File IO

YAMAN OMAR
ALASHQAR

You have a 3 gallon jug and 5 gallon jug, how do you measure out exactly 4 gallons?

Missing Numbers

$$\begin{array}{ccc} \square & 7 & 2 \\ 3 & \square & 8 \\ \hline 4 & 7 & \square \\ \hline \end{array}$$

Creating Files

```java
File myObj = new File(path);
boolean isCreated = myObj.createNewFile();
```

To create a file in a specific directory, specify the path of the file and use double backslashes to escape the “\" character

>>> "C:\\Users\\MyName\\filename.txt"

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

Write To a File

```
FileWriter myWriter = new FileWriter(filePath);
myWriter.write("STRING TO BE WRITTEN\nLINE2");
myWriter.close();
```

Read Files

```
File myObj = new File(filePath);
Scanner myReader = new Scanner(myObj);
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
myReader.close();
```

Exercise

- Write a program to find the longest word in a file.
- Write a program to count the number of lines in a text file
- Write a program to count the frequency of words in a file.
- Write a program to copy the contents of a file to another file
- Write a program to generate 26 text files named A.txt, B.txt, and so on up to Z.txt
- Combine each line from first file with the corresponding line in second file
- Write a program to count words in a text file those are ending with alphabet "e".

Java BufferedReader Class

- Java BufferedReader class is used to read the text from a character-based input stream.
- It can be used to read data line by line by readLine() method.

Method	Description
String readLine()	Reading a line of text.
int read()	Reading a single character.
boolean ready()	Test whether the input stream is ready to be read.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readAheadLimit)	It is used for marking the present position in a stream.

Java BufferedReader Example (File)

```
FileReader fr=new FileReader("testout.txt");
BufferedReader br=new BufferedReader(fr);


int i;
while((i=br.read())!=-1){
System.out.print((char)i);
}
br.close();
fr.close();
```

Java BufferedReader Example (Console)

```
InputStreamReader r=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(r);
System.out.println("Enter your name");
String name=br.readLine();
System.out.println("Welcome "+name);
```

Difference Between Scanner and BufferedReader

- Scanner and BuffredReader class are sources that serve as ways of reading inputs.
- Scanner class is a simple text scanner that can parse primitive types and strings.
  - It internally uses regular expressions to read different types
- BufferedReader class reads text from a character-input stream, buffering characters so as to provide for the efficient reading of the sequence of characters

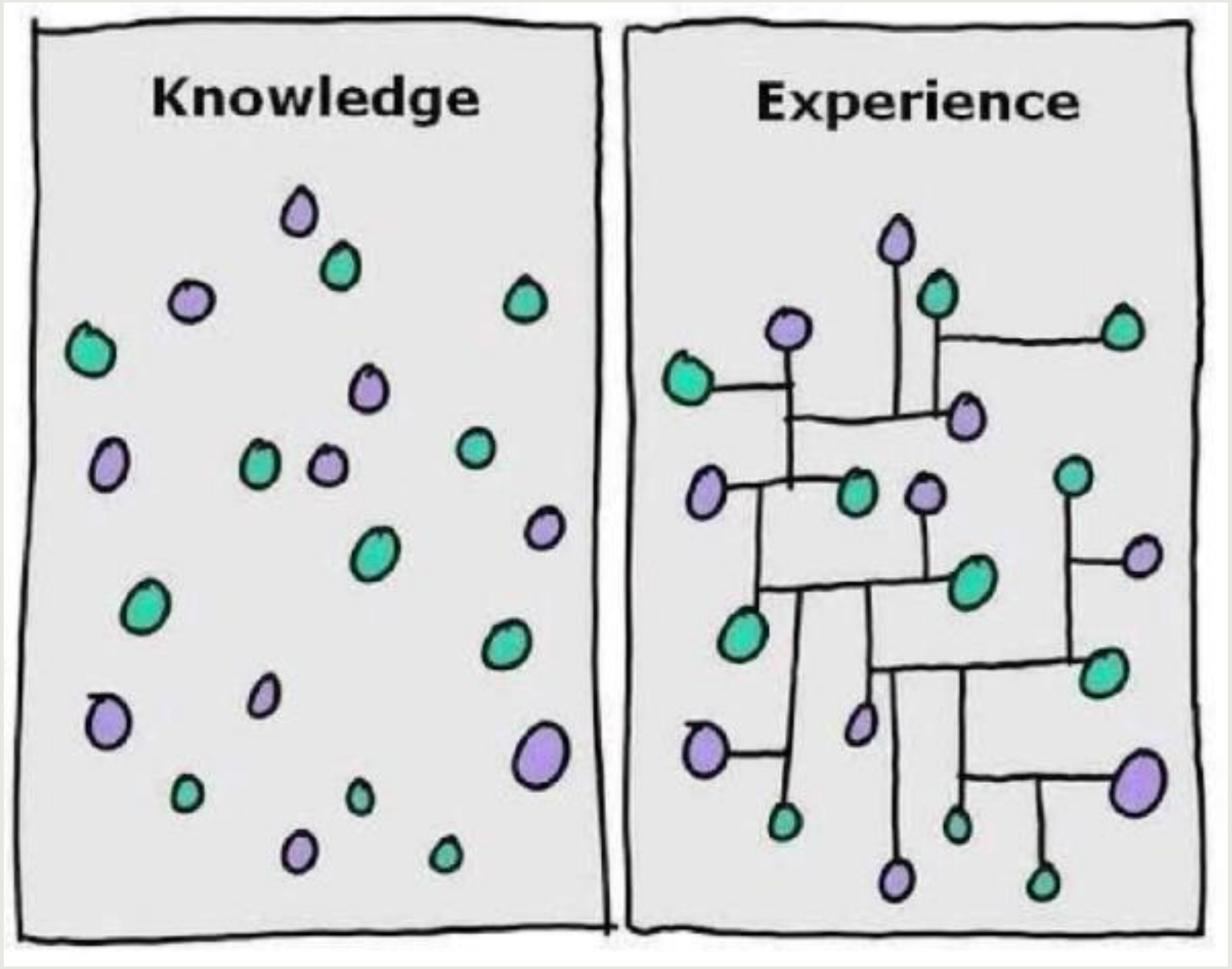
Java BufferedWriter Class

- Java BufferedWriter class is used to provide buffering for Writer instances.

Method	Description
void newLine()	It is used to add a new line by writing a line separator.
void write(int c)	It is used to write a single character.
void write(char[] cbuf, int off, int len)	It is used to write a portion of an array of characters.
void write(String s, int off, int len)	It is used to write a portion of a string.
void flush()	It is used to flushes the input stream.

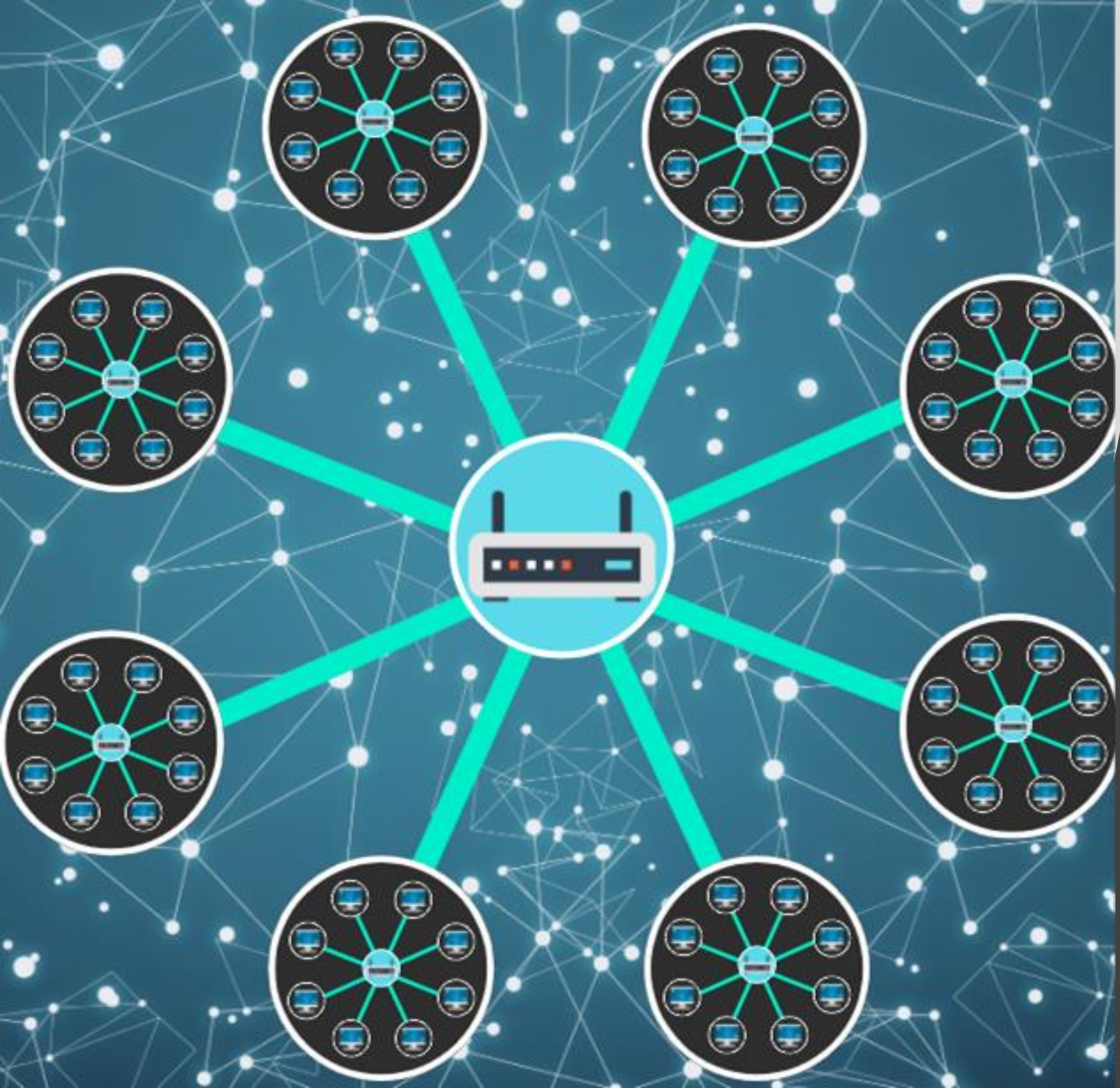
Java BufferedWriter Example (File)

```
FileWriter writer = new FileWriter("testout.txt");
BufferedWriter buffer = new BufferedWriter(writer);
buffer.write("Welcome to javaTpoint.");
buffer.close();
```


Knowledge
Experience

The Internet

A global network of multiple networks

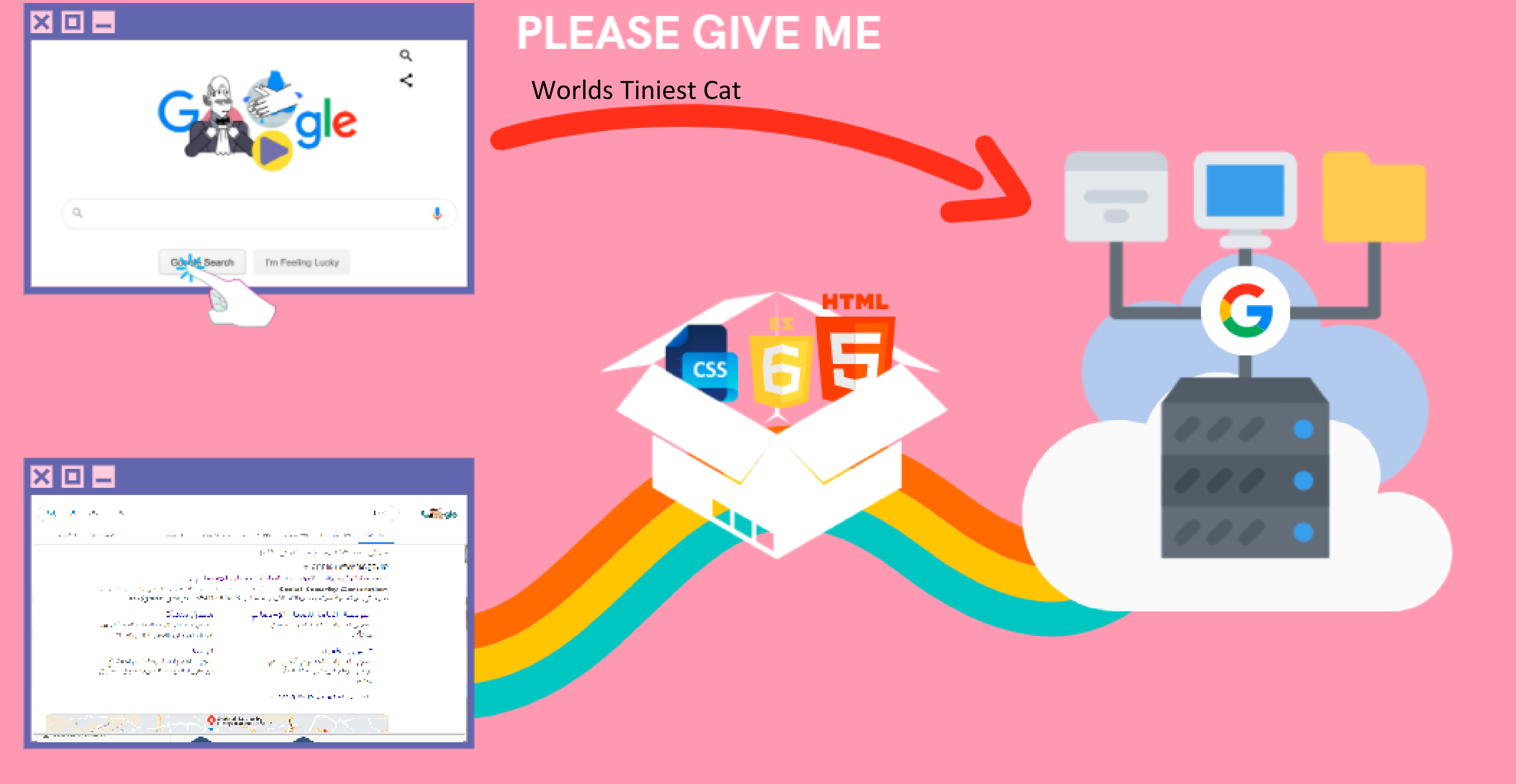


It is that infrastructure that lets you order from an online shop, share your life on social media, watch videos on streaming services, email your aunt in another country and search the web for the world’s tiniest cat.

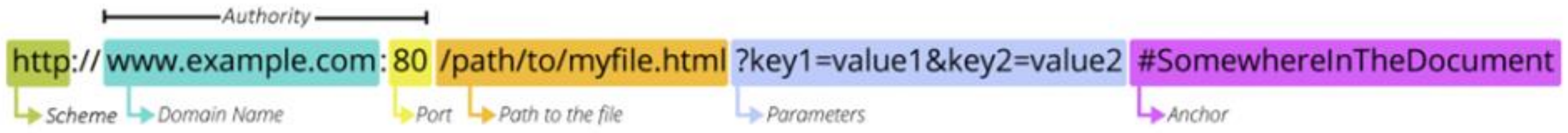
Concepts

Web	Server	Client
THE WORLD WIDE WEB IS AN INFORMATION SYSTEM WHERE RESOURCES ARE AVAILABLE OVER THE INTERNET	A COMPUTER THAT CAN SATISFY REQUESTS ON THE WEB AND STORE WEBPAGE	The computer that accesses a server

PLEASE GIVE ME



A URL is the address of a given unique resource on the Web.



- The scheme, indicates the protocol that the browser must use to request the resource.

- The authority, which includes both the domain and the port, separated by a colon
  - The domain indicates which Web server is being requested
  - The port indicates the gate used to access the resources on the web server.

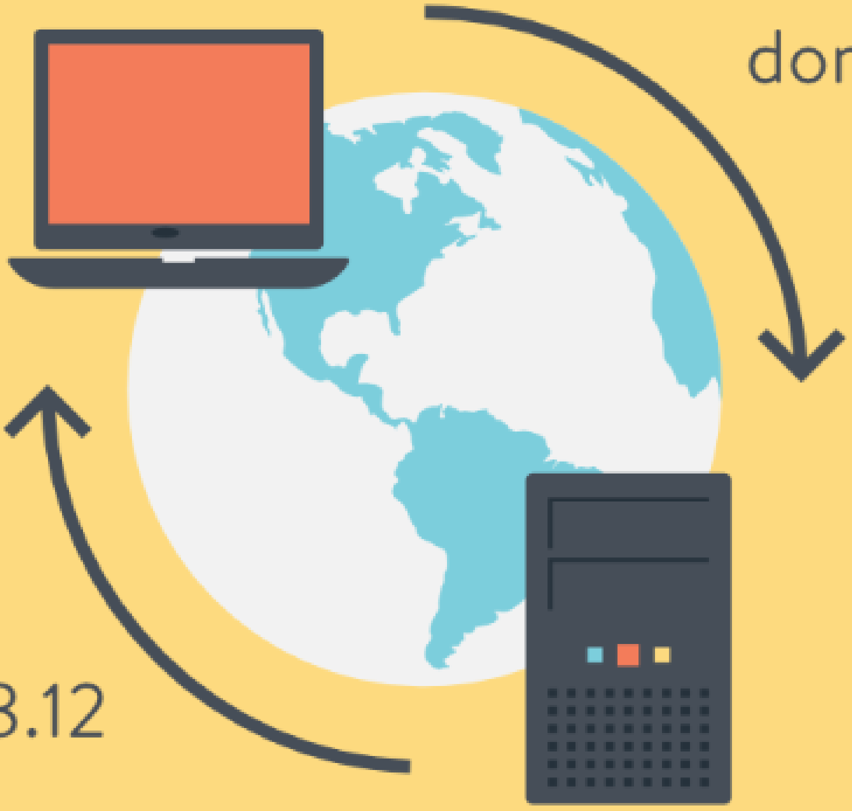
- Path to the resource on the Web server

- Parameters are a list of key/value pairs separated with the & symbol

- Anchor to another part of the resource itself

DNS – Domain Name System

How is the domain "domain.com" translated to its IP address?



There's a special type of server called "name server" or "**DNS server**" (where DNS = "Domain Name System") that translates domains to IP addresses.

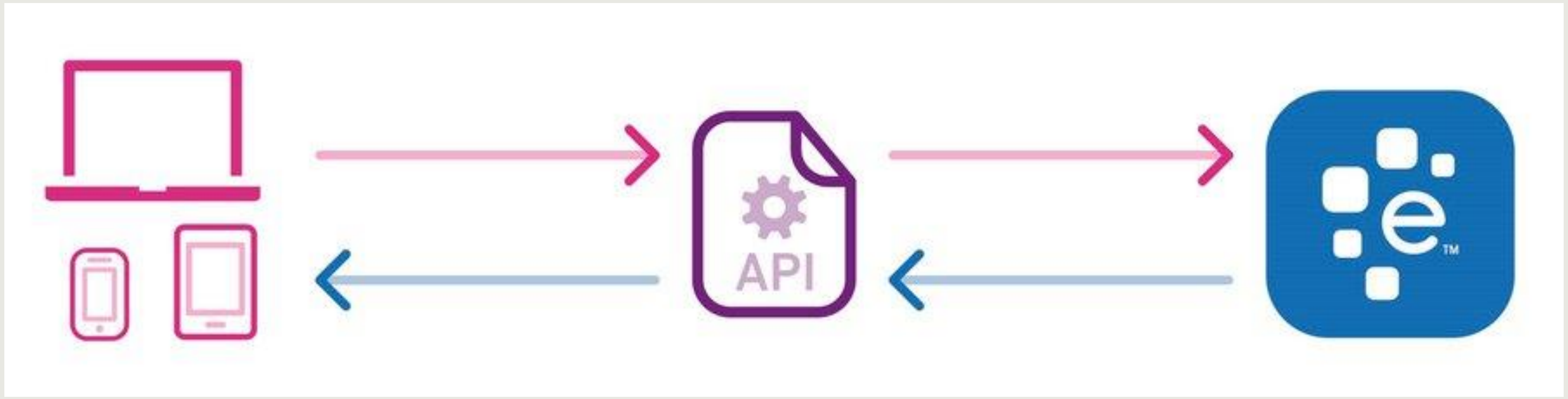
- HTTP REQUESTS
  - The Domain and the path of the resource to fetch
  - GET / POST / PUT / DELETE
    - With a optional header or body to attach some information

- HTTP RESPONSE
  - A status code (success 200 / or failure (many types))
  - Headers / Body containing the fetched resource

- The browser will parse the received response.

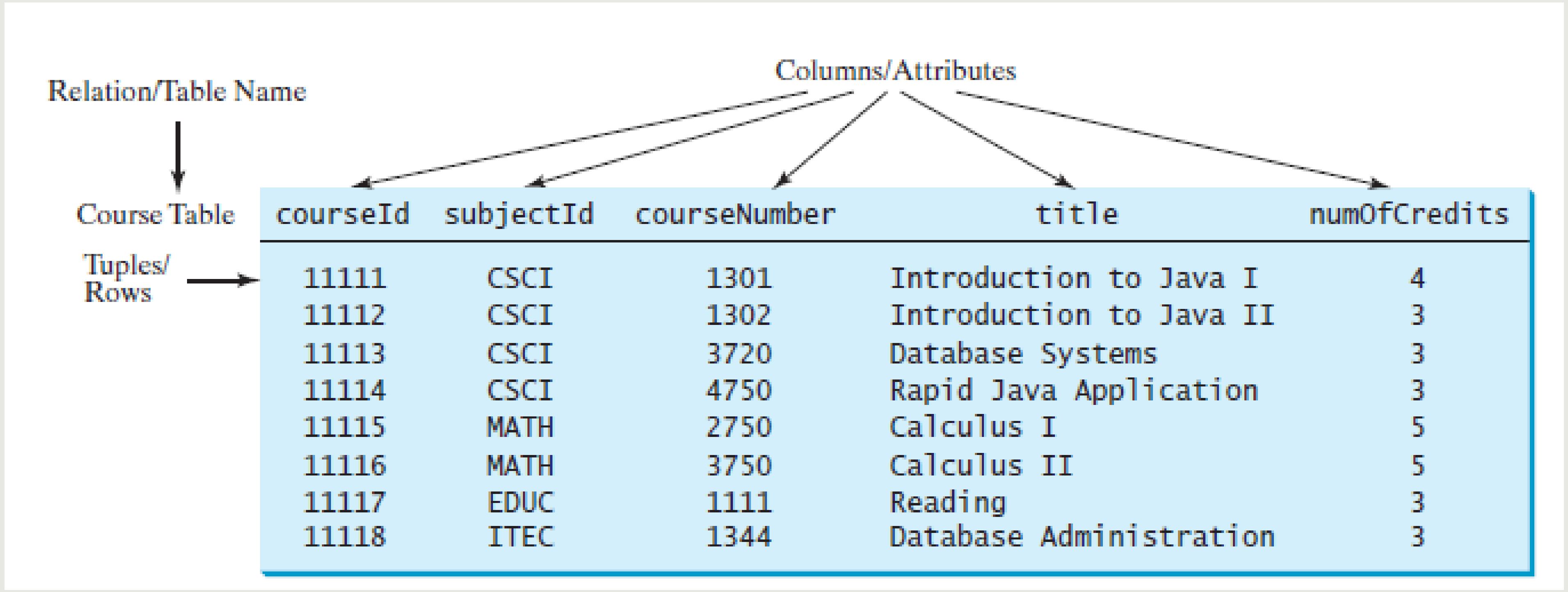
- Now we have technologies that deal with the client side.
  - Displaying webpages and data
  - Sending requests
  - HTML / CSS / JS
- And others that deal with server side.
  - Handling requests
  - Handling data and business logic
  - Sending back a response
  - JAVA / NODE JS / PYTHON / PHP

API



DB

- A row of a table represents a record
- A column of a table represents the value of a single attribute of the record.



courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
11114	CSCI	4750	Rapid Java Application	3
11115	MATH	2750	Calculus I	5
11116	MATH	3750	Calculus II	5
11117	EDUC	1111	Reading	3
11118	ITEC	1344	Database Administration	3

Creating Tables

```
create table Course (
courseId char(5),
subjectId char(4) not null,
courseNumber integer,
title varchar(50) not null,
numOfCredits integer,
primary key (courseId)
);
```

Simple Insert, Update, and Delete

- The syntax to insert a record into a table is:

  ```
  insert into tableName [(column1, column2, ..., columN)]
  values (value1, value2, ..., valueN);
  ```

-
  ```
  insert into Course (courseId, subjectId, courseNumber, title, numOfCredits)
  values ('11113', 'CSCI', '3720', 'Database Systems', 3);
  ```

Simple Insert, Update, and Delete

- update tableName
  set column1 = newValue1 [, column2 = newValue2, ...]
  [where condition];

-
  ```
  update Course
      set numOfCredits = 4
      where title = 'Database Systems';
  ```

Simple Delete

delete from tableName [where condition];

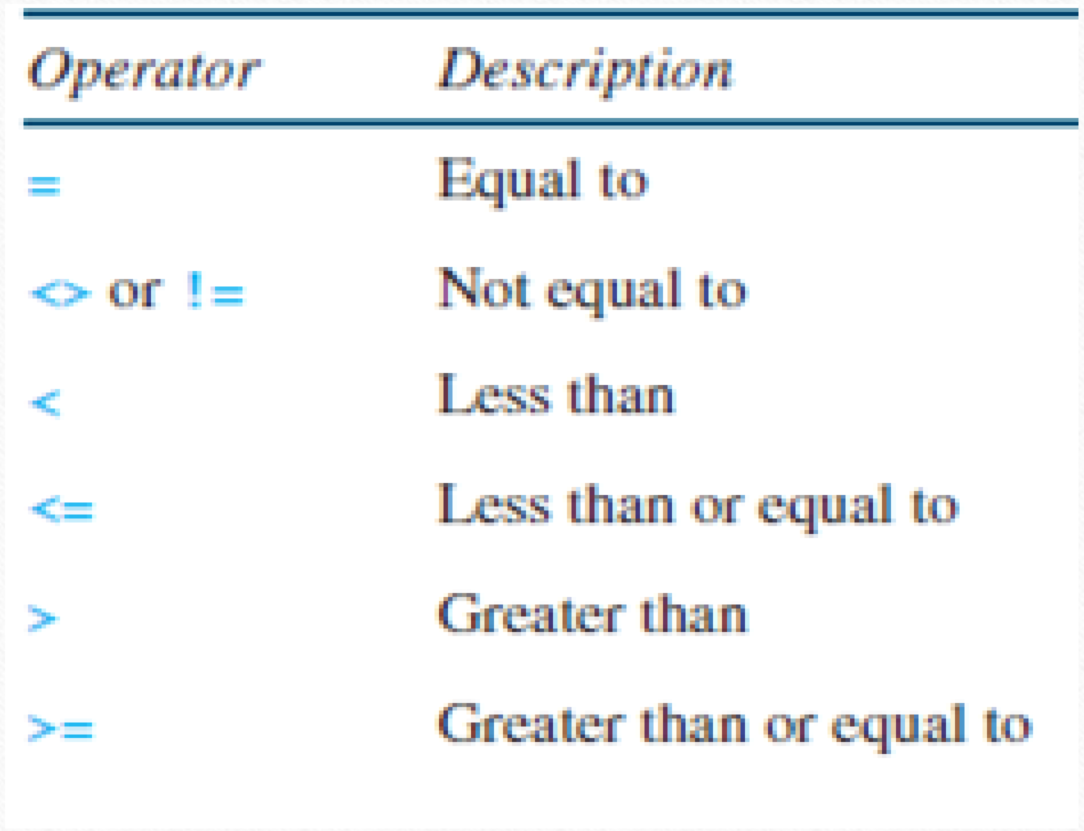
```
delete from Course
         where title = 'Database Systems';
```

Simple Queries

- select column-list
  from table-list
  [where condition];

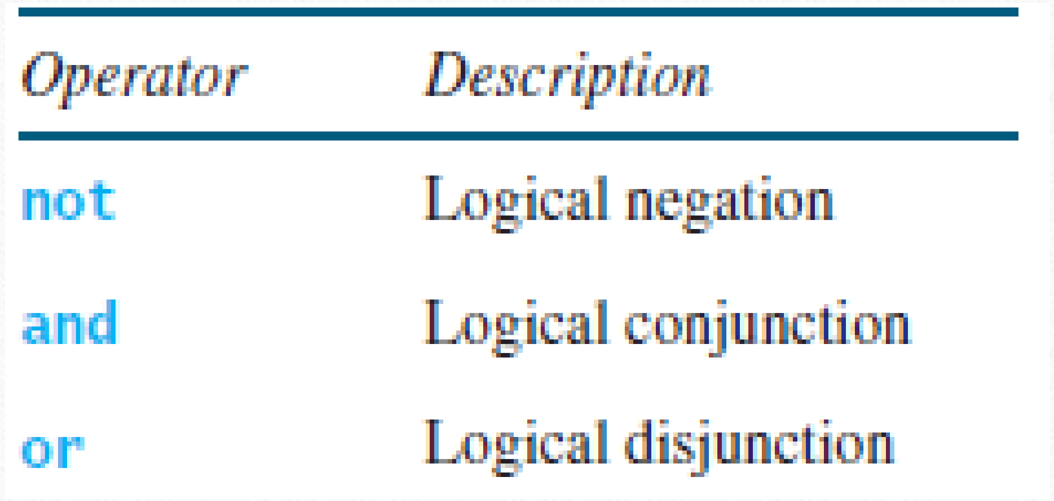
  - select firstName, mi, lastName
    from Student
    where deptId = 'CS';

Comparison Operators



Operator	*Description*
=	Equal to
<> or !=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Boolean Operators



Operator	*Description*
not	Logical negation
and	Logical conjunction
or	Logical disjunction

Example:

Get the names of the students who are in the CS Department and live in the place where the zip code is 31411.

```
select firstName, mi, lastName
from Student
where deptId = 'CS' and zipCode = '31411';
```

Distinct

- select distinct subjectId as "Subject ID" from Course;

Displaying Sorted Tuples

- select column-list
from table-list
[where condition]
[order by columns-to-be-sorted];

- select lastName, firstName, deptId
from Student
where deptId = 'CS'
order by lastName desc, firstName asc;

JDBC

- JDBC is the Java API for accessing relational database.
- JDBC provides Java programmers with a uniform interface for accessing and manipulating relational databases.
- Using the JDBC API, applications written in the Java programming language can execute SQL statements, retrieve results, present data in a user-friendly interface, and propagate changes back to the database

JDBC

- Connect to the database using the Connection interface,

Connection connection = DriverManager.getConnection(databaseURL);

- Create and execute SQL statements using the Statement interface,

Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery("select firstName, lastName from Student where lastName " + " = 'Omar'");

- Processes the result using the ResultSet interface

while (resultSet.next())
System.out.println(resultSet.getString(1) + " " + resultSet.getString(2));