Java Data Types

- byte/short/int/long
- 2^ (8/ 16/ 32/ 64)
- float [32bit] / double [64bit]
- 1.0f / 1.0d, 1.0
- char [16bit]
- "U", "±"
- String "Hello World"

Inc- / Decrement

- a++
- ++a
- a--
- --a
- ➜ return a++ / a-- ➜ return a

Variables

- int i = 5, j;
- int a = 3, b = a + 1;
- final int c; //=> Constant
- c = 22; // may be init after

Bitwise Shifts

- ~ Complement
- << Shift left
- >> Shift right
- >>> Shift right Zero fill

Naming Convention

- Constants:
- MAX, PI, MIN_TIME
- Variables, Methods, Packages:
- xVal, int1Arr, date, showDate
- Classes:
- Date, DatabaseHelper

Array

- int[] a; //Declaration
- a = new int[5] //Dimensionint[] a = new int[5];
- int[] b = {10, 20, 30};
- int[][] matrix = new int[2][3];

Array Methods

- int[] a;
- a.length; //length of array

ArrayList

- ArrayList<Double> nums = new ArrayList<>(); nums.add(2.3); nums.size() == 1double a = nums.get(0);

String Functions

- s.equals(String s2) -> bool
- s.toLowerCase()
- s.toUpperCase()
- s.replace(char old, char new)
- s.replace(String old, String new)
- s.indexOf(String s) //-1 if not availabe
- s.lastIndexOf(String s)

Java Statements

- If Statementif (expression) {
- statements
- } else if (expression) {
- statements
- } else {
- statements
- }

- While Loopwhile (expression) {
- statements
- }

- Do-While Loopdo {
- statements
- } while (expression);

- For Loopfor (int i = 0; i < max; ++i) {
- statements
- }

Exception Handling

- try { statements; }
- catch (ExceptionType e1) { statements; }
- catch (Exception e2) { catch-all statements; }
- finally { statements; }

Encapsulation

- Bundling of data and operations to be performed on that data into single unit is called as encapsulation.
- ✓ Encapsulation in Java can be achieved by including both variables (data) and methods (operations) which act upon those variables into a single unit called class

Abstract / Interface

- abstract Method
- public abstract fun();
- abstract Class
- public abstract class Test{}
- Interface
- Like abstract class, but with only abstract functions. You don't need abstract for these
- Abstract Classes and Methods are without implementation.
- You use implements for Interfaces

Connect with me:

LinkedIn | YouTube | TopMate | Medium

Japneet Sachdeva

Map Methods

- **containsKey(key**) true if the map contains a mapping for the given key
- **get(key)** the value mapped to the given key (null if none)
- **keySet()** returns a Set of all keys in the map
- **put(key, value)** adds a mapping from the given key to the given value
- **putAll(map)** adds all key/value pairs from the given map to this map
- **remove(key**) removes any existing mapping for the given key

Methods Found in both Lists and Sets (ArrayList, LinkedList, HashSet, TreeSet)

- **add(value)** adds value to collection (appends at end of list)
- **contains(value)** returns true if the given value is found somewhere in this collection
- **remove(value)** finds and removes the given value from this collection
- **removeAll(collection)** removes any elements found in the given collection from this one
- **retainAll(collection)** removes any elements not found in the given collection from this one

Methods Found in ALL collections (Lists, Stacks, Queues, Sets, Maps)

- **clear()** removes all elements of the collection
- **equals(collection)** returns true if the given other collection contains the same elements
- **isEmpty()** returns true if the collection has no elements
- **size()** returns the number of elements in the collection
- **toString()** returns a string representation such as "[10, -2, 43]"

List Methods

- **add(index, value)** inserts given value at given index, shifting subsequent values right
- **indexOf(value)** returns first index where given value is found in list (-1 if not found)
- **get(index)** returns the value at given index
- **lastIndexOf(value)** returns last index where given value is found in list (-1 if not found)
- **remove(index)** removes/returns value at given index, shifting subsequent values left
- **set(index, value)** replaces value at given index with given value

Stack Methods

- **peek()** returns the top value from the stack without removing it
- **pop()** removes the top value from the stack and returns it; peek/pop throw an EmptyStackException if the stack is empty
- **push(value)** places the given value on top of the stack

Inheritance

- Inheritance, as name itself suggests, is used to inherit properties from parent class to child class.
- Using inheritance, you can reuse existing tried and tested code.

Abstraction

- Abstraction means separating ideas from their actual implementations.
- Using abstraction, you define only ideas in one class so that those ideas can be implemented by its subclasses according to their requirements.

PolyMorphism

- ✓ Poly means many and morphs means forms. So, anything which has multiple forms is called as polymorphism

Connect with me:
LinkedIn | YouTube | TopMate | Medium
Japneet Sachdeva