Java Learnings Day 1

Object-Oriented Programming

Class and Object

In Java, a class is said to be the blueprint of an object. It specifies what can be part of an object.

The object is the real-time entity for the blueprint, that is instance of the class

Encapsulation

The encapsulation refers to an object should be consumed as a single entity. It's properties can't be accessed and modified individually

To achieve encapsulation,

- Make all the properties of a class private
- Provide necessary getters and setters for the class

Inheritance

The inheritance refers to the extending from a base class to a specialized class by allowing it to inherit parent class' attributes and methods.

The inheritance in java can be achieved using the extends keyword. The super keyword can be used to access the parent's attributes and methods

The super keyword can be used to call parent class'

- Constructor -> super(<arguments if needed>) [Can only call in first line of constructor]
- Methods -> super.methodName(<arguments>)
- Attributes -> super.attribute

only if they are public and protected. If the child class is in the same package as the parent class, default can also be accessed.

Polymorphism

The polymorphism refers to the different behaviors according to the situation for the same action.

In Java, the polymorphism is the concept of having different method bodies for the same method. There are two types of polymorphism

- Compile-time polymorphism [Method Overloading]
- Run-time polymorphism [Method Overriding]

Method Overloading refers to the same method name, but a difference in return type or arguments or both in the same class.

Method Overriding can be referred as the same method name, with same return type and arguments in the classes having Is-a relationship i.e Inheritance.

The static binding refers to the concept of having the same reference to the same object, while dynamic binding is setting the parent reference to the Child object.

Covariant return type is for the return type of the parent can be substituted with the return type of the child. [Eg: The List return type can be substituted with ArrayList, still it is considered as overridden]

Abstraction

The abstraction is the process of showing only the necessary information by hiding the implementation details from the users.

In java, Abstraction can be achieved in two ways:

- Abstract classes
- Interface [Also Helps in multiple inheritance]

The abstract class can contain both abstract and normal methods. Whereas in interface, all the methods in interface are considered to be abstract and public by default.

The abstract class cannot be instantiated. So for a class to that inherits another abstract class, it should override all the abstract methods, or it also should be marked as an abstract class.

The interface can be inherited using the implements keyword.

In Java 8, default and static methods in the interface are introduced to provide non abstract methods in the interface.

There are two types of specialized interfaces available,

- Tagged interface [Interface with 0 abstract methods]
- Functional interface [Interface with exactly 1 abstract method]

File Operations

Java I/0

In the java.io package is responsible for the purpose of file handling. The IO stands for Input and Output.

The file handling and IO operations can be done using the means of Streams. The Streams can be defined as the sequence data. The streams are of two types

- Byte Streams (InputStream/OutputStream)
- Character Streams (Reader/Writer)

The Byte Stream is used to read data as bytes, while the character stream can be used for reading characters.

The ByteStreams are categorized into InputStream and OutputStream, the InputStream is used to read data and OutputStream is used to write data as bytes.

The CharacterStreams are broadly classified as Reader and Writer, the Reader is to read and Writer is to write data as characters.

FileInputStream

The fileInputStream is used for reading data from the existing file. It reads data as bytes. To work with character data, we can go with FileReader.

Constructors:

- FileInputStream(String fileName)
- FileInputStream(File file)
- FileInputStream(FileDescriptor fdObj)

Methods:

- read()
- read(byte[] byte)

- read(byte[] byte, int offset, int len)
- skip(long len)
- getFD()

FileOutputStream

The fileOutputStream is used for writing data to the file. It write byte data into the file. To work with character data, we can go with FileWriter.

Constructors:

- FileOutputStream(String fileName)
- FileOutputStream(File file)
- FileOutputStream(FileDescriptor fdObj)

Methods:

- write()
- write(byte[] byte)
- write(byte[] byte, int offset, int len)
- skip(long len)
- getFD()

Apart from FileInputStream and FileOutputStream, there are multiple classes available for working with Input Output. Some of them are,

- Buffered<Input/Output>Stream
- Data<Input/Output>Stream
- SequenceInputStream
- PrintStream
- Object<Input/Output>Stream

Some of the character stream classes are,

- FileReader, FileWriter
- BufferedReader, BufferedWriter
- PipedReader, PipedWriter
- FilterReader, FilterWriter

The File class can be used to get metadata of a file. It is not only used for files but also for directories. It provides an abstract information of files and directories.

Protocol Buffer

The workspace of protocol buffer: Visit here