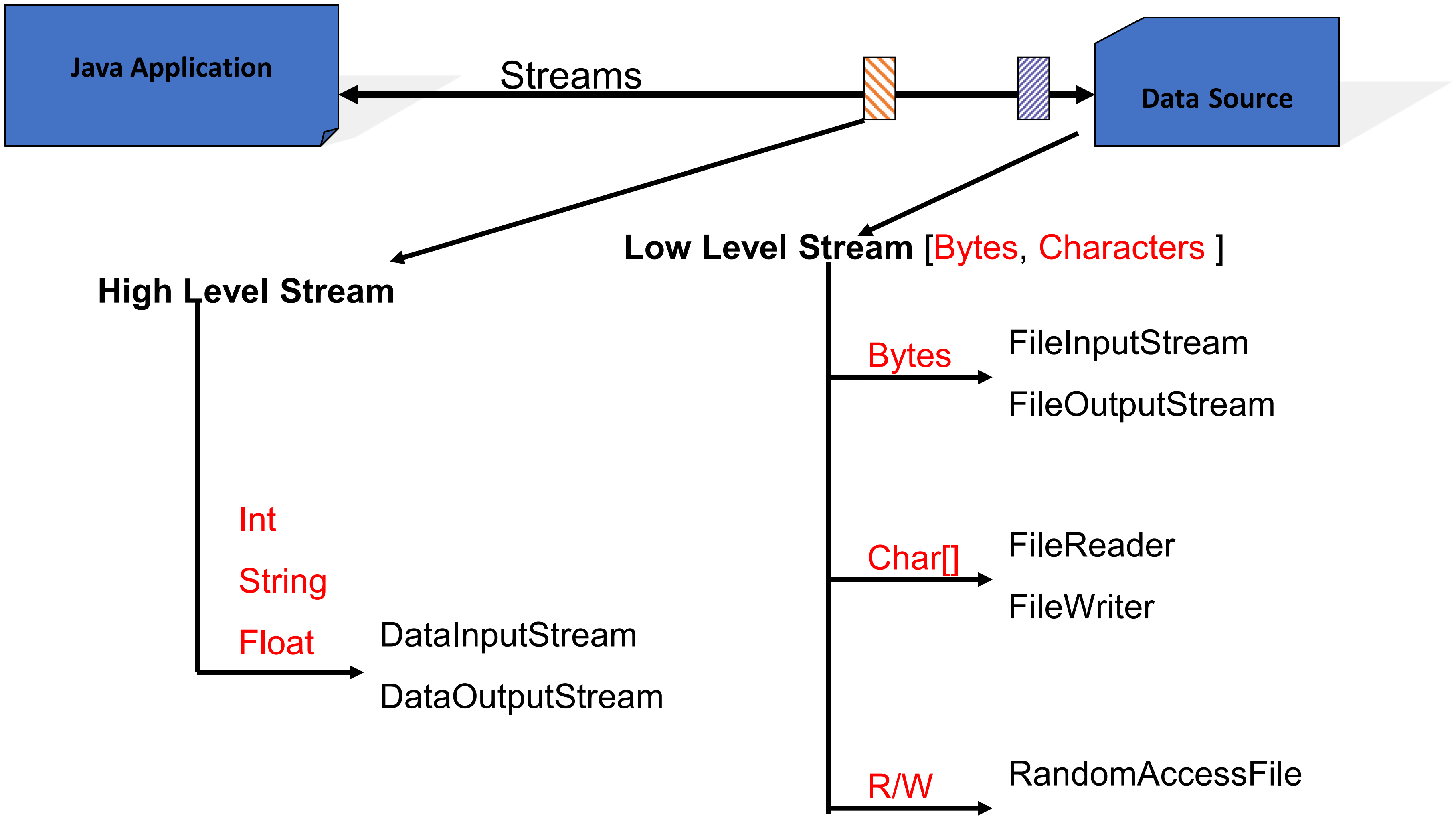
Input & Output Streams

Streams


File [Hard Disk]
Network [Remote Host]
Standard I/O [Key Board Screen]
Java Application
Streams
Data Source
Ports [parallel Serial]
?
File Information
File Class

Streams



Java Application

Streams

Data Source

High Level Stream

Int

String

Float

DataInputStream

DataOutputStream

Low Level Stream [Bytes, Characters]

Bytes

FileInputStream

FileOutputStream

Char[]

FileReader

FileWriter

R/W

RandomAccessFile

Streams

- A **stream** is a flow of data between a Data Source and a Data Sink (Destination).
- Streams are used for data input and output.
- An **Input Stream** is a stream that **reads input into the application.**
  Reading is a blocking operation (i.e., it blocks its thread).
- An **Output Stream** is a stream that **carries data out from the application**.
- Streams can be classified into two categories:
  Low-Level Streams and **High-Level Streams**.

Low Level Streams

- A Low-Level Stream is a stream that is **attached directly to the source/destination**.

- It can **ONLY** deal with raw data in the form of bytes or characters.

High Level Streams

- A High-Level Stream is a stream that is attached to a lower-level stream (i.e., layered over it).

- It can deal with higher data types such as int, float, String, or even whole objects.

- A High-Level Stream saves some conversion effort for the programmer.
  (e.g. Reading complete Strings instead of reading character by character
  then transferring them into a String).

File Class

- Commonly Used Constructor(s):
  - **`File(String path)`**
  - **`File(String parent, String child)`**
  - **`File(File parent, String child)`**

- Commonly Used Method(s):
  - **`boolean exists()`**
  - **`boolean isFile()`**
  - **`boolean isDirectory()`**
  - **`String getName()`**
  - **`String getParent()`**
  - **`String getAbsolutePath()`**

File Class cont'd

- Commonly Used Method(s):
  - **`String[] list()`**
  - **`boolean canRead()`**
  - **`boolean canWrite()`**
  - **`boolean delete()`**
  - **`long length()`**
  - **`boolean createNewFile()`**
  - **`boolean mkdir()`**

FileInputStream Class

- Commonly Used Constructor(s):
  - **`FileInputStream(String file)`**
  - **`FileInputStream(File file)`**

- Commonly Used Method(s):
  - **`int read()`**
  - **`int read(byte[] b)`**
  - **`int read(byte[] b, int offset, int length)`**
  - **`int available()`**
  - **`long skip(long)`**
  - **`void close()`**

- The **`offset`** parameter determines the start index in the destination array **`b`**.

- The **`length`** parameter specifies the maximum number of bytes to read.

FileOutputStream Class

- Commonly Used Constructor(s):
  - **`FileOutputStream(String file)`**
  - **`FileOutputStream(File file)`**

- Commonly Used Method(s):
  - **`void write(int b)`**
  - **`void write(byte[] b)`**
  - **`void write(byte[] b, int offset, int length)`**
  - **`void close()`**
  - **`void flush()`**

- The **`offset`** parameter determines the start index at the source array **`b`**.

- The **`length`** parameter specifies the number of bytes to write.

FileReader Class

- Commonly Used Constructor(s):
  - **`FileReader(String file)`**
  - **`FileReader(File file)`**

- Commonly Used Method(s):
  - **`int read()`**
  - **`int read(char[] c)`**
  - **`int read(char[] c, int offset, int length)`**

FileWriter Class

- Commonly Used Constructor(s):
  - **`FileWriter(String file)`**
  - **`FileWriter(File file)`**

- Commonly Used Method(s):
  - **`void write(int c)`**
  - **`void write(char[] c)`**
  - **`void write(char[] c, int offset, int length)`**
  - **`void write(String str)`**
  - **`void write(String str, int offset, int length)`**

RandomAccessFile Class

- Commonly Used Constructor(s):
  - **`RandomAccessFile(String file, String mode)`**
  - **`RandomAccessFile(File file, String mode)`**

- The **`mode`** parameter can assume the values **“r”** or **“rw”.**

- Commonly Used Method(s):
  - **`long getFilePointer()`**
  - **`void seek(long position)`**
  - **`long length()`**
  - **`int read()`**
  - **`int read(byte[] b)`**
  - **`int read(byte[], int offset, int length)`**
  - **`void write(int b)`**
  - **`void write(byte[] b)`**
  - **`void write(byte[] b, int offset, int length)`**

DataInputStream Class

- Commonly Used Constructor(s):
  - **`DataInputStream(InputStream in)`**

- Commonly Used Method(s):
  - **`int readInt()`**
  - **`long readLong()`**
  - **`float readFloat()`**
  - **`double readDouble()`**
  - **`String readUTF()`**

DataOutputStream Class

- Commonly Used Constructor(s):
  - **`DataOutputStream(OutputStream out)`**

- Commonly Used Method(s):
  - **`void writeInt(int i)`**
  - **`void writeLong(long l)`**
  - **`void writeFloat(float f)`**
  - **`void writeDouble(double d)`**
  - **`void writeUTF(String str)`**

Saving a Text into File Example

- The following code sample is for allowing the user to choose a text file to save the text he entered in a text area:

```
class MySaveListener implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
       JFileChooser fc = new JFileChooser();
             if(fc.showSaveDialog(NotePad.this) == JFileChooser.APPROVE_OPTION) {
                    String path = fc.getSelectedFile().getPath();
                    FileOutputStream fos = new FileOutputStream(path);
                    byte[] b = ta.getText().getBytes();
                    fos.write(b);
                    fos.close();
             }
  }
}
```

Opening a Text File Example

- The following code sample is for allowing the user to choose a text file to open in a text area:

```
class MyOpenListener implements ActionListener {
  public void actionPerformed(ActionEvent ae) {
     JFileChooser fc = new JFileChooser();
               if(fc.showOpenDialog(NotePad.this) == JFileChooser.APPROVE_OPTION) {
                      String path = fc.getSelectedFile().getPath();
                      FileInputStream fis = new FileInputStream(path);
                      int size = fis.available();
                      byte[] b = new byte[size];
                      fis.read(b);
                      ta.setText(new String(b));
                      fis.close();
             }
  }
}
```

Lab Exercise

1. Open and Save Text Files

- Continue working on the Text Editor Application.
  Write the code to open and save text files using either of the following set of classes:
  - FileInputStream and FileOutputStream
  - DataInputStream and DataOutputStream
  - FileReader and FileWriter (Bonus)

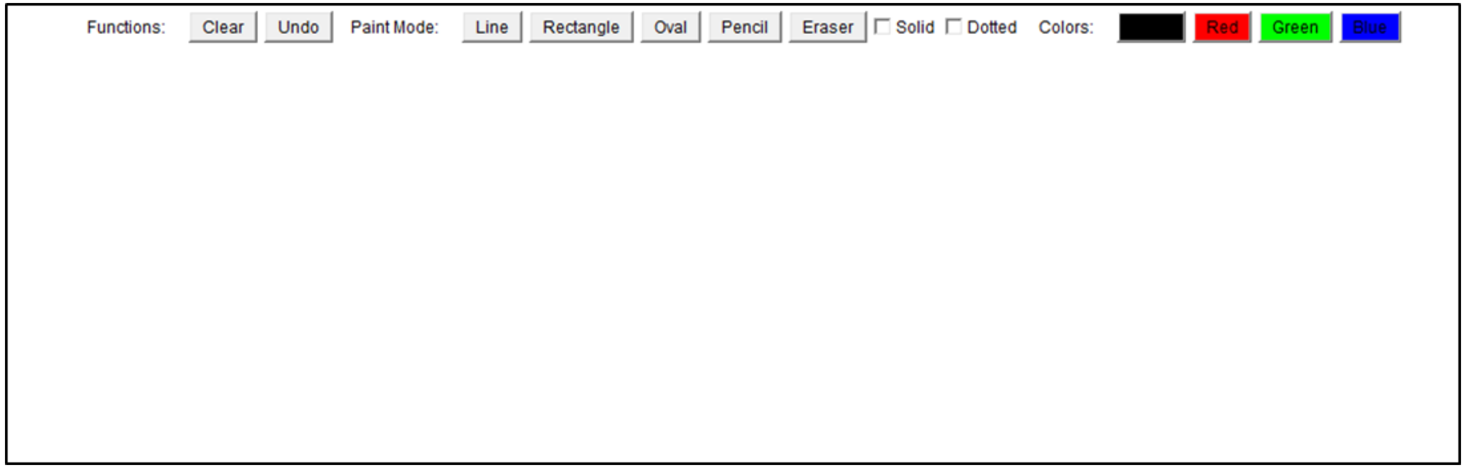
Java Certificate

Here

Challange

Paint Brush

Paint Brush is a JFrame based application that enables the user to draw basic shapes with different colors. In addition, the user should have the ability to clear all the drawings or erase some parts of them



- Apply OOP rules
- You can use: MouseListener, MouseMotionListener, ItemListener and ActionListenser
- For storing use: ArrayList