Agenda

- Reflection
- File IO
- JDBC

IO Framework

- Java IO related classes are in java.io package.
- File = Metadata (inode + directory entry) + Data (data blocks)
- These classes are related to
    - File System operations
        - Related to inode & dentry (Metadata).
        - classes: File, Path, ...
    - IO operations
        - Related to data blocks (Data).
        - IO streams
            - Byte streams
                - classes: InputStream, OutputStream and classes inherited from them.
            - Character streams
                - classes: Reader, Writer and classes inherited from them.

File system operations

- A file or directory is represented by java.io.File class.
- Important methods:
    - File(String path);
    - String getName();
    - String getParent();
    - String getPath();
    - boolean exists();
    - boolean isDirectory();
    - boolean isFile();
    - boolean canRead();

- boolean canWrite();
- boolean canExecute();
- boolean setReadable(boolean permission);
- boolean setWritable(boolean permission);
- boolean setExecutuable(boolean permission);
- String[] list();
- File[] listFiles();
- long length();
- boolean mkdir();
- boolean createNewFile();
- renameTo()
- delete()

IO operations

- In java IO framework IO is done using IO streams.
- Streams can be source (from where data can be read) or sink (from where data is written).
- Stream is stream of data -- bytes or characters.
  - Byte streams
    - java.io.InputStream (source)
    - java.io.OutputStream (sink)
  - Character streams
    - java.io.Reader (source)
    - java.io.Writer (sink)

Binary Streams

InputStream class

- It is abstract class.
- For reading from byte stream.
- Methods:
  - abstract int read(); -- read a byte and return it.
    - it returns -1 on end-of-file.
  - int read(byte[] data); -- read multiple bytes into array
  - void close(); -- from Closeable interface.

InputStream hierarchy

```
InputStream
    |- FileInputStream
    |- ObjectInputStream
    |- FilterInputStream
        |- DataInputStream
        |- BufferedInputStream
```

- FileInputStream -- read bytes from file.
- FilterInputStream -- abstract that can perform some operation while reading data from underlying stream.
- BufferedInputStream --- read the data into a memory buffer, so that we need to access underlying stream often (to improve the performance).
- DataInputStream -- read bytes from underlying stream and convert into java primitive type.
- ObjectInputStream -- read bytes from underlying stream and convert into java objects (deserialization).

OutputStream class

- It is abstract class.
- For writing into byte stream.
- Methods:
    - abstract void write(int b); -- write a byte.
    - void write(byte[] data); -- write multiple bytes from array into stream.
    - void flush(); -- force all data to be written on stream.
    - void close(); -- from Closeable interface.

OutputStream hierarchy

```
OutputStream
    |- FileOutputStream
    |- ObjectOutputStream
    |- FilterOutputStream
        |- DataOutputStream
        |- BufferedOutputStream
        |- PrintStream
```

- FileOutputStream -- write bytes into the file.
- FilterOutputStream -- abstract class that does some processing before writing data into underlying stream.
- BufferedOutputStream -- stores bytes into a memory buffer before writing into underlying stream to improve performance.
- DataOutputStream -- convert java primitive types into bytes before writing into underlying stream.
- PrintStream -- format data in textual form before writing into underlying stream.
- ObjectOutputStream -- convert java object into bytes before writing into underlying stream - serialization.

Serialization

- It is possible to convert whole java object into sequence of bytes and then to write into underlying stream. This process is called as "serialization".

- The reverse process is to reconstruct java object from sequence of bytes, is called as "deserialization".
- Serialization is done using ObjectOutputStream class. It convert data as well as metadata into sequence of bytes.
- Deserialization is done using ObjectInputStream class. It read bytes - and then reconstruct the java object using metadata. It internally use reflection.
- To serialize any object, the class must be inherited from Serializable (marker) interface. If not, exception will be thrown from objectOutputStream.writeObject() method.
- To deserizize the object, the class must be param-less constructor (default or user-defined). If not, exception will be thrown from objectInputStream.readObject() method.
- It is recommended that serializable class should have a "long" field "serialVersionUID" that keep version of serialized class. During deserialization this version is cross-checked and if mismatched, exception is thrown. This is useful for making some product (sw) and maintaining its multiple versions.

Character Streams

- For reading or writing text (characters) data.
- These streams handles characters encoding (e.g. ASCII, UTF-16 LE, UTF-16 BE, ...) implicitly. They convert char into bytes (by Writer) and vice-versa (by Reader).

Reader class

- It is abstract class.
- For reading from char stream.
- Methods:
    - abstract int read(); -- read a char and return it.
    - int read(char[] data); -- read multiple chars into array
    - void close(); -- from Closeable interface.

Writer class

- It is abstract class.
- For writing into char stream.
- Methods:
    - abstract void write(int b); -- write a char.
    - void write(char[] data); -- write multiple char from array into stream.
    - void flush(); -- force all data to be written on stream.
    - void close(); -- from Closeable interface.

Reader hierarchy

```
Reader
    |- InputStreamReader
    |    |- FileReader
    |- BufferedReader
    |- FilterReader
```

- InputStreamReader reads from InputStream (byte stream).

Writer hierarchy

```
Writer
    |- OutputStreamWriter
    |   |- FileWriter
    |- BufferedWriter
    |- FilterWriter
    |- PrintWriter
```

- OutputStreamWriter writes into OutputStream (byte stream).