Binar\_Search;  
*/\*Binary search is a searching method where it will search by  
dividing the total array length. The array should be sorted array.  
Sorted array means ascending order.*

Binar\_Search;  
*// This is linear search because, it will search one by one element from 0 index to the last index.  
//it will search linearly.*

*public class* CollectionInterfaceExmple {  
}  
*/\*  
1). Collection is an interface which is present in java.util package  
  
2). Syntax of Collection:  
  
 public interface Collection<E> extends Interface<E>  
 {  
 ---  
 ----  
 }  
  
 3). Hierarchy of collection interface  
 4). Methods of collection interface  
 -public boolean add(Object obj)  
 -public boolean addAll(Collection c)  
 -public boolean remove  
 -boolean contains(object obj)  
 -boolean isEmpty()  
 -int size()*

*public class* DataStructure {  
}  
*/\*  
Q: What is data Structure?  
Ans. Data Structure is way by which we can store the data in efficient way. As We can retrieve and access.  
Tje efficient is the matter of time and space.  
  
Types of data Structure:  
1) Primitive Data Structure  
 In Primitive Data Structure, we have (Boolean, char, byte,short, int, long, float, double)  
2). Non-primitive DS  
 Non-primitive DS is two types:  
 a). Linear DS b). Non-Linear DS  
  
 Example of Linear DS: String, Arrays, List, Set, Queue, ArrayList, LL, HS< LHS etc.  
 Example of Non-Linear DS: Graphs and trees.  
  
Primitive DS is very simple but our concentration on Non-Primitive type, where with one command we can Millions of data.  
Example: in an array or ArrayList by one variable we can store lots of datas.  
  
  
uses of Primitive DS: Calculator, snake game, currency converter  
uses of non-Primitive DS: Social media, banking website, e-commerce, for access of these all website or application we  
need to use data structure and algorithm.  
  
Collection framework:  
For using all the non-primitive DS we need a big structure where we can handle varieties of data.  
 Collection framework is a predefined API which is provided by JAVA.  
  
  
 Q: What is the difference between Arrays and Collection Framework:  
  
 Ans:  
  
 Arrays:  
 1). Arrays can store primitive and non-primitive (object) types of data.  
 such as; int [ ] a = {10, 20, 30}  
 or we can store object; Test [ ] = { obj1, obj2, obj3, obj4,........}  
 2). Array can store only homogeneous data, (same type).  
 3). Array size is fixed, we cannot increase or decrease the size of an array at runtime.  
 4). Array are inbuilt feature of java & thus we have to develop algorithm  
  
  
 Collection Framework:  
 1). Collection framework can contain only non-primitive type of data. Example: in an arrayList, if we use a.add[10];  
 here 10 is not an int it is an object. here 10 is an object not an int. It is Integer, wrapper class of int.  
 2). We can store heterogeneous (different) type of data.  
 3). We can increase of decrease the size of collections at runtime.  
 4). Collection framework is an API which provides predefined classes, interfaces and Methods.  
  
  
  
 Collection Framework:  
 It is an API.  
 Collection represents, it is the single entity or object which can store multiple data.  
 framework represents, it is the set of predefined classes & interfaces which is used to store multiple data.  
 It contains 2 main parts, a). java.util.Collection b). java.util.Map  
  
 In collection, we can enter date directly but in Map we can store data as a key value form.  
  
  
 Hierarchy of collection and Map:  
 Collection:  
 Collection are 3 types, a). List b). Set c). Queue these are interface  
 'List' divided into 4 types, 1). ArrayList (class) 2). LinkedList (class) 3). Vector (Legacy Class) 4). Stack (Legacy Class).  
 'set' divided into 5 types, 1). Hashset (class) 2). LinkedHashset (class) 3). SortedSet(interface) 4). NavigatedSet (interface)  
 5). Treeset (class)  
 'Queue' divided in 3 types 1). priorityQueue 2). Deque 3). ArrayDequeue.  
  
 Map:  
  
 map is divided into 10 parts:  
 1). HashMap (Class) 2). LinkedHashmap (Class) 3). identityHashMap (Class) 4). WeakHashMap (Class) 5). SortedHashMap interface)  
 6). NavigatedMap interface) 7). TreeMap (Class) 8). Dictionary (Class) 9). Hashtable (Class) 10). Properties (Class)  
  
Data structure and algorithm:*

*public class* ListAndSet {  
}  
*/\*  
Q: What is the difference between list and set?  
Ans:  
  
List:  
1). List is an index based data structure  
2). List can store duplicate elements.  
3). List can store any number of Null values.  
4). List follows the insertion order.  
5). We can iterate (get)the list elements by Iterator & ListIterator  
  
Set:  
1). Set is not an index-based data structure, it stored data according to the Hashcode value.  
2). Set does not allowed to store duplicate element.  
3). Set can store only one Null value.  
4). Set does not follow the insertion order.  
5). We can iterate the set elements by Iterator.*

*Selecttion or conditional statements  
 if, if-else, if-else-if, nested if.  
 Switch statement  
2. iteratio or looping statement  
 for, while, do-while  
3. Jump statement  
 break, continue, return  
  
 Syntax of Switch:  
  
 Switch(variable to be tested)  
 {  
 case (level1)  
 .  
 .  
 break;  
  
 case (level2)  
 .  
 .  
 break;  
  
 default:  
 .  
 .  
 break;  
  
 }  
  
  
Syntax for for loop:  
  
 for (initialization; condition; increment/decrement)  
 {  
 body  
 }  
  
While syntax:  
while (condition)  
{  
body  
}  
first it will check the codition, if the condition is true, then it will go to body, if codition is false, it will go  
out from the loop.  
  
do-while Syntax:  
  
do  
{  
body  
} while (condition)  
  
difference between while-do or do-while, answer: in while, first check condition then go to the body.  
on other hand, in do-while, first go to the body, then go for condition.*

*Q. Difference between Core Java and Advance java?  
  
  
Core Java:  
1). Version J2SE  
2). We learn basic fundamental concepts of Java programming.  
3). Single tier Architecture.  
4). We can develop CMD based or Desktop based application.  
5). Topics: control statement, Oops, Exception Handling, File I/O, Strings, Collection etc.  
  
Advance Java:  
1). Version J2EE  
2). We learn advance concepts like web and enterprise technology.  
3). Two tier Architecture (Client-server Architecture)  
4). We can develop web application i.e. enterprise Application.  
5). Technologies: JDBC, JSP, API, HTML, CSS, JS, etc.*

*Difference between Checked and Unchecked Exceptions?  
Always exceptions occurs in run time, never occurred in compile time.  
The exceptions which compiler can check, it is called compile time exception(Checked Exceptions) and exception compiler  
ignored or cannot check it is called runtime exceptions (Unchecked Exceptions).  
  
  
By using Try ,catch, finally, throw, and throws we can handle the exceptions.  
  
Syntax:  
  
try  
{  
//risky code  
}  
catch(ExceptionClassName ref.var.name), if we do not know what type of exception it is, we can use exception only  
{  
//handling code  
}*

*xception: It is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that  
disrupts the normal flow of the program.  
  
Exception handling is the alternative way to run the program normal flow.Not fix the error.  
  
  
  
Q: which is parent class of all classes in Java. Object is the parent class of java.  
  
Q: What is the difference between exception and error?  
Ans. Exceptions occurred by program and errors are occurred because of lack of System Resources; not by our programs and thus  
programmer cannot do anything.  
  
  
  
  
Exception hierarchy: |--> Compile-Time Exception(Checked Exception)  
 |--> Exception(1. program, 2. recoverable)--> |--> Runtime Exception(Unchecked Exception)  
Object --> Throwable--> |  
 |--> Errors(1. Lack of system resources, 2. not recoverable),it is only Runtime exception (unchecked)  
  
  
Exception:  
1. ClassNotFound Exception  
2. NoSuchMethod Exception  
3. IOExceptions (EOFExceptions, FileNotFoundExceptions, InterruptedIOException)  
4. Sequel Exception  
5. RunTime Exception (Arithmetic Exception, ClassNotFound, ...)  
6. Interrupted Exception  
  
  
  
Error:  
  
  
There are thousands of exceptions and errors available.  
 \*/*

*Identifier is any name, ti can be variable name, method name, class name or interface name.  
Example"  
  
int a = 10;  
here int is data type, a is variable = is operator and 10 is literal.  
a is variable and identifier also.  
  
Rules for identifiers:  
1. Spaces cannot be used.  
2. only \_ and $ can be used.  
3. Reserved words cannot be used.  
4. Integer value cannot be used in the first position, it can be used after the first characters.  
  
  
Naming convention:  
  
for variables: Two words can be used with '\_' or $ other symbols cannot be used. It should be in lower case letter.  
  
for method: we have to use camel case system. Example: my(), myInt(), or myIntDg() etc.  
  
for class/interface: always start with capital letter, Example: My, MyInterest, MyInterestName etc.  
 \*/  
public class* IdentifierExample {  
  
*/\*  
meaning of this structure:  
  
'class' is identifier that is a keyword. After then,  
under a single class we can make as many class as we can. But only one public class we can create.  
follow the naming convention.  
  
'IdentifierExample' is the class name.  
  
  
Another explanation of the main method  
  
public static void main (String [] args)  
  
  
here 'public' is an access modifier, there are 4 types of access modifiers available in JAVA.  
4 access modifiers are: public, private, protected and default.  
  
Why the main method is public?  
Answer: there is one reason that if we install JAVA in D drive and the JDK is available in C drive  
without public it will not able to access the JDK without public access modifier.  
  
What is the reason of Static keyword using?  
Answer: Static is a keyword also, main is the name of method, JVM can call this method without making any object due to this  
'static' keyword.  
  
What is void?  
Void is return type keyword. Void means, empty, not zero or null. So, the main method will not return anything. JDK always  
start execution from main method.  
  
What is String?  
String is a class. Why we use String type class, we could use int or other type of class. We are using the String because  
String cannot change and all processing in inside of JAVA always used String type.  
  
[] is array and arg is the array name.  
  
System.out.println();  
  
Q: What is System in the above command?  
Answer: System is a package which is available in Java.lang  
  
out: out is a static variable of print-stream, inside the System class and out for output.  
  
println is method, print Stream method. There are three types of print stream method, a) is print, b) is println and  
c) is printf method. 'print' method, prints in the same line, 'println' uses next line after printing and 'printf' is used  
  
  
If we change anything in these two lines, JVM will show compile time error. The program will not run.  
  
In main method we can make some changes like:  
  
  
a). instead of 'public static', we can write 'static public'  
b). we can declare any acceptable form like  
 (String[] args)  
 (String []args)  
 (String args[])  
c). we can give any identifier instead of 'args' .  
d). We can use var args instead of String[], like this 'String.. args'  
e). With 'main' we can use, final, synchronized, strictfp with main.  
  
  
in below the example is available.  
  
  
 \*/  
  
 static public void* main(String... abcd){  
 System.out.println("How are you?");  
 }  
}

*Q: Difference between Abstraction and Encapsulation:  
  
Ans:  
Abstraction:  
1). Abstraction is detail hiding(Implementation hiding)  
2). Data Abstraction deals with exposing the interface to the user and hiding the details of implementation.  
  
Encapsulation:  
1). Encapsulation is data hiding (information hiding)  
2). Encapsulation groups together data and methods that act upon the data.  
  
  
We can achieve abstract by :  
 a). By using Abstract class (0-100%)  
 b). By using interfaces (100%)  
  
To hide data we have to make abstract Class and Abstract Method inside that class.  
  
Note: If we make one abstract Method, the class must be abstract class, but inside of an Abstract Class it is not necessary to  
have an abstract method.  
Abstract method is that whose method; which has no body. If both method and class are abstract then we can consider that  
we achieved 100% hiding.  
  
If a regular class extends an abstract class, then the class must have to implement all the abstract methods of abstract parent class,  
or it has to be declared abstract as well.  
  
Abstract methods in an abstract class are meant to be override in derived concrete classes otherwise compile-time will be thrown.  
  
Abstract classes cannot be instantiated, means we cannot create an object of Abstract Class.  
\*/*

*Types of relationship between classes:  
  
There are two types of relationship: 1). Inheritance (it is called IS-A, relationship)  
 2). Association (it is called HAS-A relationship).  
  
 Association further can divide in two parts:  
 q). Aggregation  
 b). Composition  
  
 Advantages of relationship:  
 1). code re-usability  
 2). cost-cutting  
 3). reduce redundancy  
  
 Association (HAS-A)  
  
 Example:  
  
 class Engine  
 {  
  
 }  
 class Car  
 {  
 Engine en = new Engine(); // if wer make a new object inside other class, we can say they are  
 making association.  
 }  
  
 The main advantage of this association is that both classes are not tighlty held with each other.  
 So, there will not face any problem like Inheritance. We can choose some properties from one clas to another.  
  
  
 From association:  
  
 If there is a week relationship between two classes, then it is called aggregation  
 if there is a strong relationship between two classes, then it is called Composition.*

*\*  
  
1. Constructor is block (similar to method) having same as that of Class name.  
2. Constructor does not having return type, not even void.  
3. The only access modifier can use, public, protected, default and private  
4. constructor it executes automatically.  
  
  
Why we use constructor?  
Constructor are used to initialize an object.  
We had to write more lines in programming.  
  
How many types of constructor?  
1. Default constructor  
2. No argument constructor  
3. Parameterized constructor.  
  
  
1. Default constructor: It is also called no-argument constructor, but this constructor in not compile in JVM,  
 if there is no constructor in the program, the program itself make one constructor, that is hidden, this hidden  
 Constructor is called default constructor. This constructor is always no-argument constructor. Whenever user make  
 one constructor, then the program will not create any constructor.  
  
2. No-argument constructor: This type of constructor made by the programmer, in this constructor no argument is used.  
3. Parameterized constructor: This type of constructor made by the programmer, in this constructor parameter is passed.  
  
Q. What is the main function of constructor?  
Ans. its main function is to initialize the object. It is only initialized the object that's why there is no return.  
 That's why no need to used void.*

*What is Encapsulation:  
Ans. Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.  
  
Steps to achieve encapsulation:  
1. Declare all the variables of a class as private.  
2. Provide public setter and getter methods to modify and view the variable values.  
 \*/*

*public class* FinalKeyWordExample {  
}  
*/\*  
  
Where we can use 'final' keyword?  
variable, method that will not be overridden and class  
 \*/  
/\*  
class Test4  
{  
 // Example of use of 'final' in variable case.  
 public static void main(String[] args)  
 {  
 int i=10;  
 i += 20;  
 System.out.println(i);  
  
 //In this block, if we run the program, it will newly assign i= 30;  
 // if wer use 'final' keyword before i, then in the second line it will show error,  
 // we can use 'final' in case of constant values,like: pi, g, speed of light etc.  
  
 }  
  
}  
\*/  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
  
/\*  
'final' is used in method which cannot be overridden.  
 \*/  
/\*  
class Demo  
{  
 void mw()  
 { // if we use final before the method name 'mw()' as like final void mw(), it will show compile-time error.  
 System.out.println("I am in Demo");  
 }  
}  
  
class Test5 extends Demo  
{  
 void mw()  
 {  
 System.out.println(" i am in Test5 class");  
 }  
  
}  
  
\*/  
  
/\*  
if we wish to make a class and nobody can extend that class, at that time it will show error.  
  
Example:  
 \*/  
  
class* cv  
{  
  
 *void* bm()  
 {  
  
 }  
}  
  
*class* ih *extends* cv  
 {  
 *// if we use final before class CV. if we use final, then it will show error in the next class.*

*Q: What is Inheritance?  
Ans. It is inheriting the properties of parent class into child class  
Or  
inheritance is the procedure by which one object acquires all the properties and behaviors of a parent object.  
By this method we can call one method from other class by using extends.  
  
Child class, parent class.  
  
It is a IS-A relationship.  
  
Main advantage of inheritance is  
1. code usability.  
2. Achieve polymorphism using inheritance. i.e. method overriding.  
  
Main disadvantage of inheritance is that classes are tightly coupled, so, if we change in any class,  
that will affect all other classes.  
  
  
Types of inheritance:  
1. Single inheritance --> one class's properties inheritance to other one class  
2. Multi level --> one class's properties inheritance to other one class, that properties inheritance to other classes also.  
3. Hierarchical inheritance. --> one class's properties inheritance to other two class's at a time.  
4. Multiple inheritance --> 2 or more class's properties come in to one class.  
5. Hybrid inheritance. --> one class to multi class and multi class to one class.  
  
  
In java uses only 3 Inheritance, single, multi level and Hierarchical.  
  
In-general we can see properties of class A transfer to in class B.  
All the properties not transferred. Constructor cannot transfer and the method which one is private.*

*Using method, we can initialize the object.  
This second class is only for the training purpose  
this will show that we can initialize by method also.*

*Q: What is Interface?  
Interfaces are the blueprint of the class. It specifies what a class must do and not how.  
  
inside of an interface class all methods must be abstract.  
  
uses of interface:  
  
1. it is used to achieve abstraction  
2. it supports multiple inheritance (multiple inheritance, not supported by java).  
3. it can be achieved loose coupling.  
  
  
Syntax of interface:  
  
interface interfaceName  
{  
1. Methods it will be abstract public  
2. fields it will be public, static, final (example: if we put int s=4; it will automatically put public static final before int)  
3. in Java 8th version: we can create default concrete methods  
 we can create static methods. we cannot put protected or public.  
  
4. in Java 9th version: we can create private methods.  
  
}*

*Q. What is Opps?  
Oops means, Object Oriented Programming System / Structure  
This is a programming paradigm / methodology  
 There are lots of methods for this paradigm  
 a. Object Oriented paradigm  
 b. Procedural paradigm  
 c. functional paradigm  
 d. logical paradigm  
 e. structural paradigm  
  
Q. What are the languages follow the Oops?  
Ans.  
a). Smalltalk: this is the first Oops program which followed 100% Oops. Truly Object Oriented Programming  
b). Java c). C# d). C++ e). Python  
 a. Object-Oriented paradigm:  
  
Q. How many pillars of Oops?  
There are six main pillars of Oops  
 1. Class  
 2. Object & Methods  
 3. Inheritance  
 4. Polymorphism  
 5. Abstraction  
 6. Encapsulation  
  
  
1. Class: Collection of Objects (example vehicle, animal, Birds)  
 Class is not a real world entity. It is just a blueprint or prototype of template  
 class does not occupy memory  
  
 Syntax of Class:  
 access modifier class ClassName  
 {  
 inside the body it will contain:  
 Methods  
 Constructor  
 fields  
 blocks  
 nested class  
 }  
2. Method: A set of code which performs a particular task (Examples: run, eat, fly etc.)  
 Advantage of methods  
 code reputability  
 code optimization  
 Syntax: access-modifiers return types method name (list of parameters)  
 {  
  
  
 }  
3. Object: Object is an instance of class. O  
 Object is real world entity  
 Object occupies memory.  
 Object consists of  
 1. identity-- name -- unique  
 2. State / Attribute (Example: color, age, breed of dog)  
 3. Behaviour (Example: eat, runs) it represents methods  
 How to create object?  
 by using new keyword  
 by using newInstance Method()  
 by using clone() method  
 by using deserialization method  
 by using factory method  
  
 creation of object:  
 1). Declaration --> Animal buzo  
 2). Instantiation --> buzo = new Animal(); Here, Animal() is a constructor, and initialized. i.e. 2 & 3 combined.  
 3). initialization  
  
 Example of Syntax combined:  
 Animal buzo = new Animal();  
  
 we can call method through this object, example: buzo.run(); or buzo.eat();  
  
 \*/  
public class* OopMthd  
{  
 *public void* eat()  
 {  
 System.out.println("I am eating"); *// it will not run without object.* }  
 *public static void* main(String [] args)  
 {  
 System.out.println("1");  
 OopMthd obj = *new* OopMthd(); *// to run the method named 'eat', we have to make an object.* obj.eat();  
 obj.run();  
*// No matter what method is in outside before or after main method, it depends on the sequence of calling.* OopMthd aa = *new* OopMthd();  
 aa.eat();  
 aa.run();  
 *int* b= aa.addition(78,67);  
 System.out.println(b);  
 Birds spa = *new* Birds();  
 spa.fly();  
  
 }  
 *public void* run()  
 {  
 System.out.println("I am running");  
 }  
 *int* addition(*int* alj, *int* k)  
 {  
 *return* alj+k;  
 }  
  
 *static class* Birds  
 {  
 *void* fly()  
 {  
 System.out.println(" I am flying");  
 }  
 }  
}

*What is polymorphism?  
  
polymorphism is the many forms.  
Example: water in many forms, steam, ice and liquid water  
 Sound is forms of dog sound, cat sound, musical sound etc.  
  
 Polymorphism is two types:  
 1. Compile type polymorphism, it is called Static polymorphism. We can achieve compile type polymorphism by  
 method overloading. It is handled by compiler.  
 2. Run time polymorphism, it is called dynamic polymorphism. It is achieved by overriding method. It is handled by JVM.  
  
Method overloading: have to satisfy all conditions  
1. Same name  
2. same class  
3. Different argument  
 we can make different by :  
 a. no. of argument  
 b. sequence of argument  
 c. Type of argument  
  
Method overriding: have to satisfy all conditions  
1. Same name  
2. Different class  
3. Same argument  
 we can make same argument by:  
 a. same no. of argument  
 b. same sequence of argument  
 c. Type of argument  
4. Inheritance (IS-A) relationship must be in both methods.  
  
 \*/  
//This is the example of method loading:  
public class* PolymorphismExmple  
{  
 *void* show(*int* a)  
 {  
 System.out.println("1") ;  
 }  
 *void* show() *// Here 2 methods are same name, in same class but the argument is different.* {  
 System.out.println("2");  
 }  
 *public static void* main(String [] agrs)  
 {  
 PolymorphismExmple ob1 = *new* PolymorphismExmple();  
 ob1.show();  
 }  
}

*Super keyWord, it is also the keyword for reference variable which is used to refer immediate parent class object.  
  
1). 'super' keyword can be used to refer immediate parent class instance variable.  
  
\*/  
  
  
// The fist uses of super keyword example.  
  
  
/\*  
class A1  
{  
 int a=10;  
}  
class B1 extends A1  
{  
 int a = 20;  
 void show (int a)  
 {  
 System.out.println(a); // output = 34, this is form the method show (int a), we made object of B and initialized by a=34,  
 System.out.println(this.a);// output = 20, this is from its current class, a is its instance variable.  
 System.out.println(super.a); // output = 10, this is from the parent class instance variable.  
  
 }  
 public static void main(String [] args)  
 {  
 B1 obj = new B1();  
 obj.show(34);  
 }  
}  
  
\*/  
  
/\*  
Uses of 'super' keyword:  
  
1). 'super' keyword can be used to refer immediate parent class instance variable.  
2). 'super' keyword can be used to invoke immediate parent class method.  
3). super() can be used to invoke immediate parent class constructor.  
 \*/  
  
  
  
// 2). 'super' keyword can be used to invoke immediate parent class method. Example:  
  
  
/\*  
  
class Ab  
{  
 void ma ()  
 {  
 System.out.println("I am in Class Ab");  
 }  
  
}  
class Ba extends Ab  
{  
  
 void ma()  
 {  
 System.out.println("I am in Class Ba ");  
 }  
 void show()  
 {  
 ma(); // if we call directly then it will call own class method, its output will be (I am in Class Ba).  
 super.ma();// if we call through 'super', then it will call parent class method, the output will be (I am in Class Ab).  
 }  
  
 public static void main(String [] args)  
 {  
 Ba obj = new Ba();  
 obj.show();  
 }  
}  
  
\*/  
  
// 3). super() can be used to invoke immediate parent class constructor. Example:  
  
  
  
class* Ab  
{  
 Ab ()  
 {  
 System.out.println("I am in Class Ab");  
 }  
  
}  
*class* Ba *extends* Ab  
{  
 Ba()  
 {  
 *//here compiler use another 'super () ' constructor  
 // when we call only 'super()' it calls both constractors Ab and Ba.  
 // super();* System.out.println("I am in Class Ba ");  
 }  
 *public static void* main(String [] args)  
 {  
 Ba obj = *new* Ba();  
  
 }  
}

*Q: What is this key word?  
In Oops there is an object. The variable which refers to that object is called reference variable. "this" keyword is the  
reference variable that refers to the current object.  
  
uses of "this" keyword:  
  
1. this keyword can be uses to refer current class instance variable.  
2. this keyword can be used to invoke current class method (implicitly).  
3. this() can be used to invoke current class constructor.  
4. this can be used to pass as an argument in the method call.  
5. this can be used to pass as an argument in the constructor call.  
6. this can be used to return the current class instance from the method.  
  
Note that if we don't use the 'this' keyword, compiler automatically adds this keyword while invoking the method.*

*What is operator?  
C=A+B; here, '=' and '+' are operators, A and B are operands. Operators are more special symbols which operate operation  
on one more operands. '=' is the assignment operator. Arithmetic operators, relational operators. logical operators  
bitwise operators. miscellaneous operators.  
  
int a= 10;  
String name = "Babul";  
boolean x = true;  
  
here, int, String, boolean all are data type  
a, name and x are variables  
= is operator  
and 10, Babul, true all are literals.  
  
  
Keywords, there are 50 keywords, 48 keywords we are using but 2 keywords we cannot use, these are 'goto' and 'const'.  
 \*/*

*Local variables:  
  
1). Declaration: inside the Methods, constructor or blocks.  
2). Scope: inside the Methods, constructor or blocks not outside.  
3). When variables Gets Allocated: When method, constructor or block  
 gets executed, variables allocated memory. When gets exits, variables destroyed.  
  
4). Stored Memory: Stack Memory.  
5). Default values: There is no default values, it must be initialized. Value should be provided before used.  
6). Access pacifier: Access pacifiers cannot be used with local variables.  
  
  
Instance variables:  
  
1). Declaration: inside the class but outside Methods, constructor or blocks.  
2). Scope: Within a class, Inside all Methods, constructor or blocks not inside the Static method.  
3). When variables Gets Allocated: When object is created, variables allocated. When object destroyed variable releases memory.  
4). Stored Memory: Heap Memory.  
5). Default values: There default values available, for int, 0; for boolean, false; for object, null;  
6). Access pacifier: Access pacifiers can be used.  
7). How to use: For simple method, we can call directly.  
 For Static method, we can call through object. Example:  
 A ob = new A;  
 ob.a; where 'A' is class name and 'a' is instance variable.  
  
  
  
Static variables:  
  
1). Declaration: With 'static' keyword, inside the class but outside Methods, constructor or blocks.  
2). Scope: Similar to Instance variable i.e. Inside all Methods, constructor or blocks including Static methods, constructor or blocks.  
3). When variables Gets Allocated: When we run the program and .class (dot class) file is loaded, variable allocated  
 When .class file unloaded, variable gets deleted/ destroyed.  
4). Stored Memory: Non-Heap Memory / Static Memory.  
5). Default values: Similar to instance variables. There default values available, for int, 0; for boolean, false; for object, null;  
6). Access pacifier: Access pacifiers can be used.  
7). How to use: a. For simple method, we can call directly.  
 b. by using class name Example: A.a  
 c. by using object reference name, Example: A obj = new A;  
 obj.a; where 'A' is class name and 'a' is instance variable.*

*What is the reason for the new value of 'a', it is not changed for the new object but Static variables  
holds its value. Every new object instance variable through its original declared value. This is the difference between  
instance variable and Static variables.  
 \*/*

# Binary Search

Binar\_Search;

/\*Binary search is a searching method where it will search by

dividing the total array length. The array should be sorted array.

Sorted array means ascending order.

Binar\_Search;

// This is linear search because, it will search one by one element from 0 index to the last index.

//it will search linearly.

# Collection

public class CollectionInterfaceExmple {

}

/\*

1). Collection is an interface which is present in java.util package

2). Syntax of Collection:

public interface Collection<E> extends Interface<E>

{

---

----

}

3). Hierarchy of collection interface

4). Methods of collection interface

-public boolean add(Object obj)

-public boolean addAll(Collection c)

-public boolean remove

-boolean contains(object obj)

-boolean isEmpty()

-int size()

# Data Structure

public class DataStructure {

}

/\*

Q: What is data Structure?

Ans. Data Structure is way by which we can store the data in efficient way. As We can retrieve and access.

Tje efficient is the matter of time and space.

Types of data Structure:

1) Primitive Data Structure

In Primitive Data Structure, we have (Boolean, char, byte,short, int, long, float, double)

2). Non-primitive DS

Non-primitive DS is two types:

a). Linear DS b). Non-Linear DS

Example of Linear DS: String, Arrays, List, Set, Queue, ArrayList, LL, HS< LHS etc.

Example of Non-Linear DS: Graphs and trees.

Primitive DS is very simple but our concentration on Non-Primitive type, where with one command we can Millions of data.

Example: in an array or ArrayList by one variable we can store lots of datas.

uses of Primitive DS: Calculator, snake game, currency converter

uses of non-Primitive DS: Social media, banking website, e-commerce, for access of these all website or application we

need to use data structure and algorithm.

Collection Framework

Collection framework:

For using all the non-primitive DS we need a big structure where we can handle varieties of data.

Collection framework is a predefined API which is provided by JAVA.

Q: What is the difference between Arrays and Collection Framework:

Ans:

Arrays:

1). Arrays can store primitive and non-primitive (object) types of data.

such as; int [ ] a = {10, 20, 30}

or we can store object; Test [ ] = { obj1, obj2, obj3, obj4,........}

2). Array can store only homogeneous data, (same type).

3). Array size is fixed, we cannot increase or decrease the size of an array at runtime.

4). Array are inbuilt feature of java & thus we have to develop algorithm

# 

1). Collection framework can contain only non-primitive type of data. Example: in an arrayList, if we use a.add[10];

here 10 is not an int it is an object. here 10 is an object not an int. It is Integer, wrapper class of int.

2). We can store heterogeneous (different) type of data.

3). We can increase of decrease the size of collections at runtime.

4). Collection framework is an API which provides predefined classes, interfaces and Methods.

# Collection Framework:

Collection Framework:

It is an API.

Collection represents, it is the single entity or object which can store multiple data.

framework represents, it is the set of predefined classes & interfaces which is used to store multiple data.

It contains 2 main parts, a). java.util.Collection b). java.util.Map

In collection, we can enter date directly but in Map we can store data as a key value form.

Hierarchy of collection and Map:

Collection:

Collection are 3 types, a). List b). Set c). Queue these are interface

'List' divided into 4 types, 1). ArrayList (class) 2). LinkedList (class) 3). Vector (Legacy Class) 4). Stack (Legacy Class).

'set' divided into 5 types, 1). Hashset (class) 2). LinkedHashset (class) 3). SortedSet(interface) 4). NavigatedSet (interface)

5). Treeset (class)

'Queue' divided in 3 types 1). priorityQueue 2). Deque 3). ArrayDequeue.

# Map:

Map is divided into 10 parts:

1). HashMap (Class) 2). LinkedHashmap (Class) 3). identityHashMap (Class) 4). WeakHashMap (Class) 5). SortedHashMap interface)

6). NavigatedMap interface) 7). TreeMap (Class) 8). Dictionary (Class) 9). Hashtable (Class) 10). Properties (Class)

Data structure and algorithm:

# List and Set

public class ListAndSet {

}

/\*

Q: What is the difference between list and set?

Ans:

List:

1). List is an index based data structure

2). List can store duplicate elements.

3). List can store any number of Null values.

4). List follows the insertion order.

5). We can iterate (get)the list elements by Iterator & ListIterator

Set:

1). Set is not an index-based data structure, it stored data according to the Hashcode value.

2). Set does not allowed to store duplicate element.

3). Set can store only one Null value.

4). Set does not follow the insertion order.

5). We can iterate the set elements by Iterator.

# Conditional Statement

Selecttion or conditional statements

if, if-else, if-else-if, nested if.

Switch statement

2. iteratio or looping statement

for, while, do-while

3. Jump statement

break, continue, return

Syntax of Switch:

Switch(variable to be tested)

{

case (level1)

.

.

break;

case (level2)

.

.

break;

default:

.

.

break;

}

Syntax for for loop:

for (initialization; condition; increment/decrement)

{

body

}

While syntax:

while (condition)

{

body

}

first it will check the codition, if the condition is true, then it will go to body, if codition is false, it will go

out from the loop.

do-while Syntax:

do

{

body

} while (condition)

difference between while-do or do-while, answer: in while, first check condition then go to the body.

on other hand, in do-while, first go to the body, then go for condition.

# Core Java and Advance java

Q. Difference between Core Java and Advance java?

Core Java:

1). Version J2SE

2). We learn basic fundamental concepts of Java programming.

3). Single tier Architecture.

4). We can develop CMD based or Desktop based application.

5). Topics: control statement, Oops, Exception Handling, File I/O, Strings, Collection etc.

Advance Java:

1). Version J2EE

2). We learn advance concepts like web and enterprise technology.

3). Two tier Architecture (Client-server Architecture)

4). We can develop web application i.e. enterprise Application.

5). Technologies: JDBC, JSP, API, HTML, CSS, JS, etc.

# Exceptions

Difference between Checked and Unchecked Exceptions?

Always exceptions occurs in run time, never occurred in compile time.

The exceptions which compiler can check, it is called compile time exception(Checked Exceptions) and exception compiler

ignored or cannot check it is called runtime exceptions (Unchecked Exceptions).

By using Try ,catch, finally, throw, and throws we can handle the exceptions.

Syntax:

try

{

//risky code

}

catch(ExceptionClassName ref.var.name), if we do not know what type of exception it is, we can use exception only

{

//handling code

}

Exception: It is an unwanted or unexpected event, which occurs during the execution of a program i.e. at run time, that

disrupts the normal flow of the program.

Exception handling is the alternative way to run the program normal flow. Not fix the error.

Q: which is parent class of all classes in Java. Object is the parent class of java.

Q: What is the difference between exception and error?

Ans. Exceptions occurred by program and errors are occurred because of lack of System Resources; not by our programs and thus

programmer cannot do anything.

Exception hierarchy: |--> Compile-Time Exception(Checked Exception)

|--> Exception(1. program, 2. recoverable)--> |--> Runtime Exception(Unchecked Exception)

Object --> Throwable--> |

|--> Errors(1. Lack of system resources, 2. not recoverable),it is only Runtime exception (unchecked)

Exception:

1. ClassNotFound Exception

2. NoSuchMethod Exception

3. IOExceptions (EOFExceptions, FileNotFoundExceptions, InterruptedIOException)

4. Sequel Exception

5. RunTime Exception (Arithmetic Exception, ClassNotFound, ...)

6. Interrupted Exception

There are thousands of exceptions and errors available.

# Identifier

Identifier is any name, ti can be variable name, method name, class name or interface name.

Example"

int a = 10;

here int is data type, a is variable = is operator and 10 is literal.

a is variable and identifier also.

Rules for identifiers:

1. Spaces cannot be used.

2. only \_ and $ can be used.

3. Reserved words cannot be used.

4. Integer value cannot be used in the first position, it can be used after the first characters.

# Naming convention:

for variables: Two words can be used with '\_' or $ other symbols cannot be used. It should be in lower case letter.

for method: we have to use camel case system. Example: my(), myInt(), or myIntDg() etc.

for class/interface: always start with capital letter, Example: My, MyInterest, MyInterestName etc.

\*/

public class IdentifierExample {

/\*

meaning of this structure:

'class' is identifier that is a keyword. After then,

under a single class we can make as many class as we can. But only one public class we can create.

follow the naming convention.

'IdentifierExample' is the class name.

Another explanation of the main method

public static void main (String [] args)

# Access Modifier

here 'public' is an access modifier, there are 4 types of access modifiers available in JAVA.

4 access modifiers are: public, private, protected and default.

Why the main method is public?

Answer: there is one reason that if we install JAVA in D drive and the JDK is available in C drive

without public it will not able to access the JDK without public access modifier.

What is the reason of Static keyword using?

Answer: Static is a keyword also, main is the name of method, JVM can call this method without making any object due to this

'static' keyword.

What is void?

Void is return type keyword. Void means, empty, not zero or null. So, the main method will not return anything. JDK always

start execution from main method.

What is String?

String is a class. Why we use String type class, we could use int or other type of class. We are using the String because

String cannot change and all processing in inside of JAVA always used String type.

[] is array and arg is the array name.

System.out.println();

Q: What is System in the above command?

Answer: System is a package which is available in Java.lang

out: out is a static variable of print-stream, inside the System class and out for output.

println is method, print Stream method. There are three types of print stream method, a) is print, b) is println and

c) is printf method. 'print' method, prints in the same line, 'println' uses next line after printing and 'printf' is used

If we change anything in these two lines, JVM will show compile time error. The program will not run.

In main method we can make some changes like:

a). instead of 'public static', we can write 'static public'

b). we can declare any acceptable form like

(String[] args)

(String []args)

(String args[])

c). we can give any identifier instead of 'args' .

d). We can use var args instead of String[], like this 'String.. args'

e). With 'main' we can use, final, synchronized, strictfp with main.

in below the example is available.

\*/

static public void main(String... abcd){

System.out.println("How are you?");

}

}

# Abstraction and Encapsulation

Q: Difference between Abstraction and Encapsulation:

Ans:

Abstraction:

1). Abstraction is detail hiding(Implementation hiding)

2). Data Abstraction deals with exposing the interface to the user and hiding the details of implementation.

Encapsulation:

1). Encapsulation is data hiding (information hiding)

2). Encapsulation groups together data and methods that act upon the data.

We can achieve abstract by :

a). By using Abstract class (0-100%)

b). By using interfaces (100%)

To hide data we have to make abstract Class and Abstract Method inside that class.

Note: If we make one abstract Method, the class must be abstract class, but inside of an Abstract Class it is not necessary to

have an abstract method.

Abstract method is that whose method; which has no body. If both method and class are abstract then we can consider that

we achieved 100% hiding.

If a regular class extends an abstract class, then the class must have to implement all the abstract methods of abstract parent class,

or it has to be declared abstract as well.

Abstract methods in an abstract class are meant to be override in derived concrete classes otherwise compile-time will be thrown.

Abstract classes cannot be instantiated, means we cannot create an object of Abstract Class.

\*/

# Relationship

Types of relationship between classes:

There are two types of relationship: 1). Inheritance (it is called IS-A, relationship)

2). Association (it is called HAS-A relationship).

Association further can divide in two parts:

q). Aggregation

b). Composition

Advantages of relationship:

1). code re-usability

2). cost-cutting

3). reduce redundancy

Association (HAS-A)

Example:

class Engine

{

}

class Car

{

Engine en = new Engine(); // if wer make a new object inside other class, we can say they are

making association.

}

The main advantage of this association is that both classes are not tighlty held with each other.

So, there will not face any problem like Inheritance. We can choose some properties from one class to another.

From association:

If there is a week relationship between two classes, then it is called aggregation

if there is a strong relationship between two classes, then it is called Composition.

\*

# Constructor

1. Constructor is block (similar to method) having same as that of Class name.

2. Constructor does not having return type, not even void.

3. The only access modifier can use, public, protected, default and private

4. Constructor, it executes automatically.

Why we use constructor?

Constructor are used to initialize an object.

We had to write more lines in programming.

How many types of constructor?

1. Default constructor

2. No argument constructor

3. Parameterized constructor.

1. Default constructor: It is also called no-argument constructor, but this constructor in not compile in JVM,

if there is no constructor in the program, the program itself make one constructor, that is hidden, this hidden

Constructor is called default constructor. This constructor is always no-argument constructor. Whenever user make

one constructor, then the program will not create any constructor.

2. No-argument constructor: This type of constructor made by the programmer, in this constructor no argument is used.

3. Parameterized constructor: This type of constructor made by the programmer, in this constructor parameter is passed.

Q. What is the main function of constructor?

Ans. its main function is to initialize the object. It is only initialized the object that's why there is no return.

That's why no need to used void.

# Encapsulation

What is Encapsulation:

Ans. Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.

Steps to achieve encapsulation:

1. Declare all the variables of a class as private.

2. Provide public setter and getter methods to modify and view the variable values.

\*/

public class FinalKeyWordExample {

}

/\*

# 'final' keyword

Where we can use 'final' keyword?

variable, method that will not be overridden and class

\*/

/\*

class Test4

{

// Example of use of 'final' in variable case.

public static void main(String[] args)

{

int i=10;

i += 20;

System.out.println(i);

//In this block, if we run the program, it will newly assign i= 30;

// if wer use 'final' keyword before i, then in the second line it will show error,

// we can use 'final' in case of constant values,like: pi, g, speed of light etc.

}

}

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

'final' is used in method which cannot be overridden.

\*/

/\*

class Demo

{

void mw()

{ // if we use final before the method name 'mw()' as like final void mw(), it will show compile-time error.

System.out.println("I am in Demo");

}

}

class Test5 extends Demo

{

void mw()

{

System.out.println(" i am in Test5 class");

}

}

\*/

/\*

if we wish to make a class and nobody can extend that class, at that time it will show error.

Example:

\*/

class cv

{

void bm()

{

}

}

class ih extends cv

{

// if we use final before class CV. if we use final, then it will show error in the next class.

# Inheritance

Q: What is Inheritance?

Ans. It is inheriting the properties of parent class into child class

Or

inheritance is the procedure by which one object acquires all the properties and behaviors of a parent object.

By this method we can call one method from other class by using extends.

Child class, parent class.

It is a IS-A relationship.

Main advantage of inheritance is

1. code usability.

2. Achieve polymorphism using inheritance. i.e. method overriding.

Main disadvantage of inheritance is that classes are tightly coupled, so, if we change in any class,

that will affect all other classes.

Types of inheritance:

1. Single inheritance --> one class's properties inheritance to other one class

2. Multi level --> one class's properties inheritance to other one class, that properties inheritance to other classes also.

3. Hierarchical inheritance. --> one class's properties inheritance to other two class's at a time.

4. Multiple inheritance --> 2 or more class's properties come in to one class.

5. Hybrid inheritance. --> one class to multi class and multi class to one class.

In java uses only 3 Inheritance, single, multi level and Hierarchical.

In-general we can see properties of class A transfer to in class B.

All the properties not transferred. Constructor cannot transfer and the method which one is private.

Using method, we can initialize the object.

This second class is only for the training purpose

this will show that we can initialize by method also.

# Interface

Q: What is Interface?

Interfaces are the blueprint of the class. It specifies what a class must do and not how.

inside of an interface class all methods must be abstract.

uses of interface:

1. it is used to achieve abstraction

2. it supports multiple inheritance (multiple inheritance, not supported by java).

3. it can be achieved loose coupling.

Syntax of interface:

interface interfaceName

{

1. Methods it will be abstract public

2. fields it will be public, static, final (example: if we put int s=4; it will automatically put public static final before int)

3. in Java 8th version: we can create default concrete methods

we can create static methods. we cannot put protected or public.

4. in Java 9th version: we can create private methods.

}

# Opps

Q. What is Opps?

Oops means, Object Oriented Programming System / Structure

This is a programming paradigm / methodology

There are lots of methods for this paradigm

a. Object Oriented paradigm

b. Procedural paradigm

c. functional paradigm

d. logical paradigm

e. structural paradigm

Q. What are the languages follow the Oops?

Ans.

a). Smalltalk: this is the first Oops program which followed 100% Oops. Truly Object Oriented Programming

b). Java c). C# d). C++ e). Python

a. Object-Oriented paradigm:

Q. How many pillars of Oops?

There are six main pillars of Oops

1. Class

2. Object & Methods

3. Inheritance

4. Polymorphism

5. Abstraction

6. Encapsulation

1. Class: Collection of Objects (example vehicle, animal, Birds)

Class is not a real world entity. It is just a blueprint or prototype of template

class does not occupy memory

Syntax of Class:

access modifier class ClassName

{

inside the body it will contain:

Methods

Constructor

fields

blocks

nested class

}

2. Method: A set of code which performs a particular task (Examples: run, eat, fly etc.)

Advantage of methods

code reputability

code optimization

Syntax: access-modifiers return types method name (list of parameters)

{

}

3. Object: Object is an instance of class. O

Object is real world entity

Object occupies memory.

Object consists of

1. identity-- name -- unique

2. State / Attribute (Example: color, age, breed of dog)

3. Behaviour (Example: eat, runs) it represents methods

How to create object?

by using new keyword

by using newInstance Method()

by using clone() method

by using deserialization method

by using factory method

creation of object:

1). Declaration --> Animal buzo

2). Instantiation --> buzo = new Animal(); Here, Animal() is a constructor, and initialized. i.e. 2 & 3 combined.

3). initialization

Example of Syntax combined:

Animal buzo = new Animal();

we can call method through this object, example: buzo.run(); or buzo.eat();

\*/

public class OopMthd

{

public void eat()

{

System.out.println("I am eating"); // it will not run without object.

}

public static void main(String [] args)

{

System.out.println("1");

OopMthd obj = new OopMthd(); // to run the method named 'eat', we have to make an object.

obj.eat();

obj.run();

// No matter what method is in outside before or after main method, it depends on the sequence of calling.

OopMthd aa = new OopMthd();

aa.eat();

aa.run();

int b= aa.addition(78,67);

System.out.println(b);

Birds spa = new Birds();

spa.fly();

}

public void run()

{

System.out.println("I am running");

}

int addition(int alj, int k)

{

return alj+k;

}

static class Birds

{

void fly()

{

System.out.println(" I am flying");

}

}

}

# Polymorphism

What is polymorphism?

polymorphism is the many forms.

Example: water in many forms, steam, ice and liquid water

Sound is forms of dog sound, cat sound, musical sound etc.

Polymorphism is two types:

1. Compile type polymorphism, it is called Static polymorphism. We can achieve compile type polymorphism by

method overloading. It is handled by compiler.

2. Run time polymorphism, it is called dynamic polymorphism. It is achieved by overriding method. It is handled by JVM.

Method overloading: have to satisfy all conditions

1. Same name

2. same class

3. Different argument

we can make different by :

a. no. of argument

b. sequence of argument

c. Type of argument

Method overriding: have to satisfy all conditions

1. Same name

2. Different class

3. Same argument

we can make same argument by:

a. same no. of argument

b. same sequence of argument

c. Type of argument

4. Inheritance (IS-A) relationship must be in both methods.

\*/

//This is the example of method loading:

public class PolymorphismExmple

{

void show(int a)

{

System.out.println("1") ;

}

void show() // Here 2 methods are same name, in same class but the argument is different.

{

System.out.println("2");

}

public static void main(String [] agrs)

{

PolymorphismExmple ob1 = new PolymorphismExmple();

ob1.show();

}

}

Super keyWord, it is also the keyword for reference variable which is used to refer immediate parent class object.

1). 'super' keyword can be used to refer immediate parent class instance variable.

\*/

// The fist uses of super keyword example.

/\*

class A1

{

int a=10;

}

class B1 extends A1

{

int a = 20;

void show (int a)

{

System.out.println(a); // output = 34, this is form the method show (int a), we made object of B and initialized by a=34,

System.out.println(this.a);// output = 20, this is from its current class, a is its instance variable.

System.out.println(super.a); // output = 10, this is from the parent class instance variable.

}

public static void main(String [] args)

{

B1 obj = new B1();

obj.show(34);

}

}

\*/

/\*

# Uses of 'super' keyword:

1). 'super' keyword can be used to refer immediate parent class instance variable.

2). 'super' keyword can be used to invoke immediate parent class method.

3). super() can be used to invoke immediate parent class constructor.

\*/

// 2). 'super' keyword can be used to invoke immediate parent class method. Example:

/\*

class Ab

{

void ma ()

{

System.out.println("I am in Class Ab");

}

}

class Ba extends Ab

{

void ma()

{

System.out.println("I am in Class Ba ");

}

void show()

{

ma(); // if we call directly then it will call own class method, its output will be (I am in Class Ba).

super.ma();// if we call through 'super', then it will call parent class method, the output will be (I am in Class Ab).

}

public static void main(String [] args)

{

Ba obj = new Ba();

obj.show();

}

}

\*/

// 3). super() can be used to invoke immediate parent class constructor. Example:

class Ab

{

Ab ()

{

System.out.println("I am in Class Ab");

}

}

class Ba extends Ab

{

Ba()

{

//here compiler use another 'super () ' constructor

// when we call only 'super()' it calls both constractors Ab and Ba.

// super();

System.out.println("I am in Class Ba ");

}

public static void main(String [] args)

{

Ba obj = new Ba();

}

}

# ‘this’ key word

Q: What is this key word?

In Oops there is an object. The variable which refers to that object is called reference variable. "this" keyword is the

reference variable that refers to the current object.

uses of "this" keyword:

1. this keyword can be uses to refer current class instance variable.

2. this keyword can be used to invoke current class method (implicitly).

3. this() can be used to invoke current class constructor.

4. this can be used to pass as an argument in the method call.

5. this can be used to pass as an argument in the constructor call.

6. this can be used to return the current class instance from the method.

Note that if we don't use the 'this' keyword, compiler automatically adds this keyword while invoking the method.

# Operator

What is operator?

C=A+B; here, '=' and '+' are operators, A and B are operands. Operators are more special symbols which operate operation

on one more operands. '=' is the assignment operator. Arithmetic operators, relational operators. logical operators

bitwise operators. miscellaneous operators.

int a= 10;

String name = "Babul";

boolean x = true;

here, int, String, boolean all are data type

a, name and x are variables

= is operator

and 10, Babul, true all are literals.

Keywords, there are 50 keywords, 48 keywords we are using but 2 keywords we cannot use, these are 'goto' and 'const'.

\*/

# Variables

Local variables:

1). Declaration: inside the Methods, constructor or blocks.

2). Scope: inside the Methods, constructor or blocks not outside.

3). When variables Gets Allocated: When method, constructor or block

gets executed, variables allocated memory. When gets exits, variables destroyed.

4). Stored Memory: Stack Memory.

5). Default values: There is no default values, it must be initialized. Value should be provided before used.

6). Access pacifier: Access pacifiers cannot be used with local variables.

Instance variables:

1). Declaration: inside the class but outside Methods, constructor or blocks.

2). Scope: Within a class, Inside all Methods, constructor or blocks not inside the Static method.

3). When variables Gets Allocated: When object is created, variables allocated. When object destroyed variable releases memory.

4). Stored Memory: Heap Memory.

5). Default values: There default values available, for int, 0; for boolean, false; for object, null;

6). Access pacifier: Access pacifiers can be used.

7). How to use: For simple method, we can call directly.

For Static method, we can call through object. Example:

A ob = new A;

ob.a; where 'A' is class name and 'a' is instance variable.

Static variables:

1). Declaration: With 'static' keyword, inside the class but outside Methods, constructor or blocks.

2). Scope: Similar to Instance variable i.e. Inside all Methods, constructor or blocks including Static methods, constructor or blocks.

3). When variables Gets Allocated: When we run the program and .class (dot class) file is loaded, variable allocated

When .class file unloaded, variable gets deleted/ destroyed.

4). Stored Memory: Non-Heap Memory / Static Memory.

5). Default values: Similar to instance variables. There default values available, for int, 0; for boolean, false; for object, null;

6). Access pacifier: Access pacifiers can be used.

7). How to use: a. For simple method, we can call directly.

b. by using class name Example: A.a

c. by using object reference name, Example: A obj = new A;

obj.a; where 'A' is class name and 'a' is instance variable.

What is the reason for the new value of 'a', it is not changed for the new object but Static variables

holds its value. Every new object instance variable through its original declared value. This is the difference between

instance variable and Static variables.

\*/