# Experiment 1

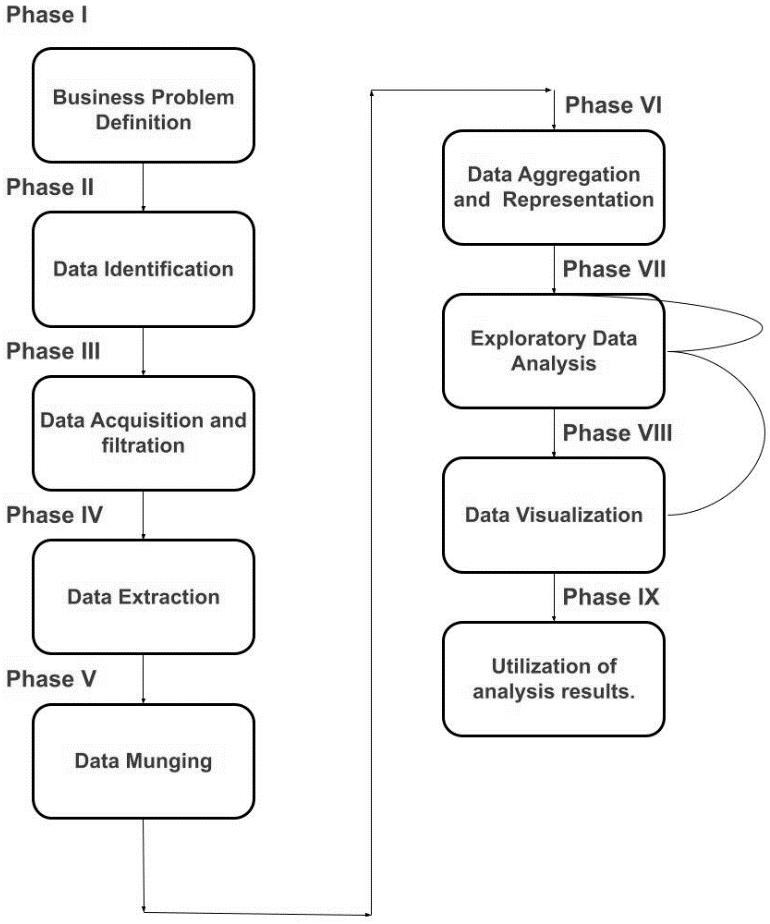
|  |  |
| --- | --- |
| **1.** | **Explain the terms “Big Data and Big Data Analytics”. Also, explain the life cycle of BDA.** |
| **Solution** | **Data-** The quantities, characters, or symbols on which operations are performed by a computer, which may be stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.  **Big data- ‘Big** Data' is also **data** but with a **huge size**. 'Big Data is a term used to describe a collection of data that is huge in size and yet growing exponentially with time. Data that are very large in size is called Big Data. Normally we work on data of size MB(Word Doc, Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 10^15-byte size is called Big Data. It is stated that almost 90% of today's data has been generated in the past 3 years. In short, *s*uch data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.  ***“Big data” is high-volume, velocity, and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”***  **Big Data Analytics-** Big data analytics is the use of advanced analytic techniques against very large, diverse big data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes.  A screenshot of a computer  Description automatically generated with low confidence  Big Data Analytics largely involves collecting data from different sources, munge it in a way that it becomes available to be consumed by analysts, and finally delivering data products useful to the organization’s business.  The process of converting large amounts of unstructured raw data, retrieved from different sources to a data product useful for organizations forms the core of Big Data Analytics. |

**Life Cycle of Big Data Analytics-** The life cycle phases of Big Data Analytics differs from traditional data analysis, mainly due to the fact that in big data, volume, variety, and velocity form the basis of data. The Big Data Analytics Life cycle is divided into nine phases-

* 1. **Business Problem Definition–** In this stage, the team learns about the business domain, which presents the motivation and goals for carrying out the analysis. In this stage, the problem is identified, and assumptions are made that how much potential gain a company will make after carrying out the analysis.
  2. **Data Definition–** Once the business case is identified, now it’s time to find the appropriate datasets to work with. In this stage, analysis is done to see what other companies have done for a similar case. Depending on the business case and the scope of analysis of the project being addressed, the sources of datasets can be either external or internal to the company.
  3. **Data Acquisition and filtration–** Once the source of data is identified, now it is time to gather the data from such sources. This kind of data is mostly unstructured. Then it is subjected to filtration, such as the removal of the corrupt data or irrelevant data, which is of no scope to the analysis objective. Here corrupt data means data that may have missing records, or the ones, which include incompatible data types. After filtration, a copy of the filtered data is stored and compressed, as it can be of use in the future, for some other analysis.
  4. **Data Extraction–** Now the data is filtered, but there might be a possibility that some of the entries of the data might be incompatible, to rectify this issue, a separate phase is created, known as the data extraction phase. In this phase, the data, which don’t match the underlying scope of the analysis, are extracted and transformed into such a form.
  5. **Data Munging –** The data is collected from various sources, which results in the data being unstructured. There might be a possibility, that the data might have constraints that are unsuitable, which can lead to false results. Hence there is a need to clean and validate the data. It includes removing any invalid data and establishing complex validation rules. There are many ways to validate and clean the data.
  6. **Data Aggregation & Representation–** The data is cleansed and validated, against certain rules set by the enterprise. But the data might be spread across multiple datasets, and it is not advisable to work with multiple datasets. Hence, the datasets are joined together.
  7. **Exploratory Data Analysis –** Depending on the nature of the big data problem, analysis is carried out. Data analysis can be classified as Confirmatory analysis and exploratory analysis.

In confirmatory analysis, the cause of a phenomenon is analyzed before. The assumption is called the hypothesis. The data is analyzed to approve or disapprove of the hypothesis. This kind of analysis provides definitive answers to some specific questions and confirms whether an assumption was true or not.

* 1. **Data Visualization –** A sort of representation is required to obtain value or some conclusion from the analysis. Hence, various tools are used to visualize the data in graphic form, which can easily be interpreted by business users. Visualization is said to influence the interpretation of the results. Moreover, it allows the users to discover answers to questions that are yet to be formulated.
  2. **Utilization of analysis results–** The analysis is done, the results are visualized, and now it’s time for the business users to make decisions to utilize the results. The results can be used for optimization, to refine the business process. It can also be used as an input for the systems to enhance performance.



It is evident from the block diagram that Phase VII, i.e. exploratory Data analysis, is modified successively until it is performed satisfactorily. Emphasis is put on error correction. Moreover, one can move back from Phase VIII to Phase VII, if a satisfactory result is not achieved. In this manner, it is ensured that the data is analyzed properly.

# Experiment 2

|  |  |
| --- | --- |
| **1.** | **Objective: Write a menu-driven program to implement t h e following data structures in java:**   1. **Linked List** 2. **Stack** |
| **Solution-** | * **Tool used: Windows OS** * **Language: Java** * **Editor: Vscode Source Code-**   **Linked List Implementation-**  Include the package java .util.\*; public class Test  {  public static void main(String args[])  {  // Creating object of class linked list LinkedList<String> object = new LinkedList<String>();  // Adding elements to the linked listobject.add("A");  object.add("B");  object.addLast("C");  object.addFirst("D");  object.add(2, "E");  // similar to above add F and G System.out.println("Linked list : " + object);  // Removing elements from the linked listobject.remove("B"); object.remove(3);  object.removeFirst();  object.removeLast();  System.out.println("Linked list after deletion: " + object);  // Finding elements in the linked list boolean status = object.contains("E"); if(status)  System.out.println("List contains the element 'E' ");  else  System.out.println("List doesn't contain the element 'E'");  // Number of elements in the linked listint size = object.size(); System.out.println("Size of linked list = " + size);  // Get and set elements from linked listObject element = object.get(2); System.out.println("Element returned by get() : " + element);object.set(2, "Y"); System.out.println("Linked list after change : " + object);}} |

**Expected Output-**

Linked list : [D, A, E, B, C, F, G] Linked list after deletion: [A, E, F] List contains the element 'E'

Size of linked list = 3 Element returned by get() : F Linked list after change : [A, E, Y]

# Stack Implementation-

java.io.\*, java.util.\*. class MyStack

{

// Pushing element on the top of the stack static void stack\_push(Stack<Integer> stack)

{

for(int i = 0; i < 5; i++)

{

stack.push(i);

}

}

// Popping element from the top of the stackstatic void stack\_pop(Stack<Integer> stack)

{

System.out.println("Pop :"); for(int i = 0; i < 5; i++)

{

Integer y = (Integer) stack.pop();System.out.println(y);

}

}

// Displaying element on the top of the stack static void stack\_peek(Stack<Integer> stack)

{

Integer element = (Integer) stack.peek(); System.out.println("Element on stack top : " + element);

}

// Searching element in the stack

static void stack\_search(Stack<Integer> stack, int element)

{

Integer pos = (Integer) stack.search(element);

if(pos == -1) System.out.println("Element not found");else System.out.println("Element is found at position " + pos);

}

public static void main (String[] args)

{

Stack<Integer> stack = new Stack<Integer>();

stack\_push(stack); stack\_pop(stack); stack\_push(stack); stack\_peek(stack); stack\_search(stack, 2);

stack\_search(stack, 6);

}

}

# Expected Output:

Pop: 4 3 2 1 0

Element on stack top: 4 Element is found at position 3 Element not found

# Experiment 3

|  |  |
| --- | --- |
| **1.** | **Objective: Write a menu-driven program to implement t h e following data structures in java:**   1. **Queue** 2. **Set** 3. **Map** |
| **Solution-** | * **Tool used: Windows OS** * **Language: Java** * **Editor: VSCode**   **Queue Implementation**  import java.util.LinkedList;import java.util.Queue; public class QueueExample  {  public static void main(String[] args)  {  Queue<Integer> q = new Queue<>();  // Adds elements {0, 1, 2, 3, 4} to queue  for (int i=0; i<5; i++) q.add(i);  // Display contents of the queue. System.out.println("Elements of queue-"+q);  // To remove the head of  queue.int removedele = q.remove(); |

System.out.println("removed element-" + removedele); System.out.println(q);

// To view the head of queueint head = q.peek();

System.out.println("head of queue-" + head);

// Rest all methods of collection interface,

// Like size and contains can be used with this

// implementation. int size = q.size();

System.out.println("Size of queue-" + size);

}

}

# Expected Output:

Elements of queue-[0, 1, 2, 3, 4] removed element-0

[1, 2, 3, 4]

head of queue-1Size of queue-4

# Set Implementation

// Java program Illustrating Set Interface

// Importing utility classes import java.util.\*;

// Main class public class GFG {

// Main driver method

public static void main(String[] args)

{

// Demonstrating Set using HashSet

// Declaring object of type String

Set<String> hash\_Set = new HashSet<String>();

// Adding elements to the Set

// using add() method hash\_Set.add("BDA"); hash\_Set.add("Ehtesham"); hash\_Set.add("Eddy"); hash\_Set.add("Example"); hash\_Set.add("Set");

// Printing elements of HashSet object System.out.println(hash\_Set);

}

}

# Expected Output:-

[Set, Example, Eddy, Ehtesham]

// Java Program to Demonstrate

// Working of Map interface

// Importing required classes import java.util.\*;

// Main class class GFG {

// Main driver method

# Map Implementation

public static void main(String args[])

{

// Creating an empty HashMap Map<String, Integer> hm

= new HashMap<String, Integer>();

// Inserting pairs in above Map

// using put() method hm.put("a", new Integer(100));

hm.put("b", new Integer(200));

hm.put("c", new Integer(300));

hm.put("d", new Integer(400));

// Traversing through Map using for-each loop for (Map.Entry<String, Integer> me :

hm.entrySet()) {

// Printing keys System.out.print(me.getKey() + ":"); System.out.println(me.getValue());

}

}

}

# Expected Output-

a:100 b:200 c:300 d:400