Day 12

Binary File

```
- If we want to write data in binary file then we should use OutputStream
and its sub classes.
- If we want to read data in binary file then we should use InputStream
and its sub classes.
- FileOutputStream instance is used to write single byte at a time inside
file. Which may degrade performance.
- FileInputStream instance is used to read single byte at a time from
file. Which may degrade performance.
- Filter stream is a stream which provides service to the underlying
stream( Console, Network, File ).
- BufferedOutputStream is a filter stream which provides buffer support
for underlying Stream. Default size of buffer is 8KB. We can change it as
per requirement.
- If we want to improve speed of write operation then we should use
BufferedOutputStream.
- BufferedInputStream is a filter stream which provides buffer support for
underlying Stream. Default size of buffer is 8KB. We can change it as per
requirement.
- If we want to improve speed of read operation then we should use
BufferedInputStream.
- DataOutputStream is a filter stream that is used to convert state of
primitive values into binary data.
- DataInputStream is a filter stream that is used to convert binary data
into primitive values.
- In case of DataOutputStream and DataInputStream, order of read/write
operation must be same.
- If we want to overcome, limitations of DataOutputStream and
DataInputStream then we should use ObjectOutputStream and
ObjectInputStream.
- ObjectOutput is a interface
        - "void writeObject(Object obj)throws IOException"
- ObjectOutputStream implements ObjectOutput I/F
- If we want to convert state of java instance ( reference type ) into
binary data then we should use ObjectOutputStream instance.
- Process of converting state of java instance into binary data is called
serialization.
- Use Of serialization:
        1. We can save state of java instance inside binary file.
        2. We can do marshalling
        - Marshalling is a process of sending state of instance from one
end of the network.
- For serialization,we shoud use writeObject() method of
ObjectOutputStream.
- if we want to serialize state of java instance then its type must
```

```
Serializable interface.
- Wihtout implementing Serializable interface, if we try to serialize java
instance then writeObject method throws NotSerializableException.
- Serializable is a marker interface declared in java.io package.
- If we want to serialize state of java instance then type of containing
object must also implement Serializable interface.
```

```
class Date implements Serializable
{       }
class Address implements Serializable
{       }
class Person implements Serializable
{
        String name;
        Date birthDate;
        Address currentAddress.
}
```

```
- ObjectInput is a interface declared in java.io package
        - "Object readObject() throws ClassNotFoundException,
IOException"
- ObjectInputStream class implemets ObjectInput interface.
- If we want to convert binary data into java instance then we should use
instance of ObjectInputStream.
- Process of converting binary data into java instance is called
deserialization.
- Use Of deserialization:
        1. To read state of java instance from binary file
        2. To do unmarshalling
- transient is a keyword in java. If we dont want to serialize state of
java field then we should declare it transient.
- JVM do not serialize state of
        1. transient field
        2. static field
-  "serialVersionUID" is considered as private static final long field of
a class. If we want to ensure that same class file is used for
serialization and deserialization then we should use serialVersionUID.
```

Text File

```
- If we want to write text inside file then we should use Writer and its
sub classes.
- If we want to read text from file then we should use Reader and its sub
classes.
- If we want to write single character at a time inside file then we
```

```
should use FileWriter instance.
- If we want to read single character at a time from file then we should
use FileReader instance.
```

Socket Programming

```
- Server :
        - It is a program which handles client's request.
        - IIS, Apache Tomcat, Sun Glassfish, Weblogic etc.
        - machine, on which we install server program is called server
machine.
- Client :
        - It is a program which consumes service provided by the server.
        - Smart Client  :       Mobile App
        - Thin Client   :       Web Browser
        - Thick Client  :       Desktop Application
- URL
        - Uniform Resource Locator
        - If we want to access resources on interanet / internet then we
should use URL
        - protocol://hostname:port/ApplicationName/ResourceName
        - http://www.sunbeaminfo.com:8080/MIS/Index.html
- Protocol
        - Set of rules that client and server must follow is called
protocol.
        - telent, udp, tcp, ftp, http, https etc.
- HostName:
        - Machine on which we deploy/install application.
        - www.sunbeaminfo.com
        - With the help of DNS hostname is mapped to IP-Address.
        - IP-Address is considered as unique identity of computer in
network.
- Port:
        - It is a logical integer number that is used to indentify process
running on server machine.
        - Reserved Port : 0-1024
- Path Name
        - /MIS/Index.html
        - MIS -> Applcation Name
        - Index.html -> Resource
- Socket is a physical memory that is allocated at client and server side
which is used to send and receive data.
- Socket is non java resource / unmanaged resource.
- If we want to do network/socket programming then we should use types
declared in java.net package.
- Types declared in java.net package:
        1. InetAddress
        2. Socket
        3. ServerSocket
```

```
        4. DatagramPacket
        5. DatagramSocket
- Transmission Mode:
        1. Simplex
        2. Half Duplex
        3. Full Duplex
- If we want to do inter process communication then we should use socket
programming.
- Using TCP protocol, if we want to create server socket then we should
use java.net.ServerSocket class.
- "accept()" is a method of ServerSocket which used to accept incomming
request. It is a blocking call.
- Using TCP protocol, if we want to create client socket then we should
use java.net.Socket class.
- Using UDP protocol, If we want to client and server socket then we
should use java.net.DatagramSocket class.
DatagramSocket clientSocket = new DatagramSocket();
DatagramSocket serverSocket = new DatagramSocket( 2025 );
- If we want to send or receive data then we should use
java.net.DatagramPacket class
- Methods of DatagramSocket:
        1. public void send(DatagramPacket p) throws IOException
        2. public void receive(DatagramPacket p) throws IOException

- Constructor's of DatagramPacket:
        1. public DatagramPacket(byte[] buf, int length);
        2. public DatagramPacket(byte[] buf, int length,
        InetAddress address, int port)
```

```
public static void main(String[] args)
{
        byte[] bs = { 83, 117, 110, 66, 101, 97, 109 };
        String str = new String(bs);
        System.out.println(str);
}
public static void main1(String[] args)
{
        String str = "SunBeam";
        byte[] bs = str.getBytes();
        System.out.println(Arrays.toString( bs ) );
}
```

Reflection

```
- Data about data or data which describe other data is called metadata.
```

Metadata of interface

```
- What is name of the interface
- Which is super interface of interface
- Which anotations are used on interface
- Which is access modifier of interface
- Which members are declared inside interface
- In which package it is declared
```

class of interface

```
- What is name of the class
- In which package it is declared
- Which is access modifier of the class
- Which is super class of class
- Which interfaces it has implemented
- Which annotations has been used on class
- Which are members declared inside class
```

Metadata of Field

```
- What is name of field
- Which is access modifier of field
- Whether filed is static, final, transient.
- Which anotations has been used on field
- Whether field is inherited or declared only.
```

Metadata of Method

```
- What is name of method
- Whether method is inherited, declared only or overriden
- Which is access modifier of method
- Whether method is static, final, abstract, synchronized
- which is return type of method
- What is the parameter information
- Which exceptions it throws
```

Advantages of metadata

```
- After compilation, java compiler generates bytecode and metadata. Due to
metadata, there is no need to include header files in java.
- To display class information in intellisense window, IDE implicitly use
metadata.
- Metadata helps JVM to serialize / to clone the object.
- To keep track of life time of object, GC implicitly use metadata.
```

If we want process metadata then we should use reflection. "Reflection" is java language feature that provides types which is used to process metadata. - If we want to use reflection the we should use types declared in java.lang and java.lang.reflect package. 1. java.lang.Class 2. java.lang.reflect.Constructor 3. java.lang.reflect.Field 2. java.lang.reflect.Method 3. java.lang.reflect.Parameter 2. java.lang.reflect.Modifier 3. java.lang.reflect.Array

Application of reflection

```
- To display type metadata, javap tool implicitly use reflection
- To display type metadata in intellisense window, IDE implicitly use
reflection
- To access value of private fields, debugger implicitly use reflection
- To manage behavior of application at runtime, reflection is used.
```

Class class

```
- It is a final class declared in java.lang package.
- Instances of the java.lang.Class<T> represent classes and interfaces in
a running Java application.
- Class has no public constructor. Instead java.lang.Class objects are
constructed automatically by the JVM.
- If we want to process metadata of any type then it is mandatory to get
reference of Class<T> class instance associated with it.
```

How to get reference of java.lang.Class instance?

```
1. Using "public final Class<?> getClass()" method.
        Integer n1 = new Integer(125);
        Class<?> c = n1.getClass();
2. Using ".class" syntax
        Class<?> c  = Number.class;
3. Using Class.forName() method.
        System.out.print("F.Q.Class Name");
        String className = sc.nextLine();       //java.util.ArrayList
        Class<?> c = Class.forName(className);
```