**Day 1**

**1) Source Code Vs Object Code**

i) Souce Code Something Which We right Using some Programming Language .

ii)In Java this this Source is Converted to Intermediate Byte Code using Java Compliler which create filename.class File.

iii)This class file is Converted to Machine Code by JVM and Executed whenever required

iv) Any OS having JDk installed in there Machine can Excuted this class file, this is why java is called Platform independent.

**2) Using Compiler with flags**

i) JDK provide java compiler with differnet Flags which can be retrived by typing javac on command line and we can get list of flags that can be used with javac command.

ii) Flag used by use were -X, set path, path, -version: To check Compiler Version

iii) Debug Mode in Eclipse can Be used to be debug the code by setting the breakpoints and retify the bug which is causing the program not work according to our need.

**3) Keywords, Variables**

i) We Rectified compile time errors by changing names of keywords which throw compile time errors.

ii) We learnt we cannot use different keywords as variables

iii) Variable are alias for memory addresses used to store some values in those Memory locations.

iv) Variables are of Different data Types like either they might be primitive like int, float or non-premitive like String, Array.

**4) Variable scope, lifetime, life time, initial value, modifiers**

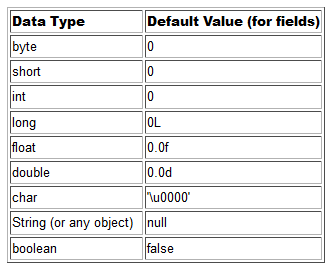
i) Place Were variables can be accessed is know as variable Scope . Scope are of 3 type

-Method Level

- Class Level

- package level

ii) Lifetime of the Variable : The **lifetime** of a **variable** is the time during which the **variable** stays in memory and is therefore accessible during program execution. The **variables** that are local to a method are created the moment the method is activated (exactly as formal parameters) and are destroyed when the activation of the method terminates.

iii) If we do not initialize a variable with some value then default value is assigned to it by the compiler. We refered Java 8 API Documentation to understand Different datatypes and their default values, below table Premitive datatypes and their Default values. 

**5) Programming Constructs (Selection constructs and Loop constructs)**

i) A selection constructs provides for selection between alternatives. We can identify three types of selection construct:

1. If statements
2. Case statements
3. Pattern matching (Not Much Used)

ii) There are times when a program needs to repeat certain steps until told otherwise, or until a condition has been met. This process is known as iteration. Iteration is also often referred to as looping, since the program ‘loops’ back to an earlier line of code. Sections of codes that are iterated are called loops. Different Types of Loop are for, While etc.

**6) Declare Array**

i) An array data structure, or simply an array, is a data structure consisting of a collection of elements, each identified by at least one array index or key. An array is stored such that the position of each element can be computed from its index tuple

ii) An array declaration has two components: the type and the name. type declares the element type of the array. The element type determines the data type of each element that comprises the array. Like array of int type, we can also create an array of other primitive data types like char, float, double..etc or user defined data type(objects of a class).Thus, the element type for the array determines what type of data the array will hold.

iii) Array Declaration int[] intArray = new Int[size] , Object[] ao = new Object[Size]

**7) Store Data in Array and Process Elements**

i) Assigning values to an element in an array is similar to assigning values to scalar variables. Simply reference an individual element of an array using the array name and the index inside parentheses, then use the assignment operator (=) followed by a value. Until an array element is first assigned a value, it has a Null (empty) value.

ii) Storing Values in Array

dataType[] arrayRefVar = {value0, value1, ..., valuek};

**DAY 2**

**1) Pointers , Declare and Initialize Pointers**

i) **Java doesn’t support pointer explicitly,  But java uses pointer implicitly:**Java use pointers for manipulations of references but these pointers are not available for outside use. Any operations implicitly done by the language are actually NOT visible.

ii) Java does not support pointer because of security concerns if pointers are not properly initialized or declared they might crash the OS or the code which is the tedious job to handle.

iii) java does not allow to Manipulate the pointer explicitly but java does it implicitly without creating any Problem.

**2)Functions definitions/invocations**

i) Functions nothing but Series of Instruction might return or do not return anything depends upon return type .In Java, all function definitions must be inside classes .

ii)Function Defination : public static void foo(parameter1 , parameter2 ……)

iii) Defining a new function does not make the function run. To execute the function, we need a **function call**. This is also known as a **function invocation**.

**3) Call functions by passing value and passing reference**

i) Call by Value means calling a method with a parameter as value. Through this, the argument value is passed to the parameter.

ii) While Call by Reference means calling a method with a parameter as a reference. Through this, the argument reference is passed to the parameter.

iii) In call by value, the modification done to the parameter passed does not reflect in the caller's scope while in the call by reference, the modification done to the parameter passed are persistent and changes are reflected in the caller's scope.

**4) Dynamic Memory allocation/deallocation**

i) Memory allocation during Runtime of the program is know as Dynamic Memory Allocation.

ii) If there are no references to the old object, then it will be garbage collected automatically. (Or, more accurately, it will be available for garbage collection and the memory will be automatically reclaimed when the garbage collector gets around to it.) This is know as memory Dellocation.

iii) We use Jvisualvm and jconsole application provided by JDK to analyze memory allocation in RAM and CPU utilization and how many time Garbage collector is called to free up the memory.

**5) Memory Leakage**

i) Memory leak is a scenario that occurs when **objects are no longer being used by the application, but the Garbage Collector is unable to remove them from working memory** – because they’re still being referenced. As a result, the application consumes more and more resources – which eventually leads to a fatal OutOfMemoryError.

**6) User Defined Structures**

i) A **structure** is a heterogeneous container object, i.e., it is an object with elements whose values do not have to be of the same data type. ... A **user**-**defined** data type is a **structure** with a fixed set of fields **defined** by the **user**.

ii) Class, Structure , Union are User Defined Structures/Datatypes.

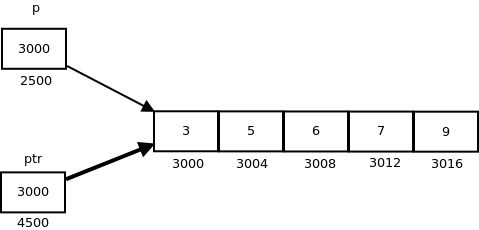
**7) Access Data Members**

i) Data Members can be accessed using objects/instances of the class or it’s a standred that make variables in class private and access them using getters and setter methods of the class.

**8) Dynamic Memory Allocation for user defined structures**

i) Allocating Memory to objects of the class during Runtime of the program using new method or class constructor calling this is know as Dynamic Memory Allocations for object/class instances.

**9) Array of Pointers**



i) an **array of pointers** is an indexed set of variables, where the variables are **pointers** (referencing a location in memory)

**10 )Iterate over array of Pointers**

i) IDK What to Write Here

**Day 3**

|  |
| --- |
| **\*Sequential Access / Random Access** |
|  |

|  |
| --- |
| **1) Array** : You can use the array index to access any element immediately. This is called random access. |
|  |

|  |
| --- |
| **2) Linked List** : We have to start at the top of the list and then move from node to node until you get to the node you want, which is termed sequential access. |
|  |

|  |
| --- |
| **3) Stack** : Use Sequentional Access Based upon FILO(First in last out) basis. |
|  |

|  |
| --- |
| **4) Queue** : Use Sequential Access Based upon FIFO (First in First out) basis. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **\*Static Allocation/Dynamic Allocation** |
|  |

|  |
| --- |
| **1)Array** : Static |
|  |

|  |
| --- |
| **2)Linked List** : Dynamic |
|  |

|  |
| --- |
| **3)Stack** : Static if we use Array and Dynamic if we use Linked List. |
|  |

|  |
| --- |
| **4)Queue** : Static if we use Array and Dynamic if we use Linked List. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \***Policy for getting/putting number** |
|  |

|  |
| --- |
| **1)Array** : Randomly insert or access data in any index of the Array and do changes accordingly. |
|  |

|  |
| --- |
| **2)Linked List** : We have to Traverse Whole list to access some data right the head node. |
|  |

|  |
| --- |
| **3)Stack** : By Pushing or Poping the data we can access or delete the data in Stack. |
|  |

|  |
| --- |
| **4)Queue** : By queing or Dequeing the list we can access or delete the data in queue. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \* **Use Case** |
|  |

|  |
| --- |
| **1)Array** : Arrays are used to implement mathematical vectors and matrices, as well as other kinds of rectangular tables. Many databases, small and large, consist of one-dimensional arrays whose elements are records |
|  |

|  |
| --- |
| **2)Linked List** :i) Implementation of stacks and queues. |
|  |

|  |
| --- |
| ii)Maintaining directory of names. |
|  |

|  |
| --- |
| **3)Stack** : i)The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack. |
|  |

|  |
| --- |
| ii)Calculators |
|  |

|  |
| --- |
| **4) Queue :**  i) Ticket counter line where people who come first will get his ticket first. |
|  |

|  |
| --- |
| ii) Serving requests on a single shared resource, like a printer, CPU task scheduling etc. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **Binary Tree** |
|  |

|  |
| --- |
| i) Sequention Access. |
|  |

|  |
| --- |
| ii) Dynamic Memory Allocations. |
|  |

|  |
| --- |
| iii) We Have to traverse the Tree to insert and Delete the node From the tree |
|  |

|  |
| --- |
| iv) Used where Large Amount of Searching operations Are done to ease the search. |
|  |

**Queue**

|  |
| --- |
| Queue Algorithm Using Array |
|  |
| Data Members MaxSize, Front =-1, Rare=-1;  Member Fuctions :  i)Enqueue()  ii) Dequeue()  iii) IsEmpty()  iv)IsFull() |
|  |

|  |
| --- |
| i) For Adding element in array we have to increment rear counter with +1 and insert element into array with elements data .While adding we also have to check weather array is full or not by comparing rear counter with max size of array. |
|  |

ii) For Deleting or Dequeing in the array we have to delete the first most element which was inserted in the array like vise .while delecting an Element we also have to check weather array is empty or not by compare fornt counter with '-1'.

**LinkedList**

|  |
| --- |
| **1. AddLast** |
|  |

|  |
| --- |
| i) Traverse Whole list till we reach Last Node of The list where Last node Pointre is pointing to NULL. |
|  |

|  |
| --- |
| ii) point the last node to the newly Created Node and make pointer of newly added node as NULL. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **2.AddFirst** |
|  |

|  |
| --- |
| i) Intially Store the address where the Head pointer is pointing in some temp variable , |
|  |

|  |
| --- |
| ii) Make head pointing toward newly Created node. |
|  |

|  |
| --- |
| ii) Make newly created node point to the address where temperiory variable is storing |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **3) Add E After B** |
|  |

|  |
| --- |
| i) Traverse Till B |
|  |

|  |
| --- |
| ii) Store the address where pointer of B is pointing in some temperiory variable |
|  |

|  |
| --- |
| ii) Change Where B's pointer is pointing and make it point at node E |
|  |

iv)And Final make E's pointer Point at the address where temperiory variable is storing