# File Read and Write using java

## Streams

- Stream is a path of communication between a source of some information and destination
- The source can be a file, computers memory or the Internet
- Input Streams allows you to read data from a source
- output streams allows you to write data to a destination
- Java programs perform I/O through streams.

- Java defines two types of streams:
    1. Byte Stream
    2. Character Stream

## Byte streams

- Byte streams provide a convenient means for handling input and output of bytes.
- Byte streams are used when reading or writing binary data.

## Character streams

- Character streams provide a convenient means for handling input and output of characters.
- They are Unicode and therefore, can be internationalized .
- In some cases character streams are more efficient than byte streams.
- We use character streams to read and write files in java.
- FileReader and FileWrite are two such character streams

    eg: FileReader fileRdr = new FileReader(“input.txt”);

- At the end of the program we need to close the opened streams.

## Buffered Stream

- Streams like FileInputStream, FileReader access the resource (disk file, client socket) for each read and write request.
- Buffered streams do read and write operations through a memory area known as “buffer” without connecting to the actual resource directly.
- We can create Buffered stream by wrapping non buffered stream object using buffered stream object.

    Eg:

    //for input operations

```
FileReader fileRdr=new FileReader("input.txt");
BufferedReader buffRdr=newBufferedReader(fileRdr);
```

```
//for output operations
        FileWriter fileWrtr=new FileWriter("output.txt");

        BufferedWriter buffWrtr=new BufferedWriter(fileWrtr);
```

## Creating Directories in Java

- A directory is a File which can contains a list of other files and directories.
- Use File object to create directories, to list down files available in a directory.
- The mkdir method creates a directory, returning true on success and false on failure.

## Listing Directories

- You can use list method provided by File object to list down all the files and directories available in a directory.

# Java Static Members

- There are two static members
  - Class Variables
  - Class Methods

## Class Variables

- For class variables it allocates only one memory location for any number of objects created for that class.

- Imagine Employee class has a class variable known as empID. For any number of objects created using Employee class it declares only one memory location in RAM and shared among all the objects.

## Class Methods

- We use static methods to invoke methods without creating objects of a particular class.

- We invoke (call ) these static methods via class name.

# Method Overloading

- The return Type, Method Name, and the Parameter list defines the Signature of the method.

- It is possible to define two or more methods with the same name within the same class (Method Overloading) with different signatures.

Eg:

```
public void CreatePoint( )

public void CreatePoint(int x , y)
```

# Overloading Constructors

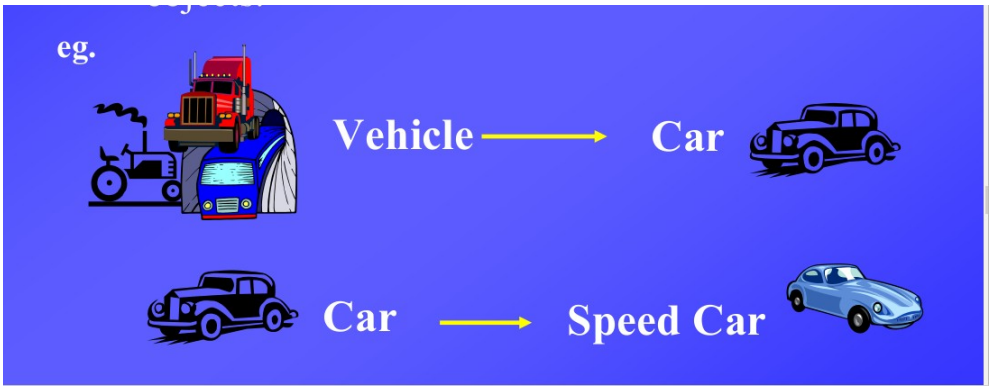
- A class can have multiple Constructors (Overloaded Constructors)

- All carries the same name

- They have either different number of arguments or different types of arguments

# Inheritance

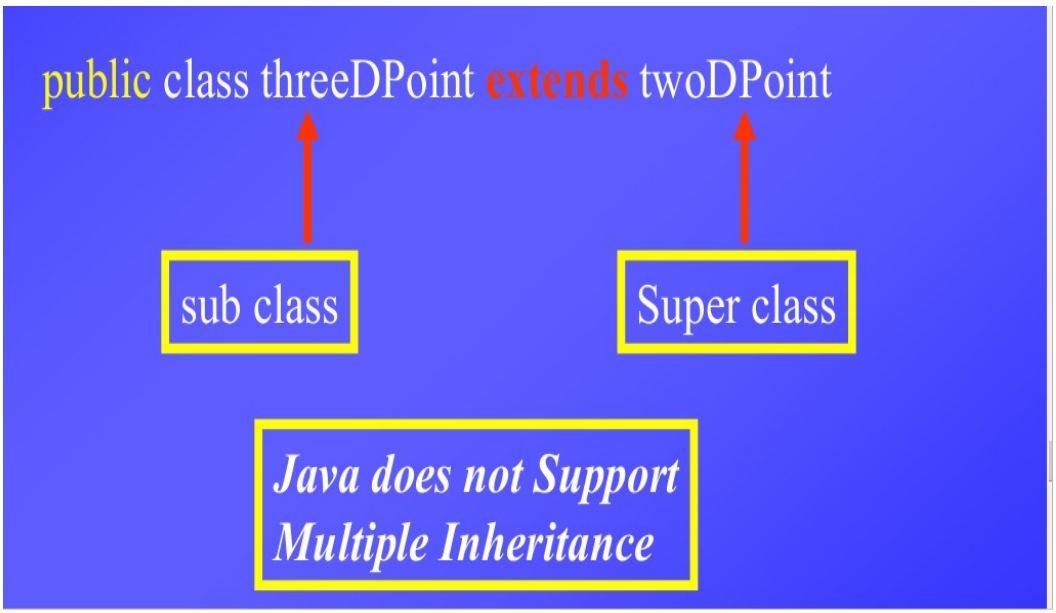
- Concept of one class of objects inheriting the data and behavior from another class of objects.




- Inheritance also helps you to reuse existing code from one or more classes by simply deriving a new class from them.

- Inheritance is the creation of one class by extending another class so that instances of the new class automatically inherit the fields and methods of its parent class.

- In Java the extends Keyword is used to achieve Inheritance




- A sub class can have its own implementation of the Super class methods.(Method

Overriding)

# Polymorphism

- In the context of Object Oriented Programming, polymorphism refers to the fact  a single operation can have different behavior in different objects.

- There are two common ways of implementing Polymorphism. Overloading, Overriding

- Overloading - Using the same method name with different parameter type lists .

- Overriding - Using different implementations of the same method in sub classes.