Java

C++ and Java:

**C++** - Pointers are present in C++, manual memory management, C++ uses compiler to convert code to machine level language

C++ is not purely object oriented as we can write some logic without a class or object

C++ has Multiple inheritance

**Java** – purely object oriented ,Do not have pointers, platform independent(WORA),   
Java uses compiler and interpreter

Garbage collection feature which takes care of memory management,

Good exception handling mechanism,Java has ways to achieve multiplt inheritance though it do not directlysupport multiple inhertance

**Features of Java:**

Simple

Object Oriented

Distributed

Robust

System Independent

Secure

MultiThreaded

Download Java:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

run the exe file and you will see Java folder with jdk and Jre in your c:/program files.

Set environment variables:

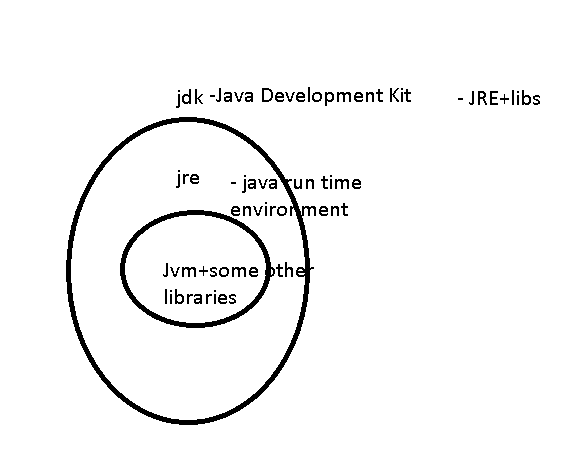
Right click on system- properties- advanced system variabled- system properties-

New-

JAVA\_HOME- C:\Program Files\Java\jdk1.8.0\_45

Edit existing variable – path: add - C:\Program Files\Java\jdk1.8.0\_45\bin

Java is system independent but JVM is not system independent.



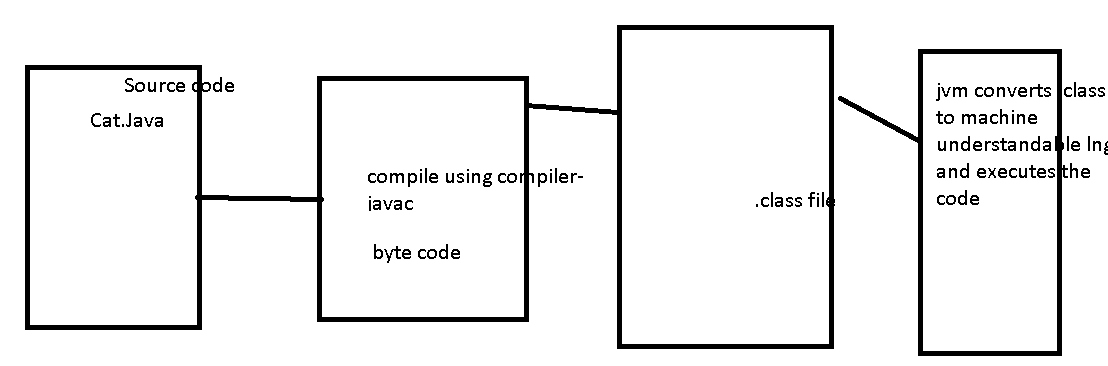
Start to Code:

Write some logic and save it as .java

Compile the code – compiler will generate .class file (byte code) only if there are no errors in the logic.

Load the class into memory – class loader

run the program- JVM will run the program- it will convert the byte code into machine understandable code to run it



Eclipse – IDE

Java is high level programming language based on c and C++

Designed to be written once and run anywhere

Runs the code in a virtualMachine(JVM)

Very similar to c#

Object Oriented

Statically Typed – it need data type declaration and you cannot change the data types

Java is a language and a platform.

Java class libraries:

I/O

Math

Networking

Collections

Logging

Graphics and UI

Text formatting

Xml  
RMI

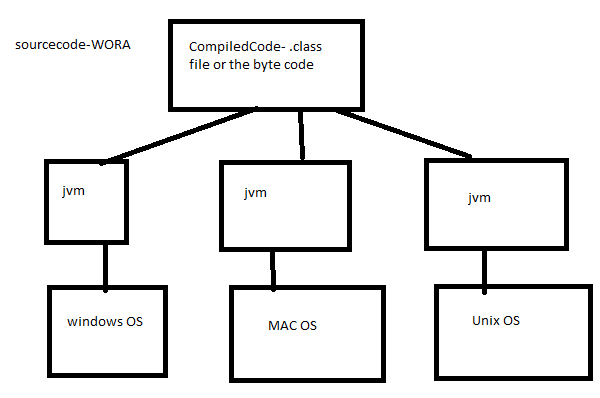
Security

DB…….

JVM- Acting as an abstraction between the OS and Programming Language.

It takes care of memory management.

It verifies that code do not try to access memory location of OS.



Class – variables and methods - logic

Object- instance of class

Variables

Scope of variables

Data types

Class

Object

Keywords

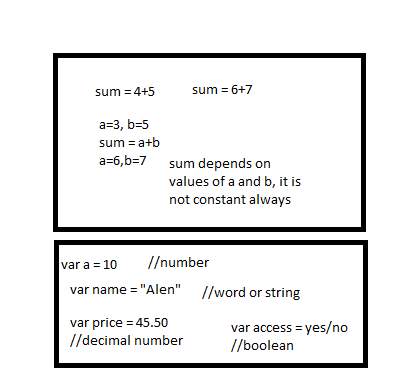
Identifiers

Operators

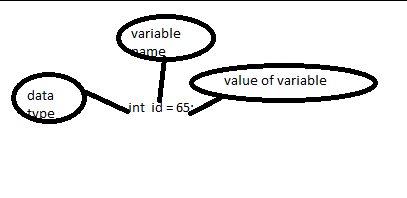
Variable – variables allow to store and retrieve data in our program, they make programs interactive.

Deals dynamically with data(number x, x can be any integer)

Without variables the data will be always constant( number 3) and non interactive.



int id = 65;



Java IDE’s – Integrated development environment- it’s a s/w that allows you

Users to easily write and debug programs.

Eclipse, Netbeans, Intellij Idea – jetbrains , RAD……

Primitive data types in java:

byte - 8 bits

short -16 bits

char – ‘ ’- 16 bit Unicode character

int – 32 bit

long – 64 bit

float – 32 bit

double – 64 bit

boolean- T/F- 1/0

All others Other than primitive types come under Object type.

Variable names should be descriptive and readable

**Scope of variables** - global - class level, local - method level , block level- within the method any block level

Package – is a folder structure to segregate and group classes logically/functionally without any conflicts.

byte , short , int, long – integers

float, double – decimals

boolean – truthness

char – keyboard character

other than primitive data type – Object

variable : variables allows to store data and whose value is not constant

3+4

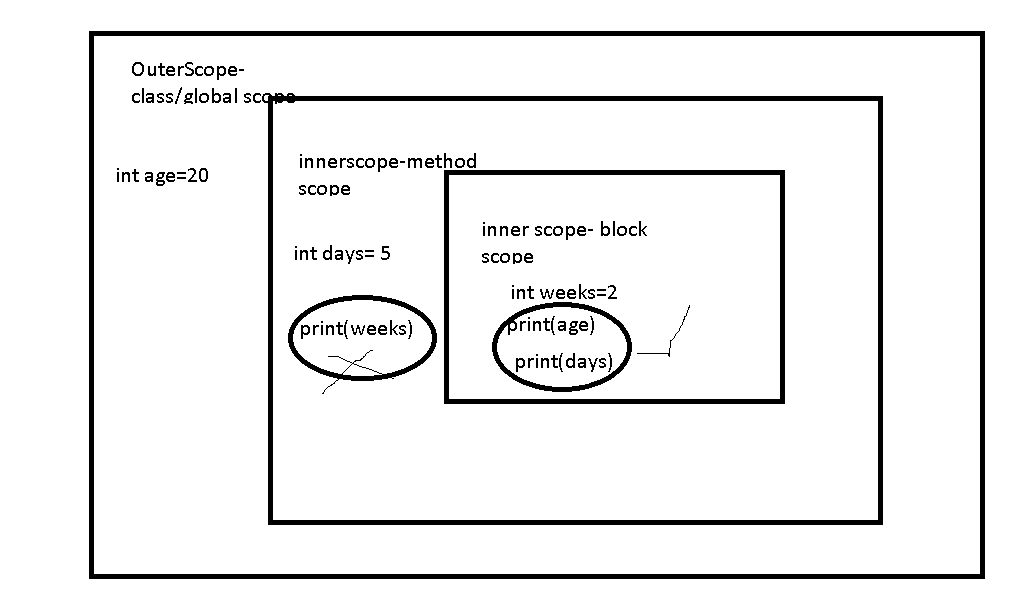
a+b

Variable scopes:

outer scope – inner scope

outer scope variables are available/visible to inner scope but inner scope variables are not visible to outer scope.

Global/class scope->method/local scope-> blocks scope



Class – A template/model/blueprint that describes the kinds of states and behaviour that objects of its type support.

Class defines state - properties/ information/data and behaviour – actions

Class - variables , methods

Dog – bella,rex, julie

Dog – name, color, height - properties/ information/data

barking,chasing,eating – actions

Object – is an instance of the class

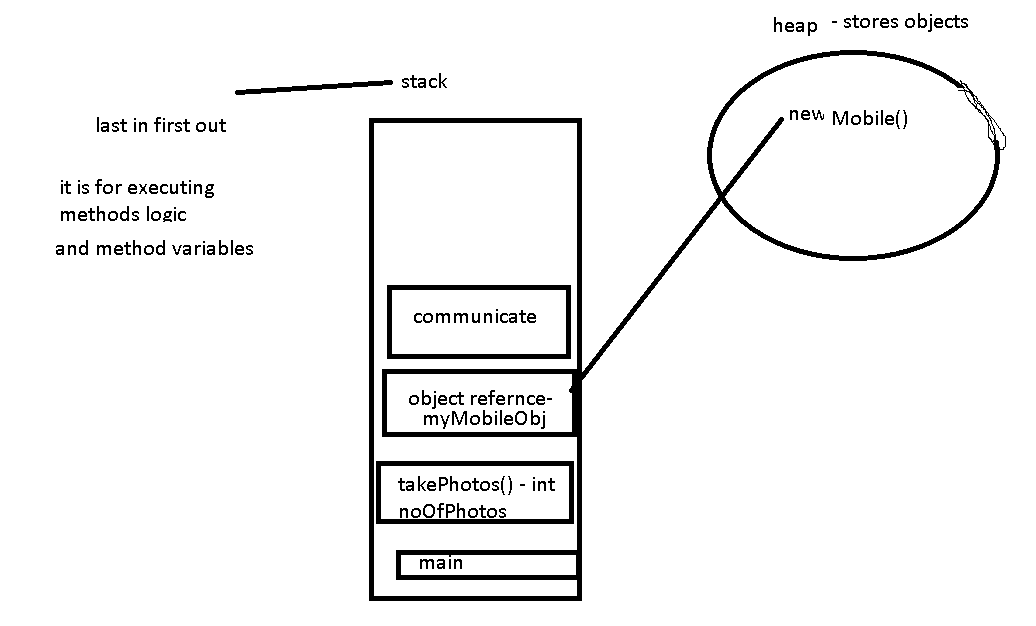
It is created using new keyword .

Object has its own state and behaviour to access all behaviours

defined by class.

State- represented by instance variables/properties

Behaviour- described by methods/ actions



Methods:

accessModifier returnType(dataType) methodName(input parameters){

//return returnTypeVlue;

}

Public void myMethd(){

Print(“helloWorld”);

}

Public int sum(int a , int b){

return a+b;

}

If u r not returning anything – then method return type is void

Identifiers – **names** given to classes or variables or methods……

Names should be always very readable and descriptive – good programming practice

Rules to declare identifiers:

1. Should not start with a number
2. It can start with letters, \_(underscore) or $(currency character)
3. After the first character/Within the identifier- can have letters, underscore, currency char and numbers
4. Should not use java keywords as identifiers
5. Identifiers are case sensitive – age, Age- 2 different identifiers

String firstName\_lastName

Int \_age

Currency $currency

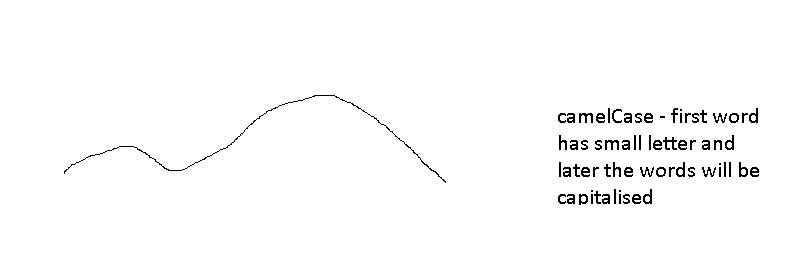
City city2

Classes& Interfaces - First letter should be capital and if it has more than one word – then first letter of inner words should be capitalized.

Methods& variables –start with lowerscase and consecuitive words will be capitalized - camelCase format

getCities

sum



Constants – all capitals letters and \_ separated b/n the words

TARGET\_TIME=10;

PI=3.134

PI\_VALUE=3.14

Keywords – All programming languages have their own set of words used within the language, java has built-in keywords – these keywords should not be used as identifiers.

Operators:

Arithmetic Operators : + ,-, \*, /- gives quotient, % - this gives remainder

+ - works as addition opr for number , concatenation operator strings

X= 4+5\*4=24

Relational Operators : > , <, >=, <=, = - assignment operator, == - comparision operator

4+5>20\*2—9>40

Logical/ short circuit Operators: And- &&, or-||

&&

X Y result

T T T

T F F

F F F

F T F

||

X Y result

T T T

T F T

F T T

F F F

^ - exclusive or…

X is multipe of 3 and 5

X=30

(x%3==0)&&(x%5==0){

Print(“yes x is multiple of 3 and 5”)

}

Else{

Print(“No its not a multile of 3 and 5”)

}

practice – class, objects, variables, identifiers

unary = they will be operated on one operand

++x, --y, !

Binary= x+y, z-x, x%a………….. 2 operands

Ternary= ? :

5>6 ? “yes it is greater”: “it is not greater”

Assignment operators:

=, +=, \*=,….

X= x+y

X+=y

X\*=y --- x = x\*y

Bitwise operators

&, |, >>, <<

4 - 00000000 00000000 00000000 00000100

2 - 00000000 00000000 00000000 00000010

4 & 2 - 0

o/p- 00000000 00000000 00000000 00000000

4|2- 6

o/p- 00000000 00000000 00000000 00000110

<<- left shift, >> - right shift

4<<2, 2<<1

00000000 00000000 000000 00 00010000 – 16

**Operators Precedence- unary, arithmetic,conditional,logical…**

**Flow control statements:**

Conditional - if else, nested if else, switch case

If(boolean expresiion – it will check a true or false){

}else{

}

Multiple if

Nested if

Switch accepts int,byte, char,short, String class, Enum types, Integer,Byte,Short,Character

Switch(conditiontomatch){

Case condn: logic to execute;

Case condn2: logic to execute;

Case condn2: logic to execute;

.

.

Case condnn: logic to execute;

Default: default logic;

}

Looping – while, do while , for

While executes the statements until condition is true…

Do..while –statement in do block executes atleast once even though the condition is not matching….

Do while evaluates the condition at the end of the do block so the statements in do block will execute atleast once surely…

But while evaluates the condition initally itself..

For – for(initializing ; conditionTo Check ; increment){

}

Branching – break, continue

Break will come out of current iteration and the for loop.

Continue will skip current iteration (next statements after continue) and goes with next iteration

**Constructor**: that is invoked always whenever a new object for the class is created.

Every class has atleast one constructot by default i.e, no args constructor.

It is similar to method syntax but it do not have a return type.

Always the constructor name should be same as your class name.

Constructors can be overloaded just like methods.

Constructor can have any access modifiers-public, private, protected, default.

Within constructor if your are using this() or super() – They should be first statements(the very first line) in constructor.

Constructors can be used to initialise the instance variables at the time of object creation itself.

**this** keyword in java – this is used to refer the member(instance variables,methods) of current object from method or constructor.

It can be used inside a method or a constructor of the class.

this is a reference to current object of execution.

**Overloading** can be achieved with methods and also constructors by varying the input arguments - we can vary by number of arguments or also by varying the data type of the arguments.

At runtime depending upon the number of arguments/ data type of the argument you are passing it will invoke respective consructor / method.

**MethodOverloading** :Reusing the method name but with **different arguments.**

Must differ in method arguments/ method input parameters – we should either vary in datatypes of arguments or number of arguments.

May or may not vary in method return types.

It can have dofferent acces modfiers and also different exceptions.

Primitive data types casting:

byte , short , int , long , float, double

left🡪right upcasting or implicit casting

right->left downcasting or explicit casting

**static** : key word in java

It is class level but not any object specific

Static can be used for variables, methods and blocks.

Every object has its own copy of instance variables but static variables are not specific to any object, they are available at class level rather than instance level.

Static variables will be shared across your objects, instance variables are specific to object.

**final**:

final is applicable to variables,methods and classes

class is final- no other class can extend this final class

variables as final – we cannot reassign/change the values of variable – constant values

methods as final – nobody can override the method

**OOPS Concepts:**

**Encapsulation** : hide implementation behind an interface.

Encapsulated code will have 2 things:

Instance variables are protected(we can use private modifier and protect it)

We provide getters and setters to access the instance variables.

**Inheritance**( IS A relation) : Inheritance allows a class to be subclass(child class)of a superclass(parentclass) – therefore subclasses inherit public and protected variables and methods of the superclass.

Car- Benz is a car, BMW is a car

Car{

Public void doARide(){

Print(“evry car can do a ride”);

}

Benz extends Car(){

Public void setBenzFeatures(){

Print(“luxury car”);

}

}

WV extends Car{

}

Polo extends WV{

}

Animal – Dog is a animal, Cat is animal

Class Animal{

Public void commonFeatures(){

Print(it has four legs)

}

Public void setSound(){

Print(“they make sounds”)

}

}

Dog extends Animal{

Public void setSound(){

Print(Dog says bow)}

}

Cat extends Animal{

Public void setSound(){

Print(Cat says meow)

}

}

**Polymorphism** – exisiting in many forms….

A singe refrence variables can take its own type of object and also its subtype object – through inheritance

A single method can be given different behaviours(different logics) by method overriding – through inheritance

Strictly speaking – polymorphism is applicable to overriding but not overloading.

**Method overriding** – redefining the superclass method logic in subclass

Overriding is only applicable w.r.to inheritance….

Exceptions- checked(compile time)- IOException – FNF exception, unchecked(run time)- nullpointer exception, numberformat exception

|  |  |
| --- | --- |
| Overloading | Overriding |
| Method arguments **must** change-either by number of args or by datatype | Method arguments **must not** change |
| Return type can be changed | Cannot change the return type except the covariant return types |
| Exceptions declared in method signature can change | Cannot change exceptions of super class in subclass method overriding-u can still add any unchecked (runtime)exceptions,narrower checked exxcpetions, but u cannot throw new or broader checked exceptions |
| Access modifiers can change | Cannot make **more restrictive** acces modifier or we cannot reduce the visibility… |
| It decides which method to call at compile time- it exhibits virtual method invocation at run time | At run time based on which object is getting(subclass or superclass) created it calls that particular method, but at compile time it just decides on object reference – so we need to make sure that the refrenec objet has the method u r calling at compile time. |

Public – protected – package – private

Less restrictive 🡪more restrictive

Overloading - static(compile time) polymorphism

at compile time only depending on my method arguments it decides which method to call

and at runtime actual method invoked is still a virtual method invocation as compiler already know depending on method arguments/signature what method to call…

Overriding – dynamic(runtime)polymorphism - at compile time your object referenece decides the method call , at run time - actual object decides which method to call

Access modifiers - public, private, protected

Access levels - public, private, protected,default/package

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Visibility | Public | Protected(package+subclass) | Private | Default/package |
| Trying to access members from same class | Yes | Yes | Yes | Yes |
| Trying to access members from different class in **same package** | Yes | Yes | No | Yes |
| Trying to access from class **outside the package** | Yes | No | No | No |
| Trying to access from **subclass** **outside the package** | Yes | Yes | No | No |

Inheritance – we achived reusablity, reomove boiler plate code to one class and remaining child classes can simply inherit the features

**AbstractClass**:

**When I need abstract class** - If we have some common functionality to be provided to sub classes and we want to declare some abstract methods which my child class should definitely implement(I am creating rules that child class should definitely follow)…

Abstract class – is a normal java class with abstract methods along with concrete methods. So this class acts as parent class.

Abstract methods – do not have method body, they only have method declarations.

Concrete methods – have method body/implementation.

Abstract class can use any of the access modifiers including private(concrete methods)…but still abstract methods cannot be private.

The first child class which is not abstract should defintely implement all my abstract methods.

Abstract class can have constructor.. but we cannot create the object for abstract class, only way the constructor is called is when the child class object is created.

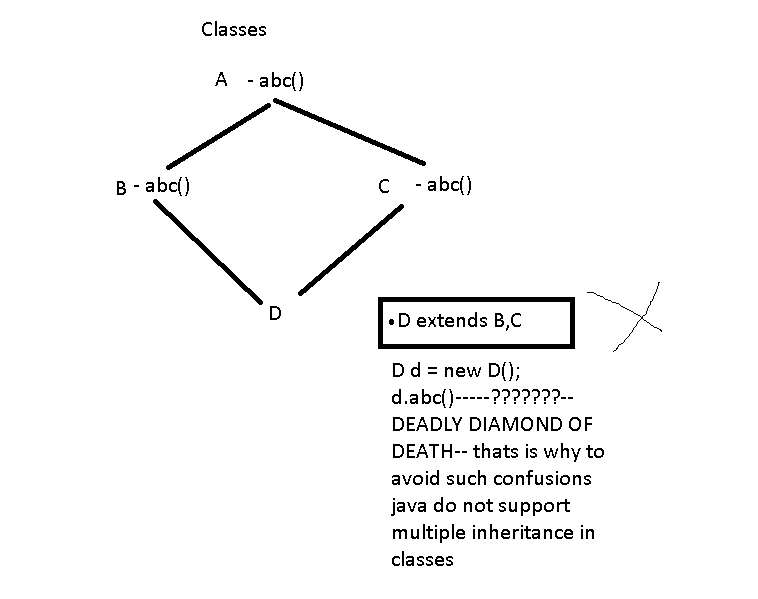
**Interface**: Interface is a contract which defines what a class should do.

It is 100% abstract class – bcz no concrete methods …

By default all the interface methods are public and abstract, we cannot have any concrete methods in interface like abstract class

By default all the variables in interface are constants- public, static and final

Interface do not have constructors.



Your interface can extends one or more other interfaces but abstract class can extend only one class.

Assuming A,B,C,E are all interfaces with some methods

Interface A extends B,C;

You inteface implementation class can implement more than one interface.

Class D implements A, E

This is a way of achieving multiple inheritance in java using intefaces.

**When to use interface:**

If you just want to declare some contracts(methods to implement) which your subclasses should follow and also if you want to implement more than one interface…

Interface - WordDocument

Interface - ExcelWorkbook – readExcelFilePath()

getSheetData()

getRowAndCellValues()

Apache Poi – HSSFWorkbook implements ExcelWorkbook,

JXL – XSSFWorkBook implements ExcelWorkbook

**When to choose abstract class and when to choose interface:**

**When we have some common functionality to share to all child classes and we want some abstract methods also which child class must implement – we choose abstract classes.**

**Interface- When we have to define only methods to implement(contract) and no need of any concrete methods, also if we need multiple inheritance – we choose interface.**

**Also we choose interface for defining Constants.**

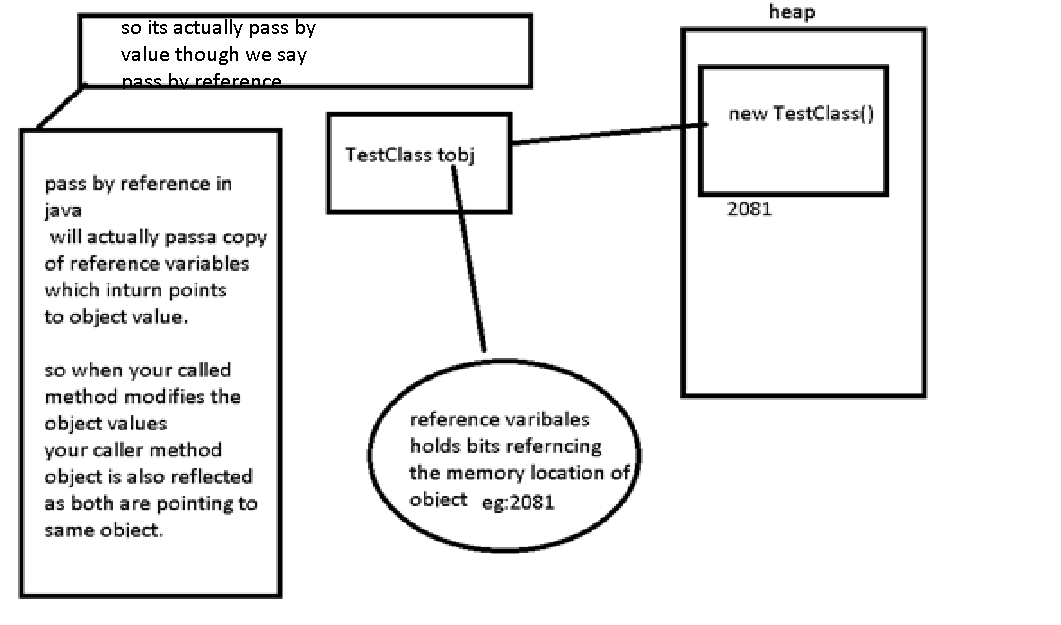
**Interface reference variable can take any child class or implementation class object, this is same with abstract class also – its just like parent-child relation ship.**

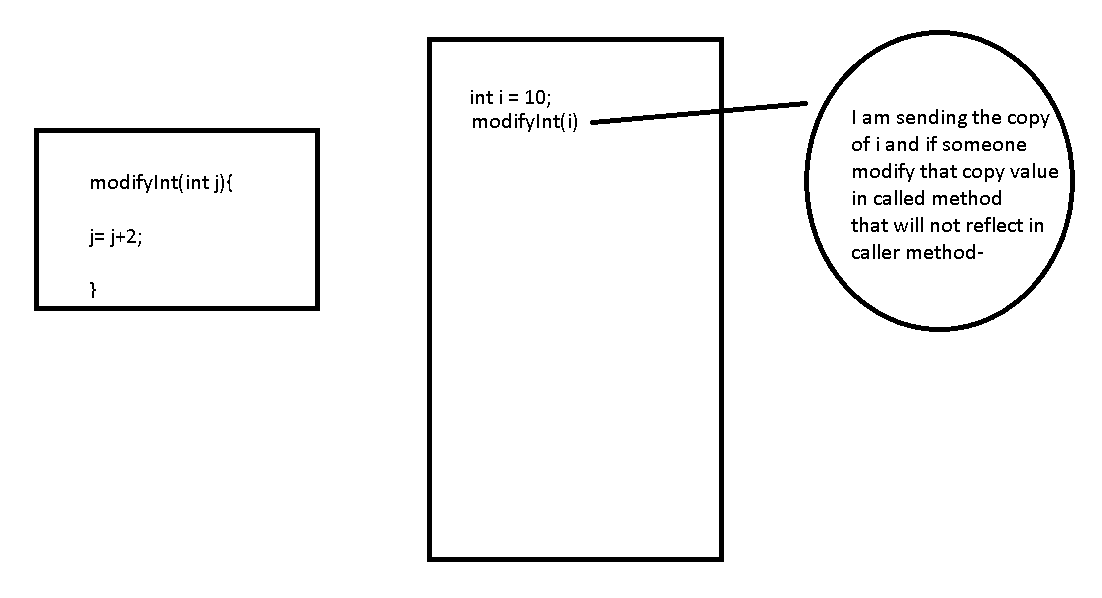
**super: is a keyword by which we can access parent class members(methds,variables,constructors) in child class.**

**Pass by value/pass by reference:**

**Primitives refer to actual value in memory location**

**ObjectsRefernce refer to memory adress of actual object.**

****

****

**Object: IT IS SUPER CLASS OF ALL CLASSES IN JAVA.**

**wait**

**notify**

**notifyAll**

**hashcode**

**equals**

**toString**

**Object-Animal-Cat,Dog**

**Object-TestClass**

**Collections : Dealing with datastructures in java**

**List of objects , iterate over the list of object and do some logic on each object, get a particular object from list,**

**Remove a object from list**

**Utils package in java – has classes w.r.to datastructures**

**List,Set, Map,Queue**

**java.util**

**Collection Interface : Set,List,Queue 🡪they extend Collection interface**

**Java.util: Map**

**java.util**

**Collections : class deals with util methods required by Collection interface-sort, toArray…….**

**collection – deals with group of data and it reprsents data structures in which objects ae stored and iterated.**

**List - list of things**

**Set – unique things**

**Map- key-value pairs(key will be unique)**

**Queue – follows an order of the things to be processed**

**sorted – sorted on a particular rule**

**sorted means data is ordered(asc/desc/userdefined order) as well**

**23, 34, 36-**

**Abc , bun, eat**

**Ordered – follows any order in which data is inserted and retrieved**

**Ordered does not mean that data is sorted**

**Eg:ArrayList orders by index, order by insertion**

**A[0]-34, A[1]-36**

**unsorted**

**unordered**

**Arrays** – you always decide the size of array whenever you are intialising the array.Arrays have both primitive data types and Object datatypes

**Collections**- No need to give size initially- like growable arrays

We have different methods like get, remove,set…..

Collections deal with only objects.

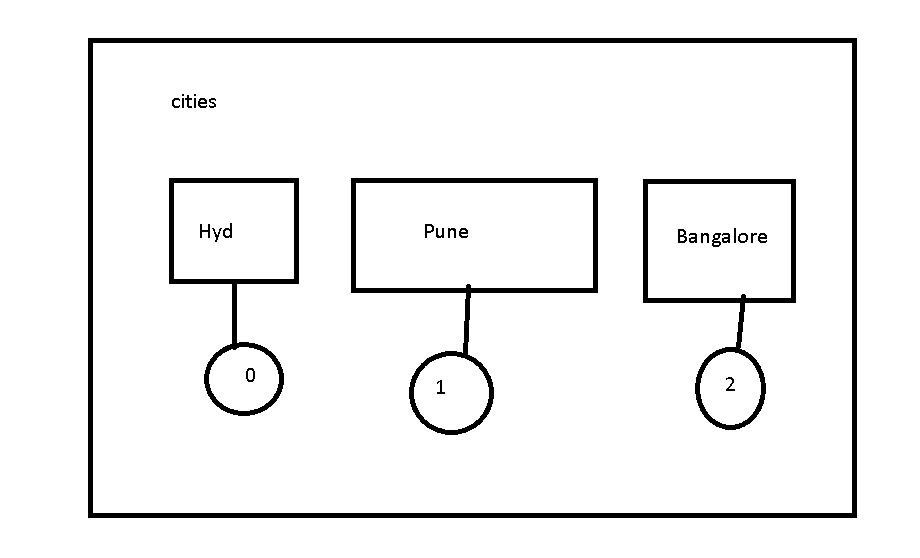
**List:**

**ArrayList –** before java 1.4ArrayList implemented only List interface

ArrayList is fast in iteration and fast in random access.

Prefer ArrayLoist when we deal with iterations but not any insertions and deletions.

But after java1.4 ArrayList also implemented a marker interface(interface with no methods) called RandomAccess interface which actually made fast random access.

****

**add(element)- add element to list**

**addAll – add another collection to existing collection**

**add(index, element)**

**remove(element) – return type is the element u removed**

**remove(index)- return type is boolean**

**contains(element)- returns boolean –gives true if element is present and false if element is not present**

**get(index) – element at specfic index**

**size()- length of list**

**java internally uses fail fast iteration for most of its collections – so we need to avoid adding or removing the elements when iterating the elements.**

**Object – equals(), hashcode()**

**Few collections internally use Hashing principle which makes them efficient..**

**Vector:**

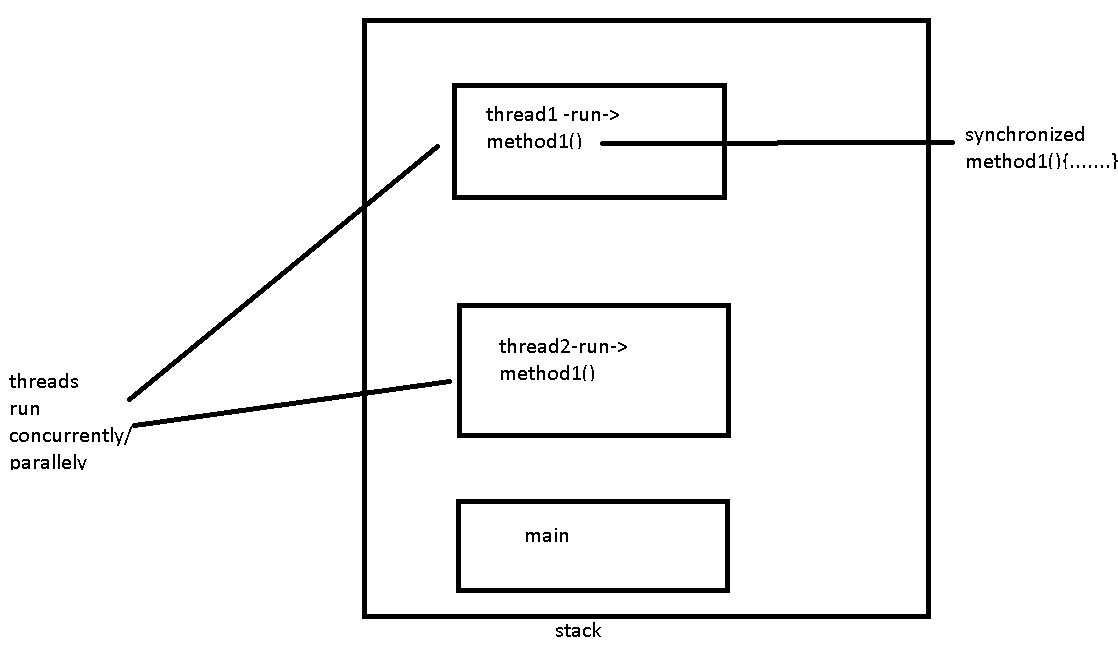
**Vector has synchronized methods which makes it slower than arraylist**

**Vector allows to specify capacity and increment capacity.**

**If you do not specify capacity by default it takes as 10 and increments by double the capacity.**

**Vector :** similar to ArrayList but the methods in vector are synchronized(making thread safe)

Vector is comparitively slow than ArrayList because methods are synchronised, so we should go with vector only when required(eg: may be a multi threaded environment), but normally we prefer ArrayList over vector.



Vector will double the size fo itself when it grows.

ArrayList will grow by half of its size when resized.

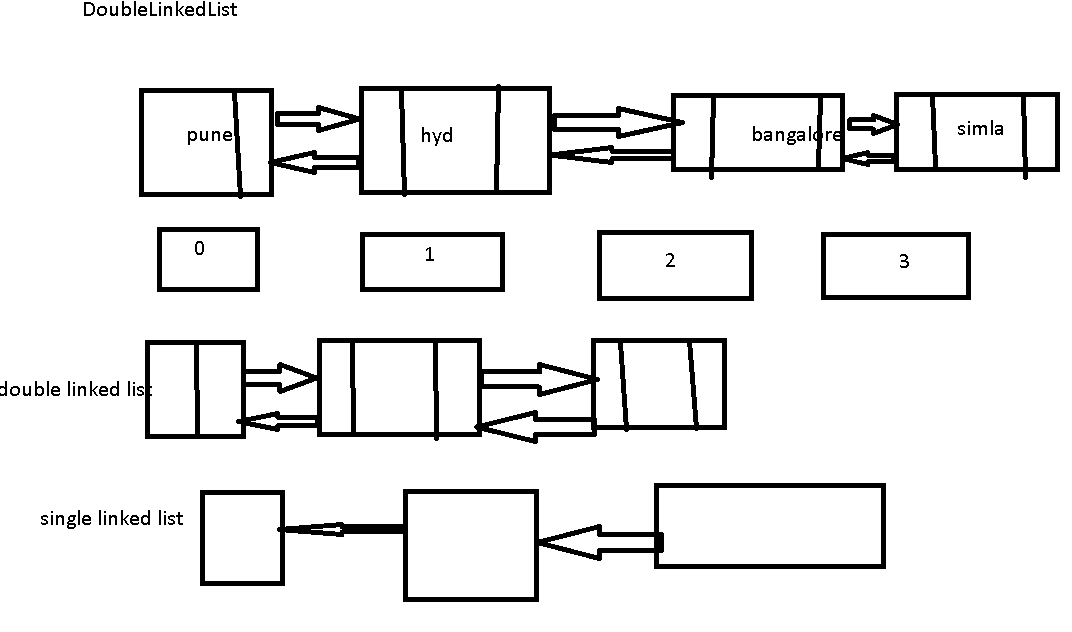
List of data- one thread is trying to do update add operation and some other thread is trying to remove the data from List.

**LinkedList** : prefererred when you need to deal with insertion and deletion of data, it implements **both List & Queue** interfaces, so it also has Queue methods peek(),poll(),offer()

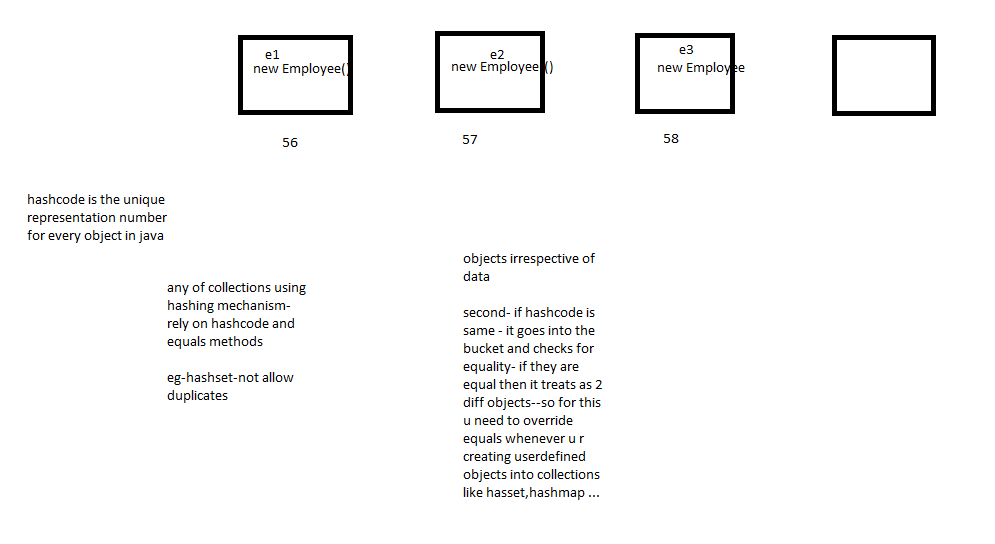
It uses doubled linked list structure to store data

It is ordered by index like array list

It iterates slower than ArrayList



**Hashcode and equals importance with usedefined objects in hashing collections:**



**Set**:

**HashSet** : do not allow duplicates, add method returns false when u try to add duplicate and it will not add to collection.

Set do not maintain any order but list maitains index order.

Set is faster because it uses hashcode() mechanism.

HashSet will allow one null– if you add more than one null it considers as duplicates

**LinkedHashSet:**

LinkedHashSet maintains order using doublelinked list structure

Internally.

It iterates elements in the order they are **inserted**.

**TreeSet**:

Treeset maintains natural sort order or **ascending order**

Internally Treeset uses redblack tree structure

TreeSet will not allow null.

**Map**:

**HashMap**:unsorted and unordered

Store data in key value pairs.

Key is unique and values can be unique or duplicates

Key can take only one null value and values can be multiple null values

HashMap is faster because it uses hadcode mechanism

**LinkedHashMap**:

It maintains order of insertion, it is slower than HashMap for iterartion but

For insertion and deletion scenarios its faster.

**Hashtable**:

Similar to HashMap but methods are synchronized.

HashTable will not take any null values

LinkedHashMap- faster w.r.to insertions and deletions of data

**TreeMap:**

**Ordered by natural sort order.**

**Queue**:

ProrityQueue

All our classes which start with Hash-HashSet,HashMap,Hashtable they use hashcode mechanism and they are faster because of hashcode

hashcode()

equals()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class(C) | List(I) | Set(I) | Map(I) | ordered | sorted | WhenToUse |
| ArrayList | \* |  |  | ByIndex | No | Its fast for iteration purpose |
| LinkedList | \* |  |  | ByIndex | No | Its fast for insertions and deletions- follows doublelinkedlist structure |
| Vector | \* |  |  | ByIndex | No | Similar to arraylist but it has synchronised/threadsafe methods,it increases double the initial capacity when it tries to increment the size, it is slower than arraylist |
| HashSet |  | \* |  | No | No | Follows hashcode mechanism and its faster than any other set. |
| LinkedHashSet |  | \* |  | ByInsertionOrder | No | Faster for inserions and deletions. |
| TreeSet |  | \* |  | bySorting Order | ByNaturalAscOrder/custom comparisionRules-Comparator,Comparable | We use only when we need data to be sorted. |
| HashMap |  |  | \* | No | No | Follows hashcode mechanism and its faster than any other Map.It allows one null key ,mutiple null values |
| HashTable |  |  | \* | No | No | Like hashMap but it has threadsafe or synchrosied methods.it is slow than hashmap, it does not allow any null values |
| LinkedHashMap |  |  | \* | byInsertionOrder | No | Faster for inserions and deletions. |
| TreeMap |  |  | \* | bySortOrder | ByNaturalAscOrder/custom comparisionRules-Comparator,Comparable | We use only when we need data to be sorted. |
| PriorityQueue©- Queue(I) |  |  |  | bySortOrder | ByNatural or custom order | For implementing  Queue –fifo or any priority based queue. |

Equals and hashCode overriding is important when you are inserting user defined object into HashSet and HashMap(as keys) and any other collection which reliues on hashing algorithm..

Comparable and Comparator are required only if you need sorting of the data when you have user defined objects…

\*\*\*If you are using your own class objects as keys to your HashMap or any Hash related classes – you need to override equals and hashcode- else hashmap/any hash classes will fail to work properly.

When you want to add a user defined object into a sorted collection like TreeSet or TreeMap we need to implement any one of below interfaces:

Comparable

Comparator

To write custom sort rules –

O1<o2 = -1

O1>o2 = 1

O1==o2 -> 0

**Example scenario:**

**List of Products:**

**Class Product- id, name, price**

**Class Order- orderNum,orderData,customerId**

**I want to store and retrieve list of details - Each product and the order w.r.to that product**

|  |  |
| --- | --- |
| **Product1- 58** | **Order1** |
| **Product2- 40** | **Order2** |
| **Product3-100** | **Order3** |
| **Product4- 700** | **Order4** |

**HashMap with Product as key and Order as value**

**We need to override equals and hashcode here because Product is user defined object.**

**TreeMap- for sorted keys- again u need to implement comparable for sorting Products by its id.**

**Queue- FIFO principle**

**PriorityQueue- as opposed to queue- this goes with the priority defined but not FIFO principle- if u want to set priority again u can implement comparable or comparator else it goes with natural sort order.**

**Stack: LIFO principle**

package-java.util:

Collection interface

Comparable(I),Comparator(I)

Collections class

Arrays class

Properties class

Calendar, Date classes

Aug batch – class notes

JDK- lib, Jre, JVM

.java - .class – jvm

Variable

Class

Car

Manufacturer

Color

Engine

brakes

Class - state and behaviour

Object – is instane of the class which has instance variables and methods

Car car = new Car();

Car.color = black

Car.manufacturer= BMW

Car car 1= new Car();

Car1.color = white

Car1.manufacturer= BMW

Package – folder/directory structure that helps to segregate things(classes) logically….

Class in java – has 2 access modfiers –

1. public
2. default/package

Variables and methods:

public

private

protected

default/package

primitive datatypes:

number/integers:

byte - 8 bits – 0000 0010

short – (-2^8 to 2^8 -1)

int

long

decimal: 45.56, 56.7%

float

double

char – ‘a’

boolean – true or false – truthness of the expresion/statement

String – its an Object

Other than primitive datatypes every other datatype is an Object in java.

Java is statically typed language - because we need to give the appropriate data type at the tine of declaration of variables.

Static : keyword in java – class scope i.e., it is specific to class but not object….

Using class name – we can access static variables and methods

We do not need to create objects to access these variables unlike instance variables/ methods..

We cannot acces non static variables /methods inside a static method.

Why because – static variables do not need objects and they are loaded at the time your class gets loaded—they don’t need objects but instance var’s need objects..

Different types and scopes of Variables in Java Class:

Global variables:

Instance variables – variables out side the method and within the class.

Class/Static variables

Method variables /local variables– which are inside the method

Method parameters

Block variables - Variable inside Block

Operators:

Arithmetic -> (binary operators- u generally perform these actions with 2 operand) + , - , \*, / - quotient , % - remainder

A+b

a-b

a%b

a/b

a\*b

Unary- > (on single operand)++, --, + , - , ! , ~

++a

a++

--b

b- -

-5, -6

+5,+6

00001111 – 11110000

!true -> false

!false 🡪 true

Conditional /relational operators:

<, > , <= , >=, ==, !=

Bitwise:

Left shift - << - 2 – 0000 0010 –

2 << 1🡪 0000 0100 🡪4

Right shift - >> 0000 00010

2>>1 -> 0000 00001 -> 1

unsigned right shift - >>>

&

|

^

ShortCircuit Operators:

&&

||

Operators

Control flow

This

Constructor

Inhertance

byte, char, int,short,Enum, String, Byte, Character, Integer, Short

Switch (input){

Case 1: logc to be executed

Case 2:

Default:

}

Loop statements- while, do-while, for loop

Execute set of statements repeatedly or until a condition is satisfied

Employess – 10- list

Iterate the data one bye one – and perform the logic

Do while – execute the statements in while block first and then check for the condition in the do…

Constructor – is invoked at the time of creation of object using new keyword

Constructor Name is same as class name

We can have more than one constructor in same class- constructor overloading- this is possible when you change either the no of args or the datatype of the args of constructor

Constructor will not have return type

this – is key word which refers to instance variables or methods of the current object of execeution….

Public, private, protected, default

OOPS Concepts:

Encapsulation

Abstraction

**Inheritance – common features of parent class (instance vars or methods) can be inherited to child class. Reusabilty of code**

Overloading – having same method name with different input parameters

different – you should differ parameters either by count or by datatype

Overriding

You may or may not change return type

You may or may not change acces modfiers

You may or may not throw same exceptions.

Polymorphism

Car – sportCar,

BiKe – mountainBike, gearless

Animal – Dog, Cat

Write a program to find given number is even or odd

Write a program to check whether a number is prime or not

3,5,7-

20-

Write a program to demonstrate constructor overloading?

Also write a program to understand this keyword?

Constructors – you can use constructors to set the data but you should prefer this only when you create one set of data for one object and u do not want to change data again

Bcz if u want to change data again - u need to again create the object

Setters – using setters you can set data n no of times with same object /instance.

Animal – makeSound

eatFood

noOfLegs=4;

sound;

food;

color;

name;

Dog extends Animal – makeSound(“bow”)

eatFood();

Cat extends Animal- makeSound(“meow”)

Car – brakes

Clutch

Engine

Model

Color

Stearing

SportsCar extends Car

Engine – design according to sprtscar

default-protected-public

Overrding- u can change acces modifier to be less restrictive but not more restrictive

Stack & heap

Pass by value

WrapperClasses

Int id = 4;

Int id1=5;

Arrays – arrays are objects that store multiple variables of **same** type.

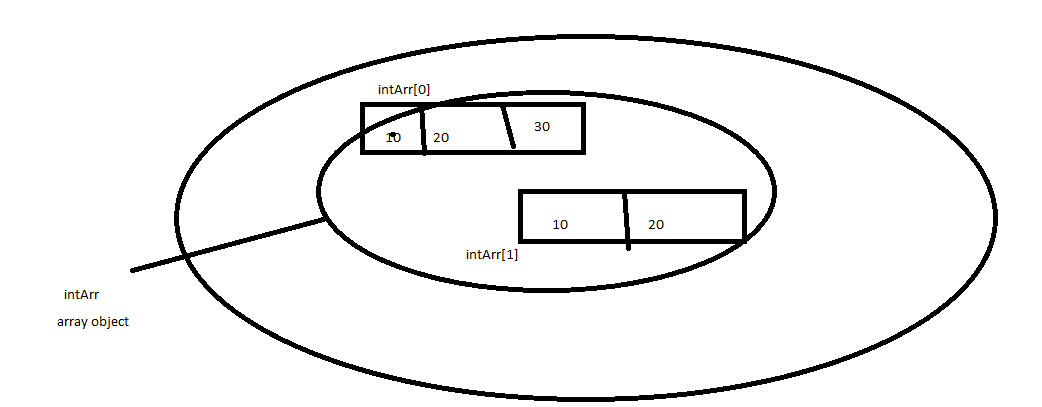
Arrays can be used to store primitive datatypes and also objects.

Array itself is also an object.

Declare an array

Int id ;// declaration

Id=5; // initialsing the variable



Class Animal{

}

Class Dog extends animal{

}

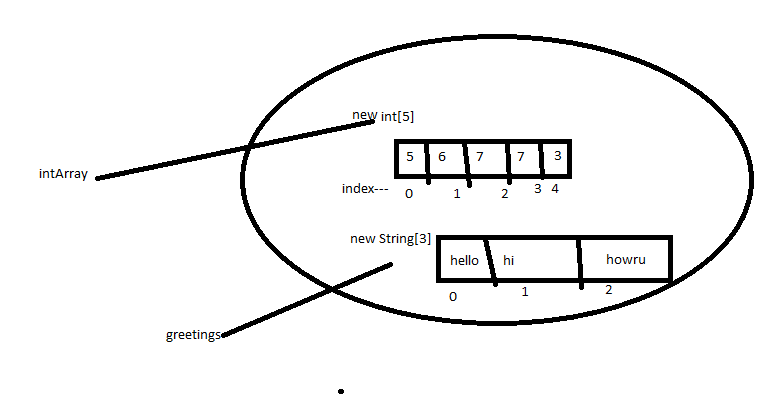
Animal[] arr = new Animal[4];

Arr[0] = new Dog();

Object[] objArr = new Object[3];

objArr[0]= 1;

objArr[1]=”hello”;



String – String is a class in java.lang package , it is under Object hierarchy like all other classes in java.

String is immutable in java.

String class is declared final in java.

String is set of 16 bit unicode characters….

String is set of char array data

String can be created using literals or using new Object

String can be compared using == and equals but it is always good to use equals method bcz == works only for your string literals as == compares references but not the actual value.

equals compares actual value but not the reference unlike ==

String literals are created in string constant pool – if there is exisiting same value then new reference points to same value instead of creating new object again unlike string objects creation

String objects are create in heap memory

Keeping efficient memory managment in mind- java people created string constant pool mechanism.

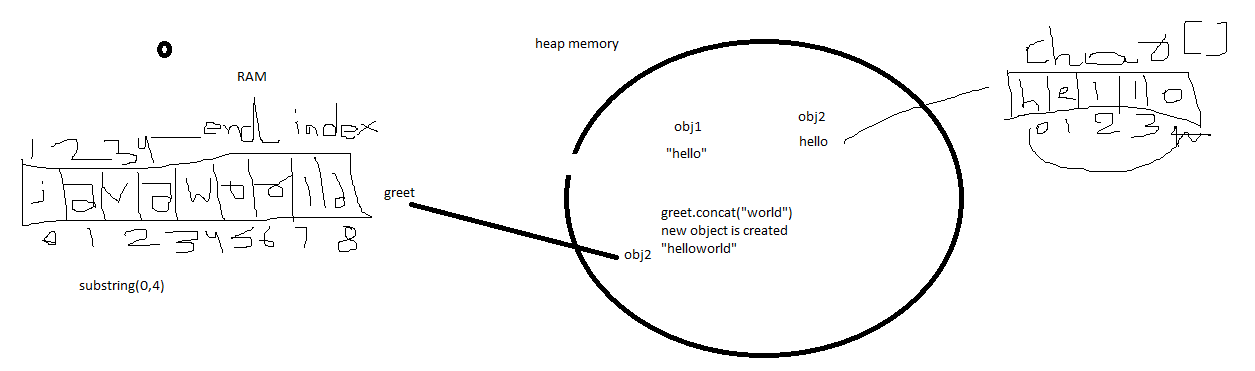
Why string immuatble?

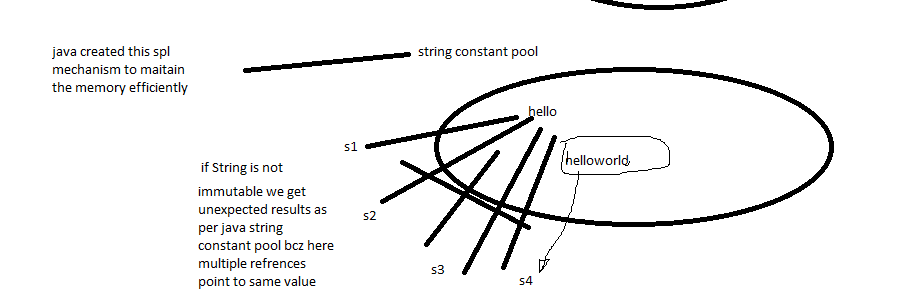
Keeping stringconstant pool in mind they made string immutable.

Other reasons are for security, cache etc…

How u can make any class immutable in java?

By making class final and there are some other rules…





■ charAt() Returns the character located at the specified index

■ concat() Appends one String to the end of another ( "+" also works)

■ equalsIgnoreCase() Determines the equality of two Strings, ignoring case

■ length() Returns the number of characters in a String

■ replace() Replaces occurrences of a character with a new character

■ substring() Returns a part of a String

■ toLowerCase() Returns a String with uppercase characters converted

■ toString() Returns the value of a String

■ toUpperCase() Returns a String with lowercase characters converted

■ trim() Removes whitespace from the ends of a String

StringBuffer and StringBuilder:

These classes are mutable unlike String class.

StringBuffer methods are synchronized , they are thread safe.

StringBuilder methods are not synchronized, they are faster.

StringBuffer st = new StringBuffer(“hello”);

st.append(“world”);

StringBuilder st = new StringBuilder(“hello”);

st.append(“world”);

eg: when dealing with file data appending and whenever we have to modify string frequently we prefer stringbuffer and stringbuilder classes.

append

delete

insert

reverse

Write a program to declare and intilase single dimensional and 2 dimensional arrays in all possible ways..

Declare 2 different arrays with primiives and other with objects and print the data in array?

Int[]

Integer[]

Animal[]

Write a prgoram to find sum of all elemnts in it array?

{2,3,4,5}//output – 14

Arrays –

Store 10 id’s

Store 10 Animal objects

We need to specify the size at the time of array creation which is not always feasible..

Collections:

DataStructures and storing the set of similar datattypes

Techniques to achieve specific task:

Search – linearsearch, binarysearch…

Sorting structure---Redblack tree structure

Employee – id, name, address

Employees- 100 employees

Insert a new employee

Delete an employee

Retrieve/search a particular employee details

Sort the employee data w.r.to their id’s

Customers

SearchItems

You can store- Integer, Float, all wrapper classes, String..all that satisfy Object hiererachy

You want to store without any duplicates

Allow duplicates

Store data in a key value pair

Collection – is super interface for List,Map and Queue

Collections – is a class which have predefined utility methods whoch can be used by any of collections

collections /collection - terms used to refer different types of collection classes in java

collections only deal with java objects but not primitives…

List :

List will allow duplicates

Set:

Will only store unique data / it does not allow duplicates

Map:

It allow to store data in a key value pair

Ordered --

Sorted

Write a program to replace the given character in the input string?

// input String, inputChar, replaceWith

Write a program to check the extension of the given filename for xls, pdf and doc and do some logic accordingly…

If we want to sort the arraylist or any list data – we need to implement comparable or comparator and use Collections.sort(listObj, comparatorObj) or

use Collections.sort(listObj)

For Set and Map if you need sorting for your userdefined objects you need to use TreeMap And TreeSet and make sure you implement Comparable interface.

Wheneever u add userdefined objects as keys into hashmap make sure u override equals and hashcode…

Write a program to display data in dictionary format?

Home: a place to live

Book : that stores data

Write a program to count duplicate characters in String and display character and count of the character?

Write a program to sort list of Student data with std name?

List<Student> stdList = new ArrayList<Student>();

Student{

Id

Name

}