Java Bootcamp

DATTA DIWARE

Agenda

- Introduction to Java
- Basic input output
- Data Structures in Java
- Assignments

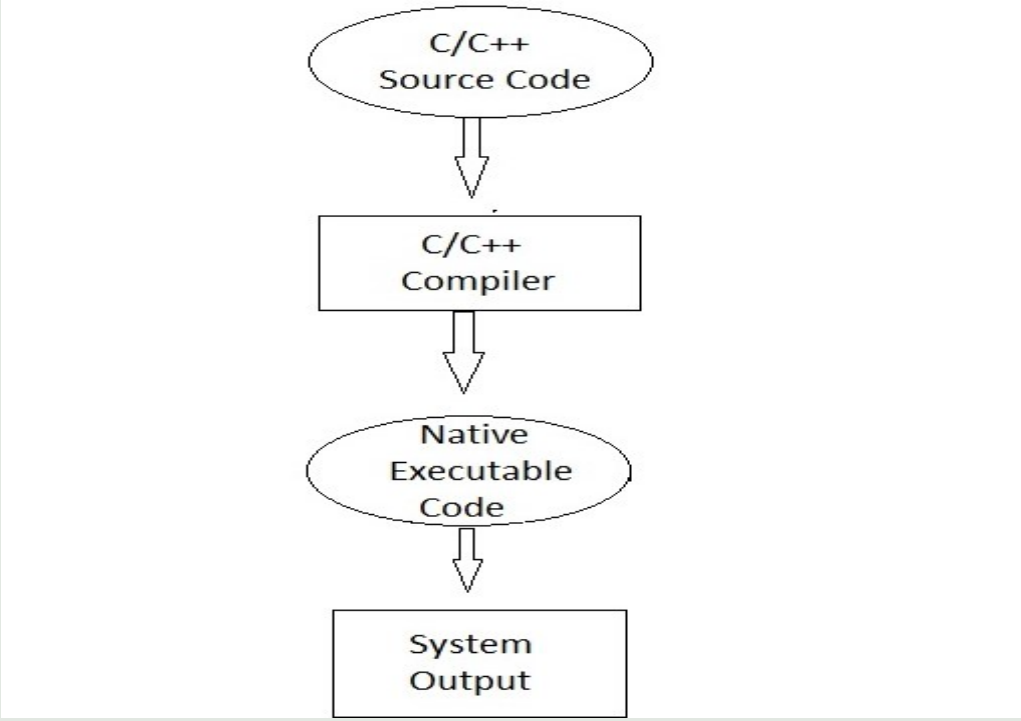
Introduction to Java

What is Java ?

- Java is **high-level object-oriented** programming language.
- Java is simple and easy to learn
- Java is platform independent
- Java has **automatic** memory management
- Java supports **Multi-threading and Concurrency**
- Java has strong community support
- Java has built-in security mechanisms
- **Its popularity ensures a high demand for Java developers, offering ample job opportunities.**

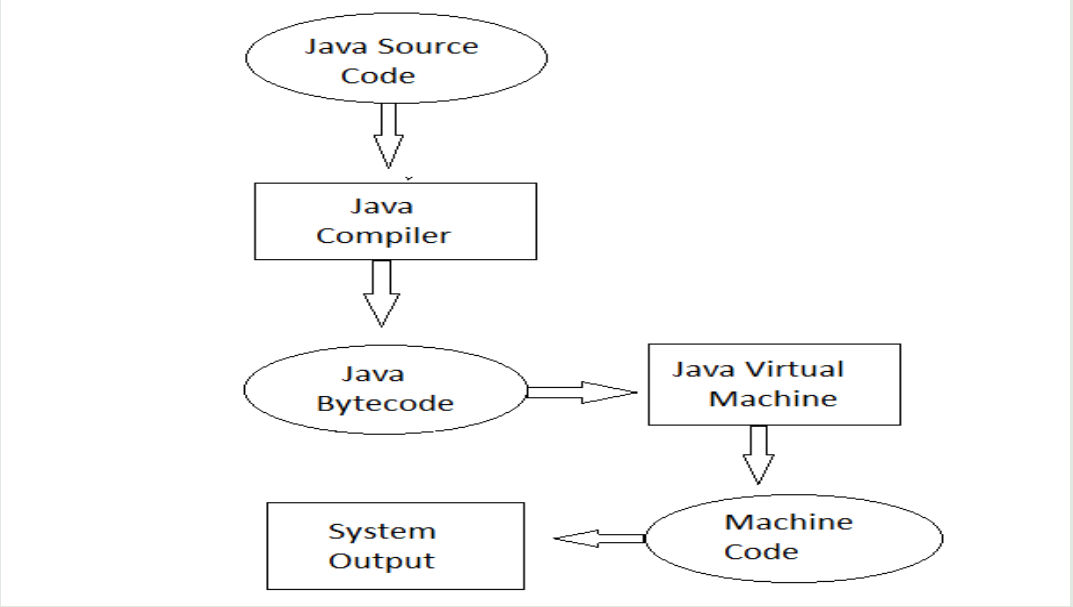
What is platform Dependent/Independent ?

- Platform Dependent means the program / software that we have developed can run /execute (show results) on a specific platform. That is on a specific operating system.



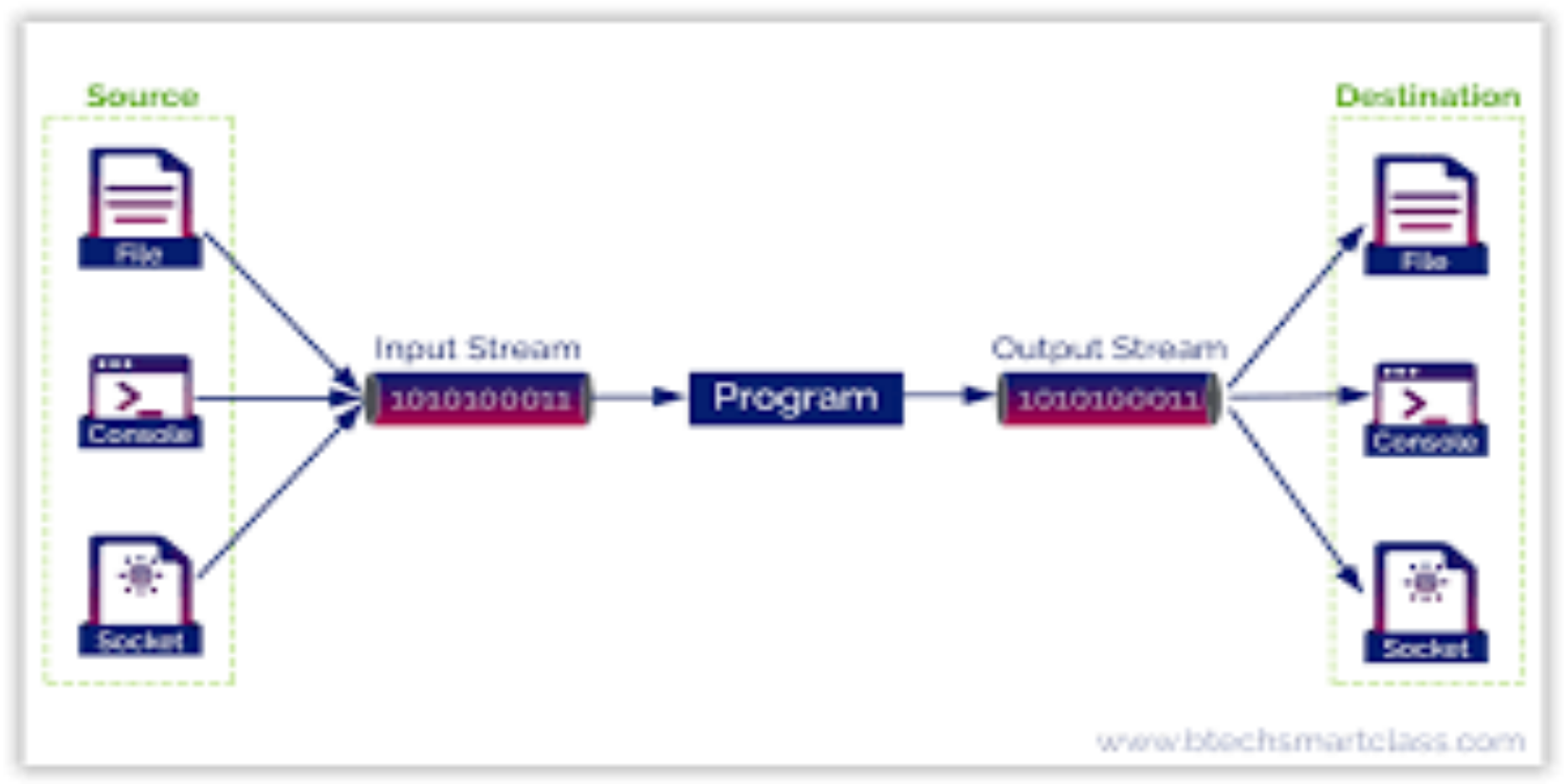
Contd.

Java is called Platform Independent because programs written in Java can be run on multiple platforms without re-writing them individually for a particular platform, i.e., Write Once Run Anywhere (WORA)

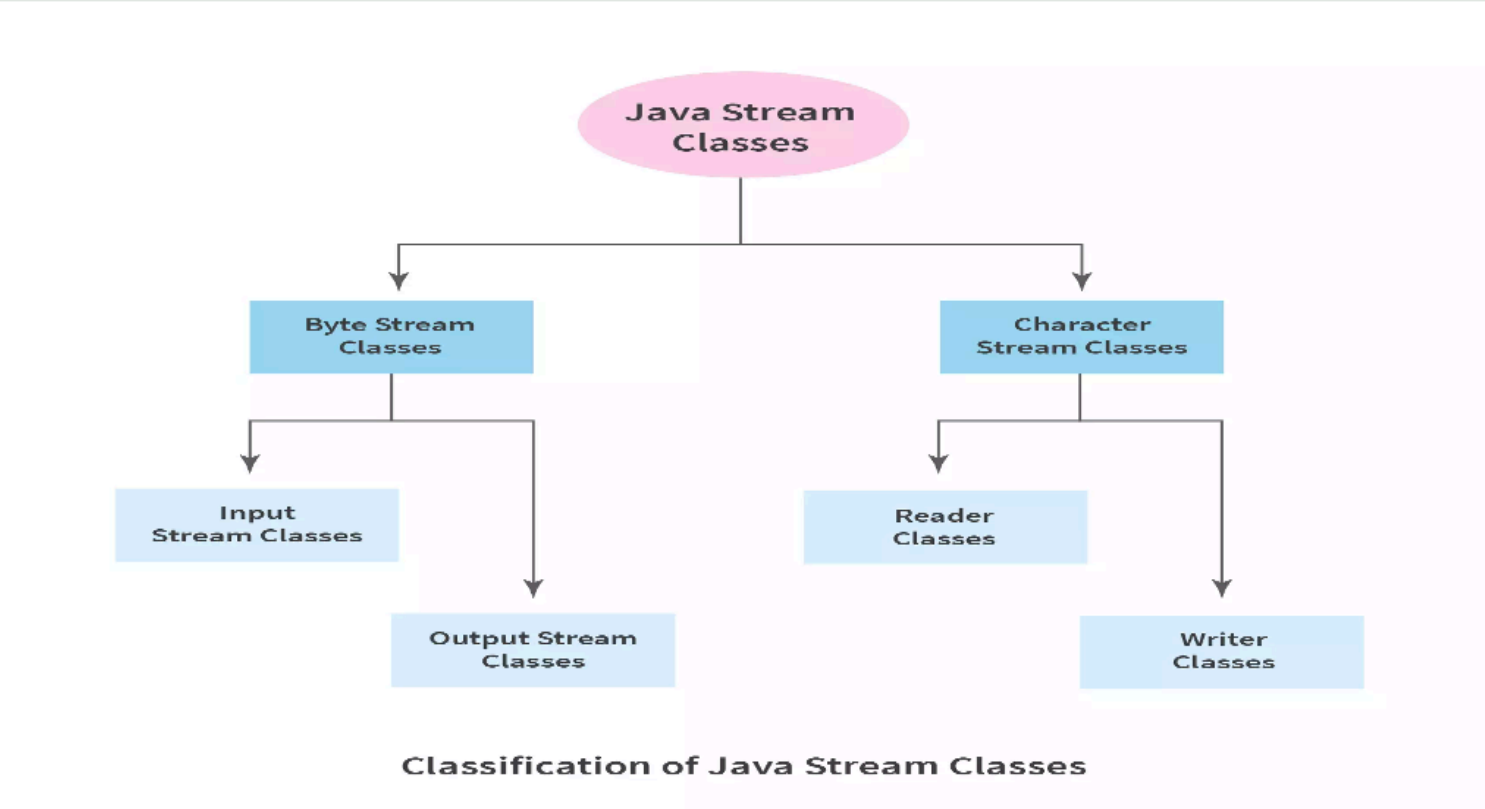


Let’s get hands dirty

Java I/O


Source
File
Console
Socket
Input Stream
1010100011
Program
Output Stream
1010100011
Destination
File
Console
Socket
www.btechsmartclass.com

Java I/O



Classification of Java Stream Classes

Java I/O

Byte Stream	Character Stream
They process the data byte by byte.	They process the data character by character.
They read/write data 8 bits maximum at a time.	They read/write data 16 bits maximum at a time.
They are most suitable to process binary files.	They are most suitable to process text files.
All byte stream classes in Java are descendants of InputStream and OutputStream.	All character stream classes in Java are descendants of Reader and Writer.

Java Data Structures

- What are the data types ?
- Integer, Float, String are the datatype
- A data structure is a specialised format for organising, processing, retrieving and storing data.
- Example : Array, LinkedList, Queue, Stack

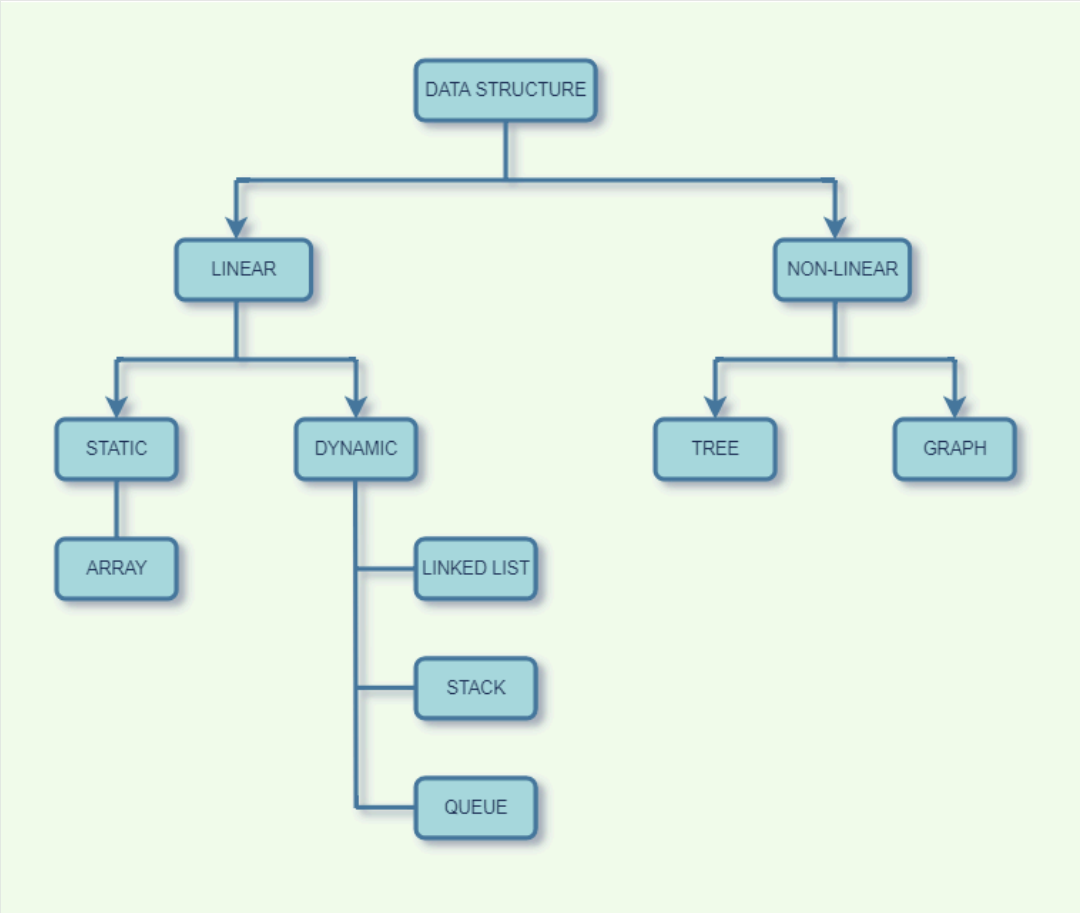
Data Structure

- **Five factors to consider when picking a data structure include the following:**
  - What kind of information will be stored?
  - How will that information be used?
  - Where should data persist, or be kept, after it is created?
  - What is the best way to organize the data?
  - What aspects of memory and storage reservation management should be considered?

Characteristics of Data Structure

- Linear or non-linear
- Homogeneous or heterogeneous
- Static or dynamic

Data Structure Hierarchy



Data Structure - Array

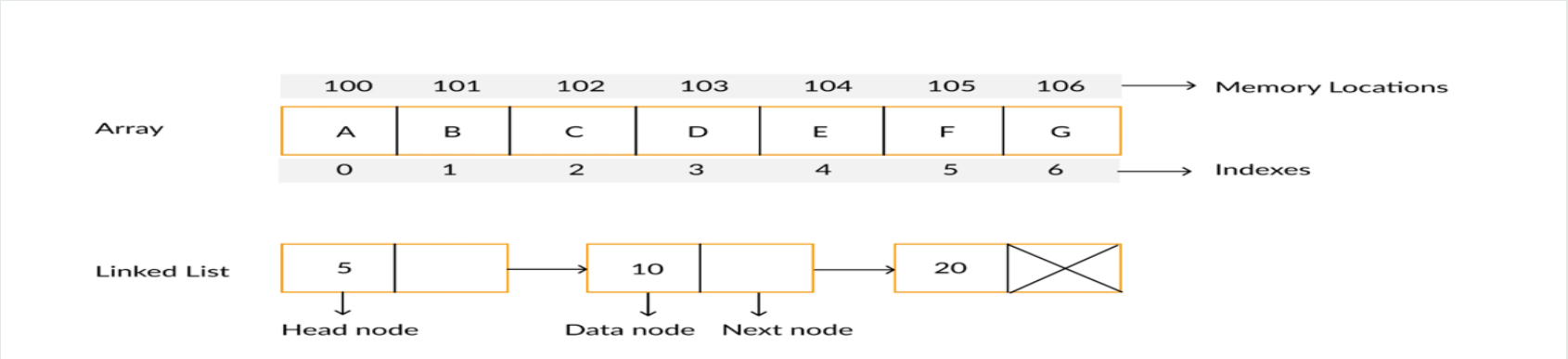
- An array stores a collection of items at adjoining memory locations.
- Arrays are homogeneous data structure
- Element can be calculated or retrieved easily by an index
- Mostly fixed in size

Data Structure - Array

- Ways to declare arrays in Java
- Ways to initialize arrays in java
- How to add elements in Array
- How to delete element from array
- How to print elements from Array
- Operations we can perform on arrays
- Multidimensional Arrays

Data Structure – LinkedList

- A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



- First object of the Linked List is known as the head and the last object is known as the tail of the Linked List.

Array vs LinkedList

Arrays	Linked Lists
An array is a collection of elements of a similar data type.	Linked List is an ordered collection of elements of the same type in which each element is connected to the next using pointers.
Array elements can be accessed randomly using the array index.	Random accessing is not possible in linked lists. The elements will have to be accessed sequentially.
Data elements are stored in contiguous locations in memory.	New elements can be stored anywhere and a reference is created for the new element using pointers.
Insertion and Deletion operations are costlier since the memory locations are consecutive and fixed.	Insertion and Deletion operations are fast and easy in a linked list.
Memory is allocated during the compile time (Static memory allocation).	Memory is allocated during the run-time (Dynamic memory allocation).
Size of the array must be specified at the time of array declaration/initialization.	Size of a Linked list grows/shrinks as and when new elements are inserted/deleted.

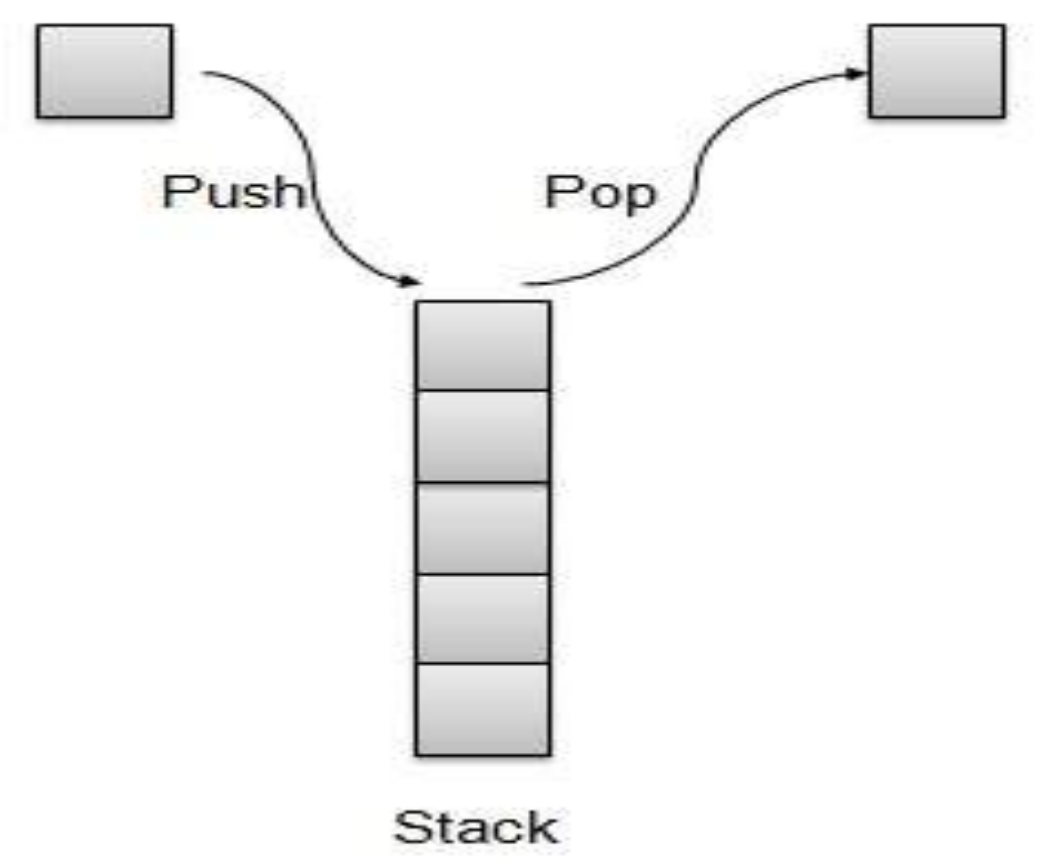
Assignments

- Write a Java program to take your name as input and print “Hello ${name}”
- Write a Java program add() two numbers and print sum
- Write a Java program to sum all the elements from array
- Write a java program to print 2D array
- Write a program to count number of elements in 2D array
- Write a program to implement Queue in Java

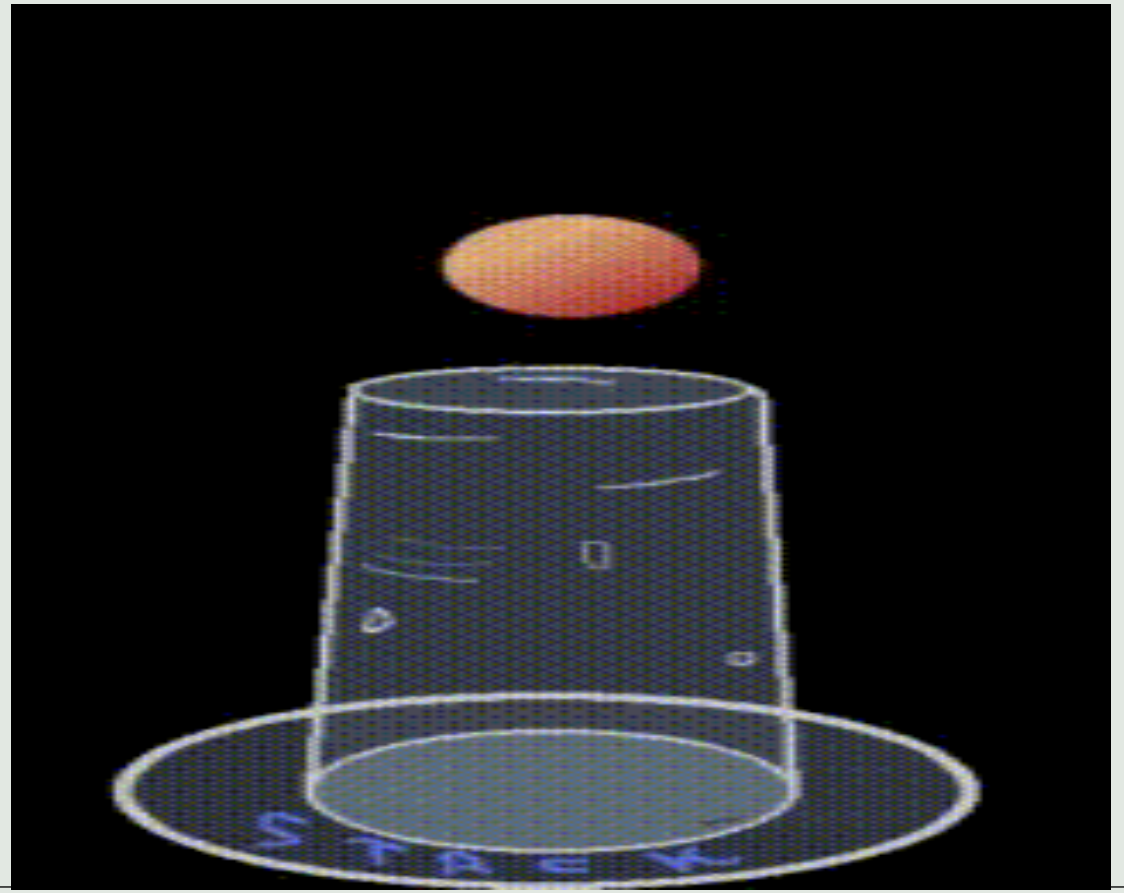
Data Structure : Stack

- A stack is a data structure that follows a last in, first out (LIFO) protocol
- The stack data structure has three main methods: push(), pop() and peek().
- The push() method adds a node to the top of the stack.
- The pop() method removes a node from the top of the stack.
- The peek() method returns the value of the top node without removing it from the stack.

Data Structure : Stack



Stack contd.



Stack contd.

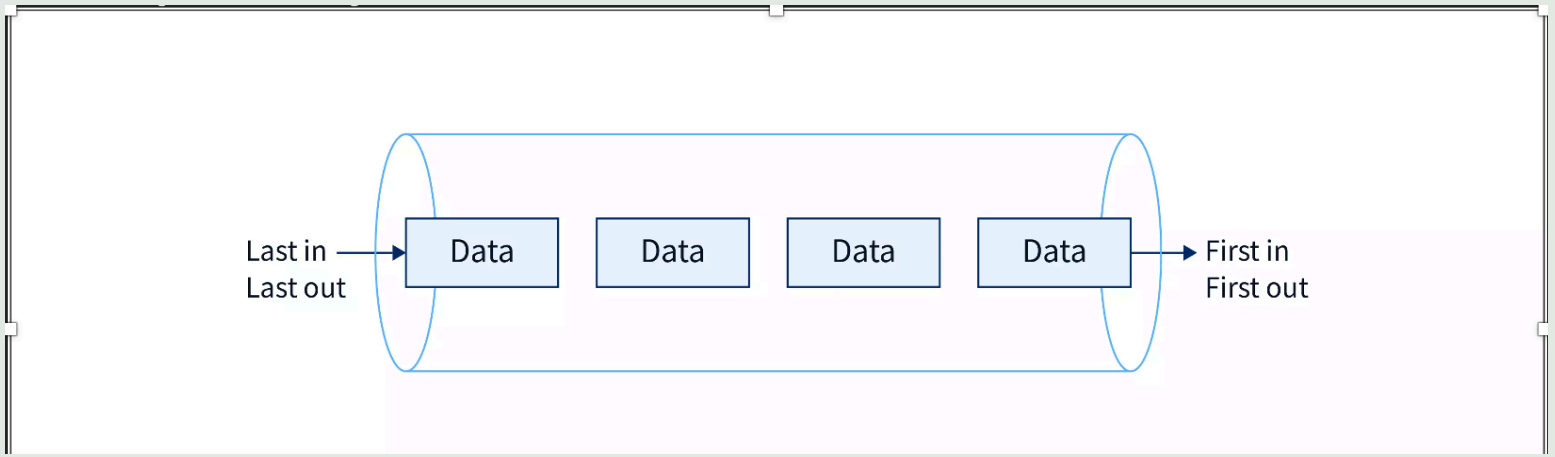
- Can u tell me few examples of Stack use in real life
- Back and Forward button on web browser
- Undo Redo functionality
- Compilers bracket balancing

Java Stack Implementation

- Push 10 elements in stack
- Pop all elements from stack
- Check stack is full while pushing
- Check stack is empty while popping

Data Structure : Queue

- What is Queue?
- The Queue in data structure is an ordered, linear sequence of items. It is a FIFO (First In First Out) data structure, which means that we can insert an item to the rear end of the queue and remove from the front of the queue only.



Data Structure : Queue

- The following are some basic functions defined for a queue.
  - enqueue(): It's used to add an element to the back of the queue.
  - dequeue(): This method removes an element from the queue's front.
  - IsEmpty(): This method is used to determine whether or not the queue is empty.
  - IsFull(): This method is used to determine whether or not the queue is full.
  - peek(): It's used to get the front value without eliminating it from the equation.

Java Class

- A class is a blueprint or a template for creating objects in Java.
- It defines the structure and behaviour of objects.
- A class can contain various members, including fields, methods, constructors, and nested classes.

```
public class Student {
    // member Variable declaration
    // Constructors
    // member methods
    // optional main() method
}
```

Java Class : Access Specifiers

- Java provides access modifiers to control the visibility of class members.
- Public: Accessible from anywhere.
- Private: Accessible only within the class.
- Protected: Accessible within the class and its subclasses.
- Default (no modifier): Accessible within the same package.

Java Class : Member Variables

- Member variables are variables declared within a class.
- They hold data that represents the state of an object.
- Member variables can have different access modifiers ex private, public, protected, default
- They are accessible within the class and can be used in its methods.
- Member variables store object-specific data and can be accessed through the object's reference.

```
public class Student {
  private int rollNo ;
  private String name ;
  private float marks = 0.0 ;
}
```

Java class : Methods

- Methods are blocks of code that perform specific tasks or actions.
- They are declared within a class and have a name, parameters, and an optional return type.
- Access modifiers control the visibility of methods.
- Method parameters provide inputs to the method.
- Return type specifies the type of value returned by the method.
- Methods are invoked or called to execute their code

Example

```
public class Student {
    private int num1;
    private int num2;

    public void printMessage();

    public int returnNum();

    public void addNumber(int num1, int num2);

    public int addNumber(int num1, int num2);
}
```

Java class : Constructors

- Constructors are special methods used to initialise objects of a class.
- They have the same name as the class and no return type.
- Constructors are automatically called when objects are created.
- Constructors can be overloaded to have different parameter sets.
- Default constructors are provided by Java if no constructors are defined.
- Initialisation blocks can be used for additional initialisation logic.

Example

```
public class Student {
    private int num1;
    private int num2;

    Student(){}

    Student(int num1Local, num1Local){
     this.num1 = num1Local;
     this.num2 = num1Local
    }

    public void printMessage();

    public int returnNum();

    public void addNumber(int num1, int num2);

    public int addNumber(int num1, int num2);
}
```

Java Object

- Objects have state (member variables) and behaviour (methods).
- Objects are created using the "new" keyword.
- Object-oriented programming emphasises the use of objects for building programs.
- ```
  Student student = new Student() ;
  ```
  ```

Example

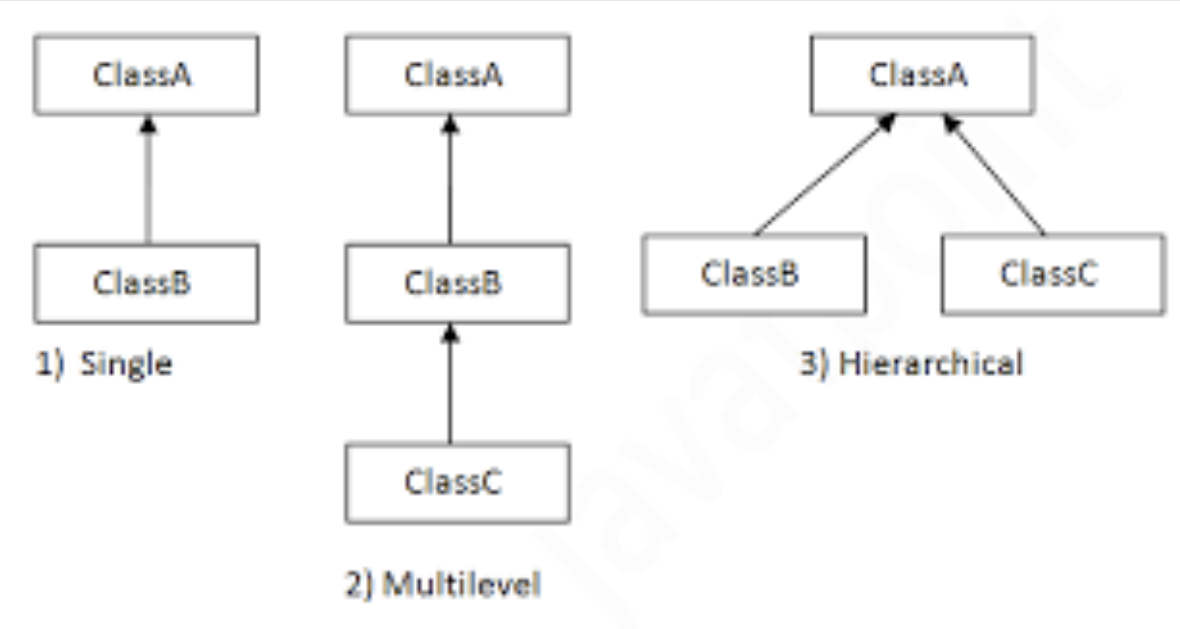
```
public class Student {
    private int num1;
    private int num2;
    Student(){}
    Student(int num1Local, num1Local){
     this.num1 = num1Local;
     this.num2 = num1Local
    }
    public int addNumber(int num1, int num2){
      int addition = num1 + num2 ;
      return addition ;
    }
   public static void main(String[] args){

    Student student = new Student();
    student.addNumber(10,20);
   }
}
```

Inheritance

- Inheritance is a mechanism in Java that allows classes to inherit properties and methods from other classes.
- Superclass is the class being inherited from, while subclass is the class that inherits from the superclass.
- Inheritance is declared using the "extends" keyword.
- Access modifiers control the visibility of inherited members.
- Inheritance supports different types such as single, multiple (not supported in Java), multilevel, and hierarchical.

Inheritance Contd.

- 

Single inheritance

- Single inheritance is a type of inheritance in Java where a subclass extends only one superclass.
- The subclass inherits all non-private members from the superclass.
- Inherited members become part of the subclass and can be accessed and used within it.
- Method overriding allows the subclass to provide its own implementation of inherited methods

Assignments - session 1

- Write a Java program to take your name as input and print “Hello ${name}”
- Write a Java program add() two numbers and print sum
- Write a Java program to sum all the elements from array
- Write a java program to print 2D array
- Write a program to count number of elements in 2D array
- Write a program to implement Queue in Java

Assignments - session 2

1. Bank Account Class:
2. Create a Java class called BankAccount that represents a bank account. The class should have fields such as account number, account holder name, and balance. Implement methods for depositing and withdrawing money, checking the account balance, and displaying the account information.

4. Bank Account Inheritance:
5. Build upon the previous Bank Account class assignment. Create a Java class hierarchy to represent different types of bank accounts such as BankAccount, SavingsAccount, and CheckingAccount. The base class should have common properties and methods for all accounts, while the derived classes should inherit from the base class and provide specific implementations. Implement methods to calculate interest, handle withdrawals, and display account information.

6.

Thank you !!

Connect@ https://dattadiware.github.io/