# MVC:

Architectural pattern for implementing user interface.

**Model**: Manages the data of the application. Responds to requests from the view and listens to instructions from the controller.

**View**: Responsible for showing all portions of the data to the viewer. (ASP, PHP, JSP)

**Controller:** Responsible for responding to the user input and performing logic on the data model objects.

# Databases:

### Relational Database

* Means of storing in such a way that information can be retrieved from it.
* The ability to retrieve similar data from a table is the bases for a relational database.
* **Database schema** refers to the organization of data. It is a way of logically grouping objects such as tables, views, and stored procedures.

### NoSQL Database

* Non-relational, distributed and horizontally scalable (increasing capacity by adding resources).
* Its schema free and able to store huge amounts of data.
* Example, **Apache Hadoop** which is a framework that allows distributed processing of large data across clusters.
* **MongoDB**

# OOD Concepts JAVA specific:

## Concurrency

* Two basic units of concurrency **Threads** and **Processes.**
* Java is mostly concerned with **Threads**
* **Processing** time is shared through an OS via time slicing.

Process:

* A self-contained execution environment.
* Each process has its own memory space.
* Most Java applications run on a single process. Additional processes can be created using *ProcessBuilder*.

Threads:

* *Lightweight processes*
* Creation of a new thread requires less resource. The main thread can create more threads.
* Every process has at least one thread. Threads share processes resources.

## Sockets

* Sometimes a program requires a lower level communication.
* Client-server applications: Server provides a service such as processing database queries.
* TCP is used in client-server application, by establishing a connection to one another.
* A program binds a socket to its end of the connection.
* Each client and server writes to and reads from the socket bound to the connection.
* Two classes *Socket* (client side)and *ServerSocket* (server side).

## Inheritance

public class B extends A

In Java classes can be derived from other classes. The derived class is called a *subclass,* and the main class is called the *superclass.* A class is only allowed a single **inheritance**, thus can only have one superclass. All classes implicitly are a subclass of the object class.

## Override

A method with the exact same parameters and return in a *subclass* will ***@Override*** the method in the *superclass*.

## Interface

public interface Bicycle

class ACMEBicycle **implements** Bicycle

A reference type that can only contain, constants, **method signatures**, default methods, static methods and nested types. Interfaces cannot be instantiated, they can only be implemented.

## Polymorphism

Ability of an object to take on many forms. Different types of polymorphism.

**Method Overloading:** Two or more methods with the same name with different arguments.

**Method** **Overriding**: Declaring a method in a subclass which is already present in the parent class.

## Encapsulation

Hiding the variable of a class from other classes. Declare variables as **private** and use **public** *setters* and *getters* to modify and view variables.

## Recursion

A function calling itself.

## Runnable

Type of class that can be put into threads and describes what the thread is supposed to do.

public class MyRunnableTask implements Runnable {

public void run() {

// do stuff here

}

}

Thread t = new Thread(new MyRunnableTask());

t.start();

# JAVA Nitty Gritty:

Import java.util.\*;

<https://www.hackerrank.com/> for practice

## COLLECTION Types Implementation Summary;

* For the Set interface, HashSet is the most commonly used implementation.
  + It is a Collection that, unlike List, does not allow duplicates.
  + HashSet allows at most one null element.
  + HashSet is faster than other implementations of Set, TreeSet and LinkedHashSet.

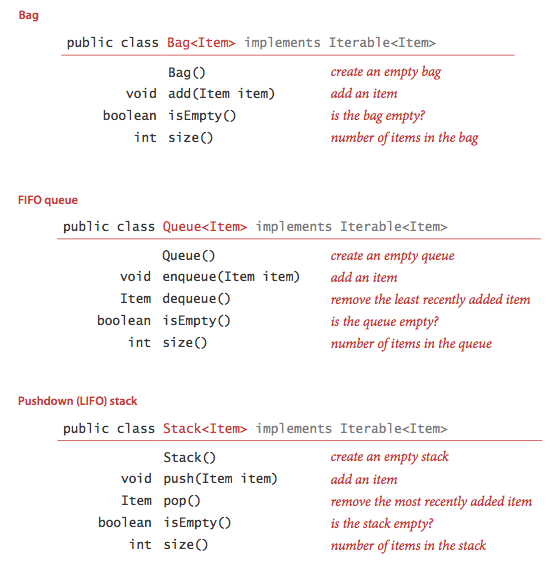
Set<Integer> set = new HashSet<Integer>();

* For the List interface, ArrayList is the most commonly used implementation.

List<MyObject> list = new ArrayList<MyObject>();

* For the Map interface, HashMap is the most commonly used implementation.

Map m1 = new HashMap();

peak/top returns first element without removing it.

## Reading Inputs

**Reading from file**

|  |  |
| --- | --- |
| import java.io.File;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileNotFoundException;  import java.io.IOException; | File file = new File("grid.txt");  BufferedReader reader = new BufferedReader(new FileReader(file));  String tempLine = reader.readLine(); |

**Reading from Console**

|  |  |
| --- | --- |
| import java.util.Scanner; | System.out.println("Enter your username: ");  Scanner scanner = new Scanner(System.in);  String username = scanner.nextLine();  System.out.println("Your username is " + username); |
| Import java util.regex.Matcher;  Import java util.regex.Pattern.  Splitting strings | String[] parts = string.split(Pattern.quote("."));  ([^-]+)-([^-]+) // Each part consists of characters other than - |

## SORTS

<http://www.java2novice.com/java-sorting-algorithms/>

|  |  |
| --- | --- |
| Bubble sort | Worst and average О(n2) |
| Selection sort | Θ(n2) |
| Insertion sort | О(n2) |
| Quick sort | Average Θ(n log(n)) worst Θ(n2) |
| Merge sort | O(n\*log(n)) |

TreeSet sortedSet = new TreeSet<Integer>(set); //can sort a set.

## SEARCHES

|  |  |
| --- | --- |
| Binary Search (Divide and Conquer) |  |
| Binary Search (Recursive) |  |

## GRAPHS

Topological Sort

BFS for finding friendship levels

DFS

## Methods

|  |  |  |
| --- | --- | --- |
| Char | C = "abcdefghijklmn".toCharArray(); | Converts to char arrary |
| all | Arrays.toString(array); | Prints out array with ease |
| Maps | Map.get(“key”);  Maps.put(“key”, data);  **Maps.**[**size**](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html#size())() | Gets element  Puts data in map. |
| Import java.util.LinkedList<E> | add(E e)  [**clone**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#clone())()  [**element**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#element())()  [**remove**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#remove())() | Appends the specified element to the end of this list.  Retrieves, but does not remove, the head (first element) of this list.  Retrieves and removes the head (first element) of this list. |
| Queue | Queue queue = new LinkedList();  queue.add("Java");  queue.remove("Java");  queue.poll();  queue.peek(); | Add to the back, remove from the front  Poll: retrieves and removes the head of this queue  .peek() just returns the current element |
| Stack | Stack st = new Stack() |  |

## Readers

*BufferedReader* is the main tool for inputting external data.

* *InputStreamReader* for reading from console.
* *FileReader* for reading from file!