An ISO 9001 : 2008 Certified Company

NARESHi technologies

SOFTWARE TRAINING & DEVELOPMENT

040-2374 6666 | 23734842

info@nareshit.com

www.nareshit.com

nareshit nareshitech

nareshit

Find Latest IT Jobs SeshaJobs.com

Opp. Satyam Theatre, Durga Bhavani Plaza, Ameerpet, Hyd-16

CORE JAVA

- Oops Types of Classes
- Arrays Packages
- String Handling Collection
- Reflection API
- Regular Expressions

by

Mr. Hari Krishna

for Online Training Call / WhatsApp : 8179191999

19/08/2017

Object Oriented Programming (OOP)

1. OOP Basics.
2. object design models.
3. Object relations (IS-A, HAS-A, USES-A)
4. OOP Principles (E, I, P, A)
5. Coupling and Cohision
6. LC-RP Archiectecture
7. Real time project develop based on OOP concepts.

- The abbriviation of OOP is "Object Oriented Programming". we do not have a word called "OOPs" or "OOPS" in programming world.
- The wrong abbreviation created by industry for the word OOPS is
  "Object Oriented Programming System"
  "Object Oriented Programming Structure" X
  "Object Oriented Programming Synopsies".

  All above abbrivation are wrong
  There in not meaning for the character 's' in OOPs
  So OOPs also stand for
  "Object Oriented Programming"
- We have 4 types of Programming languages based on the Style of programming (structure)
  1. Monolithic
  2. Procedural / Structured programming language
  3. Object Oriented
  4. Functional
- Machine language, Assembly language are come under the Monolithic programming languages
  Cobol, Pascal, Fortron are comes under Procedural PLs.
  C comes under Structured PLs.
  C++, Java, .NET, PHP are comes under OOPL's.
  Java. NET Python Scala are comes under Functional PL's.

Note:- Every OOP language will support Functional Programming.

- Problems of Monolithic Programming language (ML, AL)

1.) Meant for developing System level (OS and Processor) Operations performing software.

2.) We can not use these language for developing end-user required mathematical calculations, because these languages will not support mathematical operations directly.

To perform Mathematical operations

(i.) ML, we must use 0's and 1's

(ii.) In AL, we must use mnemonics (add, sub, store....) which not new to every human being.

3.) Program is not human being readable, because english like Characters and mathematical operators are not used in these languages programs.

4.) Modularity is not supported, it means we can not develop big task as seperate individual programs.

5.) No reusability.

Based on all above programs ML and AL can not be used for developing mathematical operations.

- Advantages and Problems of Procedural/structured PL

Procedual/structured PL are meant for developing mathematical operation based program for end-user (humen being) like adding, subtracting.....

These language will used english like Character and directly mathematical operators in developing programs

Advatages are:-

1.) Program is easily readable

2.) Modularity is supported

3.) Reusability is supported

4.) Easily we can find and solve error in programs.

Problems are:-

1.) No security to data.

2.) No Dynamic binding between programs.

No security to data means

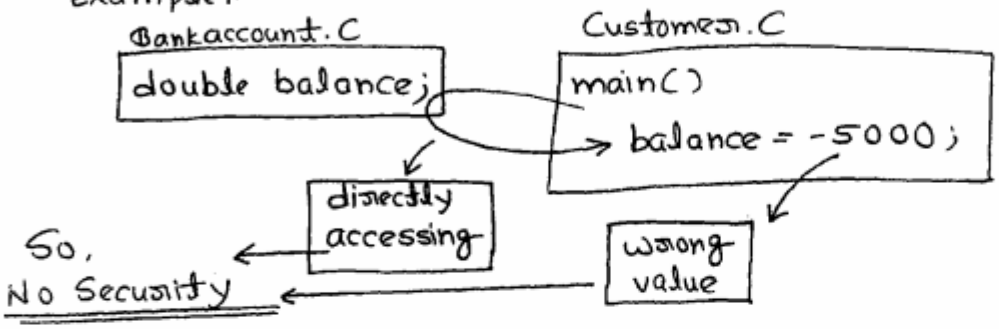
The data (values) available in a program, is freely / directly accessible to other programs.

So ~~we can stop~~

1.) We can not stop unauthorised people to access our data.

2.) We can not stop storing wrong values.

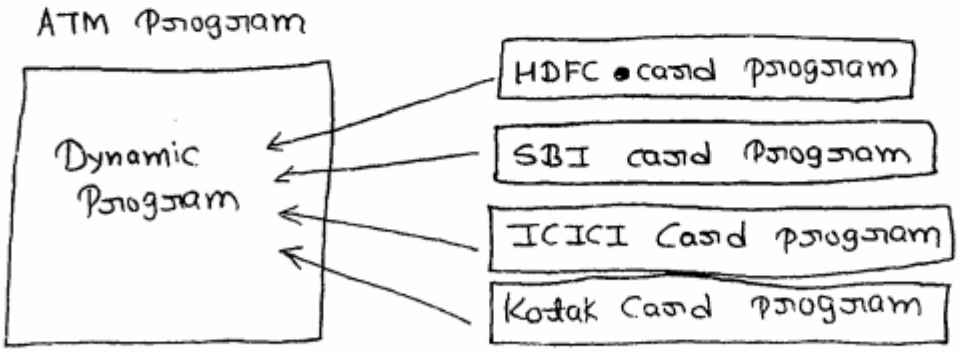
Example:-



For Example :- we can not access money in ATM directly, It means ATM has security.

- Dynamic Binding Between program means,

A program can able to change from one program to another program to access operations at run time without modifying code, then this program is called dynamic binding program.



ATM program is a dynamic program because it can accept any bank's ATM card without modifying its logic

CDMA Mobile → Static program.

GSM Mobile → Dynamic Program.

Q. Why Procedural / Structured Programming language doesn't support Security and ~~in~~ dynamic binding

- Because, structure Programming language are meant for performing mathematical operations it doesn't required security and ~~or~~ dynamic binding.
- In Mathematical operation we do not have any confidential or security related data. In Mathematical operation we will take input, performs ~~a~~ calculations and ~~result~~ return the result data. For example:- Calculator.
- Because, We are not storing any data, security concepts are not required.
- ~~Also~~ dynamic binding concepts are also not required in Procedural or structured programming language because once we develop program by accessing other program functionality in future this functionality never change. For e.g. Calculator
- Observe below program, You can understand above explanation.

Calc. C

```
main() {
    add(6,5);
    sub(6,5);
    mul(6,5);
}
```

Add. C

```
int x=10
int add(int a, int b)
{
    return a+b;
}
```

Sub. C

```
int sub(int a, int b){
    return a-b;
}
```

Mul. C

```
int mul(int a, int b){
    return a*b;
}
```

As you can observe in above program there ~~is~~ is no data storing requirement so doesn't required security.

There is no requirement of changing addition, subtraction, multiplication logic to different addition, subtraction, multiplication

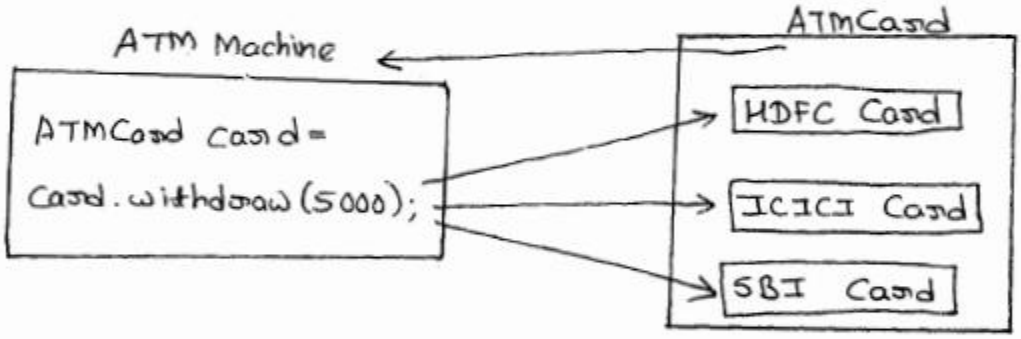
Q. Procedural and Structured programming are not suitable to develop business application, why?

Ans:- Because business application will required security and dynamic binding, because in business we will store data and time to time we will change one product to another product of same type.

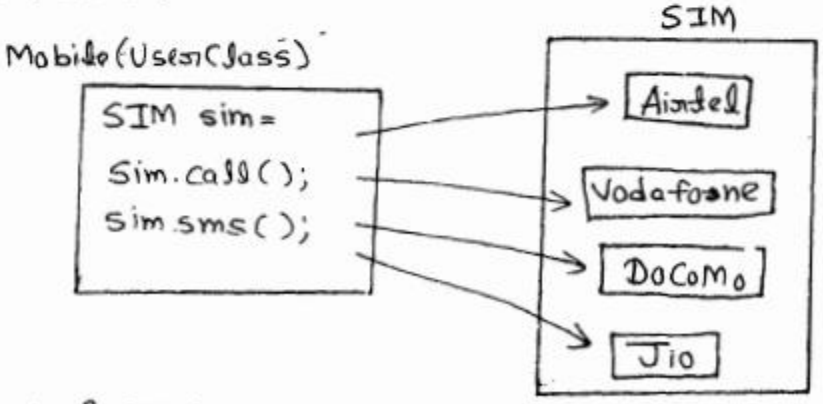
For. Example, Bank application contains BankAccount, inside BankAccount we contain data, This data should not be accessible and visible to other people, Here we required <u>security</u>.

For withdrawing and depositing money we will use ATM Machine, ATM Machine should accept any ~~bank~~ bank ATM card, it means ATM Program should change from one ATM card program to another ATM card program without modifying software in ATM, So here ATM software required <u>dynamic binding</u>.

Below program will show you dynamic binding :-



Below diagram will show you dynamic binding between Mobile and Sim Card :-



In above 2 programs ATM and Mobile we achived dynamic binding because we are accessing withdraw call and SMS operations by using SUPER type program name ATMCard & SIM.

In structured programming language we do not have facility for creating one program deriving from another program, it means ~~inheritance~~ inheritance.

Advantages of Object Oriented Programming

OOP supports → Modularity
→ Security
→ Dynamic binding

Hence to develop either mathematical operation or business operation based project we will use Object Oriented progra. languages.

22/08/17

~~What is Object Oriented Programming?~~

Summary on Types of languages:-

01. Monilithic PL (ML, AL)
→ Problem: ~~Does~~ Does not support Modularity.

02. Procedural/Structural PL (P, C, F/C, C++)
→ Adv.: Support Modularity
→ Problem: Does not support Security & Dynamic Binding.

03. OOP Language (C++, Java, Smalltalk, .NET)
→ Advantage: Supports Modularity, Security & Dynamic Binding.

Advantages of Functional Programming

- Functional programming is an extension to Object Oriented Programming so its syntaxes are similar to OOP, but it reduces number of lines of code. We will write only consized code.
- We can pass the function logic directly to other function as argument without creating class and object.

For more programming details on above 2 points Refer Java 8 Features chapter.

From Java 8 version onwards, java language is Functional programming language.

Q What is OOP ?

→ Choose one option :-
1. Language. ×
2. Technology ×
3. Technique ✓

Ans:- OOP is a technique or methodology. It Provides set of suggestions for creating ~~Rel~~ Real world Object in Programming world. By Achieving Security and Dynamic binding. By using OOP concept we can develope Highly Scalable Applications to accept or plug future Enhancements as per industry requirements.

For Example:- Switch Board with socket to plug diff electronic devices can be considerd as OOP Based Project, Scalable or Dynamic binding project.

For e.g, Swith board without socket only with switch can be considerd as POP based project. It is not Scalable project, means It can't accept future changes or connections to the other electronic devices. without changes.

Q. Why OOP ?

We must develop projects by using OOP concepts

1. To create real world objects in programming world.
2. To acchieve security to object data.
3. To achieve Dynamic binding between objects communication (scalability)

*. OOP Concepts

To represent real world object in programming world and for achieve Security and dynamic binding, OOP provides below Nine concepts, They are:-

1. class
2. object

← For representing real world object

3. Encapsulation ← For Security

4. Inheritance
5. Polymorphism
6. Abstraction

7. Message passing
8. Loose coupling
9. Runtime Polymorphism

← For Dynamic Binding

Q. Differences between POP and OOP.

POP	OOP
1. ~~Autom~~ Meant for developing mathematical operations.	1. Meant for developing Business operations.
2. Have programs are developed around operations.	2. Have programs are developed around object and its data.
3. It is function based programming language.	3. It is class based programming language.
4. It doesn't support security and dynamic binding.	4. It support security and dynamic binding
5. It doesnot support Access specifiers and inheritance.	5. It supports Access specifiers and inheritance.

Q. What is An Object, Class and Instance, Relationship between these three things ?

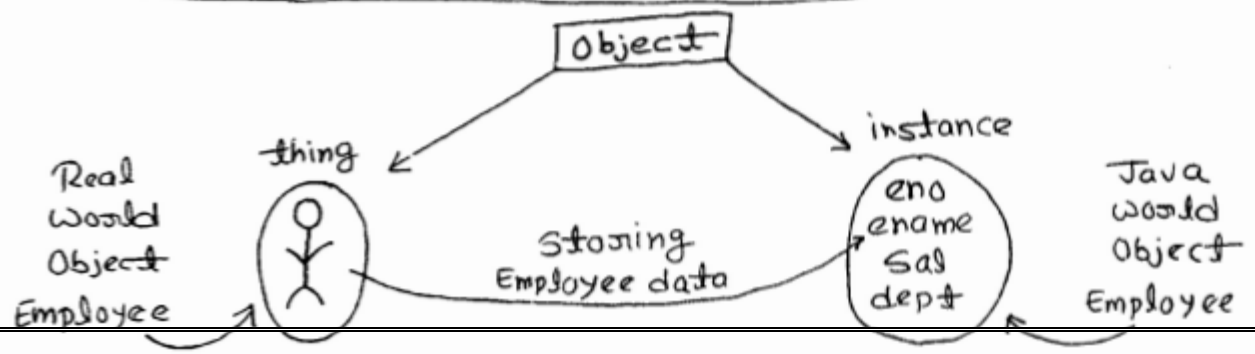
Ans:- 1. What is an object and why should we create object in programming world ?

Ans:- Object is a Real world thing that we can see & use for e.g. Person, Animal, Bike, Mobile, Laptop, bank account etc. are objects.

Technically in programming world object is an Instance of a class.

Final Definition of object (Interview)

"Object is a real world thing which is an Instance of a class in programming world.



23/08/17

In above diagram,
2nd circle ~~with sno, sname~~ is the memory created from the class emp with the variables eno, ename, sal, department.
In the memory we will store values of a particular employee.
For ex:- HK employee value.

As many real world object we want to create those many seperate new instances we must create from class employee in this new instance ~~i want~~ we must store values of the particular object.
For e.g. if you want to store HK object values and balayya object values. we must create 2 instances.

Q. Why should be create real world objects in programming world ~~we must will use~~ ?

- We will use OO programming to develop business operations in program world and to perform those operations by computer.
- As part of business we will have objects and we will store that object values perform operations of this object. Hence, in programming world also to ~~create the~~ store and perform operations of real world ~~business~~ object, we must create same objects in programming world as program.
- As many <u>types of objects</u> exist in business those many classes we must create.

Every object has 2 things :-

1. Type of object. (student, employee, ...)
2. Instance of object. (Student1, Student2, Emp1, Emp2....)

1. Type of object means the particular name that represents multiple objects comes under one group.
For ex. student is a type of object.
Employee is a type of object.
Animal is a type of object.

2. Instance of an object means a particular object of this type available in its group.
For ex. Raju is an instance of student object
Rani is an another instance of student object.

We will create type of object by using "class".
We will create instance of this object by using "new" keyword.
Below diagram will show you creating Employee objects in programming world using Java.

Procedure to create Employee object in java

Creating real world object in programming world is a three step process :-

step 1 : Create a class with required no. of variables and methods for representing object, for storing its values and implementing its operations.

step 2 : Create an Instance (memory) from this class

step 3 : ~~Stor~~ In this Instance store values of particular object.

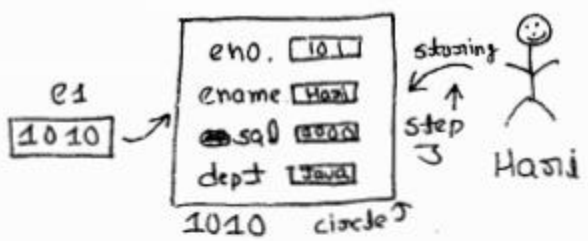
// Example.java

```
class Employee{
    int eno;
    String ename;
    double sal;
    String dept;
}
```

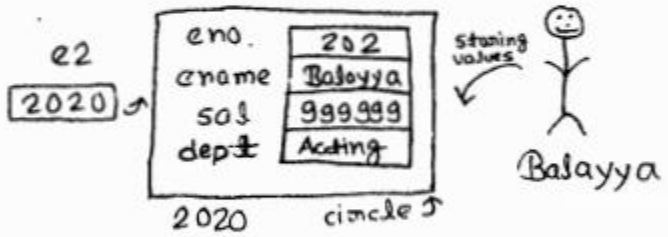
step 1 ↑

```
Employee e1 = new Employee();
```

← step 2



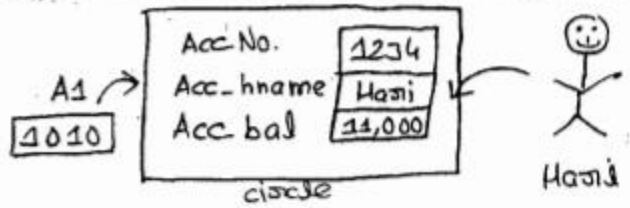

```
Employee e2 = new Employee();
```



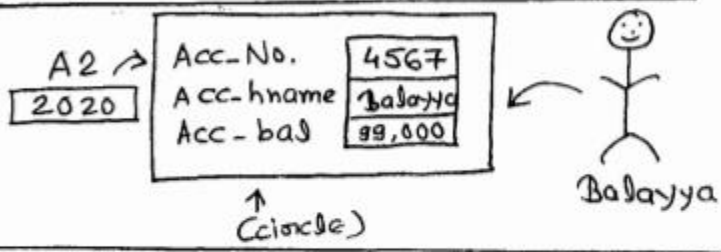
Q. Create Real world object Bank Account with values Ac_No, Acc_hname, Acc_bal. Create 2 accounts for 2 different customers

```
class BankAccount {
    long ~~int~~ Acc_No;
    string Acc_hname;
    double Acc_bal;
}
```

```
BankAccount A1 = new BankAccount();
```




```
BankAccount A2 = new BankAccount();
```

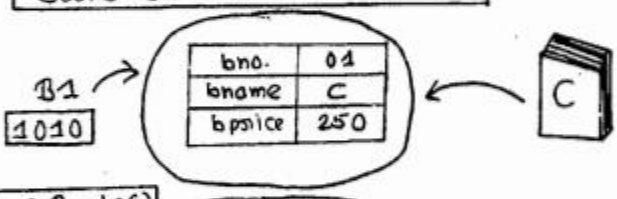


Assignment:- Look around you, you will find many objects of different types, Create minimum 5 Objects of your choices. In each type of object create 2 instances with its values.

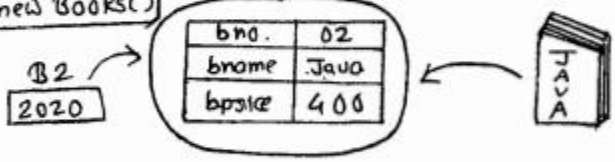
Answer:-

1.
```
class Books {
    int bno;
    string bname;
    int bprice;
}
```

```
Books B1 = new Books();
```

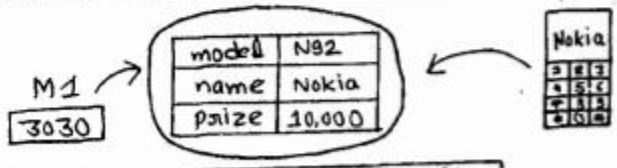



```
Book B2 = new Books()
```

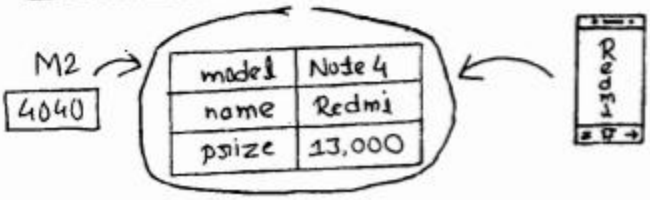


2.
```
class Mobile {
    int model;
    string name;
    double psize;
}
```

```
Mobile M1 = new Mobile();
```



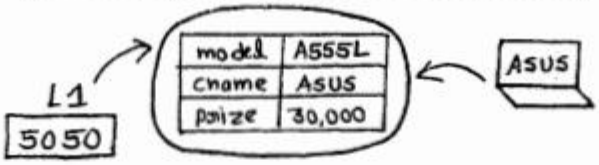

```
Mobile M2 = new Mobile();
```



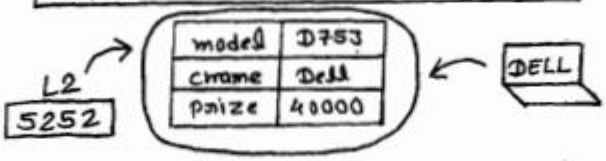
3.

```
class laptop{
    string Model;
    string cname;
    double price;
}
```

```
laptop L1 = new laptop();
```



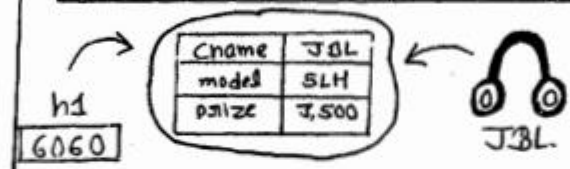

```
laptop L2 = new laptop();
```



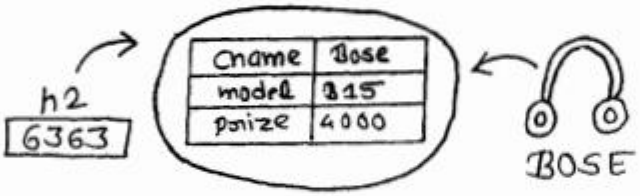
4.

```
class Headphone{
    string cname;
    string model;
    double price;
}
```

```
Headphone h1 = new Headphone();
```



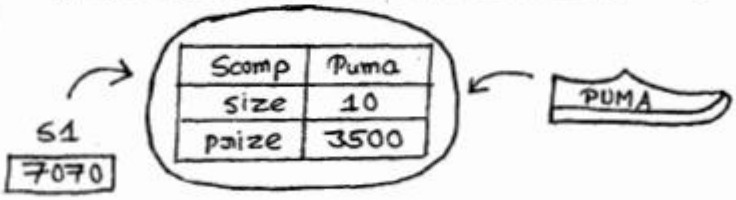

```
Headphone h2 = new Headphone();
```



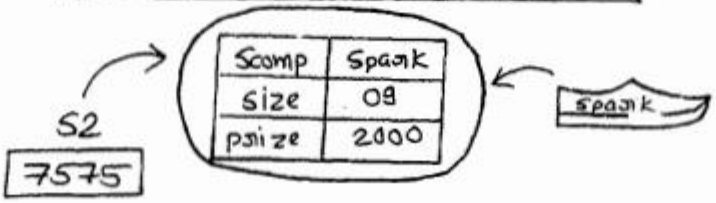
5.

```
class Shoes{
    string Scomp;
    int size;
    double price;
}
```

```
Shoes S1 = new Shoes();
```




```
Shoes S2 = new Shoes();
```



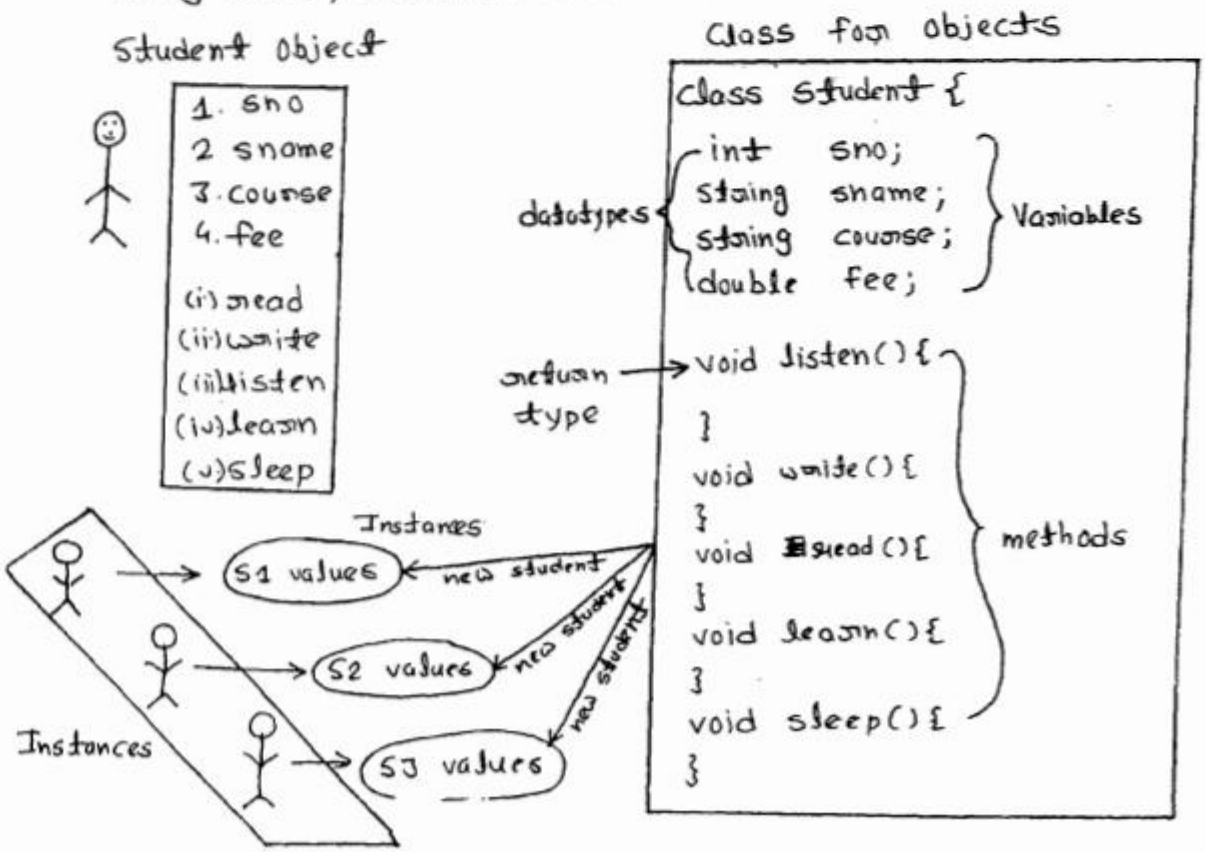
Q2 What is Class? Why Class?

A class is an user defined data type, it is used for storing multiple values of an object and also used for implementing operations of an object. So we can say Class is a collection of variables and method of an object.

Class is a blueprint or specification of an object, It specifies what an object contains and performs an operations by declaring variables and method. Class is called blueprint of an because from the same class we can create multiple Object of same type.

Class also can be called as Logical construct of an object because it construct the object logically by providing design here design means specifying object values and operations by using variable and method.

Below program will show you representing real world object Student in programming world with it properties by using class, variable and methods.



Q 3. What is an Instance? Why Instance?

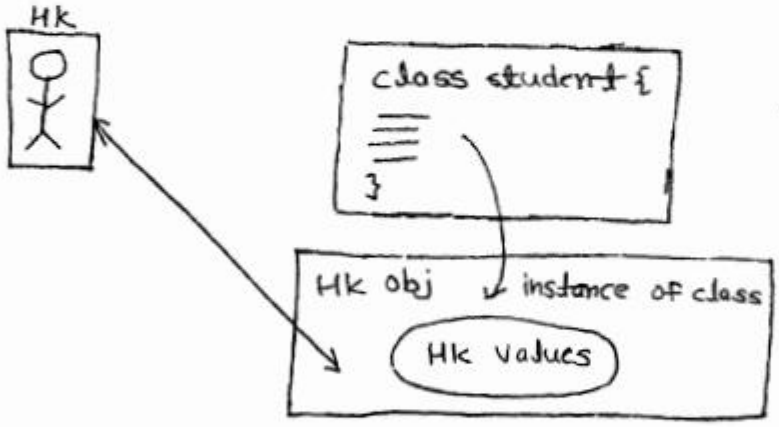
An Instance is a memory that is created from a class, used for representing a real world particular object of this class type by storing this object values.

In above diagram, we created 3 instances from the class Student to represent 3 students by storing their values.

Instance of a class is created by using "new" keyword and constructor.

Q. Why object is called Instance of a class?

Because the real world particular object is created in programming world using Instance of a class, thats why object is called Instance of a class.



Summary :-

1. Object is real world thing which is an instance of a class.
For e.g. Student, Employee, BankAccount, Bike.

2. Class is a blueprint of an object from which we will create multiple objects of same type, class specifies what an object will contain and what operations object perform.
For e.g. student { sno, sname, listen(), write() }

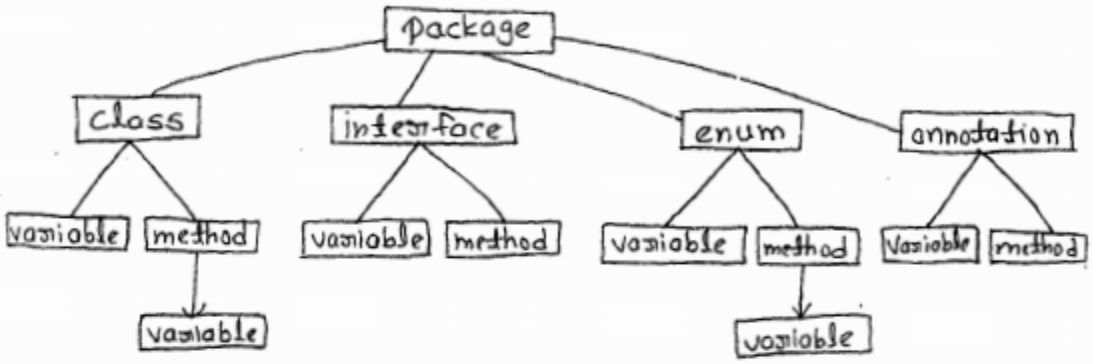
3. instance is a memory created from a class for storing particular object values of this class type.
For e.g. student (101, "Hk");
Student (102, "BB");

Programming Elements & Types of classes

1. object designing at different levels
2. use of interface, creation Rules
3. use of ~~class~~ abstract class, creation rules,
4. use of concrete class, creation rules,
5. use of final class, creation rules.
6. use of enum, creation rules
7. use of annotation, creation rules
8. sample project using all 6 types of classes
9. Interview questions, Java 8, 9 enhancements

Java Programming Elements

Java supports 7 programming elements, for representing real world object in the programming world, all the 7 programming elements are organized as shown in below diagram:-

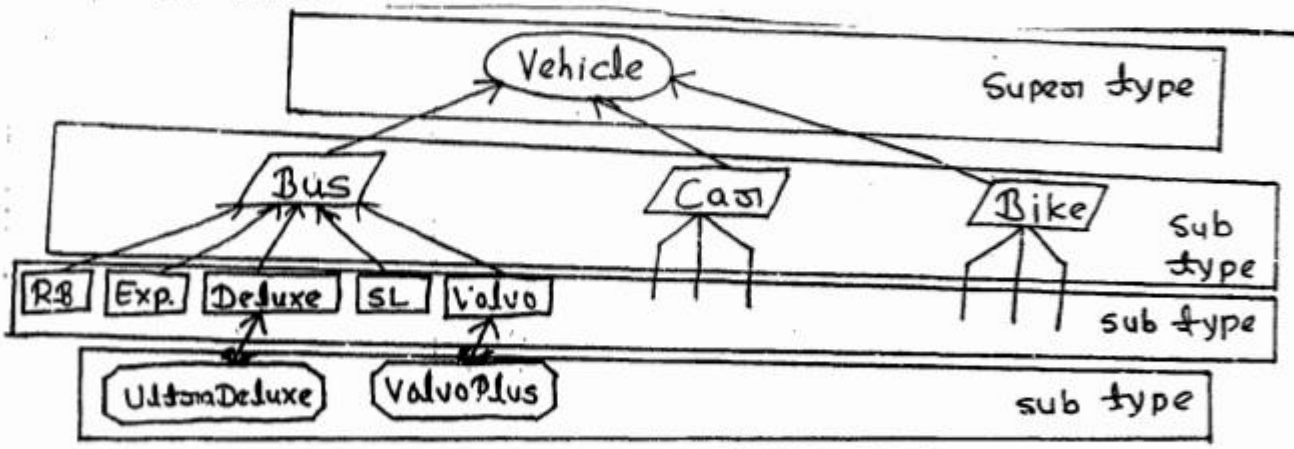


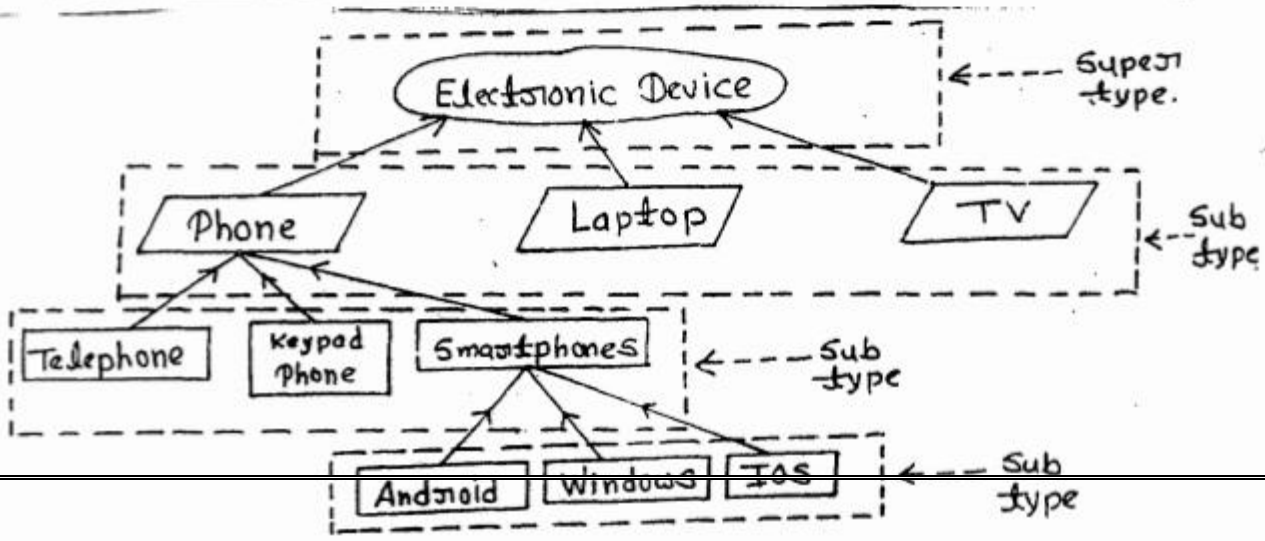
From the above diagram we can conclude, we have 7 different concepts we must learn to understand OOP and to create real world object in programming world by storing its values and by implementing its operations.

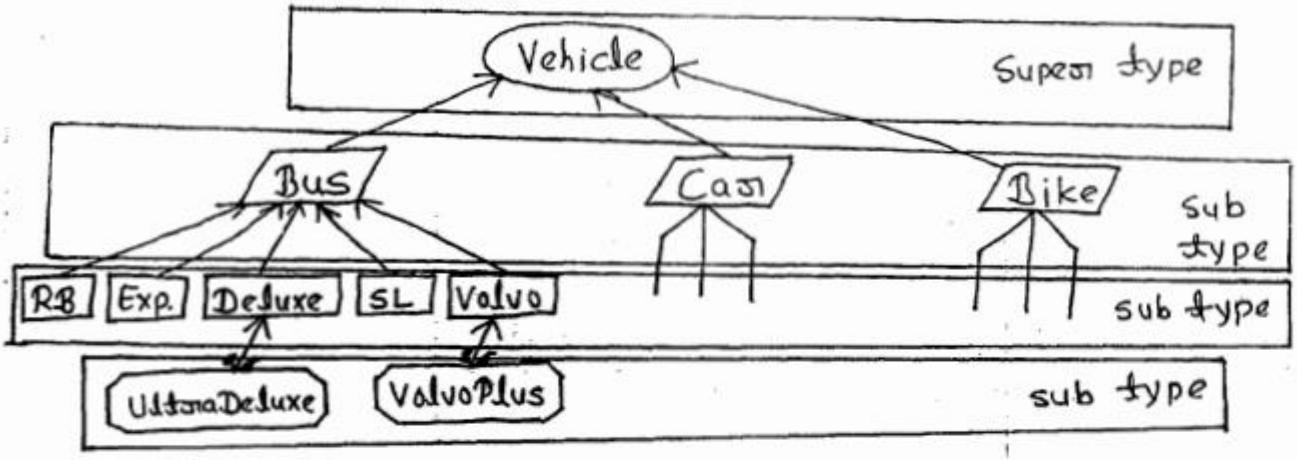
1. Package and types of packages and their purpose.
2. class and types of classes and their purpose.
3. variable and types of variables and their purpose.
4. block and types of blocks and thier purpose.
5. construnctor and types of constructor and their purpose.
6. methods and types of methods and their purpose.
7. this keyword and its forms and their purpose.
8. sharing object to other methods, its modification effect
9. pass by value and different test cases.
10. setter method, getter methods, mutator, accesser methods.
11. mutable and immutable objects.
12. Factory method, Factory class, factory design pattern.
13. Singleton design pattern class.
14. Reading run-time values from keyband.
15. OOP block diagram, steps to create real world object in programming world, developing a project to create real world object in programming world using all above OOP concept.

Class and Types of Classes

- Java supports 6 types of classes :-
  - (i.) interface
  - (ii.) abstract class
  - (iii.) concrete class.
  - (iv.) final class.
  - (v.) enum.
  - (vi.) annotation.
- All above 6 classes are used to represent real-world object in programming world.
- An object implementation will come at multiple levels because every real-world object will contain sub-types.
- Then to represent the object and its sub-types we have 6 types of classes.
- For example, consider Vehicle and Electronic Device objects, they have multiple levels & sub-type object. as shown below :-



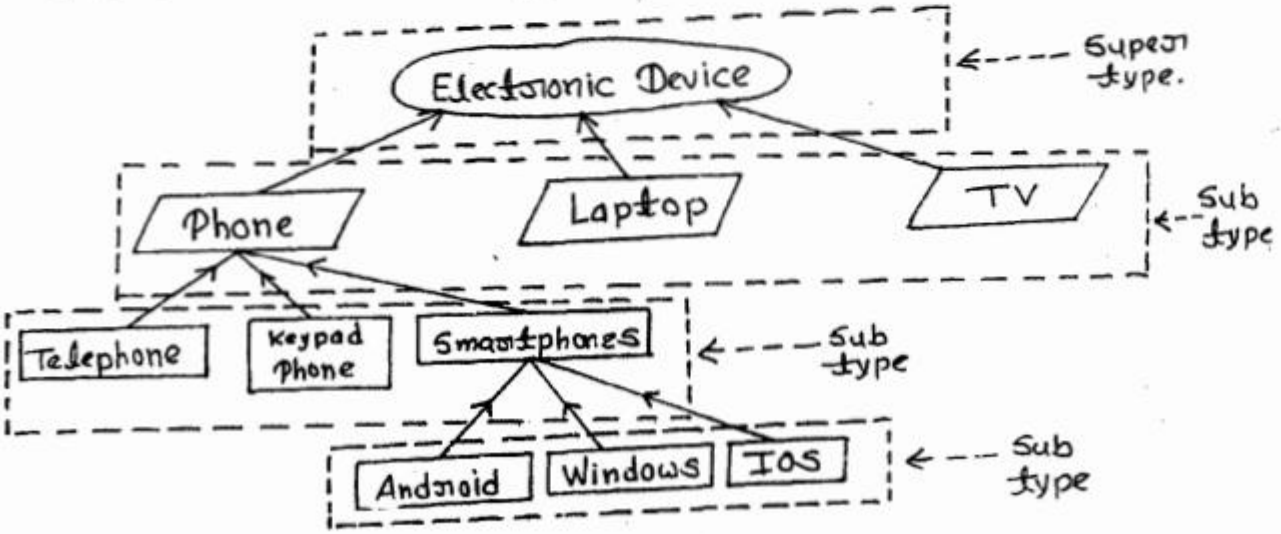




An object is considered as one type of object based on the operations it is supporting for example, we call a Bus is a vehicle, because it support Transport facility by contain different things like engine, breaks, wheels, seats etc.

We are not called vehicle, because we do not have transport operation required above properties.

By following above diagram, Design Electronic Device object with its sub-type object at different levels.



21/09/17

To create an objects main type and its sub type. we must use interface, abstract class, concrete class and final class.

Q. What is interface, purpose of interface, Creation syntax, Example and its set of rules?

⇒ • Interface is a fully unimplemented class, it is used to represent an object main type by declaring all its operations to be implemented by subtype program.

• Mainly interface is invented to support multiple type ~~interface~~ inheritance.

• interface created by using keyword "interface".

• Inside interface we can create only 5 different types of members :-

1.) public static final variables.
2.) public ~~static abstract class~~ abstract methods.
3.) public default methods.
4.) public static methods.
5.) public static inner classes.

• When an object's all operations implement logic will be changing from one sub type to another sub type of this object, then we must create this object by using interface.

• In project, when we want to declare operations of an object and we want sub-type programmer implement those operations, we must choose interface. We want declare operations of an object, because this object's operations implementation logic is changing from one sub type object to another sub-type object.

- we create interface by using the keyword "interface" and we must create sub class from interface by using "implements" keyword.

Syntax:-

optional mandatory

AM ELM interface <interfaceName> {

--------(members)

} ← optional

AM ELM class <className>

implements <interfaceName> {

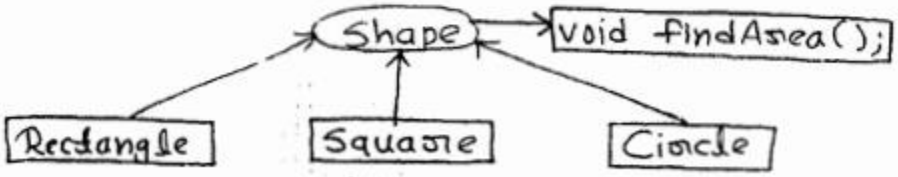
--------- interface methods

--------- implementation

}

implementing interface abstract method in its sub class is mandatory otherwise compiler throw error

- Sample program developement from real-world, consider the real-world object shape with with its sub types Rectangle, Square, Circle.



The operation findArea() implementation logic is changing from one type of shape to another shape, so we must declare findArea() operation by using abstract method, so that is implemented by all its sub type object.

Hence we must create the object ~~type~~ shape by using interface as shown below:-

```
//Shape.java
interface Shape {
    void findArea();
}
```

Here we declare abstract method findArea() or here we have created interface for the object shape by declaring its operation findArea(). Remember compiler will automatically adds the keyword public and abstract to this method.

```
//Rectangle.java
class Rectangle implements Shape {
    double l, b;
    public void findArea(){
        Sopln("Rectangle area:" + (l*b));
    }
}
```

Here we implement ~~the~~ interface shape and created sub type class deriving from interface. for implementing its operation for Rectangle object with required logic.

```
//Square.java
class Square implements Shape {
    double l;
    public void findArea(){
        Sopln("Square Area:" + (l*l));
    }
}
```

Here we created sub class deriving from interface for implementing its operation for Square object with required logic.

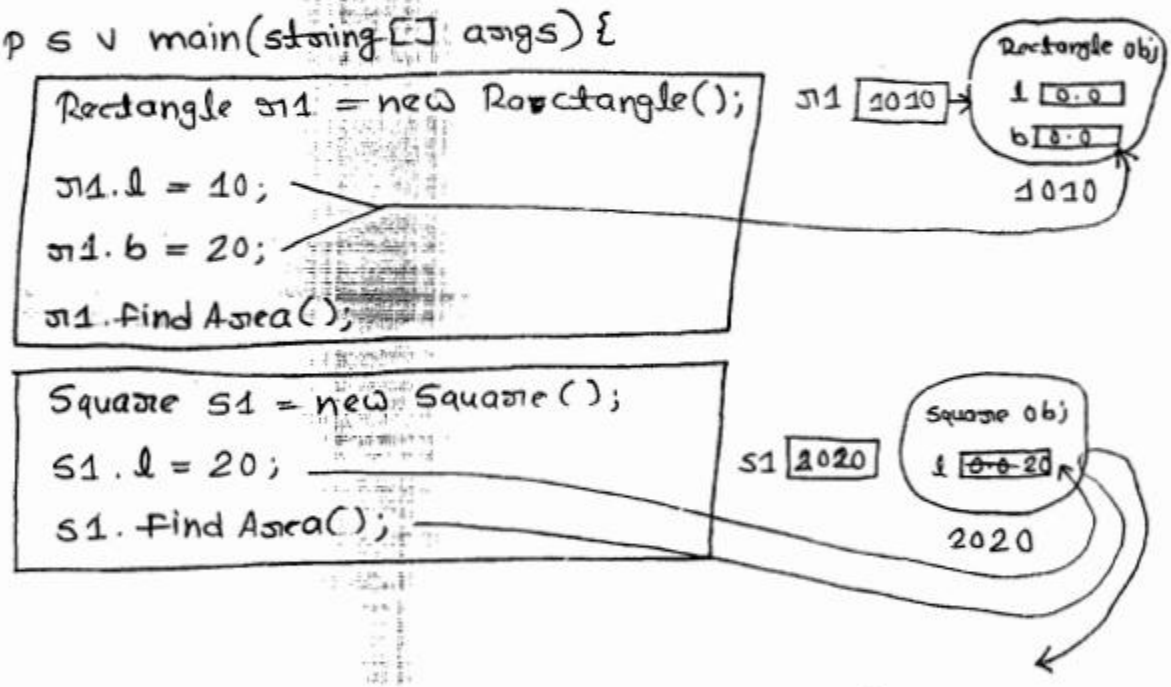
```
//Circle.java
class Circle implements Shape {
    double r;
    public void findArea(){
        Sopln("Circle Area:" + (3.14*(r*r)));
    }
}
```

Here, another sub-class deriving from interface for implementing its operation for circle object

Just by creating shape object and its sub-types, is not sufficient, because shape operation implementation logic is not executed automatically.

To execute findArea() logic from its different subtype object we must define main method class, inside main method we must create sub-type object instance, should initialize it with required values and then, should invoke findArea() method.

```
//painter.java
class painter {
    p s v main(string[] args) {
        Rectangle r1 = new Rectangle();
        r1.l = 10;
        r1.b = 20;
        r1.findArea();
        Square s1 = new Square();
        s1.l = 20;
        s1.findArea();
```



O/p:-
Square Area: 400.0

What is the Meaning of 22/09/17

```
r1.findArea();
s1.findArea();
c1.findArea();
```

in the above program:-

⇒ In above 3 statements we are invoking findArea() of shape interface to executing from its sub classes. In the statement r1.findArea(); compiler and JVM will search findArea() method in rectangle class and executes its logic in rectangle class with the values available in r1 object.

In the statement s1.findArea(); we are asking compiler and JVM execute findArea() from square class with the values available in s1 object.

In the statement c1.findArea(); compiler and JVM will search findArea() method in circle class and execut

You yourself speak the meaning of below 6 statements:-

```
r2.findArea();
r3.findArea();

s2.findArea();
s3.findArea();

c2.findArea();
c3.findArea();
```

Q. What are the complete set of rules in creating interface?

1. interface created by using keyword "interface".
2. ~~We can save~~ Like a class, we must also save interface by using .java extension. The name of the file can be same or diff. from interface name, but it is recommended to save java file & interface name with same name for easy identifying & auto compile.
3. Like as class when we compile interface java file, we will get .class file with the interface name.

We can compile interface, but we can not execute interface, because interface does not contain main method.

But java 8v onwards interface also can contain static methods, hence inside interface we can define main method and we can execute interface by using java command.

Sample Program:-

Example.java

```
interface Example {
    public static void main (String [] args) {
        System.out.println(" From interface MM");
    }
}
```

Compile and Execute:-

```
>javac Example.java
    |- Example.class
>java Example
o/p: From interface MM
```

Note:- You must compile and execute above program by using Java 8v compiler & JVM. If you use lesser version compiler & JVM throw error.

2. To an interface we can apply only default, public, abstract and strictfp modifiers.

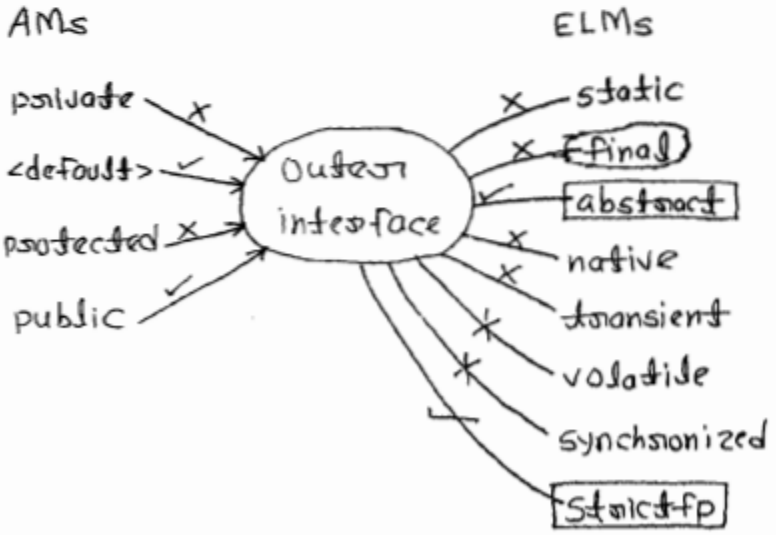
We can not apply other modifiers to outer interface if we try to apply other modifiers to interface compiler will throw error.

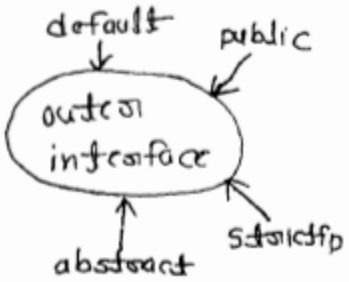
We can not apply static to outer interface, because by default we can access interface name without created object, basically static keyword is only allowed to the members created inside a class or interface.

we can not apply final to interface (outer/inner) because interface should have sub class.

we can not apply native, synchronized to interface because these two modifier will ~~not~~ work on logic of method.

we can not apply transient and volatile to interface because these two ~~methods~~ modifiers will work on variables values.





3. We can not derive an interface from a class, because a class contains concrete method, that can not be exist in interface.

But we can derive an interface from other interface because an interface contains abstract method, it can be inherited and can exist in another interface.

Example:-

```
class A {
    void m1(){}
}
class B extends A {
    ✓
}
interface C extends A {
    X CE
}
```

```
interface I1 {
    void m1();
}
interface I2 extends I1 {
    ✓
}
```

Rule:- To derive an interface from other interface, we must use "extends" keyword.

4. We must derive an interface from other interface by using extends keyword only, we can not use implements keyword.

extends means reuse the members available in super class.

implements means we must implement (provide logic) to the methods inheriting from an interface.

Sub interface can not implement method of super interface, so we can not derive sub interface by using implements keywords.

```
interface I1 {
    void m1();
}
```

```
interface I2 extends I1 {
}
```

```
interface I3 implements I1 {
}
```

23/09/17

Java does not support multiple inheritance
It means we can not derive a class from multiple classes
Compiler will throw error.

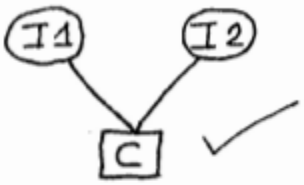
```
class A { int a = 10; }
class B { int a = 20; }
class C extends A, B { } X CE '{' expected.
```

Java does not support multiple inheritance, because of ambiguous error in accessing constructor, variables and method from super classes by using super keyword.
In Multiple Inheritance compile can not decide from which super class constructor, variable and method should execute. Hence to eliminate this ambiguous error, Multiple inheritance is not given java and more over this inheritance is not required in real life and real time project.

Java does not support multiple inheritance with class but supports multiple type inheritance with interfaces.

It means we can not derive a class from multiple classes but we can derive a class from multiple interfaces.

```
interface I1 {}
interface I2 {}
class C implements I1, I2 {}
```

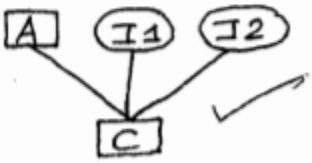


We will not get ambiguous error with interfaces, because interface doesnot contain constructor, and does not contain implemented method and super key-word will not references to interface rather it reference to super class that java.lang.object.

In java

1.) We can not derive a class from multiple classes.
2.) We can derive a class from multiple interfaces.
3.) We can derive a class from one class and one interface.
4.) We can derive a class from one class and multiple interfaces.

```
class C extends A
        implements I1, I2 {}
```



5.) We can not derive a interface from a class.
6.) We can derive a interface from other interface.
7.) We can derive an interface from multiple interfaces.

In java we can not create a class without inheritance
If we create a class without inheritance, it is automatically derived from java.long.object class by compiler software.

For e.g. if we create a class as

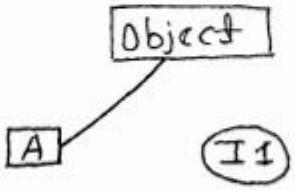
```
class A {}
```

compiler change it as

```
class A extends java.long.object
```

But we can create interface without inheritance, it means if we create a interface it will not be derived from any other class or interface.
It means interface is not sub type of java.long.object.

For e.g. if we create `Interface I1 {}`
no changes are added by compiler software.



Q. If we create a class deriving from an interface is also deriving from any other class ?

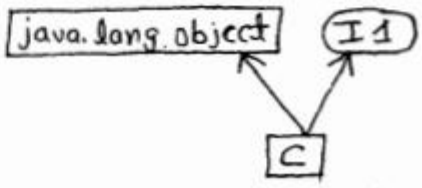
⇒ Yes, it will be also derived from java.long.object class automatically by compiler software.

For example:-

```
Interface I1 {}
class C implements I1 {}
```

It is changed by compiler as

```
interface I1 {}
class C extends java.long.object
        implements I1 {}
```



4. Inside interface we can create only :-

(i) public static final variables.

(ii) public abstract methods.

(iii) public default implemented methods.

(iv) public static implemented methods.

(v) public static inner classes.

Then interface we can not create :-

(i) instance variables.

(ii) non-static implemented methods (CM).

(iii) blocks (SB/NSB).

(iv) constructor.

If we create above 4 members inside interface compiler will throw error.

Note:- If we create instance variable inside interface, we do not get any compile time error, because compiler will convert to public static final variable automatically.

For example:-

```
interface Example {
    int x = 10; ✓
    void m1 () { } ✗
    Example () { } ✗
    static { } ✗
    { } ✗
    void m2(); ✓
    default void m3() { } ✓
    static void m4() { } ✓
    class A { } ✓
```

>javap Example

```
interface Example {
  public static final int x=10;
  public abstract void m2();
  public default void m3() {}
  public static void m4() {}
  public static class A {}
}
```

5. Inside interface we must assign value to variable in its declaration statements itself, because interface variable is static final variable.

Static final variable must be initialized, else compiler will throw error.

For e.g.

```
interface Example {
    int x; x CE: "=" expected
}
```

6. we can not create object from interface or we can not instantiate interface, because it not fully implemented class, it contains abstract methods.

But we can create ref. var. by using interface, because for storing its implemented class object, and executing its method from implemented class.

For example:-

```
class Mobile {
    psv main(String[] args) {
    SIM sim; ✓
    //sim = new SIM(); X
    sim = new BSNL();
    sim.call(); ✓
    }
}
```

```
interface SIM {
    void call();
}
```

```
class BSNL
    implements SIM {
public void call() {
    Sopln("Calling using BSNL");
}
}
```

7. We must derived sub class from interface by using implements keyword.

```
interface SIM {}
//class BSNL extends SIM {} X
class BSNL implements SIM {} ✓
```

8. If interface contains abstract methods, its sub class must implement all abstract methods declared in this interface, else we should declare sub class as abstract, otherwise compiler throw error.
For example:-

```
interface SIM {
  void call();
  void sms();
}
```

```
class Airtel implements SIM { } X
```

```
class Vodafone implements SIM {
    public void call () { }
}
```
X

```
abstract class DoCoMo
        implements SIM { }
```
✓

```
abstract class JIO
    implements SIM {
        public void call ()
        { }
    }
```
✓

25/09/17

Note:-

1. If we derive a class from an interface we must implement all abstract methods.

2. If we do not want implements some of the methods of this interface, we must declare sub class as abstract

3. If we declare sub class as abstract
   (i) we can ignore implementing all methods.
   (ii) We can ignore implementing some of the method it means we can implement some methods and we can ignore implementing some other methods.
   (iii) By declaring ~~class~~ sub class as abstract, we can implement all methods.

Identify compile time error in below programs:-

```
interface SIM {
    void call();
    void sms();
}
```

```
class Airtel implements SIM {
}
```
✗ CE

```
class Airtel implements SIM {
    public void call() {}
}
```
✗ CE

```
abstract class Airtel
    implements SIM {
}
```
✓

```
abstract class Airtel
    implements SIM {
    public void call() {}
}
```
✓

```
abstract class Airtel
    implements SIM {
    public void call() {}
    public void sms() {}
}
```
✓

```
class Airtel implements SIM {
    public void call() {}
    public void sms() {}
}
```
✓

9. We must implement interface methods with public keyword, because interface methods are public. If we implement interface methods without using public keyword, we get error.

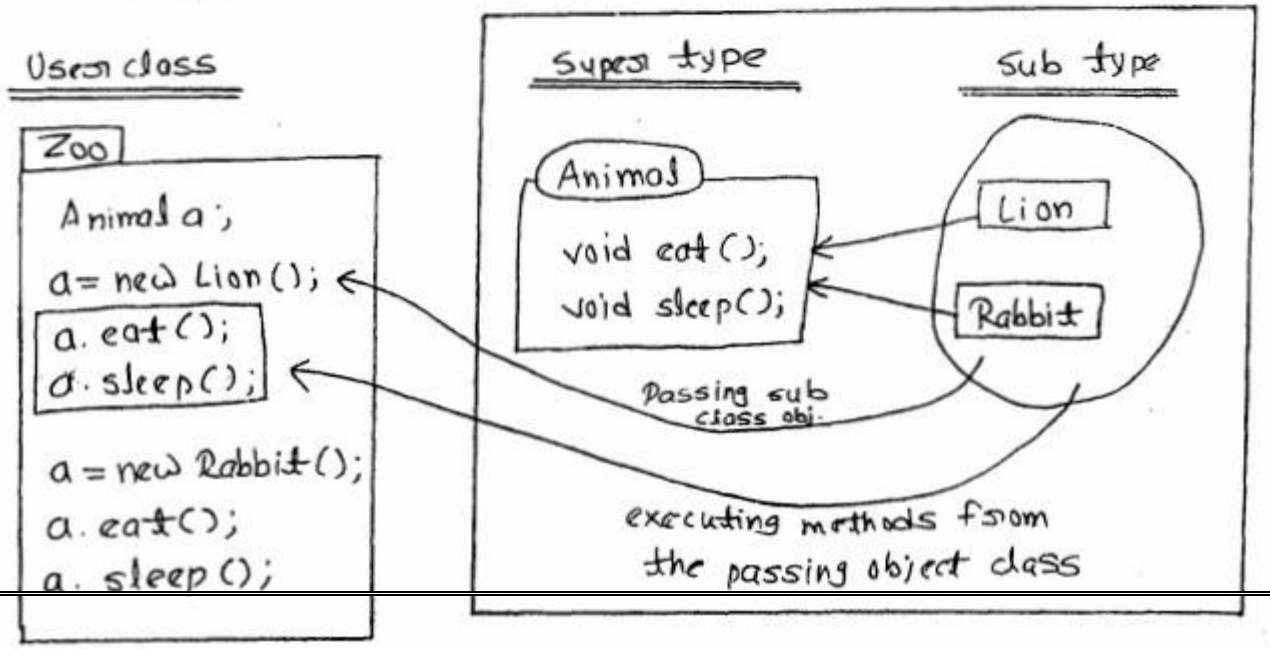
```
interface SIM {
    void call();
    void sms();
}
```

```
class Airtel implements SIM {
    void call() {} X CE
    void sms() {} X CE
}
```

```
class Airtel implements SIM {
    public void call() {} ✓
    public void sms() {} ✓
    void mms() {} ✓
}
```

Airtel class own method
No need to declare as public.

Que:- With the knowledge you gain on interface, its creation syntax, with its implementation class syntax and its rules try to design and develop Zoo_Animal Project.



Ans:-

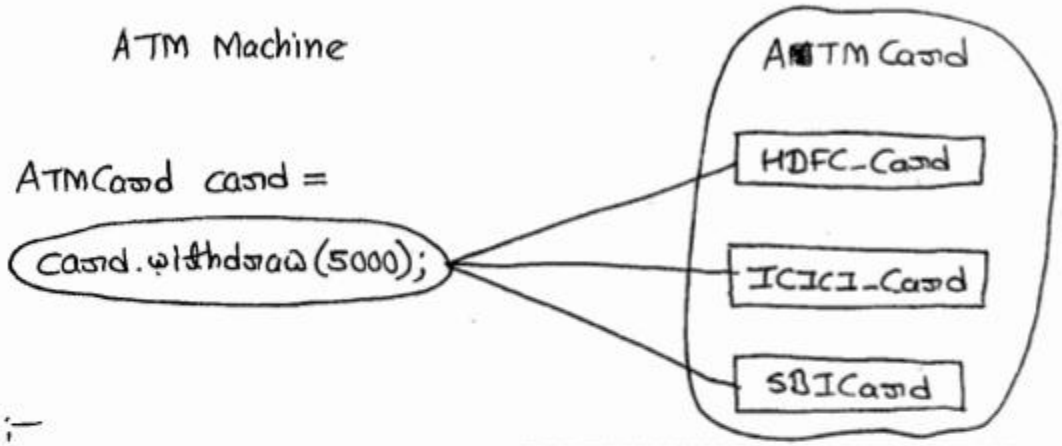
```java
//Animal.java
interface Animal {
    void eat();
    void sleep();
}
```

```java
//Lion.java
class Lion implements Animal {
  public void eat(){
    Sopln("Lion eat Non-Veg");
  }
  public void sleep(){
    Sopln("Lion sleeps in Cave");
  }
}
```

```java
//Zoo.java
class Zoo{
  psvm main(String[] args){
    Animal a;
    a= New Lion();
    a.eat();
    a.sleep();

    a = New Rabbit();
    a.eat();
    a.sleep();
  }
}
```

```java
class Rabbit implements Animal {
  public void eat(){
    Sopln("Rabbit eat Carrot");
  }
  public void sleep(){
    Sopln("Rabbit sleeps in Hole");
  }
}
//Rabbit.java
```

Q. Develope interface based program, to achieve Dynamic binding ~~wit~~ between the objects ATM Machine and ATM Card.



Ans:-

```
//ATMCard.java
interface ATMCard {
        void withdraw(int a);
}
```

```
//ATMMachine.java
class ATMMachine {
    p.s.v main(String[] args) {
        ATMCard card;
        card=new HDFCCard();
        card.withdraw(5000);

        card = new ICICICard();
        card. withdraw(5000);

        card = new SBICard();
        Card. withdraw(5000);

    }
}
```

```
//HDFCCard.java
class HDFCCard implements ATMCard
{
    public void withdraw(int a)
    {
        S.o.printf("%d rs. withdrawn
                    from HDFC Bank", a);
        S.o.pln();
    }
}
```

```
//ICICICard.java
class ICICICard implements ATMCard {
    public void withdraw(int a) {
        S.o.printf("%d rs. withdrawn
                    from ICICI Bank\n", a);
    }
}
```

```
//SBICard.java
class SBICard implements ATMCard {
    public void withdraw(int a) {
        S.o.printf("%d rs. withdrawn
                    from SBI Bank\n", a);
    }
}
```

Q. When should we store sub class object reference in super ~~type~~ class type referenced variable?

⇒ For achieving loose coupling & Runtime polymorphism (Dynamic binding) we must store sub type object ref in super class type ref. variable.

- If we store sub class object ref. directly in the same sub class type ref. variable, user class will tightly coupled with particular sub type object. It means user class can not change current sub type object to another sub type object without modifying source code.

  For example:- CDMA Mobile is tightly coupled with one SIM (Reliance) and it can not changed to another company SIM (TATA).

- If we store sub class object ref. in super class type ref. var, user class will become loosely coupled with sub type objects. It means user class can change one sub type object to another sub type object without changing source code.

  For example:- GSM Mobile is loosely coupled with SIM object. It can accept any company SIM and can change to another company SIM without changing its software.

Q. Identify whether below program is tightly coupled or loosely coupled.

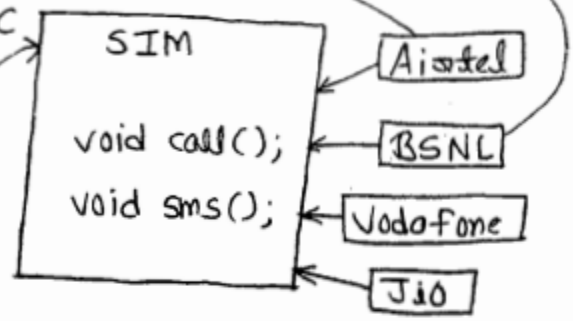
```
class Mobile {
    void insertSIM(Airtel a) {
        a.call();
        a.sms();
    }
}
```

TC → Airtel ✓

BSML ✗

Vodafone ✗

```
class Mobile {
    void insertSIM(SIM sim){
     sim.call();
     sim.sms();
    }
}
```



To achieve loose coupling and dynamic binding we have to develop sub type objects with inheritance and user class we must call methods using super class type referenced variable.

Abstract Class :-

1. What is abstract class ?
2. Why abstract class ?
3. When should we choose abstract class ?
4. Syntax to create abstract class ?
5. What are the diff. members allowed in abstract class ?
6. Saving, Compiling and Executing abstract class ?
7. Rules in creating abstract class, interview and OCP questions ?
8. Creating sub class from abstract class ?
9. Rules on creating sub class ?

1. What is abstract class ?

⇒ An abstract class is a partially implemented class.

2. Why abstract class ?

⇒ Abstract class is used for declaring some of the operations and implementing some of the operations of an object common to all its sub types.

3. When should we choose abstract class ?

⇒ When we want to represent an object by implementing some of the operations common to sub types and by declaring some of the operations those to be implemented by sub types, then we should choose abstract class. Mainly abstract class is used for representing sub type objects for providing partial implementation. common to next subtypes.

4. Syntax to create abstract class.

⇒ we can create abstract class by using the keywords abstract and class

```
abstract class classname
```

For example:-

```
abstract class Bus {
    public abstract void engine();
    public void break() {
        Sopln("Bus have 2 breaks");
    }
}
```

abstract class Bus

method engine() is declared

method breaks() is implemented

The method engine is implemented by Bus sub type object as per their requirements.

Hello, chitti, closely look at engine() method declaration We have placed the keyword abstract explicitely.

We didn't place abstract keyword explicitly in the interface.

Interface by default contains abstract methods, so we no need to add abstract modifier to abstract methods, compiler will add automatically.

But abstract class is a class, it contains concrete methods also, so compiler can not add abstract modifier to abstract methods automatically. We must add abstract modifier to abstract methods in abstract class, else compiler will throw error.

*. To implement logic to abstract method of an abstract class, we must define sub class from this abstract class.

To create sub class from an abstract class, we must use extends keyword.

The rule on sub class is :-

1. We must use extends keyword.
2. We must implement all abstract methods declared in this abstract class, else we must declare this sub class also abstract otherwise compiler will throw error.

```
abstract class Bus {
  abstract void engine();
  void breaks(){
    Sopln("Bus has two breaks");
  }
}
```

```
class RB extends Bus {
    void engine(){
      Sopln("RB run at 40 KMPH");
    }
}
class volvo extends Bus {
    void engine(){
      Sopln("volve runs at 110 KMPH");
    }
}
```

we no need to use public for Implementing abstract class methods, bcoz they are not public.

```
class Driver {
   psv main(string [] args){
      Bus b;
      b = new RB();
      b.engine();
      b.breaks();

      b = new volvo();
      b.engine();
      b.breaks();
   }
}
```

executed from RB

executed from Bus (reusing)

Compile and execute Driver class.

```
> javac Driver.java
> java Driver.
```

Output:- RB runs at 40 KMPH.
Bus has two breaks.
Volvo runs at 110 KMPH.
Bus has two breaks.

27/9/17

SCJP and Interview questions on Abstract class :-

Rules

Rule 1 :- We must create an abstract class by using the keywords "abstract" and "class".

For example :-

```
abstract class Example {

}
```

It is an abstract class.

We can save, compile and execute an abstract class. Abstract class is also saved with .java extension and when we compile it using javac command. Compiler software will generate class file with the class name.

If we define main() method inside the abstract class, we can also execute it by using java command.

For example :-

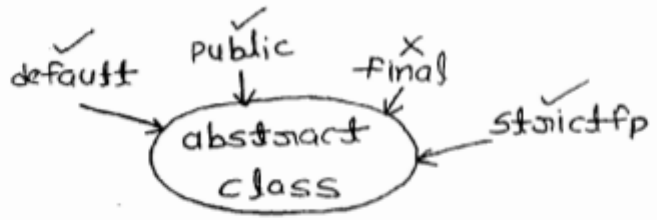
```
//Example.java
abstract class Example {
    p s v main(string[] args) {
        Sopln("Hi");
    }
}
```

>javac Example.java
↳ Example.class

>java Example

Output :- Hi

Rule 2 :- The allowed modifiers (AM and ELM) to an abstract class are only default, public and strictfp.



If we declare abstract class as final, compiler will throw Error :- illegal combination of modifiers.

Identify errors in the below list :-

abstract class A { } ✓
public abstract class B { } ✓
final abstract class C { } ✗ → we can not declare abstract class as final, because abstract class must have sub class.
strictfp abstract class D { } ✓

Rule 3 :- We can derive abstract class either from a class or from an interface. or from both.

- Remember, java does not support multiple inheritance using classes, it supports multiple inheritance by using interfaces.
- Hence we can derive an abstract class from only one class and can derive abstract class from multiple interfaces.
- If we derive an abstract class only from interfaces, internally it is also derived from java.lang.object class.
- If we derive an abstract class from an interfaces, we no need to implement all abstract methods of this interface.

To derive Abstract class from other class, we must use extends keyword.

To derive an Abstract class from an interface, we must use implements keyword.

```
class A {}
interface I1 {}
```

```
abstract class C extends A {}
abstract class D implements I1 {}
```

```
abstract class E extends A
                implements I1 {}
```

```
abstract class F implements I1 extends A {}
```
X

implements should use after extends.

Rule 4:- Inside abstract class we are allowed to define all regular static and non static 10 members which are allowed to inside a regular class. In addition to those 10 members also allowed to define abstract method.

```
abstract class Example {

    static variable        Non-static variable
    static block           Non static block
    static method          Non static method
    Main Method            constructor
    static Inner class     Non static Inner class

                +
            abstract method
}
```

Note :- Compared to interface, abstract class members are by default are not public, they are default AM members.

If we want to change default AM to private/protected/public we are allowed to change.

But in interface we are not allowed to change from public to protected/default/private.

Rule 5 :- Compared to interface,

1.) Inside abstract class we must explicitly place abstract modifier in the creation of abstract method else compiler will throw error. But in case interface we no need to place abstract modifier in abstract method, compiler will add it automatically.

2.) Inside abstract class abstract method is not public, its AM is default. If we want to change its AM, we are allowed to change it, but only to either protected/public.

We can not declare as private

```
abstract class Example {
    void m1(); X
    abstract void m2(); X
    abstract void m3(); ✓
    private abstract void m4(); X
    protected abstract void m5(); ✓
    public abstract void m6(); ✓
    final abstract void m7(); X
    static abstract void m8(); X
    strictfp abstract void m9(); X
```

Note:- Inside an abstract class to an abstract method, we are allowed to apply only default, protected, public member.

Rule 6:- We can not instantiate abstract class, it means we can create object from an abstract class, compiler will throw error, because abstract is not fully implemented class.

For example:-

```
abstract class Example {
    abstract void m1();
}
```

```
class Test {
  psv main(string[] args) {
    Example e1 = new Example(); X
    Example e2; ✓
  }
}
```

Note:- We can create reference variable by using abstract class. It is used for storing its sub class objects.

28/09/17

Rule 7:- We must derive subclass from abstract class by using the keyword extends because abstract class is a class.

We must derive a class from another class by using abstract class

```
abstract Bus {
}
class RB extends Bus {} ✓
class RB implements Bus {} X
```

Rule 8 :- When we derive a class from abstract class, we must override/implements all abstract methods of this abstract class else, we must declare this sub-class also abstract otherwise compiler will throw error.

```
abstract class Example {
    abstract void m1();
    abstract void m2();
}
```

```
class sample extends Example {
    void m1() {}
}
```

```
abstract class Sample
    extends Example {
    void m1() {}
}
```

```
class sample extends Example {
    void m1() {}
    void m2() {}
}
```

```
abstract class sample
    extends Example {
    void m1() {}
    void m2() {}
}
```

Rule 9 :- The default AM of abstract method in abstract is "default" but not public like in interface.

Hence in sub class while overriding abstract method of abstract class, we no need to use public always.

We must choose AM in sub class, based on the AM used in super class.

AM in super class	allowed AM in subclass
default →	default / protected / public
protected →	protected / public
public →	public

For example:-

```
abstract class Example {
    abstract void m1 ();
}
```

```
class S1 extends Example {
    private void m1() {}
}
```
X CE

```
class S2 extends Example {
    void m1() {}
}
```
✓

```
class S3 extends Example {
    protected void m1() {}
}
```
✓

```
class S4 extends Example {
    public void m1() {}
}
```
✓

Rule 10:- We can declare class as abstract, even though we have not created any abstract method in it.

If class is declared as abstract there is no rule that it must have abstract method, but if class contains abstract methods, this class must be declared as abstract, else compiler will throw error.

```
abstract class Example {
    void m1() {}
    void m2() {}
}
```

abstract →

```
class Sample {
    void m1() {}
    abstract void m2();
}
```
X

03/10/17

Q.) What is the need of declaring a concrete class as abstract class ?

⇒ For stop creating object of this class and to execute this class non-static methods are created by using sub-class object for executing those methods logic for sub class object.

Q.) Do you have predefined abstract concrete class ?

⇒ Yes, we have two pre-defined abstract concrete class :-

(i.) java.awt.Component.

(ii.) Javax.servelet.http.HttpServelet.

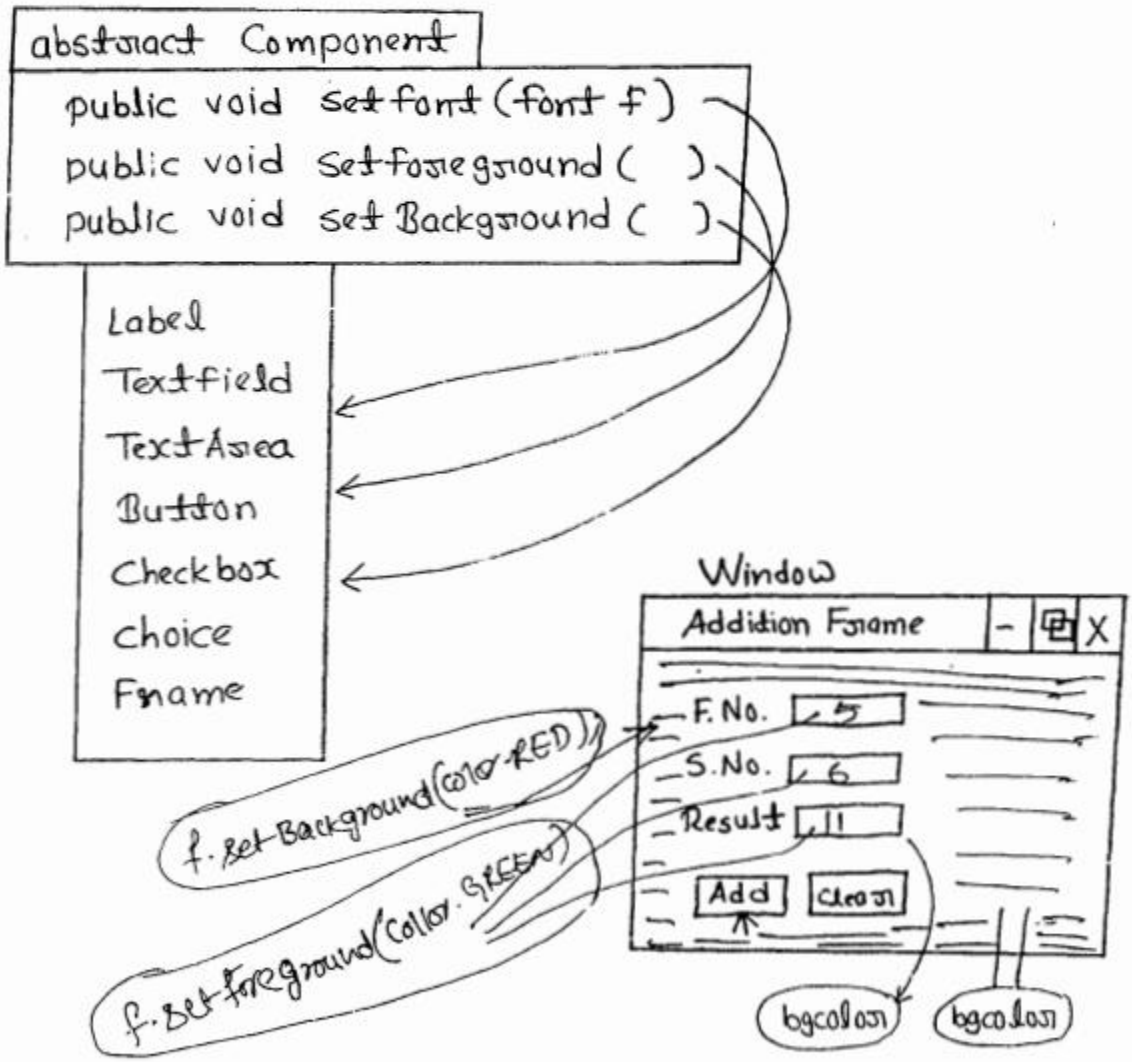
In above 2 classes Component and http servelet all methods are concrete methods, but still these two classes are declared as abstract for stop creating objects from these two classes and executing methods defined in these two classes for their sub type object.

[P.T.O.]

For example :-

The class component contains methods like setfont(), setforeground(), setBackground(), methods for setting font and color to the file components (sub-types) those added to windows.

Hence these methods must be called and executed for components sub-type objects. This class component is declared as abstract.



*. Concrete class points :-

1. What is concrete class ?
2. When should we choose concrete class ?
3. Give me real time example for concrete class ?
4. Creation syntax of concrete class ?
5. Example program, saving, compiling and executing.
6. Rules in creating concrete class ?
7. Interview and OCJP questions on concrete class ?

1. What is a concrete class ?

⇒ Concrete class is a fully implemented class.

A class that contains only concrete methods to provide full implementation to an object all operations is called concrete class.

Anything you are seeing and using in real world is an instance of a concrete class.

For example :-

RB, Exp, Volvo buses are instances of a concrete class, Lions, Rabbits, Tiger are instances of a concrete class.

2. When should we Create concrete class ?

⇒ We must create concrete class in 4 senarios :-

(i.) For creating sub-type object with all operations implementation.

(ii.) For creating main type of an object by providing all operations implementations directly without thinking of next sub-types and multiple inheritance.

(iii.) For developing mathematical operations calculations.

For example :- Adding, subtracting, multiply, dividing, find palindrome, displaying in diff. pattern.

(iv.) For developing executable class with main method to start project execution.

3. Syntax to create concrete class ?

⇒ should not apply abstract modifier to create Object.

```
AM ELM class classname extends parent_class
                                implements interface_name {

    (i.) variables
    (ii.) blocks
    (iii.) constructors
    (iv.) methods
    (v.) inner classes.
}
```

Sub class 1 | Sub class 2 | Sub class n

Note:- like the keyword "abstract" we do not have the keyword "concrete" to create concrete class.

* Refer previous pages to see concrete class usage for creating sub-type objects for providing implementation to abstract methods those are declared in super type class.

For example:-

In Shape project

(i) Shape is super type, we created it using interface.
(ii) Rectangle, square, circle are sub types, we created them by using concrete class.

In Animal project

(i.) Animal is super type, It is interface.
(ii.) Lion, Rabbit are sub types, they are concrete class.

*. Below Example will show you concrete class creation for mathematical operation implementation and executable class creation.

Q. Develope a program to find given number is even or odd ?

⇒

```
class Test {
    public static void main(String [] args) {
        boolean even = EvenorOdd.find (10);
        if (even) {
            Sopln("The num. 10 is even");
        }
        else {
            Sopln("The num. 10 is odd");
        }
    }
}
```

True

False

```
class EvenorOdd {
    static boolean find (int n) {
        return (n%2);
    }
}
```

04/10/17

Rules on concrete class.

⇒ R1:- We must create concrete class by using the keyword "class".

R2:- The allowed modifiers to a concrete class are:-

(i.) <default>
(ii.) public.
(iii.) final.
(iv.) strictfp.
(v.) abstract.

Note:- as per syntax we are allowed to apply the modifier abstract to a concrete class, then we should call this concrete class as "abstract concrete class".

R3:- We can create a concrete class as sub class deriving from another concrete class or from an abstract class or from an interface.

- We must use "extends" keyword to derive it from a concrete class or abstract class.
- We must use "implements" keyword to derive it from an interface.

If there is any abstract method declared inside the derived abstract class or inside interface we must implement this method, else we must declare our class also as abstract, otherwise compiler will throw error.

R4:- Inside a concrete class abstract method is not allowed.

Inside a concrete class we can create:-

(i.) variables (static & non static)
(ii.) blocks (static & non static)
(iii.) constructors (param & noparam.)
(iv.) Only concrete method. (static & non static)
(v.) inner classes (static & non static)

Like in interface, in concrete class the variables are not by default public, static and final, the default modifier to a concrete class member is "default". We must apply the additional modifier like public, static, final to concrete class variables if we needed.

Like in interface, in concrete class the methods are not public and abstract, we must declare the method as public explicitly if we needed.

R5 :- We can create reference variable and also object from a concrete class because a concrete class is an fully implemented class.

Example :-

Rectangle	r1	=	new Rectangle();
Lion	l1	=	new Lion ();
RB	rb	=	new RB ();
Student	s1	=	new Student();
Emp	e1	=	new Emp();

ref. var. — object / instances

R6 :- We can derive sub class from a concrete class by using "extends" keyword.

R7 :- We no need to overwrite / implement a method from the concrete class, because all methods in concrete class are already implemented. Inside sub class we just need to reuse the logic given in super class, concrete class.

```
class Person {
    void eat() {
        Sopln("Veg");
    }
}
```

```
class Student extends Person {
    here we are using eat() method
    from person class.
}
```

```
class Party {
    ps v main(s[] args) {
        Student s1 = new Student();
        s1.eat();
    }
}
```

R8:- If you don't like the implementation given in super class, you are allowed to override this super class method in sub class with your required logic.

Like in interface, in concrete class concrete method is not public by default. Them while overriding the concrete method in sub class, we no need to use public. We must use AM in sub class to the overriding method based on the AM used in sub per class to this concrete method.

If it is declare as public, we must override this method in sub class by applying "public" keyword.

```
interface SIM {
    void call();
    void sms();
}
```

```
class BSNL implements SIM {
    public void call(){ }
    public void sms(){ }
}
```

```
class Person {
    void eat () { }
    public void sleep () { }
}
```

```
class student extends Person {
    void eat() { } ✓
    public void sleep () { } ✓
}
```

```
class emp
    extends person {
public void eat () { } ✓
void sleep () { } X
}
```

In sub class to the eat() method can add default/ protected/ public.
but to sleep() method we can add only public, because it is declared as public in parent class.

Final Class points :-

1.) What is a final class ?
2.) Give me real world exmple for final class ?
3.) When should we choose final class ?
4.) Creation syntax ?
5.) Developing, saving and execution final class ?
6.) Rules on final class ?
7.) Interview and OCJP question on final class.

1. What is a final class ?

A class that is declared with the modifier "final" is class final class. A final class is a concrete class, fully implemented class, which doesn't have sub-class.

For e.g.:-

```
Final class VolvoPlus {

}
```

```
class Benz
    extends VolvoPlus
{ can't inherit
}  from final class X
```

3. When should we choose final class ?

⇒ If we want to stop inheritance / creating further multi level sub classes,

We must choose final class or we must declare concrete class as final.

We don't want further sub-classes from Volvo type buses from VolvoPlus, then we declared VolvoPlus class as final.

Then when we try to create sub class from VolvoPlus, compiler thrown error : can't inherit from final class.

05/10/17

4. Creation syntax of final class.

⇒

```
AM ELM final class ClassName
 (R2)      (R1)     extends ClassName  <-- (R3)
                    implements PI1, PI2 {

  ----variables
  ----blocks
(R4) ----constructors
  ----methods (abstract method not allowed)
  ----inner classes
}
                (R5) - instance creation

                (R6) - sub class
```

S1 S2 S3

6. Rules on final class.

⇒ ~~Rules~~ The rules we learn on concrete class are all applied on final class, except two additional rules:-

(i) We can not declare final class as abstract, compiler will throw error: illegal combination of modifiers abstract, final.

(ii) We can not create sub class from final class, compiler will throw error: can't inherit from final class.

R1:- We must create final class by using the keywords "final" and "class".

R2:- We can apply only ~~3~~ 2 modifiers in combination with final keyword, they are:-

(i) public

(ii) strictfp

abstract final class Ex { } X

final class Ex { } ✓

default → public final class Ex { } ✓

public strictfp final class Ex { } ✓

R3:- We can derive a final class from other concrete class or abstract class or interface.

We must use extends keyword for class and implements keyword for interface.

If there are any abstract methods exist in abstract class or interface, we must implement all those methods (abstract methods) in this final class, else compiler will throw error.

We must implement all those abstract methods in final class, bcoz we can not declare final class as abstract.

R4:- Inside final class we can not create abstract method. Except abstract method we can create all members which are allowed inside concrete class including main method. refer above diagram for the allowed members list.

R5:- We can instantiate final class, it means we can create object from final class.

```
final class Example {
    void m1() {
        Sopln("Hi");
    }
}
```

```
class Test {
    p s v main (String [] args) {
        Example e1 = new Example();
        e1.m1();
    }
}
```

Output :- Hi

R6:- We can not create sub class from final class.

```
final class Example {

}
```

```
class Sa extends Example {
                ^
}       X CE
```

Conclusion from all above points

To create an object's main type and sub types we must use interface, abstract class, concrete class and final class.

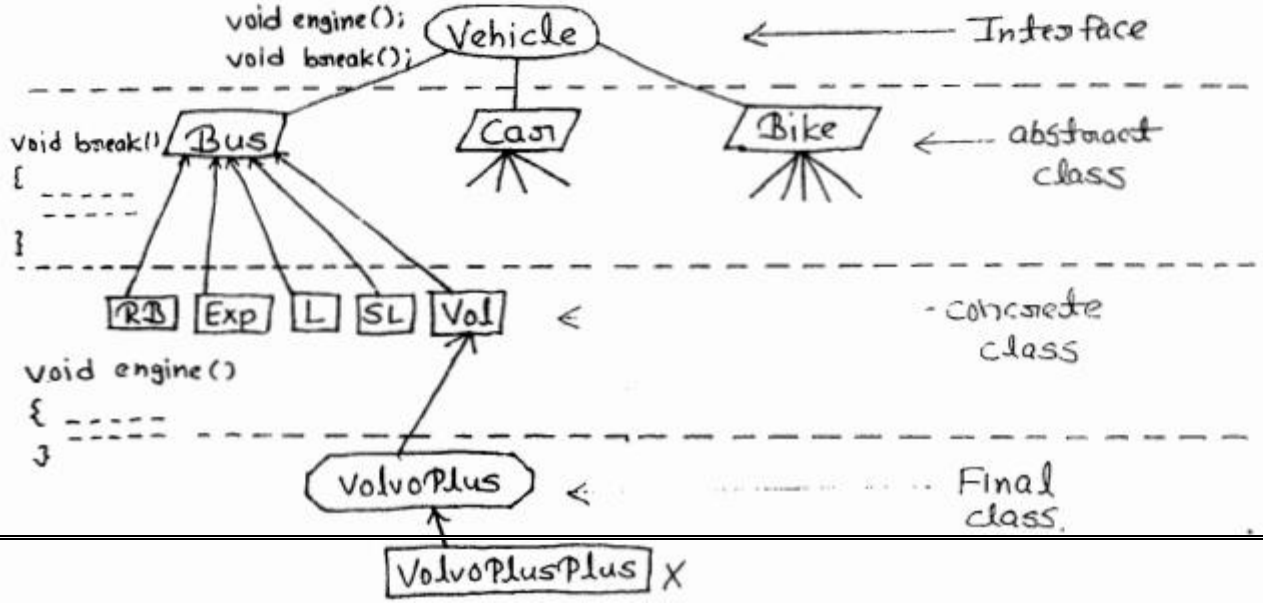
1. Interface :- for creating an object's main type by declaring all operations.

2. abstract class :- for creating an object's sub types by implementing some of the operations common to next sub types.

3. concrete class :- for creating an object's sub types by implementing all operation of this object common to next sub types.
It is also used for developing mathematical operations to be performed by computer.

4. final class :- for creating an object's final sub types by implementing all operations of this object.
final class can stop multilevel inheritance

Que :- With the knowledge you gain in the above pages, design and develope Vehicle object at multiple levels and develope its user object ~~in loose~~ Driver in loosely coupled run-time polymorphic approach.



Vehicle.java

```
interface Vehicle {
    void engine();
    void breaks();
    void seat();
}
```

Bus.java

```
abstract class Bus implements Vehicle {
    public void breaks() {
        System.o.pln("Bus has 2 breaks");
    }
}
```

Volvo.java

```
class Volvo extends Bus {
    public void engine() {
        Sopln("Volvo runs at 110KMPH");
    }
    public void seat() {
        Sopln("Volvo has 30 seats");
    }
}
```

VolvoPlus.java

```
final class VolvoPlus
        extends Volvo {
    public void seat() {
        Sopln("Volvo Plus has
                40 seats");
    }
}
```

Driver.java

```
class Driver {
    p s v main (String [] args) {
        Vehicle V;
        V = new Volvo();
        V.engine();  ---------->  executed from Volvo.
        V.breaks();  ---------->  executed from ~~vehicle~~Bus.
        V.seat();    ---------->  executed from Volvo.

        V = new VolvoPlus();
        V.engine();  ---------->  executed from Volvo (Reusing)
        V.breaks();  ---------->  executed from Bus.
        V.seat();    ---------->  executed from VolvoPlus.
    }
}
```

{ Thats all about Types of classes
interface, abstract class, concrete class &
final class }

Refer next page for
Summary table on
Types of classes and
all its points

Rules / Tcs	purpose	Allowed modifiers
1. interface	For creating main type of an object by declaring all operations of this object and forcing sub-types programmers to implement these operations as their requirement. For supporting multiple type inheritance	3 Modifiers allowed:- <default> public abstract strictfp
2. abstract class	In general we will use abstract class for creating sub type objects for a given object by implementing some of the operations of this object given in interface. We don't use abstract class for creating main type, because it does not support multiple inheritance.	2 modifiers:- <default> public abstract strictfp
3. Concrete class	Concrete class is also used for creating sub type objects by providing full implementation to all operations declared in an object's interface or in an object's abstract class. It is also used for developing mathematical operation to be performed by computer.	4 modifiers:- <default> public final abstract strictfp
4. Final class.	final class is also used for creating sub type objects by providing full implementation to all operations declared in an object's interface or in an object's abstract class or in an object's concrete class, but for not allowing further sub class from it in multilevel way. It is also used for developing mathematical operations to be performed by computer.	2 modifiers:- <default> public final strictfp

Allowed members	Instantiation allowed ?	super interf., class sub interf., class allowed ?
6 types :- 1. p s f var. 2. p a methods 3. p s innerclass. 4. p default methods. (1.8) 5. p static methods. (1.8) 6. private method (1.9) (default static)	Not allowed.	super interface :- Yes super class :- No sub interface :- Yes sub class :- Yes
11 types of members :- Static variable — Non static variable static block — Non static block static Method — Non static method Main method — constructor static inner class — Non-SIC abstract method.	Not allowed	super interface :- Yes super class :- Yes sub interface :- No sub class :- Yes
10 types of members :- SV — NSV SB — NSB SM — NSM MM — constructor SIC — Non-SIC	Yes Allowed.	super interface :- Yes super class :- Yes sub interface :- No sub class :- Yes
10 types of members :- SV — NSV SB — NSB SM — NSM MM — constructor SIC — Non-SIC	Yes Allowed.	super interface :- Yes. super class :- Yes. sub interface :- No. sub class :- No.

So, far we have learn below terminology and implementation code :-

1. Object class instance.
2. Class & types of classes.

1. Super Class / super interface.
2. sub class / sub interface (implementation class)
3. implements keyword.
4. extends keyword

5. implementing method.
6. overriding method.
7. overloading method.
8. hiding method.

9. implementing object functionality by achieving security to object data in one class

10. Accessing object functionality
sub class refvar = sub class object.
super class refvar = sub class object.
Tight coupling and loose coupling

OOP Principles

- Abstraction
- Inheritance (IS-A relation)
- Encapsulation
- Cohesion
- Association relation (HAS-A relation)
  - Composition
  - Aggregation
- Coupling to achieve 100% loose coupling & Runtime polymorphic App. development, use reflection API

LC-RP Architecture

- to represent real world object in Programming world
- by achieve security &
- achieve Dynamic Binding

Arrays in Java

19/10/2017

HAPPY DIWALI

1. What is an Array ?
2. Why array ?
3. How to create array ?
   - (i) new [ ]
   - (ii) { }
   - (iii) new [ ] { }

   - declaration
   - object creation
   - initialization
4. Problems of premitive variables and advantage of array ?
5. Limitation of Array or problems of array, solution.
6. Sample programs.
7. Compile time errors and run time errors.
   - assignment wise
   - size wise
   - Comparision with "C" language.

   - creation wise → NASE
   - reading wise → AIOOBE
   - storing wise → ASE
8. Types of Array :- (i) Single Dimensional Array.
   (ii) Multi-Dimensional Array.
   - Jogged Array.
9. Algorithms :-
   - Swapping Algorithms (No special name for this) algorithm
   - Sorting Algorithms (IS, SS, BS, HS, QS, MS, ...)
   - Searching Algorithms (Linear search, Binary Search)
10. Interview and OCJP Questions
11. Sample programs
12. Project developement using Arrays and database
    - OOP basics, operators control flow statements, Arrays and JDBC will be included.

16/10/17 Arrays in Java

Q. What is the different in array creation in C Language and in Java language?

⇒

Array Creation	In C	In Java
double[5] da;	✓	×
double [ ] da;	×	✓
double [] da = new double [4];	×	✓

* In C language, Array is premitive type whereas in Java Array is refferenced type.
* In C language, we create Array with just variable declaration, we don't use "new" keyword, whereas in Java we declare Array variable and Array object seperately, we will use "new" keyword in array object creation.
* In C language, we will specify Array size in variable declaration side, whereas in Java we must specify Array size object creation side.

double[5] da = new double [4]; ×

Q. How much memory occupied by a string?

⇒ Number of characters * 2.

For e.g. "hari" occupies 8 bytes.

Q. What are the different ways we have in java program to store value?

⇒ We have 3 ways :-

(i) Using premitive type variable.

(ii) Array object.

(iii) Class object.

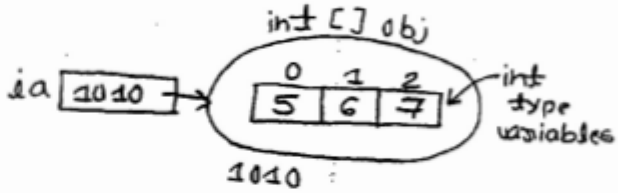
For storing one value or one object reference, we must use variable.

For storing multiple values or multiple objects of same type we must use Array object. Here we must create Array object using a datatype based on the type of value you want to store.

For e.g. for storing int type values we must use int[] as below:-

```
int[] ia = {5,6,7};
```

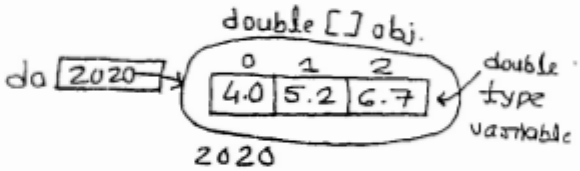
Array var. Array Obj



For storing double type values we must use double[] as below:-

```
double[] da = {4.0, 5.2, 6.7}
```

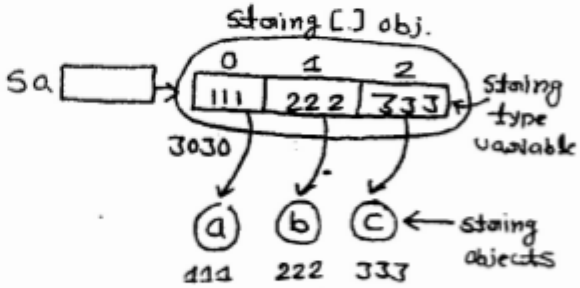
Arr. var. Arr. Object



For store multiple string objects we must create string [] as below:-

```
String [] sa = {"a", "b", "c"};
```

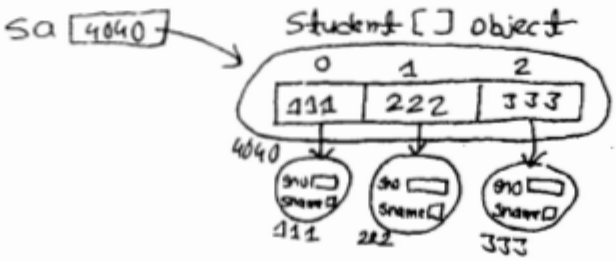
Arr. var. Arr. object



For storing multiple student objects, we must create student[] object.

```
Student[] sa = {new Student(), new student(), new student()};
```

```
class student{
  int sno;
  String sname;
}
```



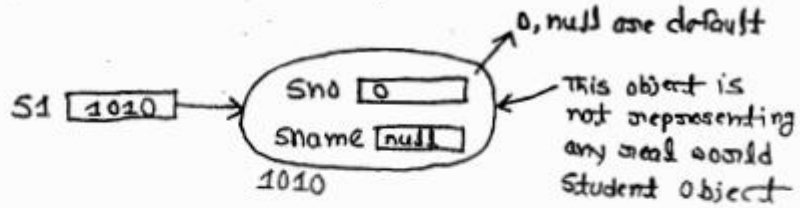
If you want to store multiple emp object, we must create emp[] object :—

For storing 1 student multiple values we must create student class object.

```
Student S1 = new student();
```

Here student class obj. is created with non-static variable memory declared in the class student.

```
class student {
    int sno;
    String sname;
}
```



To store multiple student objects, we must create student class array objects with the size equal to no. of objects you want to store. **Array Creation Syntax 1 :—**

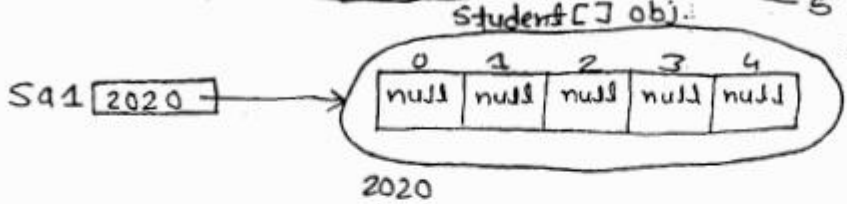
Note:- If we create array object using any class, this class objects are not created rather Array obj. is created with this class type ref. variable's. No. of variables = the length we specified.

e.g. :—

```
Student[] Sa1 = new Student[5];
```

1 student[] obj. is created

5 student type ref. variable.



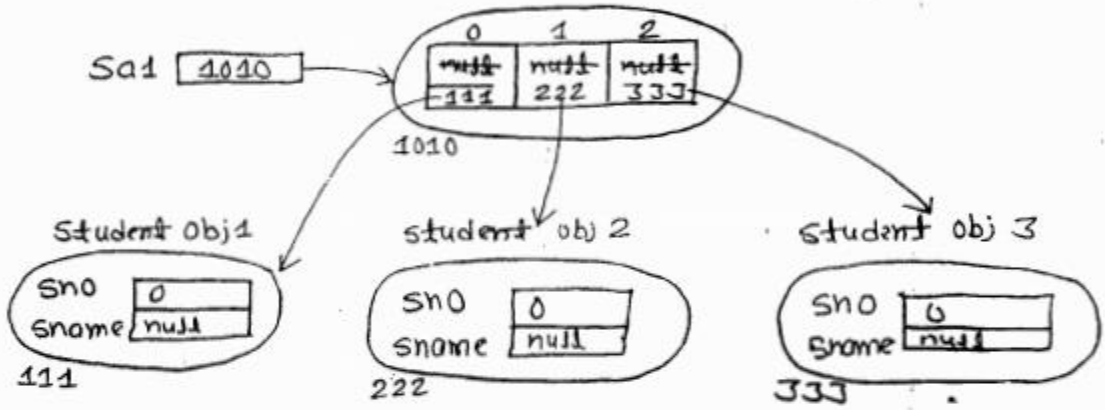
Here student class obj. not created, it means student class non static variables memory is not created.

In above syntax of Array creation only we can specify size, it creates only variables, it will not allows us to store values or objects in the Array creation syntax itself.

If you want to store values on object, we must perform initialization operation in the next line seperately, as shown below:—

```
Student[] sa1 = new Student[3];
```
← student array object creation

```
Sa1[0] = new student();
Sa1[1] = new student();
Sa1[2] = new student();
```
← student class objects creation



Array Creation Syntax 2 (with size & values):—

If you create array obj with new keyword as shown above, only array object is created with location. hence we must declare array in one line and we must initilized

Using Syntax 2 '{}' we can create array and initiazed array in single line.

Syntax:- `DT[] ref. var. = {e1, e2, e3, -----};`

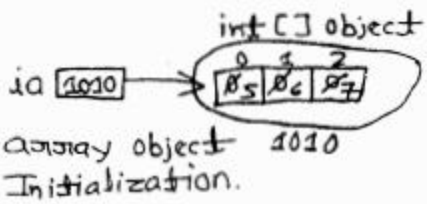
PDT RDT

Example:- below statement will create array object with Student Obj. in single line.

Creating int[] with values

```
int[] ia = {5, 6, 7};
```

array ref. var — array obj. creation



array object Initialization.

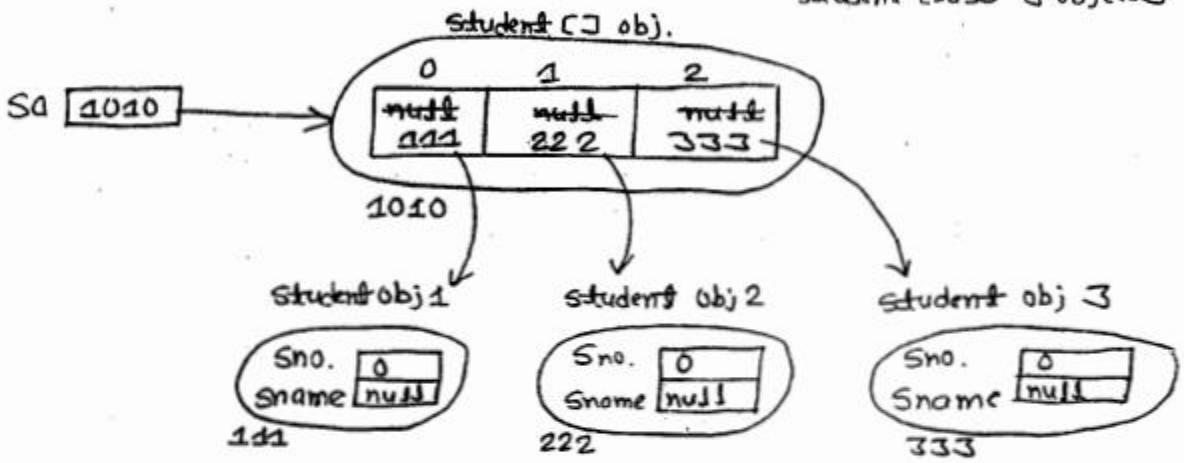
E.g. 2 Creating and initializing array object with students objects.

```
Student[] sa = { new Student(), new Student(), new Student() };
```

student[] type ref. var

student[] object creation with size.

student[] object initialization with student class 3 objects



Q. When should we choose 1 syntax and 2nd Syntax in Array object creation ?

⇒ When we know only type of values and no. of values should store then we must choose first syntax with new keyword array creation.

We know type of values, no. of values and also the values then we must choose 2nd syntax using '{ }'.

Consider :2 senerios :-

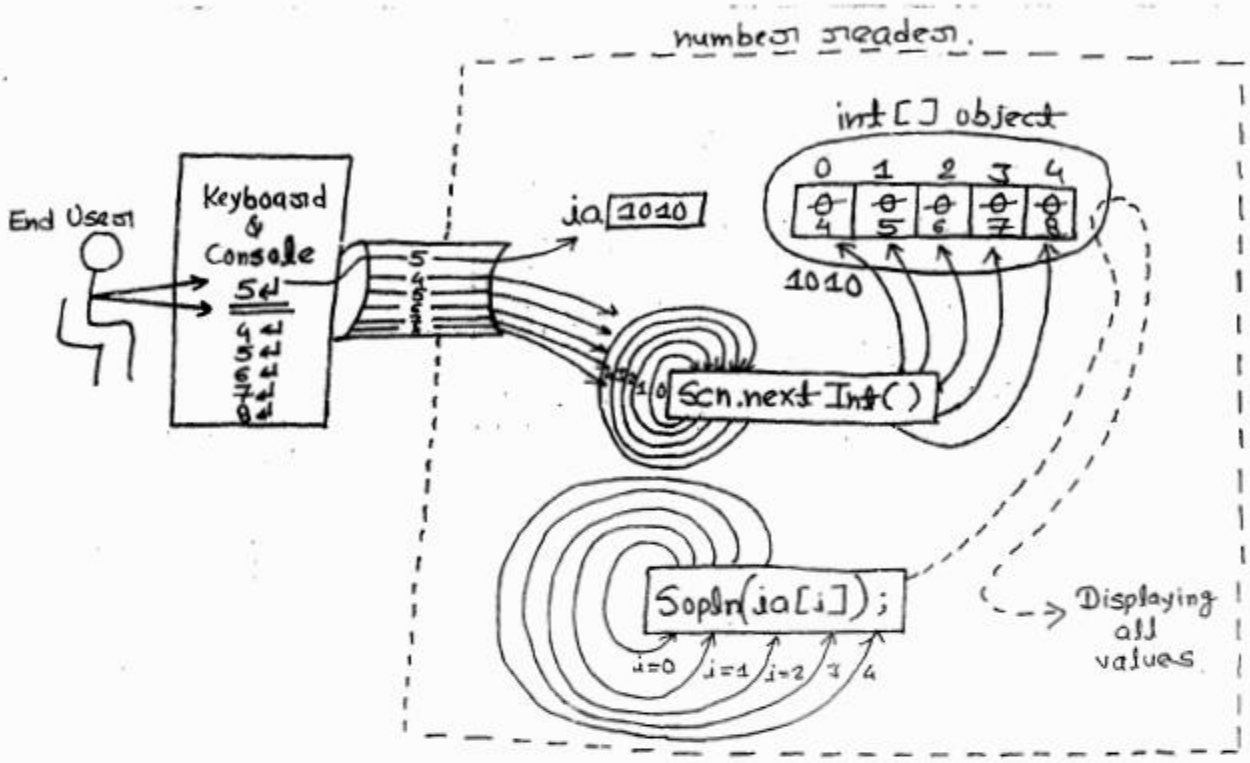
1. We want to read 10 int. from end user from keyboard and store in Java program.
2. We must take 2 num. into a method as argument, add them and subtract them, return both operation result.

To develope 1st senerio program you must create int[] object with 1st syntax.

To develop 2nd senerio, we must create int[] with 2nd Syntax.

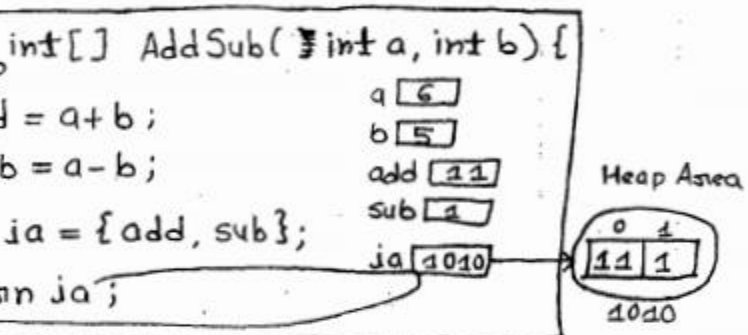
1.

```
import java.util.scanner;
Class NumbersReader {
    public static void main(string[] args) {
        Scanner scn = new Scanner (System.in);
        System.out.print(
            "How Many Numbers do you want to enter ?:");
        int noOfValues = scn.nextInt();
        int[] ia = new int[noOfValues];
        System.out.println(
            "array object is created with length" + noOfValues);
        System.out.println(
            "\n Reading No. from end-user from keyboard");
        for(int i=0; i<noOfValues; i++){
            Sopln("Enter number" + (i+1) + ":");
            ia[i] = scn.nextInt();
            Sopln("  " + ia[i] + " is stored in array \n");
        }
        System.out.println("\n Reading and displaying
                                values from array");
        for(int i=0; i<ia.length; i++){
            Sopln(" number at index" + i + ":" + ia[i]);
        }
    }
}
```


numbers reader.
int[] object
0 1 2 3 4
0 0 0 0 0
4 5 6 7 8
1010
End User
Keyboard & Console
5↵
4↵
5↵
6↵
7↵
8↵
ia 1010
Scn.nextInt()
Sopln(ia[i]);
i=0
i=1
i=2
3
4
Displaying all values

2. We must take 2 numbers into a method as argument, add and subtract them, return both operation result.

```
class AddSub {
    static int[] AddSub(int a, int b) {
        int add = a+b;
        int sub = a-b;
        int[] ia = {add, sub};
        return ia;
    }
    public static void main(String [] args) {
        int[] res = AddSub(6,5);
        Sopln(res[0]); sopln(res[1]);
    }
}
```



Summary :-

In java we can create array object in 3 ways :-

1. Using "new" keyword and square bracket [].
2. Using {} bracket.
3. Using "new" keyword, [] and {}.

1 If we know only Type of values and no. of values to stored in array, we must choose 1st syntax. Refer 1st program (Numbers Reader) for understand.

2. If we know Type of values, no. of values and also values then we should use 2nd syntax i.e. {} bracket. 2nd syntax has a limitation i.e. we can use {} only with variable declaration.

It means `int[] ia = {5,6,7}` ✓

We can not use this syntax as variable assignment it means array variable declared in one place and performing assignment in another place with {}.

`int[] ia;` Array variable declared.

`ia = {5,6,7};` X

Variable ia assignment, here compiler will throw error because array object type is not mentioned before

If we use {} in variable declaration, compiler software will take the type to the array from the variable declaration. As shown below :-

Below 2 statements are equal :-

```
int[] ia2 = {11, 12, 13, 14, 15};
int[] ia3 = new int[] {11, 12, 13, 14, 15};
```

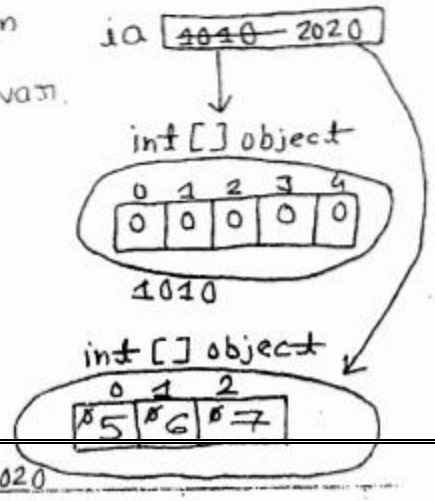
Here we created, 3rd Syntax Array object.

If we use {} in variable assignment compiler software can not infer or substitute type for array creation.

Hence if you want to declare array variable in one place and if you want to assign this array variable in next line by creating new array object with values we must choose 3rd syntax. that is "new [] {}."

For example :-

```
int[] ia;   // Array variable declaration
// Array obj. creation & assigning to var.
ia = new int[5];
// ia = {5,6,7}; X CE
   No type information, ∴ error.
ia = new int[] {5,6,7};
```



Define main method class to call and execute below method :-

```
Class Example {
    void m1 (int[] ia) {
        for (int i=0; i < ia.length; i++) {
            Sopln(" Element "+(i+1)+ "in: "+ ia[i]);
        }
    }
}
```

Explicit values

default values

```
class test {
    p s v main(string[] args) {
        Example e1 = new Example();
        e1.m1(new int[5]);
        // e1.m1({5,6,7});
        e1.m1(new int[]{5,6,7});
    }
}
```

1st way method call output :-

element 1 is : 0
2 is : 0
3 is : 0
4 is : 0
5 is : 0

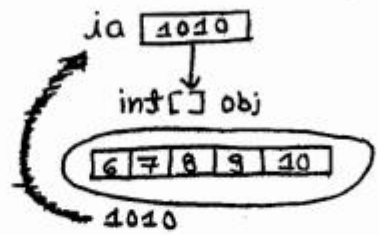
3rd way method call output :-

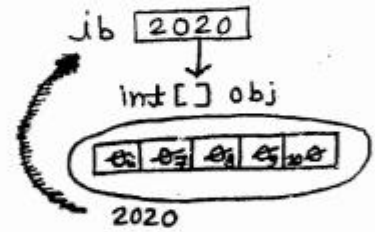
element 1 is : 5
element 2 is : 6
element 3 is : 7

Q. Develop a program to create int[] object with 5 values copy elements of this array into 2nd array. [Array Copy Program]. Display Both array values.

⇒

```
class ArrayCopy {
    p s v main(String [] args) {
    int [] ia = {6,7,8,9,10};
    int [] ib = new int[ia.length];
    for(int i=0; i< ia.length; i++)
    {
        ib[i] = ia[i];
        Sopln(" 1st array, index "+i+": "+ ia[i]);
        Sopln(" 2nd array, index"+i+": "+ ib[i]+"\n");
    }
  }
}
```





Output :-

1st array, index 0 : 6
2nd array, index 0 : 6

1st array, index 1 : 7
2nd array, index 1 : 7

1st array, index 2 : 8
2nd array, index 2 : 8

1st array, index 3 : 9
2nd array, index 3 : 9

1st array, index 4 : 10
2nd array, index 4 : 10

Above program is Ameerpet standard program,
The problem in this program is no code reusablility, it can copy only one array, and can not be accessed from other classes.
To make this code reusable, we have to develop seperate class with ~~this~~ ~~an~~ required no. of methods with this logic. As shown below :-

```
class ArrayCopyTest {
  p s v main(String [] args) {
   ArrayOperations ao = new ArrayOperations();
    int[] ia = {5, 6, 7, 8, 9};
    int [] ia2 = ao. CopyArray(ia1);
   Sop("ia1 elements");
     ao. displayarray(ia1);
   Sop("\n ia2 elements");
     ao. display array(ia2);

    - - - -
    - - - -
   //repeat above statements
   //with another new array
    - - - - -
    - - - - -
  }
}
```

```
class Arrayoperation {
  int [] CopyArray (int [] ia) {
    int[] ia2 = new int [ia.length]
    for(int i=0; i< ia.length; i++){
      ia2[i] = ia[i];
    }
  return ia2;
  }

void displayArray (int [] ia){
  for(int i=0; i < ia.length; i++){
    Sop(" "+ia[i]);
  }
}
```

21/10/17

Program 5:- Develop a program to store multiple employee Objects using an array.

In this project we must develope 2 classes :-

(i.) Class Employee - for defining employee type.

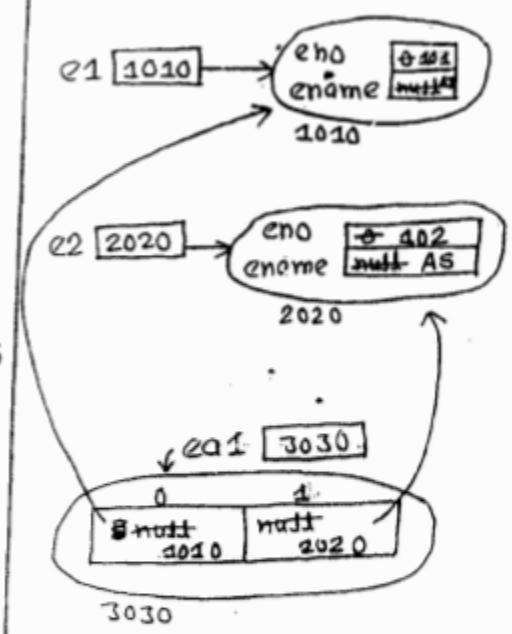
(ii.) Class Test - for creating Employee array object with employee objects.

// Employee.java

```
class Employee {
    int eno;
    String ename;
}
```

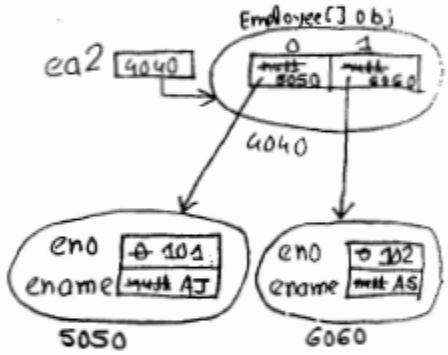
// Company.java

```
class Company {
  p s v main(String [] args) {
  //Approach 1 : Emp obj. than Array object.
  Employee e1 = new Employee();
   e1.eno = "101";
   e2.ename = "AJ";
  Employee e2 = new Employee();
   e2.eno = 102
   e2.ename = "AS";
   Sopln("Two employee objects are ready");
  Employee[] ea1 = {e1,e2};
   Sopln("The ~~empl~~ two Employee objects
          are stored in array");
   Sopln("Displaying e1 objects values");
   Sopln("By using e1 variable & ea1[0]");
   Sopln(" e1.eno:" + e1.eno);
   Sopln(" e1.ename:" + e1.ename);
   Sopln(" ea1[0].eno:" + ea1[0].eno);
   Sopln(" ea1[0].ename:" + ea1[0].ename);
```



The both objects 1010 & 2020 have 2 references, e1, ea1[0] and e2, ea1[0] respectively.

```
Sopln();
Sopln("Displaying e2 objects value");
Sopln("By using e2 variable");
Sopln("e2.eno : " + e2.eno);
Sopln("e2.ename: " + e2.ename);
Sopln("By using ea1[1] variable");
Sopln("ea1[1].eno" + ea1[1].eno);
Sopln("ea1[1].ename" + ea1[1].ename);

// Approach 2: Emp. obj. are created in future
//             at program execution time.

Employee[] ea2 = new Employee[2];
ea2[0] = new Employee();
ea2[1] = new Employee();

ea2[0].eno = 101;
ea2[0].ename = "AJ";

ea2[1].eno = 102;
ea2[1].ename = "AS";

Sopln("Displaying 1st Employee value");
Sopln("ea2[0].eno : " + ea2[0].eno);
Sopln("ea2[0].ename: " + ea2[0].ename);

Sopln("Displaying 2nd Employee value");
Sopln("ea2[1].eno: " + ea2[1].eno);
Sopln("ea2[1].ename: " + ea2[1].ename);
```



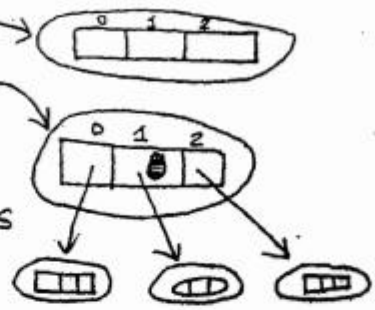
Types of Arrays :-

23/10/17

Every language will supports 2 types of arrays :-

(i) Single Dimensional array.

(ii) Multi Dimensional array.



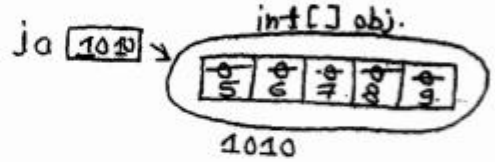
- An array that can store multiple values or multiple objects of a class is called Single Dimensional array.

For example:-

```
int[] ia = new int[5];
```

```
ia[0]=5;
ia[1]=6;
ia[2]=7;
ia[3]=8;
ia[4]=9;
```

```
//Printing values.
Sopln(ia[0]);
Sopln(ia[1]);
Sopln(ia[2]);
Sopln(ia[3]);
Sopln(ia[4]);
```



Output :-

5
6
7
8
9

- An array i.e. meant for storing other array is called multidimensional arrays. Here the Dimensions means level in the array object memory. If only one level is there, it is single Dimensional array. If two levels are there, it is two dimensional array, If three levels are there, it is three dimensional array etc.

The Dimension or level is represented by [].

SDA contains [].

TDA contains [][].

Three DA contains [][][].

Syntax for Two Dimesional array :-

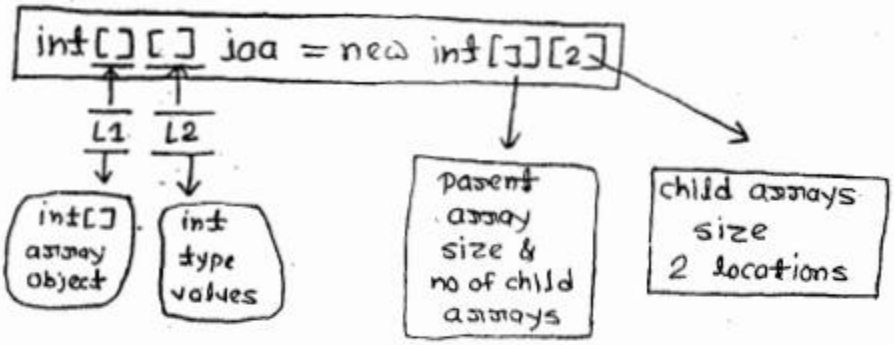
```
DT[][] var.name = new DT[][]
```

size

Child array size.

parent array size + no. of child array.

For example:-



~~In C language~~,

* Above statement will create one referenced variable with name iaa of type int[][] and then it creates 3 locations array object of type int[][] to store 3 int[] objects.
* And in each location child Array is created with two location. Each location of type int. to store 2 int values.
* Ultimately in the 1st level parent array, we store int[] objects. In 2nd level array objects we store int values.

Syntax :- (for store and read values from MDA)

In SDA,

ia[0]
ia[1]

ia[index]

In MDA,

1st location { iaa[0][0];
iaa[0][1];

2nd location { iaa[1][0];
iaa[1][1];

3rd location { iaa[2][0];
iaa[2][1];

iaa [parent array location index] [child array location index]

Develop a program to create single and multi dimensional array for storing int values, reading and updating those values.

```
class TestSDAMDA {
    p.s.v. main(string[] args) {
    // SDA Creation, storing, reading, modifying values.
    int[] ia = new int[3];

    ia[0] = 5;
    ia[1] = 6;
    ia[2] = 7;

    Sopln(ia[0]);
    Sopln(ia[1]);
    Sopln(ia[2]);

    ia[0] = 8;
    ia[1] = 9;
    ia[2] = ia[1]+3;

    for(int i=0; i< ia.length; i++)
    {
        Sopln(ia[i]);
    }
```

ia 1010 → int[] obj. (0, 1, 2) 1010

insert → storing values

reading & displaying values → static logic

select →

updating or modifying → update

Dynamic logic for reading & displaying

```
    // MDA Creation, storing, reading, modifying values.
    int[][] mia = new int[3][2];

    mia[0][0] = 5;
    mia[0][1] = 6;
    mia[1][0] = 7;
    mia[1][1] = 8;
    mia[2][0] = 9;
    mia[2][1] = 10;

    Sopln(mia[0][0]);
    Sopln(mia[0][1]);
    Sopln(mia[1][0]);
    Sopln(mia[1][1]);
    Sopln(mia[2][0]);
    Sopln(mia[2][1]);

    mia[0][0] = 3;
    mia[0][1] = 4;
    mia[2][0] = mia[0][0] + mia[1][1];

    for(int i=0; i< mia.length; i++){
        for(int j=0; j< mia[i].length; j++){
            Sopln(mia[i][j]);
        }
    }
}
}
```

mia 1010 → int[] object (0, 1, 2) 1010 → int[] obj. (0, 1) 111; int[] obj. (0, 1) 222; int[] obj. (0, 1) 333

storing values in child array

reading & displaying child array values → static logic

updating array values

Dynamic logic for reading & displaying

Develope a program to create multi-dimensional array with 2nd syntax for initializing MDA with values.

⇒

```
DT[][] var.name = { {e1,e2,e3,e4}, {e1,e2,e3,e4}, {e1,e2,e3,e4} }
```

Child Array

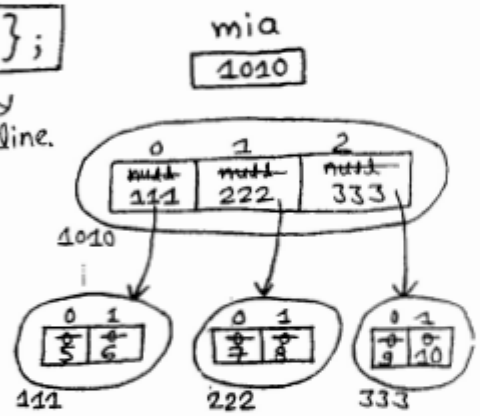
values of CA1 | values of CA2 | values of CA3

Parent array

Example :-

```
int[][] mia = { {5,6}, {7,8}, {9,10} };
```

Array object variable declaration, Array object creation & initialization in single line.

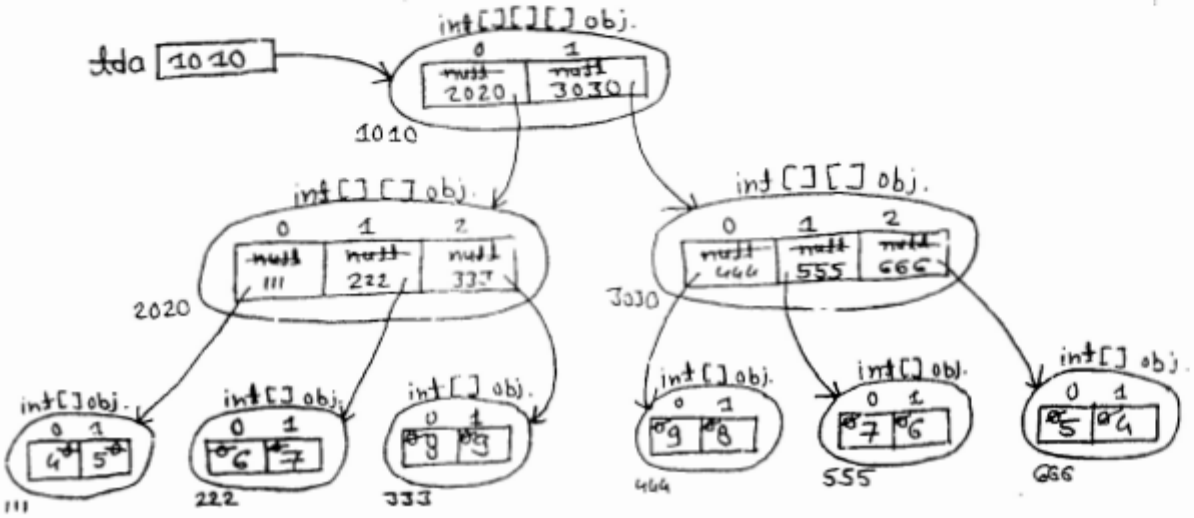


// Reading & displaying

```
for (int i=0; i< mia.length; i++){
    for(int j=0; j < mia[i].length; j++){
        Sop ln (mia[i][j]);
    }
}
```

```
// or
sop.ln(" mia["+i+"][j+"]value:"+mia[i][j]);
```

Q. Create 3DA and initialize and read values from both syntaxes.



Syntax #1 :-

```
int[][][] tda = new int[2][3][2];
```

```
tda[0][0][0] = 4;
tda[0][0][1] = 5;
tda[0][1][0] = 6;
tda[0][1][1] = 7;
tda[0][2][0] = 8;
tda[0][2][1] = 9;

tda[1][0][0] = 9;
tda[1][0][1] = 8;
tda[1][1][0] = 7;
tda[1][1][1] = 6;
tda[1][2][0] = 5;
tda[1][2][1] = 4;
```

```
for(int i=0; i< tda.length; i++){
    for(int j=0; j < tda[i].length; j++){
        for(int k=0; k < tda[i][j].length; k++){
            Sopln(tda[i][j][k]);
                //or
            Sopln("tda["+i+"][]["+j+"][]["+k+"]
                value :" + tda[i][j][k]);
        }
    }
}
```

Syntax #2 :-

```
int[][][] tda = { { {4,5}, {6,7}, {8,9} }, { {9,8}, {7,6}, {5,4} } };
```

child array

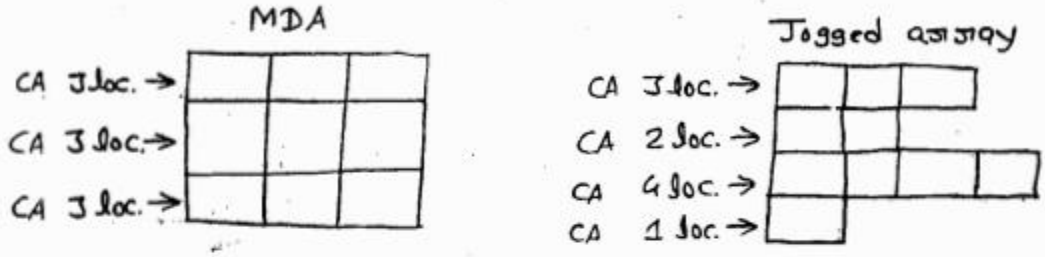
child array

parent array

Grand Parent array

* Jogged Array :-

A Multi-Dimensional array with different sizes child array is called Jogged Array.



25/10/2017

- Syntax for jogged array :-

#Syntax 01 :- (New keyword wise jogged array creation)

To create jogged array with this new keyword syntax, don't mention child array size leave it empty. Then in the next lines initialize parent array locations with child array instances with different sizes.

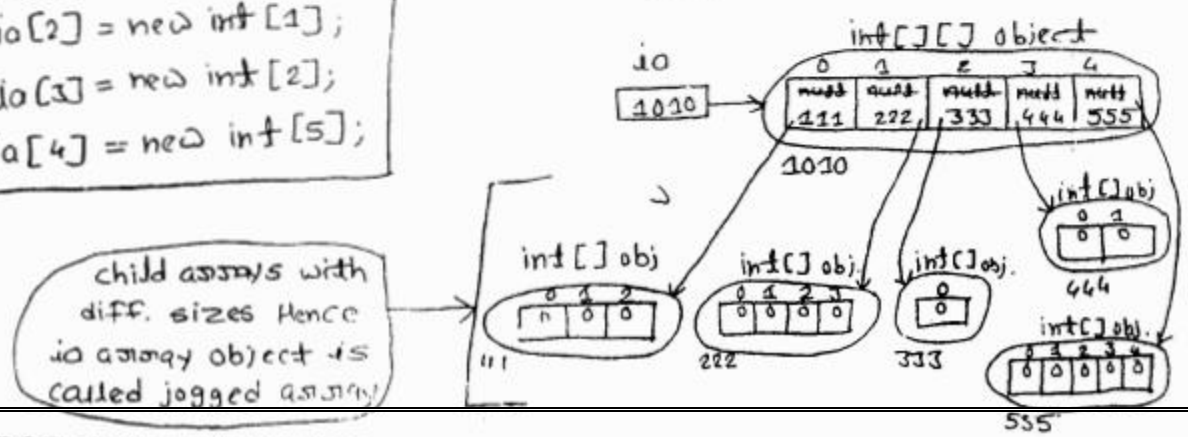
For e.g.

```
int[][] ia = new int[5][];
```

parent array size

child array size is not mentioned for creating child arrays with diff. sizes as shown below

```
ia[0] = new int[3];
ia[1] = new int[4];
ia[2] = new int[1];
ia[3] = new int[2];
ia[4] = new int[5];
```



Syntax #02 :- ({} wise jogged array creation)

Q. Create jogged array for the below creation :-

ja[0] →	5	6	7		
ja[1] →	15	16	17	18	
ja[2] →	25				
ja[3] →	35	36			
ja[4] →	45	46	47	48	49

⇒

```
int[][] ja = {{5,6,7}, {15,16,17,18}, {25}, {35,36}, {45,46,47,48,49}};
```

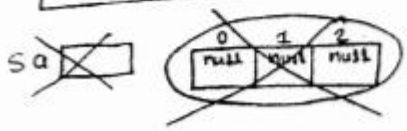
// Displaying jogged array values

```
for(int i=0; j<ja.length; i++){
    for(int j=0; j< ja[i].length; j++){
        Sopln(ja[i][j]);
    }
}
```

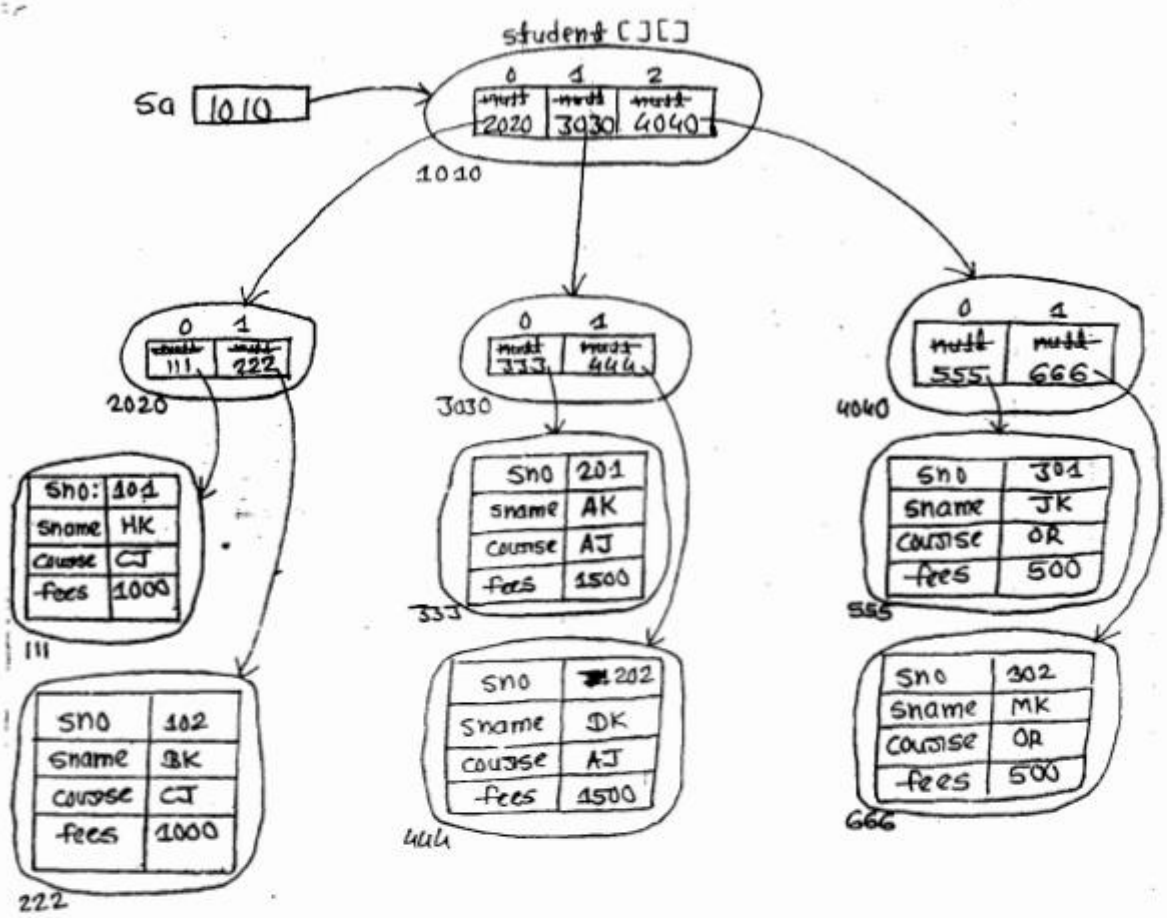
Q. Develop a program to create MDA to store student objects on course wise.

⇒ ~~student~~

```
class student {
    int sno;
    string sname;
    string course;
    double fees;
}
```




```
class College {
    p s v main(string[] args){
        student[][] sa = new student[3][2];
        sa[0][0] = new student();
        sa[0][0].sno = 101;
        sa[0][0].sname = "HK";
        sa[0][0].course = "CJ";
        sa[0][0].fees = 1000;

        sa[0][1] = new student();
        sa[0][1].sno = 102;
        sa[0][1].sname = "BK";
        sa[0][1].course = "CJ";
        sa[0][1].fees = 1000;
```




```
Sa[1][0] = new student();
Sa[1][0].sno = 201;
Sa[1][0].sname = "AK";
Sa[1][0].course = "AJ";
Sa[1][0].fees = 1500;

Sa[1][1] = new student();
Sa[1][1].sno = 202;
Sa[1][1].sname = "DK";
Sa[1][1].course = "AJ";
Sa[1][1].fees = 1500;
```

```
Sa[2][0] = new Student();
Sa[2][0].sno = 301;
Sa[2][0].sname = "JK";
Sa[2][0].course = "OR";
Sa[2][0].fees = 500;

Sa[2][1] = new student();
Sa[2][1].sno = 302;
Sa[2][1].sname = "MK";
Sa[2][1].course = "OR";
Sa[2][1].fees = 500;
```

Q. When to choose SDA student[] and MDA student[][] for storing student objects?

⇒ * For storing multiple students objects as one group and then for sending all these objects from one method to another method as method argument or return type we must choose single dimensional array.

* If you want to store the same students with coursewise seperation then we must choose multi-dimensional array.

* If we want to store multiple values of different type using array we must choose SDA with class object.

* Java to Database Mapping :-

```
class student {
  int sno;
  String sname;
  String course;
  double fee;
}
```

Java	Database
class ⟷	table
fields ⟷	columns
instances ⟷	rows
array of instances ⟷	multiple rows

Database

Table

sno	sname	course	fee
101	HK	CJ	1000
102	SK	CJ	1000
103	PK	AC	2000
104	MB	AJ	2000
105	PB	Oracle	1500
106	SD	Oracle	1500

Step 1:- Create no. of classes as many tables as we have in database with the same table name.

Step 2:- Inside the class create fields as many columns as we have in table. Field type and name should be same as table column type and name.

Step 3:- Create one instance seperately for each row means create no. of instances from this class as many rows as we have, copy each row values in its respective instance variables.

Step 4:- Store all instances inside an array or collection.

Step 5:- Finally, pass/return this array/collection of instances to another method. In that another method retrieve instances from array/collection, then use or display the values available in this instance.

26/10/17 Que:- Develop a project to store NIT Students info. in DB.

To develope this project, we must follow below steps:—

Step 1:- Create new schema (user) NSIS (Naveshit Student Information System) with the password NSIS in oracle database then create student table.

Step 2:- Then create a table student with the colums sno (PK), sname, course, fee, email, mobile.

Run below commands in SQL Plus window:-

(i) Login to system user.

(ii) RUN below queries:-

```
SQL> CREATE USER nsis
     IDENTIFIED BY nsis;

SQL> GRANT DBA to nsis;

SQL> CONNECT nsis/nsis

SQL> SHOW USER

SQL> CREATE TABLE student (
        sno NUMBER(4) PRIMARY KEY,
      sname VARCHAR2(20),
     course VARCHAR2(10),
        fee NUMBER(10,2),
      email VARCHAR2(20),
     mobile NUMBER(10)
     );
```

The required DB setup is completed.

Step 3:- Create a class with name Student with fields same as columns.

// Student.java

```
class Student {
    int sno;
    String sname;
    String course;
    double fee;
    String email;
    long mobile;
}
```

- This class is called user defined datatype as per core java.
- It is also called as DTO/bean object/Domain object/POJO/Entity object from projects point of view.
- This class is only meant for storing student object values, tranferring all values at a time from view to model & model to view in MVC architecture while performing CRUD operations.

Step 4:- Next we must develope DAO (Data Access Object) class to connect to DB and perform CRUD operations on student table with the given values.

In this class we must atleast 6 methods:-

(i.) For opening connection.
(ii.) For closing connection.
(iii.) For performing insert operation.
(iv.) For performing select operation.
(v.) For performing update operation.
(vi.) For performing delete operation.
etc...

```
//StudentDAO.java (DAO/model/persistance operation class)
~~class~~ ~~Student~~
import java.sql.*;
class StudentDAO {

    Connection con;

    void openConnection() throw SQLException {
        String url = "jdbc:oracle:thin:@localhost:1521:orcl";
        string usn = "nsis";
        string pwd = "nsis"
        con = DriverManager.getConnection(url, usn, pwd);
    }
```

storing the connection object in class level con variable to make this connection available to all methods.

```
    void closeConnection() throw SQLException {
        if (con != null) {
            con.close();
            con = null;
        }
    }

    void insert (Student S) throws SQLException

        String iQuery = "INSERT INTO student
                        (sno, name, course" +
                        " fee, email, mobile") + "values(?????)";

        PreparedStatement pstmt = Con.prepareStatement(iQuery);
```

```
pstmt.setInt(1, s.sno);
pstmt.setString(2, s.name);
pstmt.setString(3, s.course);
pstmt.setDouble(4, s.fee);
pstmt.setString(5, s.email);
pstmt.setLong(6, s.mobile);

pstmt.executeUpdate();

} // End of insert

Student[] select() throws SQLException {

String sQuery = "SELECT sno, sname, course, fee, email, mobile"
                + " FROM student";

PreparedStatement Selectpstmt =
                Con.prepareStatement(
                        sQuery,
                        ResultSet.TYPE_SCROLL_INSENSITIVE,
                        ResultSet.CONCUR_READ_ONLY);

ResultSet rs = Selectpstmt.executeQuery();

rs.last();                    <- Move cursor to last Row
int noofrows = rs.getRow();   <- Retrieving cursor row no.
rs.beforeFirst();             <- Move cursor to first
// Counting no. of rows.
```

```
Student[] Students = new Student[noofrows];
int index = 0;
```

/* Creating new array
with size equals to
number of rows */

```
While (rs.next()) {
    Student s = new Student();
        s.sno = rs.getInt(1);
        s.name = rs.getString(2);
        s.course = rs.getString(3);
        s.fee = rs.getDouble(4);
        s.email = rs.getString(5);
        s.mobile = rs.getLong(6);

        Student[index++] = s;
} // while close
```

Moving on each row.

new instance created.

Reading Values

Storing the instances in array object to return all row values to main method class.

```
return Students,

} // select method close.
```

```
Student select (int sno) throws SQLException {

    String swQuery = "select sno, sname, course, fee, email,
                      mobile" +
                      " FROM Student" +
                      " WHERE sno = ?";

    PreparedStatement pstmt = con.prepareStatement(swQuery);

    pstmt.setInt(1, sno);

    ResultSet rs = pstmt.executeQuery();

    if(rs.next()){ <------------------ // because the current select
                                          query returns only one
                                          row, hence we used if
                                          condition.
                                          If select query returns
                                          multiple rows, then we
                                          must use while condition.
      Student s1 = new Student();

      s1.sno     = rs.getInt(1)
      s1.sname   = rs.getString (2);
      s1.course  = rs.getString(3);
      s1.fee     = rs.getDouble (4);
      s1.email   = rs.getString (5);
      s1.mobile  = rs.getLong(6);

      return s1;
    }
    else {
      throw new SQLException("No student found with given sno:"+
                                                        sno);
    }
} //select(sno) method close.
```

31|10|17

```
Student[] select(String course) throws SQLException {
    String scQuery =
        "SELECT sno, sname, course, fee, email, mobile "+
        "FROM student "+
        "WHERE course = ? "+
        "ORDER BY sno ASC";

    PreparedStatement pstmt = con.prepareStatement(
                                scQuery,
                                ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_READ_ONLY
                              );
    pstmt.setString(1, course);
    ResultSet rs = pstmt.executeQuery();
    if(rs.next()){
        rs.last();
        int noOfRows = new Student[noOf rs.getRow();
        Student[] students = new students[noofRows];
        rs.beforeFirst();
        int index = 0;
        while (rs.next()){
            Student s = new Student()
            s.sno = rs.getInt(1);
            s.sname = rs.getString(2);
            s.course = rs.getString(3);
            s.fee = rs.getDouble(4);
            s.email = rs.getString(5);
            s.mobile = rs.getLong(6);
            students[index++] = s
        } // while close
        return students;
    }
    else{
        throw new SQLException("Given course "+course"+" is not exist");
    }
} // select(course) throws method close.
```

```
//MainOffice.java

import java.util.*;
import java.sql.*;

class MainOffice {
    public static void main(String[] args) throws SQLException {
        Scanner scn = new Scanner(System.in);
        StudentDao sdao = new StudentDAO();
        sdao.openConnection();

        System.out.println("Choose option:");
        System.out.println("1 Insert");
        System.out.println("2 Select");
        System.out.println("3 Update");
        System.out.println("4 Delete");
        System.out.println("5 Exit");

        System.out.print("\n Enter option number:");
        int option = scn.nextInt();
        scn.nextLine();
        switch(option){
            case 1: { //insert operation code
                student s1 = new Student();

                System.out.println("Enter student Values");
                System.out.print("sno :");
                s1.sno = scn.nextInt();
                scn.nextLine();

                System.out.print("sname : ");
                s1.sname = scn.nextLine();
```

101

```
System.out.print("Course");
s1.course = scn.nextLine();

System.out.print("fee :");
s1.fee = scn.nextDouble();
scn.nextLine();

System.out.print("email : ");
s1.email = scn.nextLine();

System.out.print("mobile : ");
s1.mobile = scn.nextLong();
scn.nextLine();

sdao.insert(s1);
System.out.println("\n Student values are stored
                    successfully");

break;
} //insert case close.

case 2 : { // select operation

Sopln("Choose one Option");
Sopln("1. All students");
Sopln("2. One student");
Sopln("3. Course wise students");

Sopln("\n Enter option ~~name~~ number");
option = scn.nextInt();
scn.nextLine();
```

```
switch (option) { //inner switch for select options.
  case 1: { //all student selection
    Sopln("\n Retriving all rows and displaying");
    Student[] students = sdao.select();
    for(int i=0; i<students.length; i++){
      Student s = students[i];
      Sopln("\nstudent +(i+1)+" values are");
      Sopln("sno :"+s.sno);
      Sopln("sname : "+s.sname);
      Sopln(" course:"+s.course);
      Sopln(" fee : "+s.fee);
      Sopln(" email:"+s.email);
      Sopln(" mobile:" + s.mobile);
    } // for close
  break;
} // all students case is close.

case 2 : { // one student selection
  Sopln("Enter sno.");
  int sno = scn.nextInt();
  scn.nextline();
  Sopln("Retrieving one student values and displaying");
  student s = sdao.select(sno);
  Sopln("\nstudent "+sno+" values are:");
  Sopln(" sno:"+s.no);
  Sopln(" sname:"+s.sname);
  Sopln(" course:"+s.course);
  Sopln(" fee:"+s.fee);
```

```
        Sopln("email: "+s.email);
        Sopln("mobile: "+s.emobile);

        break;
      } // one student

    case 3: { // course wise selection.
        Sopln("Enter course name:");
        String course = scn.nextline();
        Student[] students = sdao.select(course);
        for(int i=0; i<students.length; i++) {
          Students s = students[i];

          Sopln("\n Student "(i+1)+" Values are");
          Sopln(" sno  : "+ s.sno);
          Sopln(" sname : "+ s.sname);
          Sopln(" s course : "+ s.course);
          Sopln(" fee : " + s.fee);
          Sopln(" email: "+s.email);
          Sopln(" mobile : " + s.mobile);
        }
        break;
      } // course wise selection case close.

    } // inner switch for select close.

    break;
  } // select operation case close.
} // outer switch close.
sdao.closeConnection();
} // main close.
} // class close.
```

~~Variable and Types of variables~~

PACKAGES

BY

Mr.HARI KRISHNA SIR

NARESH technologies

An ISO 9001 : 2008 Certified Company

AMEERPET HYDERABAD

sri raghavendra Xerox

All software language materials available

beside sathyam theatre line balkampetroad ameerpet Hyderabad

cell :9951596199

Packages & accessibility modifiers

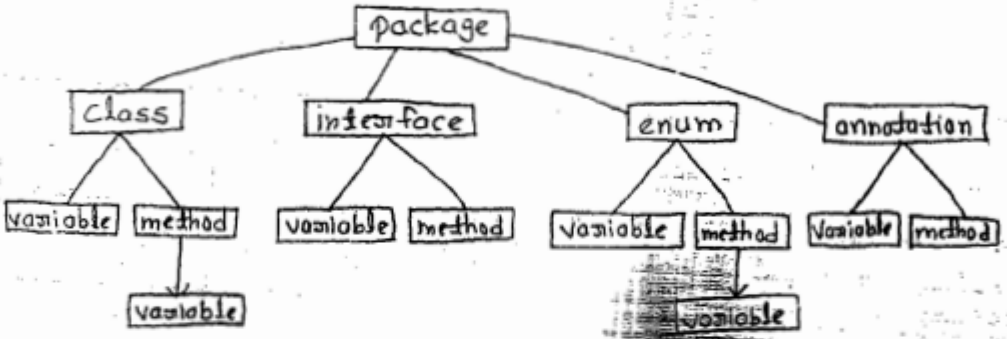
1) what is a package & sub package
2) creating package & sub package
3) -d option
4) package & sub package creation as per project standards
5) use of import statement
6) differences between import P1.*;
   import P1.A;
7) importing parent & sub packages
8) Diff ways of setting classpath
9) Accessing

NON-PACKAGE class member → from ~~non~~packaged class member
NON-PACKAGE class member → from packaged class member

20/09/17 ~~Classes & Types of Classes~~

Java Programming Elements & their purpose

Java supports 7 programming elements, for representing real world object in the programming world, all the 7 programming elements are organized as shown in below diagram:-



From the above diagram we can conclude, we have 7 different concepts we must learn to understand OOP and to create real world object in programming world by storing its values and by implementing its operations.

1. Package and types of packages and their purpose.
2. class and types of classes and their purpose.
3. variable and types of variables and their purpose.
4. block and types of blocks and their purpose.
5. construnctor and types of constructor and their purpose.
6. methods and types of methods and their purpose.
7. this keyword and its forms and their purpose.
8. sharing object to other methods, its modification effect
9. pass by value and different test cases.
10. setter method, getter methods, mutator, accesser methods.
11. mutable and immutable objects.
12. Factory method, Factory class, factory design pattern.
13. Singleton design pattern class.
14. Reading run-time values from keyband.
15. OOP block diagram, steps to create real world object in programming world, developing a project to create real world object in programming world using all above OOP concept.

Complete list of Java Programming Elements

For supporting object oriented programming concepts and implementing real world objects in programming world, Java supports 7 programming elements, They are :-

1. package → 2 different packages.
2. class → 6 different types of class.
3. variable → 4 different types of variables.
4. method → 2 different types of method.
5. constructor → 3 different types of constructor.
6. block → 3 different types of block.
7. inner class → 4 different types of inner class.

\+

= 24

Learning OOP is nothing but learning above 7 programming elements and its subtypes and ~~their purpose and~~ their creation syntax, and their purpose in object creation.

1. Package & Types of packages.

a*. What is package ?

b* Need of package or Why package ?

c*. Types of packages ?

d*. Syntax to create package.

e*. Saving, Compiling and executing packaged class.

f*. Structure of a class with package and subpackage ?

g*. Naming convention of a package as per project standards ? and different standard package names we use in project.

h*. Need of -d option of javac command and importance of a classpath in executing packaged class ?

i*. Different in setting class path in below options

1. Using java command options -cp/-classpath
2. Using cmd command set classpath
3. Using environment variable classpath.

~~import and static~~

import and static import statements

j *. What is import and use of import ?

k *. What is the meaning of fully qualified class name and what is the difference between FQCA and import statement.

l *. What is difference between
import p1.*; and import p1.example;

m*. Compiler algorithm in finding classes ?

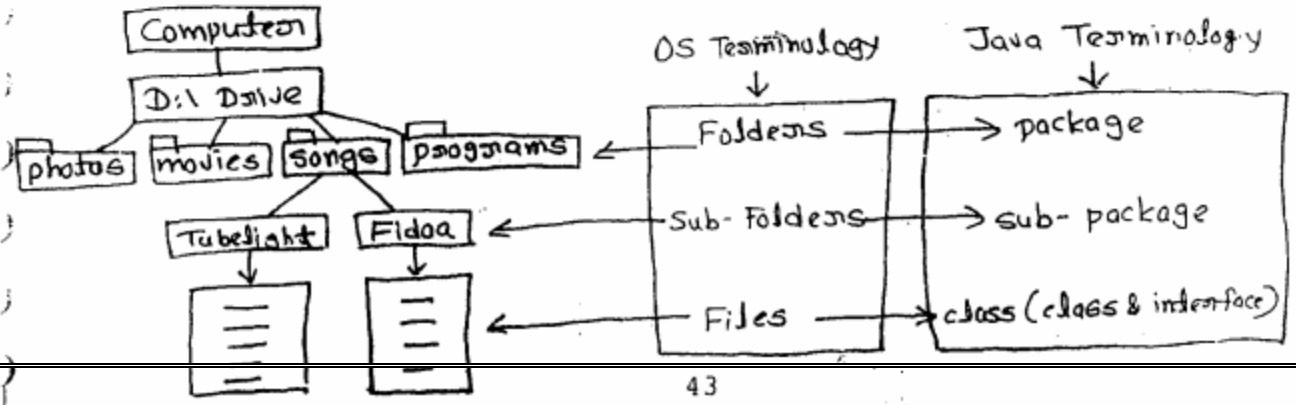
n *. Difference in accessing packaged class and subpackaged class ?

o * Interview and OCJP questions on package and import ?

p *. Need of Java 5 new feature "static import", its creation syntax, rules, sample programs and FAQs ?

a&b: What is package and why package ?

A package is a folder i.e linked with a class, it is used for organizing same functionality related classes and interfaces as one group and also used for separating one functionality multiple classes from other functionality group of classes.
And also it is used for seperating new classes from existing classes when both have same name.
For Example :- In computer we generally store different type of files as separate groups by using folder and directory concept as shown below. This folder wise organization is nothing but package concept in java and files are nothing but classes and interface in java.



- Like in computer, in projects we will create different types of classes and interfaces based on project functionality
- All these classes and interfaces we must create as seperate groups by using package and sub-package.

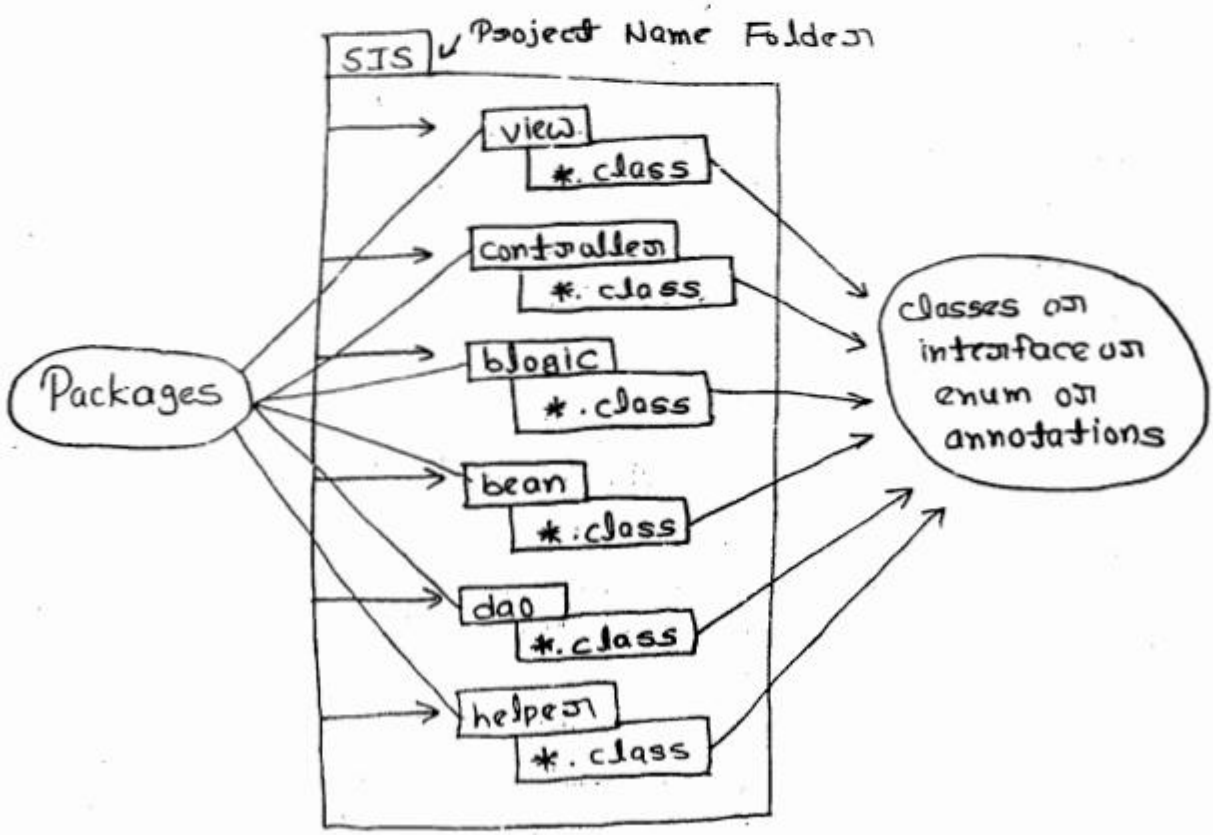
For example:-

→ We will have DB interaction classes (Model)/(dao)
→ We will have controller classes (controller)
→ We will have view classes (view)
→ We will data storeing classes (Bean/Domain)
→ We will have reusable/helper/util classes (helper)

to store all these classes as seperate groups we will create folder/Packages with the same functinality names. That means we will create packages with the names

model, controller, view, bean, helper, blogic, ...

↑ package names in projects.



c.) Types of package. 26/08/17

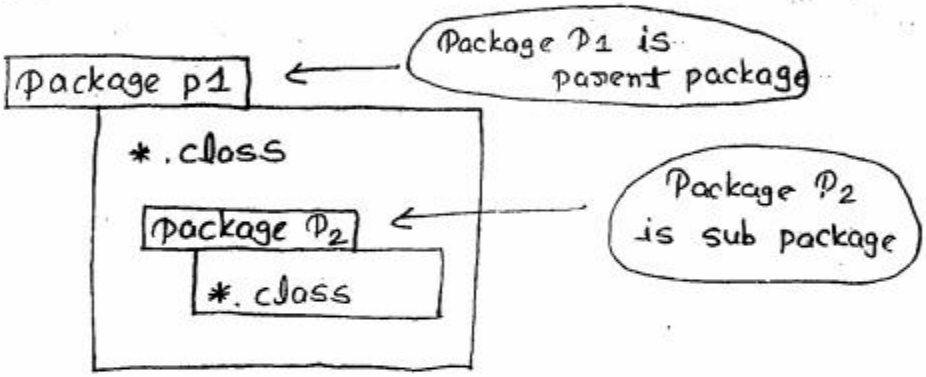
Java supports two types of package creation

1.) Parent package.

2.) Sub package.

The main package is nothing but parent package, A package created inside other package is called as sub-package.

For example :-



Parent package is used for representing main functionality of the project classes.

We will use sub-package for representing sub-functionality of the parent package functionality.

For example, in Java we have

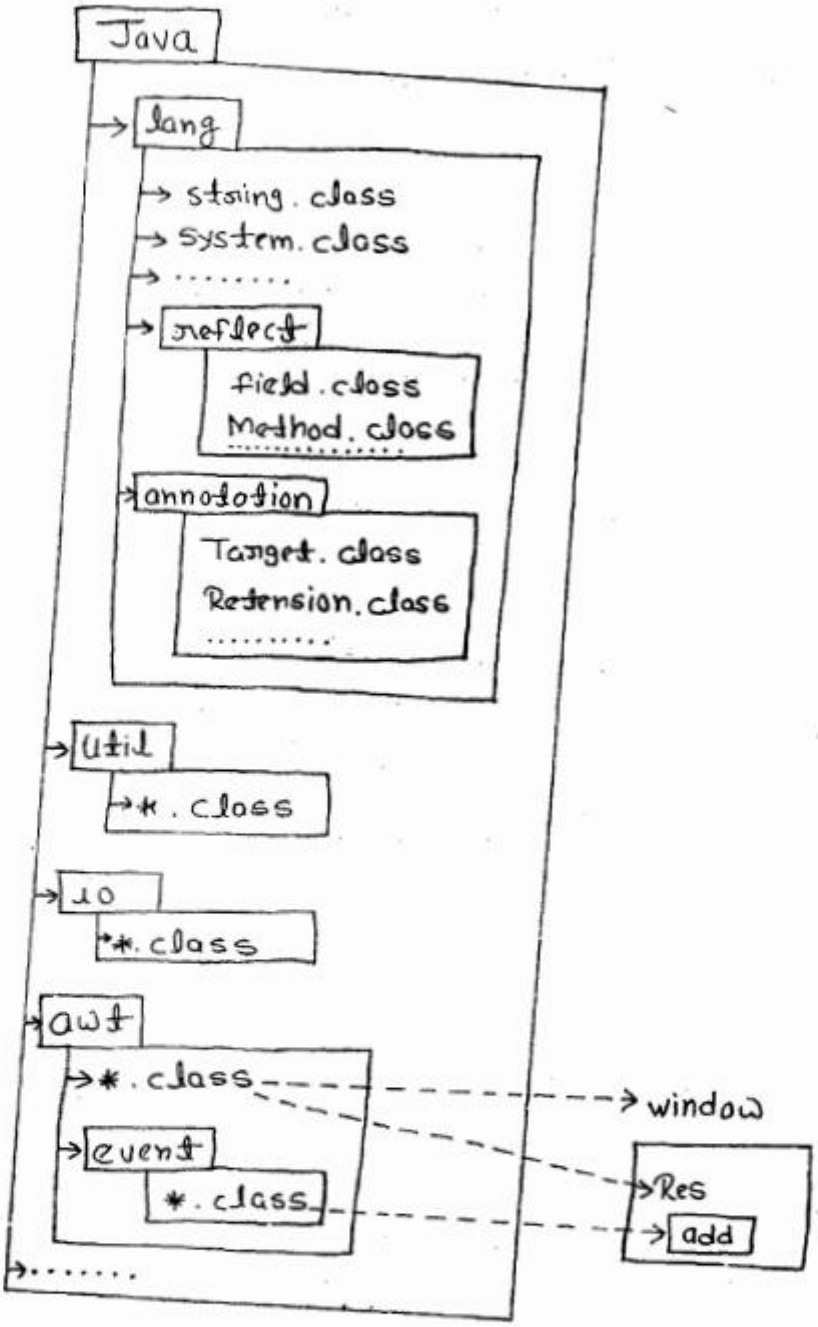
parent package java and

sub package lang, util, io, awt, sql.....

for specifying sub functionalities of Java language.

Note:- Inside sub package we are allowed to create some more sub packages.

For e.g. Inside java.lang subpackage we have subpackages annotation, reflect..... to represent lang level sub-functionality.



awt programming has 2 functionality

(i.) Components creation logic

(ii.) Components Event handing logic.

- To create window and its component like label, textfield, button etc.... we will use <u>awt</u> package classes.

- To handle components event and to execute the respective background logic, we will use <u>awt.event</u> sub-package classes.

Note :- According to Ameerpet, we have 2 types of packages

(i.) pre-defined package.

(ii.) user-defined package.

This answer is dirty answer or stupid answer, because these are not types these words are prepared based on who created and when created. The package (parent or sub) which is already created by SUN or by other team member is called pre-defined (parent or sub) package.

For example, java, java.lang, java.io, java.util .... are pre-defined (parent and sub) packages given by SUN as part of Java API.

For example service, service.dao, service.impl are user defined (parent and sub) packages.

The package created newly by our self is called user defined package.

Summary :-

Q. How many types of packages java support ?

- Java supports 2 types of packages : (i.) parent package and (ii.) sub package, the main folder we created for storing the main functionality classes is called parent package. The sub-folder created inside main folder for storing sub-functionality classes is called sub-package.
- The parent package and sub-package can be a predefined package or can be a user defined package.

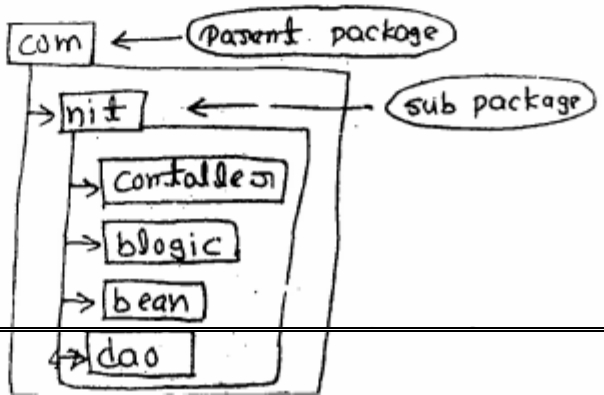
Q. What is the folder structure for the below package statements :-

com.nit.controller

com.nit.blogic

com.nit.bean

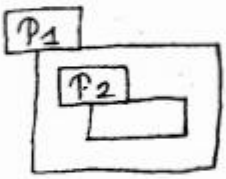
com.nit.dao



d.) Syntax to create package.

- For organizing classes as groups we must create our own packages (user defined packages).
- To create either predefined or user-defined, parent or sub packages in java language we will use a "package" keyword.
- Syntax:-

parent package creation
package packageName;
Example: package p1; creates a folder with name P1

sub package creation
package parentpackageName.subpackageName;
Example:- package P1.P2; creates two folders with the name P1, P2 

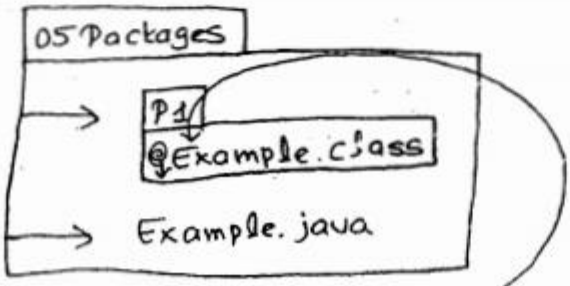
Note:- if parent package p1 is already existed, above statement will create only subpackage p2 inside P1 folder.

e.) Procedure to save, compile and execute package java file class

29 | 08 | 17

1. You can save java file in current directory.
2. Open cmd prompt -> change path to current working directory.
3. Run javac command as below:-

```
javac -d . Example.java
```



4. Run java command as below

```
java P1.Example
```

5. We must develop a java file package with a class as shown below:-

//Example.java

```
package p1;
class Example {
    public static void main (String[] args) {
        System.out.println("Hello World");
    }
}
```

Save above program with the name "Example.java" inside current working directory "05 packages" folder.

05 Packages
Example.java

Wrong Compilation and Execution

- We should not compile and execute packaged class with a regular syntax as :-

```
javac Example.java
java Example
```

X Wrong Syntax

- if we compile packaged class only by using javac command package folder is not created, Example.class file is directly saved in current working directory, as shown below

05 packages
Example.java
Example.class

- if we compiled packaged class directly by using javac command, we don't get any error, but if execute packaged class directly by using java command we will get error: "could not find or load main class Example"
- We get this error because inside Example.class file the name of the class is not Example, rather it name is p1.Example.

- At the time of compilation compile software will attach package name to class name. Then to execute packaged class we must use class name as packagename.classname as java p1.Example

- For the above command we will get error again "could not find or load main class p1.Example" because the p1 folder is not available.

- The meaning of java Example command is:
  (i.) search for Example.class file in current working directory.
  (ii.) if Example.class file is available, verify the class name is Example.

  Note:- given name and name of class in .class file must be same.

  (iii.) if the class name is different, JVM will throw error "Could not find or load main class"
  If class name is same as mentioned after java Command, JVM will start execution with main method.

- The meaning of p1.Example command is :-
  (i.) Search for package folder with the name p1 in CWD
  (ii.) if p1 is not available throw error "cfolmc", if p1 is available search for Example.class file in p1 package.
  (iii.) If Example.class file is not available, JVM will throw error "cnflmc"
  If Example.class is available, JVM will verify whether the class name is p1.Example or not.
  (iv.) If class name is different, JVM will throw error "cnfoLmc"
  If class name is same as the name mentioned after java command, JVM will start execution with main method.

- So, to execute a packaged class we must place its class file inside its packaged folder, then we should run the command

```
java packagename.classname
```

For example:-

```
java p1.Example
```

Que:- Will package folder created automatically by using the keyword "package"?

Ans:- No, package keyword will not create package folder automatically at the time of compilation. We must run javac command with its option "-d" as shown below:-

```
javac -d . Example.java
```

Summary:- developing, saving, compiling and executing packaged class and its structure

1. // Example.java

```
package p1
class Example{
    p s v main(String[] args){
        Sopln("H,W!");
    }
}
```

2. 05 packages (save.)

```
05 packages
  Example.java
  p1
    Example.class
```

3. 05package>javac -d . Example.java

4. 05 package> java p1.Example

5. O/P: H,W!

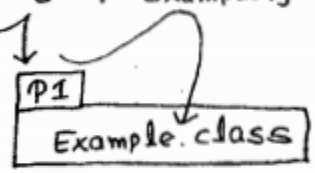
h.) Need of -d option with javac command. 30/08/17

-d option will creates folder for the package statement with the given name and it will move the generated .class file automatically into this package folder.

```
package p1;
class Example {

}
```

case#1: javac Example.java
↳ Example.class
here package folder is not created.

case #2: javac -d . Example.java



here with -d option package folder is created with the given name p1, and then Example.class file is moved into p1 folder.

Here . represent the folder path where this package folder must be saved along with its .class file

here . means CWD, so the package p1 folder is saved in CWD along its .class file.

Syntax to copy package into different folder :-

different folder :- C:\test

step 1: Create the folder test in C drive.

step 2: in the command prompt, in 05Package folder path run below command

javac -d C:\test Example.java

Executing above .class from current directory 05package

05package>java p1.Example

Error: Could not find or load

when we place package folder in different directory, We can not execute this package class from current working directory, JVM throw error, Because JVM will search this package P1 only in CWD (05 package), it can't search in other directory.

Solution :- To run classes from different folder we must set classpath by storing package's parent directory path.

This means to run P1.Example class from current working directory 05package, we must set classpath by storing C:\test, as shown below :-

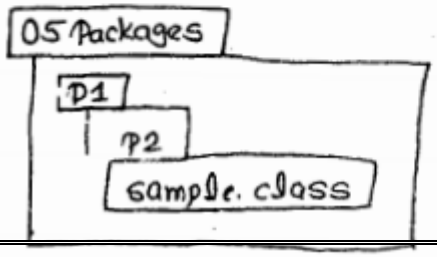
```
05Package > javac -d C:\test Example.java
05 Package > java p1.Example
Error: Could not find or load main class p1.Example
05Package > set classpath = C:\test
05 package > java p1.Example
Hello, World !
```

Note :- In classpath, we should not include package name, if we include package name, than JVM search for package inside package, it is not available it will throw error.

Que :- Develop a class with sub-package, compile and execute by saving in current directory and by saving in different directory.

Ans :- ⇒ Open Edit plus
⇒ write below code :-

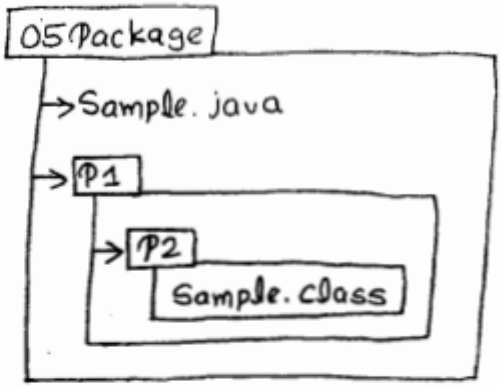
```
package p1.p2;
class sample{
    public static v m(SC] args){
        S.o.p.ln("HelloWorld");
    }
}
```



⇒ Save with the name Sample.java in CWD.

⇒ Compile and Execute by saving package in CWD.

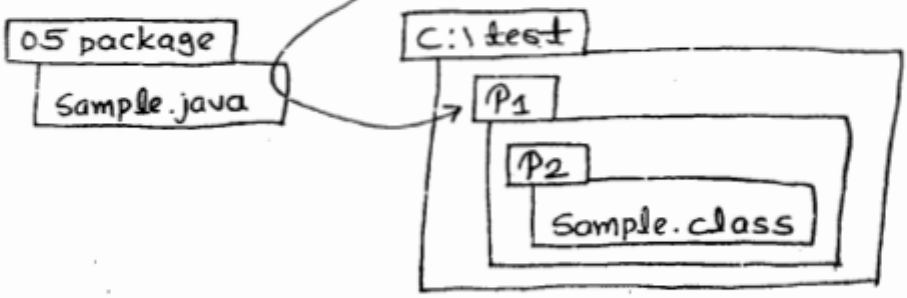
05package > javac -d . Sample.java




```
05 package > java p1.p2.Sample
Hello World
```

⇒ Compile and Execute by saving package in different directory (C:\test)

```
05 package > javac -d C:\test Sample.java
```




```
05 package > set classpath = C:\test
05 package > java p1.p2.Sample
Output :- Hello World.
```

Note:- If parent package folder already existed, only sub-package folder and its .class file are created directly inside this parent package folder

Q. What will happen for the below commands execution:-
05 package > javac -d C:\test Example.java

31/08/17

i.) Different ways of setting classpath and difference between them.

We can set classpath in 4 ways :-

(1.) By using java command option -cp

(2.) By using java command option --classpath

(3.) By using temporary setting in command prompt
set classpath=

(4.) By using permanent setting in environment variable window.

If we set classpath by using -cp/-classpath, this classpath setting value is available only for this cmd line.
If we run java program in the next cmd line, we will get Error: c f o l m c.
If we want to maintain this classpath setup value for next cmd lines we must set classpath using set classpath option.

If we set classpath by using set classpath= option, this classpath setup value is available for all cmd lines, but only in this cmd prompt window.
If we close this cmd window, classpath setup values lost.

If we set classpath by in Environment variable window, this classpath setup values are available permanently to all cmd windows, in all cmd lines even after system restart.

Usage:-

```
05Package> java p1.Example
Error: could not find or load main class p1.Example
```

```
05 package> java -cp C:\test p1.Example
Hello World! from the class p1.Example
```

← Not available for next cmd line.

```
05package > java p1.Example
Error: C n f o l m c p1.Example
```

```
05package > set classpath = C:\test
05pakcage > java p1.Example
Hello World! from the class p1.Example
05package > java p1.Example
Hello World! from the class p1.Example.
```

Need of . in classpath :- The . represents CWD path, By Using . placed in classpath, compiler and JVM will find, compile and execute classes from CWD.

If we donot place . in classpath, the classes placed in CWD are not found, compiled and executed by compiler and JVM.

> **Note:-** If we do not create classpath variable in our system, the default value of classpath is . means CWD path.
> Then compiler and JVM automatically find, compile and execute classes from CWD.

If we explicitly create classpath, we must place . in classpath value, else classes in CWD are not executed.

Refer page no. 221 (volume-1B) for more details on classpath and . in classpath execution cases.

Role of ; in classpath :- The character ; will act as

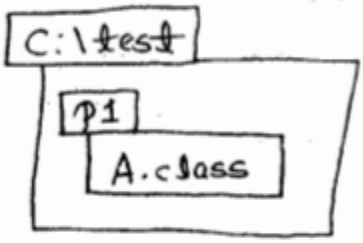
(i) Seperator when we use in between two paths.

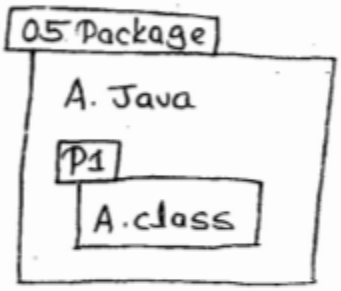
(ii) . (CWD) when we use at end of classpath value or when we use multiple time as ;;

E.g.
```
set classpath = C:\test; D:\abc
set classpath = C:\test;
set classpath = C:\test;; D:\abc
set classpath = Hari;;Krishna
```

Create a class called A in package p1, compile and save in C:\test and also in CWD. Display message loaded from C:\test when you compile and save in C:\test folder and Display message loaded from CWD when you compile and save in CWD.

With this program test all above classpath cases





01|09|17

Shocking point on "-d" option :-

We can use -d option for compiling normal java files, it means we can use -d option to compile a java file even though it doesn't have package statement.

The basic need of -d option is saving .classfile in the given folder path.

If the java file contains package statement,

-d option will create package folder, generate .class file in this package folder and then saves package folder in the given directory path.

If the java file doesn't contains package statement,

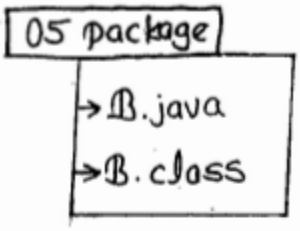
-d option will save .class file directly inside the given folder path.

For example,

```
// B.java
class B {
    public s v m(string[] args) {
        System.out.println("Hi");
    }
}
```

```
05package > javac -d . B.java
    B.class is directly saved in CWD
05 package > javac -d C:\test B.java
    B class is saved in C:\test folder
```

Case #1 :-




```
05package > javac -d . B.java
05package > java B
Hi (.class file is loaded from and
        executed from CWD)
```

Case #2 :-

```
05package > java -d C:\test B.java
05package > java B
Error: could not find or load main class
```

If we save .class file in different directory, then to execute this class, we must set classpath.

Here, B.class file is saved in other directory, so, classpath is mandatory.

```
05package > set classpath=C:\test
05package > java B
Hi (.class file is loaded and executed
        from C:\test)
```

Que :- Ameerpet style of creating package is wrong, Why ?

- In Ameerpet, while learning packages chapter we will save all java files inside one directory, and then later we will compile these java files by using -d option. then, Now package folders are created, .class files are generated and saved inside these package folders.
- Saving java files inside project folder directly leads to two problems :-
  - (i.) All hundreds of different functionality java files are saved in single folder. So that finding a particular functionality java file to modify is very tough.
  - (ii.) And more over we can not create 2 classes java files with a same name.
- Solutions to the Above two problems is :-

  We must create package folder not only for .class files but also for .java files.

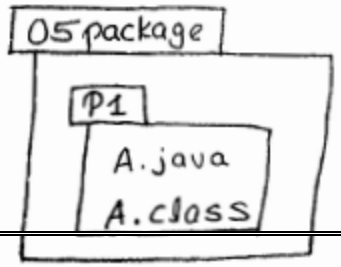
Project Style of package Creation

We must create package folder manually for java files, programmatically for .class files

Procedure :-

1. Create a folder with packagename p1.
2. Save source file in this package folder.
3. then compile these java file without using -d option.
4. then .class file is directly created inside package folder (p1).

05pakage> javac p1\A.java



5. 05 package> java p1.A

In this approach also, we have a problem, that is separating .class file from .java is very tough for sending project executable file to client.

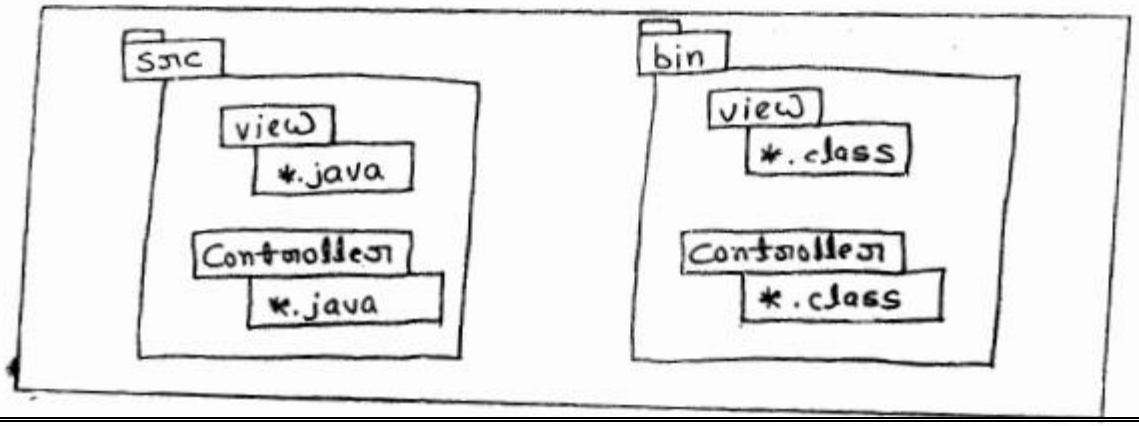
So the right approach is we must save java files and class files with package folder in seperate directories as shown below

Procedure:-

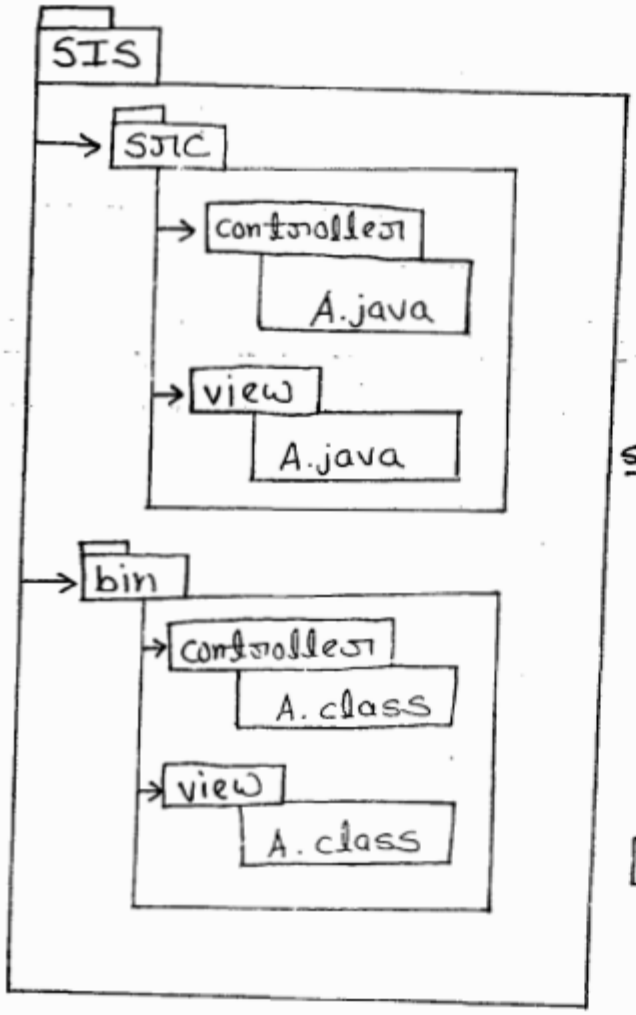
1. Create project folder with two ~~seperate~~ sub-directories "src" and "bin"
2. Inside src folder create package folder with its java files.
3. ~~Compile~~ Compile the java file with -d option by given bin folder path.
4. Then -d option will create package folder inside bin directory with the .class files generated.
5. Then change directory path to bin folder and execute class files
   (or) send bin folder all packages with its class file to client.

This is the right procedure of ~~creating~~ packaged java files and class files.

Not only we are following, eclipse also follows same procedure if we use eclipse IDE for java file creation, it automates this packages creation for java file and class files seperately in src and bin folders.



Example :-



SIS> javac -d .\bin src\view\A.java

SIS>javac -d .\bin src\controller\A.java

SIS > cd bin

bin> java view.A
↳ Load view's class

bin > java controller. A
↳ Load Controller's class

02/09/17

g.) Naming convention of a package as per projects standards.

1. Inside package name all character should be small letters. And package name should be as short as possible.
2. Package name should represent a group name generally functionality name like view, blogic, dao.....
3. In project we will create package name with as below

Company web domain reverse. functionality

E.g. com.nit. view
com. nit. blogic
com. nit. bean
com. nit. dao

Package statement Rule :-

In a single java file only one package statement is allowed also the pakage statement must be first statement in the java file.

Example.java

```
class Example { }
package p1;
```

X CE : Package statement must be first statement in java file.

Example.java

```
package com.nit.view;
package com.nit.blogic;
package com.nit.bean;
class Example {
}
```

→ X Compile time Error

Reson:- A class must exhibit only one functionality.

Accessing One package class from another packaged class :-

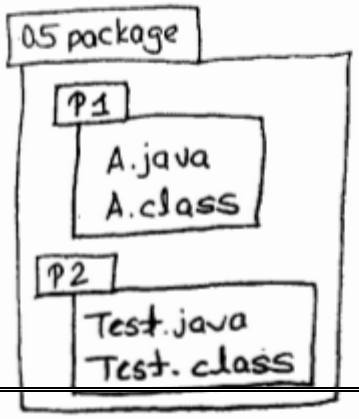
A.java

```
Package p1;
public class A {
(R1) public void m1() {
(R2)     S.o.pl.n ("Hi");
    }
}
```

Test.java

```
package p2;
class Test {
  P.s.v.main(s[] args) {
    P1.A a1 = new p1.A();
(R3) a1.m1();
  }
}
```

```
05 package > javac p1\A.java
05 package > javac p2\Test.java
05 package > java p2. Test
O/p: Hi
```



If you want to access one package class from another package class, we must follow below three rules :—

R1 :- Class should be public.

R2 :- The method or variable we want to access should be public.

R3 :- In another class, we must access this class by using fully qualified name. Means we must access class with its package name as p1.A.

Run above program with the below 4 cases :-

Case 1 :- Call m1() method directly, without class A reference, as shown below :-

m1(); X CE: not find symbol m1

If we access method directly without reference then compiler searches m1() method definition in same class i.e. Test class. It can not search in class A. If you want to search m1() in class A we must invoke m1() with its class A object reference. As shown in above program.

Case 2 :- Remove package name in accessing class A. Means replace p1.A a1 = new p1.A(); as A a1 = new A(); X CE: can not find symbol class A

If we place class name directly, compiler searches class A definition in current method, in current class, in current package folder. But not in another packages.

If you want to access class from another package we must access it using package name as shown in above program.

Case 3 & Case 4 :-

Remove public to class A and method m1() then we will get compile time error, "Class A is not public", "method m1() is not public".

If we want to access a class and or its member from different package, it should be public. As shown in above program.

Introduction to import statements

1. Import is a package concept related keyword.
2. When we want to access a class from one package to another package, we must use import statement in the accessing class java file. For example in previous program we must use import statement in Test class to access class A from the package p1 to package p2.

Test.java

```
package p2;
import p1.*;
class Test {
    p.s v. main (String [] args) {
    A a1 = new A();
    a1.m1();
    }
}
```

3. import statement must be placed after package statement and before class declaration.
4. import statement will not import (c and p) classes from one package to another package, rather import statement will just give info to compiler in which package the class is available.

04/09/17

Compiler Adding changes to our class with respect to packages and import statement

```
package p2;
import p1.*;
class Test {
  P S v m(S[] args){
    A a1 = new A();
    Sopln (a1.x);
}
```

```
class p2.Test{
  P S v main(java.lang.S[] args){
    P1.A a1 = new p1.A();
    java.lang.System o p ln (a1.x);
  }
}
```

1. If we create a class with a package and import statements, at the time of compilation compiler software will remove package and import statements and attaches package name to class name and then finally save the modified compiled byte code inside .class file. as shown above ↑
2. Compiler will attach package statement package name to class name and constructor name.
3. Compiler will attach import statement to the classes names those are accessing from this package or this imported package

For more details, read bytecode by using javap command.

Syntax :- `javap -verbose classname`

Example :- 05packge > javap -verbose p2.Test

Que :- What is the name of a package class ?

An :- Fully qualified name It means package_name.class_name

For example :- p1.A, p2.test,
java.lang.String
java.lang.System
java.sql.connection

Example :— If we need to pass classname as input to another program, we must pass name with fully qualified name. as shown below:—

Enter class name: A ✗ java.lang.CNFE

Enter class name: p1.A ✓

l. Difference between import p1.* and import p1.A

- When we need to access class A from p1 package to p2 package, we can use either import p1.* or import p1.A.
- There are 3 differences between these two statement :—

import p1.*	import p1.A
1. Using p1.* we can access all public classes available in the package p1.	1. Using p1.A, we can access only class A in the package p1.
2. p1.* has less priority, it means if class A available in both p1 and p2 package, class A is loaded are executed from current package p2.	2. p1.A has First priority, it means class A is loaded and executed from package p1 even though class A is available in current package p2.
3. with p1.* we can create class A in current java file.	3. with p1.A we can not create class A in current java file.
4. with p1.* compiler doesn't search for .class/.java file in package p1, until we use class name.	4. with p1.A, compiler will immediately search A.class/A.java file in pacakage p1 if not available, it will throw error

For programming point on above table, Refer page No. 222 & 223 in volume 1(B)

06/09/17

m. Compiler algorithm in finding classes

import p1.*	import p1.A;
1. Searching class A {} in current method main method.	1. Searching class A {} in current method i.e. main method.
2. Searching class A {} in current class Test{} body.	2. Searching class A {} in current class Test{} body.
3. Searching class A {}, in current java file Test.java	3. Searching class A {} in current java file Test.java.
4. Searching A class {}, in current java file package p2.	If class A {} is available compiler will throw error for import p1.A statement
5. Searching A.java file in current current java file package p2.	4. Searching A class {} file in imported package p1.
6. Searching A class {} file in imported package p1.	5. Searching A.java file in imported package p1.
7. Searchin A.java file in imported package p1.	

if here also not available, then compiler ~~with~~ throw error could not find symbol class A

Here the error thrown for the statement A a1 = new A();	Here the error thrown for the statement import p1.A;

Note:- If we remove import p1.A; from this java file, compiler will search A.class then A.java files in current package p2.

Auto Compilation

In a project, we no need to compile every class java file. When we compile main method class java file all classes java files those are accessing from this main method class are automatically compiled by compiler software.

The phenomina or algorithm, compiling internally using classes java file automatically by compiler software, is called auto compilation.

For example, observe below program

A.java

```
package p1;
public class A {
    public void m1(){
        Sopln("p1.A. m1");
    }
}
```

Test.java

```
package p2;
import p1.*;
class Test {
    p.s.v main(S[] args){
        A a1 = new A();
        a1.m1();
    }
}
```

p1
- A.java
- A.class

p2
- Test.java
- Test.class

```
05package> javac -d . p2\Test.java
```

If both .class and .java files are available, compiler will the time stamp of .java and .class file.

If .java file time stamp > .class time stamp
    then recompile

else
    Compiler will directly uses existing .class file.

Autocompilation algorithm all steps:- [Page No. 222]

When we are using a class from another class, No need to compile that class first, Compiler automatically compiles that class.

For example assume we are calling Example class method from sample class method we can compile sample class directly without compiling Example class.

Compiler follows below procedure to compile Example class:-

1. First it searches that Example class definition in Sample .java, if not found
2. It searches for Example.class in Sample class package, if not found
3. It seaches for Example.java in Sample class package, if not found
4. It searches for Example.class in imported packages, if not found
5. It searches for Example.java in imported packages, if not found
6. Then compiler terminate Sample.java file compilation by throwing CE: cannot find symbol.
7. If Example.java is found, it searches for ~~exap~~ Example class definition in Example.java file. If it is found, compiler compile entire java file, it means it also compile other class definitions and generate those class's .class files. Else terminates Sample.java file compilation with above compilation error.
8. If Example.class is found, it also searches for Example.java. If not found, compiler uses Example .class file directly.

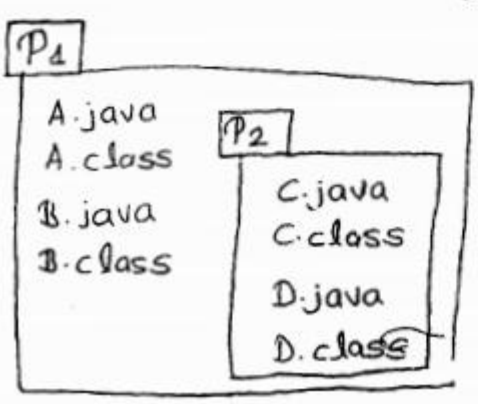
9. If Example.java is also available, it checks modified time of both files, if Example.java file modified date is greater than Example.class modified date, it compiles Example.java again for generating Example.class with its latest changed java code.

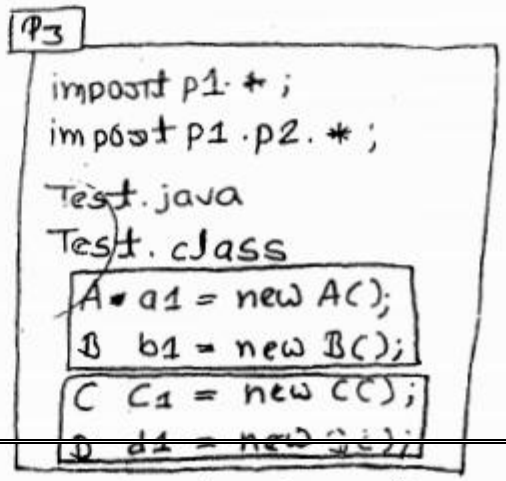
Test all above cases/points using below two programs.
[Refer Volume - 1B, Page No. 222 to 223]

09/09/17

Importing Subpackage and accessing sub-package classes from our own class from different package

- For accessing classes from sub-package we must place import statement for subpackage classes.
- If we write only parent package import statements, we can access only the classes those are available in parent package.
- By adding sub-package import statement, we can not access classes from parent package, we can access classes only from subpackage.
- If we want to access classes from both, parent package and sub-package, we must add import statement for both parent package and sub-package.
- Consider below diagram and identify from which package classes are accessing from which import statement :-





A.java

```
package p1;
public class A {
    p.s.v.m(S[] args)
}
```

B.java

```
package p1;
public class B {
}
```

C.java

```
package p1.p2;
public class C {
}
```

D.java

```
package p1.p2;
public class D {
}
```

Test.java

```
package p3;
import p1.*;
import p1.p2.*;
class Test {
    p.s.v.m(S[] args){
        A a1 = new A();
        B b1 = new B();
        C c1 = new C();
        D d1 = new D();
    }
}
```

- In above project we should not write main method in every class, its not compile time error or run time error, It is a wrong design, because the classes A, B, C, D must be executed from class Test
- So we should not define main method in these 4 classes.
- We should define all 4 classes public, then only we can access them from Test class

- To compile and execute above project classes, we need to compile and execute Test.java file, As per Auto Compilation algorithm the classes A,B,C,D are Automatically Compiled.

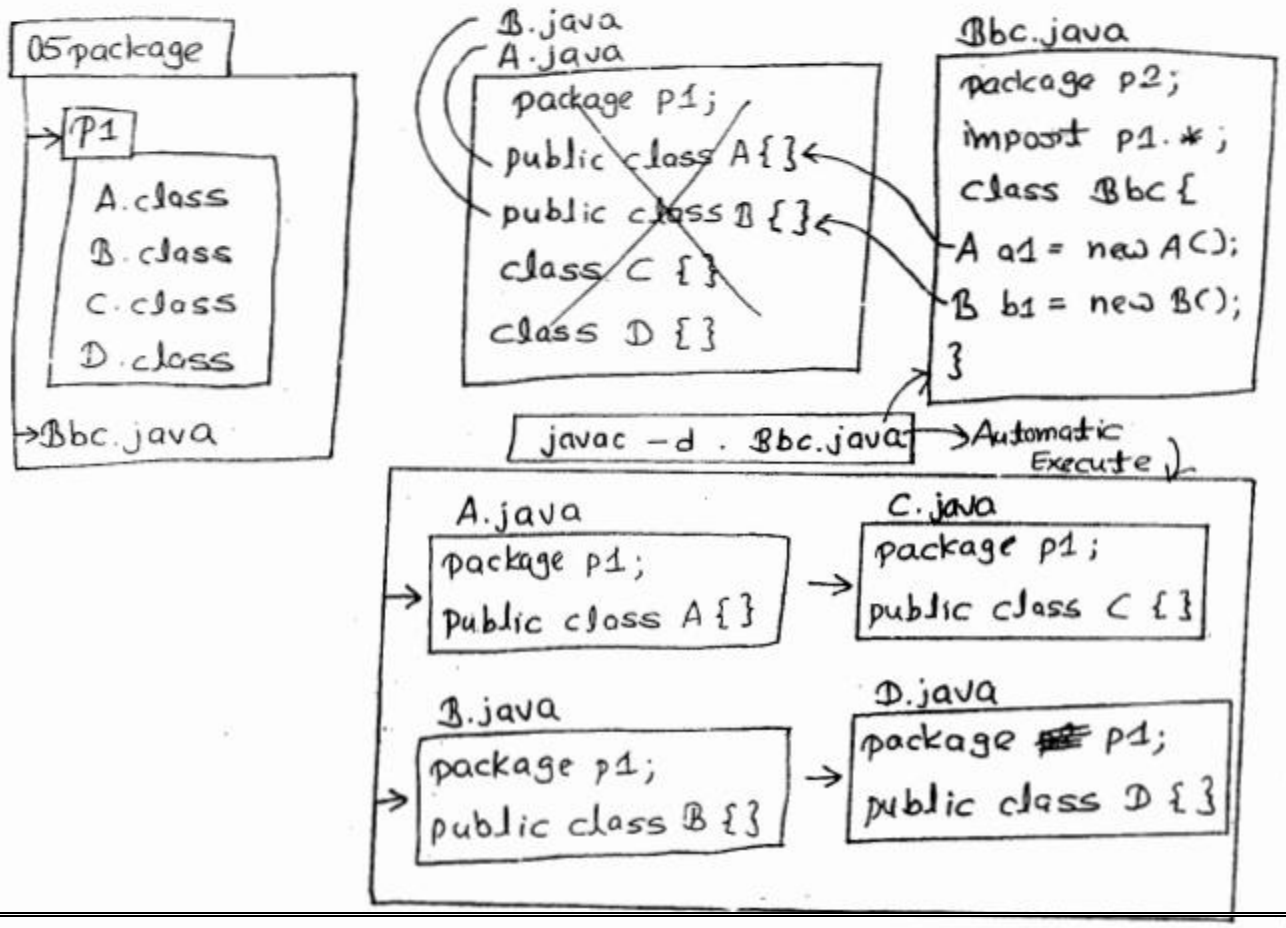
  Use below commands to compile and execute test class.

```
05package> javac p3\Test.java
05package> java p3.Test
```

Q. How can we place multiple classes in single package folder ?

Ans:- We have 2 approaches to place multiple classes in single package

1. Declare all classes in single java file
2. Declare Every class in Separate java file with the same package statements.



- If we save all classes in single java file, all class files are created in single package but the problem in this approach is we can not access these classes in another package classes, because these classes are not public.
- To access these classes in another packaged classes, we should declare classes as public.
- If we declare class as public, classname and java-file name must be same, then in a single java file we can not declare multiple public classes. Hence it is not a good approach creating multiple classes in single java file.
- Hence to reuse a class, it means to access a class from different package classes and also to achieve auto-compilation we must create each class in seperate java file by declaring class as public. If we want to place .class files of all these in a single package we must place same package name statement in all java files.

> **Rule:**— To get auto compilation of these java files we must save java file also inside single package.

Q. Why public class name and Java file name must be same and why non public class name and Java file name no need to be same?

⇒ For 2 reasons, public class name and java file name should be same:-

1. For easy Identifying in which java file the class is defined either for reading or modifying source code.
2. Also for achieving Auto-compilation.

These 2 rules are not applicable for non-public class, because non-public classes are accessible to some programmer who defined them, So he/she knows in which java file class is defined, hence non-public class name and java file name no need to be same.

- Either class is public or non public we must create java filename and class name must be same and more over we must create one class per java file for autocompilation.

Que:- What is the output from the below program ?

```
public class A {
    public void m1(){
            sopln("Hi");
        }
    }
```

```
package p1;
class B {
    p s v main(s[] args){
        A a1 = new A();
        a1.m1();
    }
}
```

Ans:- Compilation Error: could not find symbol class A.

Different Combinations to access one class from another class.

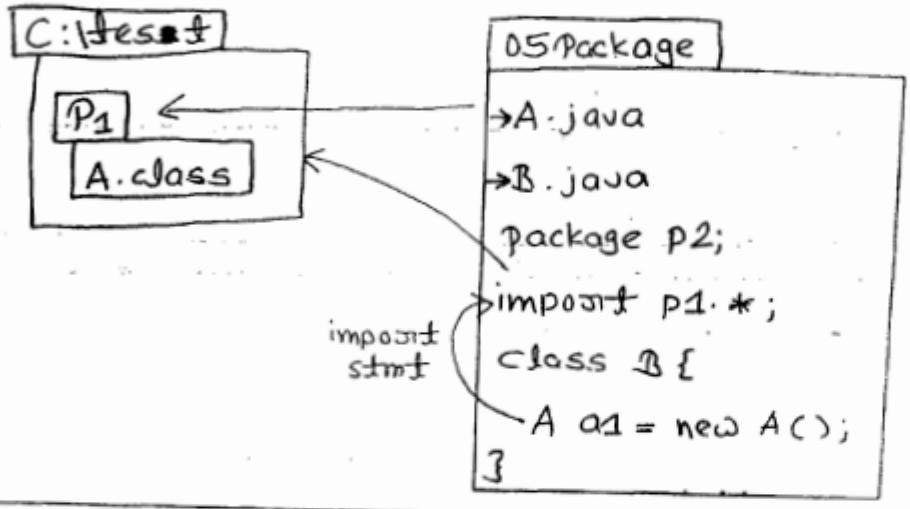
Non Packaged class ——✗——> Packaged Class

Non Packaged class ——✓——> Non packaged Class

Packaged class ——✓——> packaged class

Packaged class ——✓——> Non packaged class

Q. From Compiler point of view, what is the difference between import statement and classpath ?
What is the output of below program:-




```
05package > javac -d C:\test A.java
05 package > javac -d C:\test B.java
```

Ans:- 1. Compiler uses class path to find package and ofcourse also uses for finding classes.

2. Compiler uses import statement to find classes those we are accessing in our class.

12/09/17

Q. How can we access parent package classes in the same parent package other classes and parent package classes in its sub-package classes and sub-package classes in other sub-package classes and in parent package classes ?

Ans:- If we want to access a class from other classes of same package we no need to use import statement. we can access the class directly. So parent package classes we can access directly by their name inside other classes in the same parent package.

For Example:-

A.java

```
package p1;
public class A {
}
```

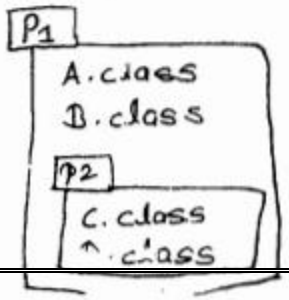
B.java

```
package p1;
public class B {
    A a1 = new A();
}
```

← Import statement is not used

If we want to access a class from other package classes this class should be public and we must use import statement of this class package in another java file in which we want to access.

Hence if we want to access parent package classes in subpackage classes or if we want to access sub-package classes in other sub-package classes or in its parent package classes we must import statement of its package and also the class should be public.



A.java

```
package p1;
import p1.p2.*;
class A {
    C C1 = new C();
}
```

D.java

```
package p1.p2;
import p1.*;
class D {
    B b1 = new B();
}
```

B.java

```
package p1;
public class B {

}
```

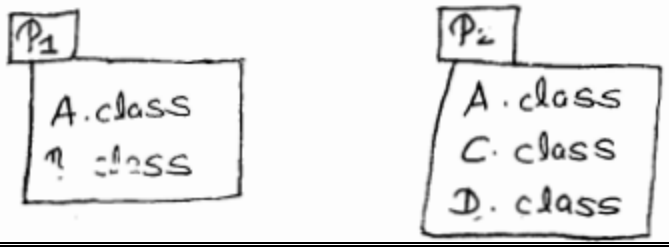
C.java

```
package p1.p2;
~~import~~ public class C {

}
```

Q. We can access classes of one package from another package in two ways using fully qualified name and by using import statement. Then when to use import statement and when to use FQN ?

Ans:- We will use FQN only if there is any name conflict that means if a class is defined in more than one imported packages we must use FQN (Fully qualified Name) to diffrentiate this class from the imported packages.

Otherwise, means if there is no name conflict, we always access classes by using Import statement. Consider below Example, we access class by using FQN and other classes by using Import statement.



P3

```
//Test.java
package p3;
import p1.*;
import p2.*;
class Test {
    psv main(s[]args){
    //A a1 = new A();  ---> CE: Ambiguous
      B b1 = new B();
      C c1 = new C();
      D d1 = new D();
      p1.A a1 = new p1.A();
      p2.A a2 = new p2.A();
      a1.m1();
      a2.m1();
  }
}
```

Pending Topics :-

1. Static import

2. Working with jar files

~~Refer next chapter~~.

Refer volume 1B

STRING HANDLING

BY

Mr.HARI KRISHNA SIR

NARESH technologies

An ISO 9001 : 2008 Certified Company

AMEERPET HYDERABAD

sri raghavendra Xerox

All software language materials available

beside sathyam theatre line balkampetroad ameerpet Hyderabad

cell :9951596199

String Handling

22/7/15

In this chapter we will learn storing string data in JVM from a program & further performing several operations on this string data. The operations such as comparing, concatenating, replacing, changing case, trimming, retrieving characters, splitting, etc operations.

In this chapter we will learn below things

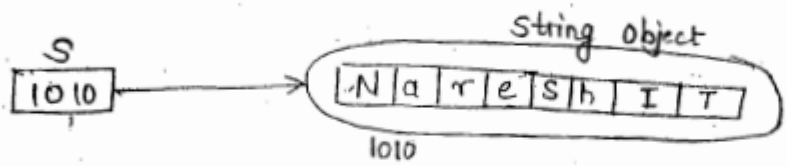
1. What is string?
2. What is string handling?
3. Different ways to store string data in JVM.
4. Difference bt^n String, StringBuffer, StringBuilder classes.
5. Immutable and Mutable objects.
6. ThreadSafe and Non ThreadSafe objects.
7. StringPolling
8. String constant Polled Area.
9. Garbage Collection with String Object
10. String class constructors and methods
11. StringBuffer class constructors and methods.
12. Programs
13. Interview & SCJP Questions.

What is a String?

→ String is a sequence of characters which is placed in double quotes. It is an instance of a predefined class java.lang.String. That means every string literal is an object of string class by default.

for e.g. "NareshIT" is an object of String class.

```
String s = "NareshIT";
```



→ The class String is used for storing string data in java program.

What is String Handling?

→ Performance different operations on string data is called String Handling.

→ To perform several operations on string data the reqd. methods are already given by sun in the classes String, StringBuffer, StringBuilder.

→ As a programmer we just need them to access to perform reqd. operations.

For e.g.

- If we want to compare two string having same characters or not, we must call a method 'equals' available in String class.

  For e.g.

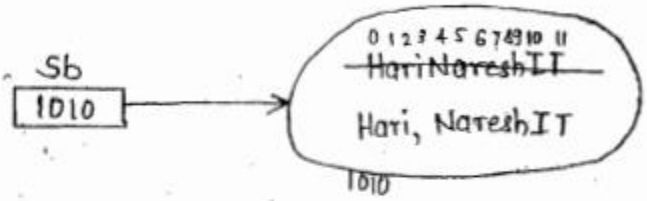
```
String S1 = "Naresh IT";
String S2 = "Naresh IT";
if (S1.equals(S2)){
    sopln("Hi");
}
else{
    sopln("Hello");
}
```

→ Comparing characters in S1 & S2.

- If we want to insert some characters in the middle of the string we must a method called '<u>insert</u>' from StringBuffer class.

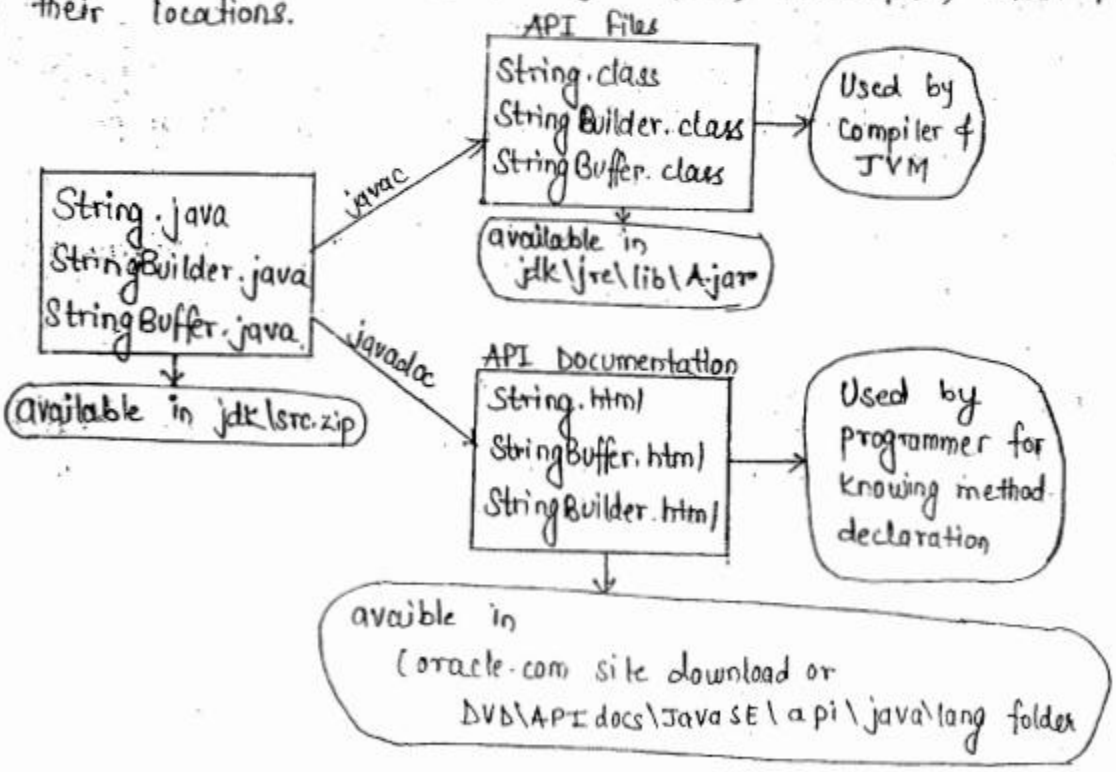
e.g.

```
StringBuffer sb = new StringBuffer("HariNareshIT");
sb.insert(4,',');
```



How can we find or where can we find method declarations available in String, StringBuffer & StringBuilder classes?

→ We must use Api Documentation files of these 3 classes. Below diagram shows above 3 classes java files, .class files, .html files & their locations.



What are the different ways we have to store string data in JVM?

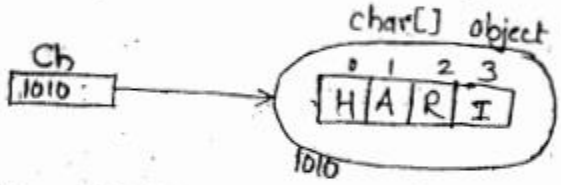
29/7/15

→ We have

1. Char[] object
2. String object
3. StringBuffer object
4. StringBuilder object

→ The actual way of storing string data is using char[] object. for e.g.

```
char[] ch = { 'H', 'A', 'R', 'I' };
```

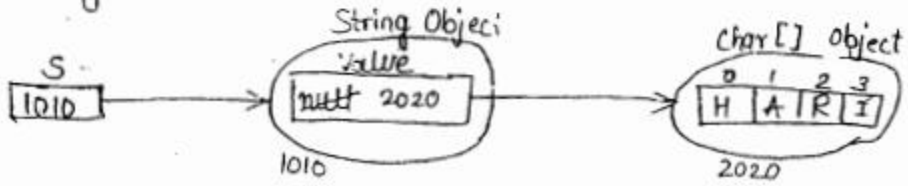


→ Inside String, StringBuffer, StringBuilder classes also char[] object is used for storing string characters.
for e.g.

String class looks like below

```
public final class String {
        private final char[] value;
                ===
}
```

```
String s = "HARI";
```



→ String, StringBuffer, StringBuilder objects has a char[] object for storing the given string characters.

When we have char[] object for storing string characters why String class is given?

→ char[] has two problems

1. Length is fixed
2. Operations reqd. inbuilt methods are not available.

→ To solve above two problems every programmer in projects must define their own class & methods.

→ Since this logic is same for every programmer, sun has decided to define a class with a name string with reqd. no. of methods to operate on char[] object.

→ String class is defined as <u>immutable</u> object that means for the modifications we are doing on string data result will be stored in another null string object, but not in current string object.

→ For e.g.

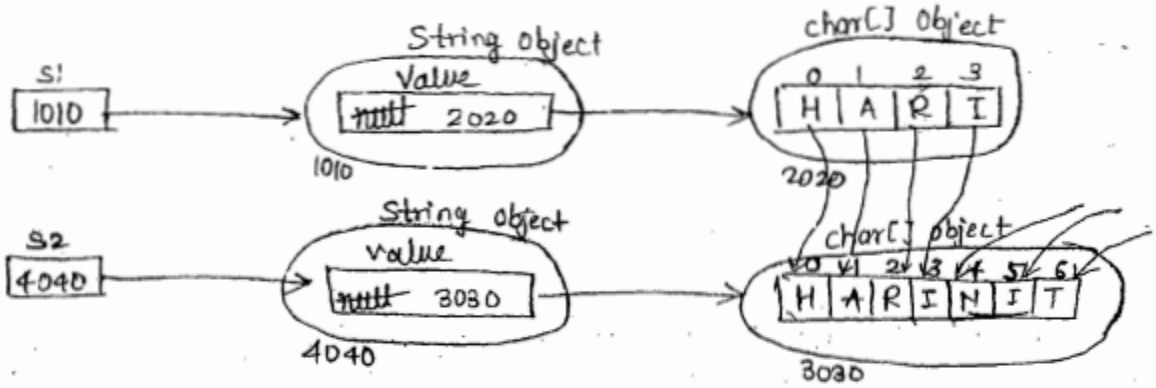
If you want to concat more characters to an existing string object characters we must use a method called 'concat'. This method internally concat argument string characters to current string characters, stores result in new string object, then returns its reference.

e.g.

```
String s1 = "HARI";
String s2 = s1.concat("NIT");
Sopln(s1);        ---------->  o/p HARI
Sopln(s2);        ---------->  o/p HARINIT
```

~~Memory diagram:~~

Memory Diagram:



Why String is given as immutable?

→ For using String object as Key in Map collection it is created as immutable.

→ Once we store an object as key in Map, its data should not be modify. If its data is modified the value mapped to this object cannot be retrieved from this Map because other programmer will try to retrieved value using old data.

→ In order to not stored ~~modified~~ modification result in current object, this object should be immutable.

Why StringBuffer class is given when we have String for storing string data?

→ Since String is immutable, for every modification on String data new String object will be created. This leads to more memory consume, performance is reduced.

→ To solve this problem StringBuffer class is given as a mutable object.

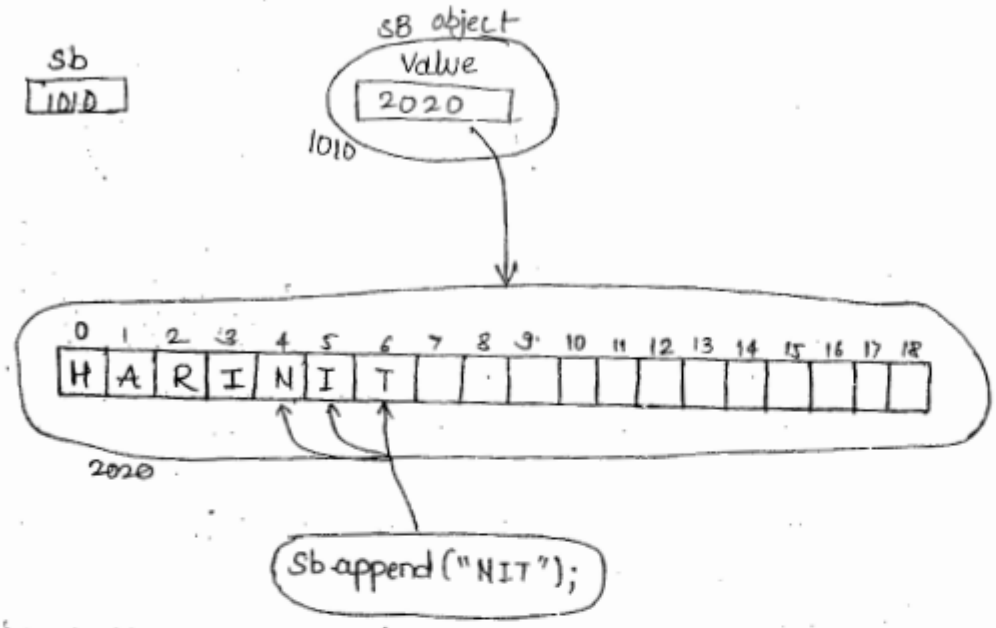
→ That means the modifications we are doing on string data in StringBuffer object the result will be stored in the current StringBuffer object, new StringBuffer object will not be created.

for e.g.

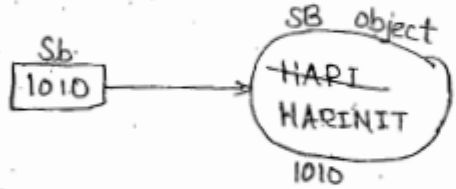
```
StringBuffer sb1 = new StringBuffer("HARI");
StringBuffer sb2 = sb1.append("NIT");
Sopln(sb1);  ———————→  o/p : HARINIT
```

StringBuffer Memory Diagram

25/7/15



Shortcut Memory



Why StringBuilder class has define when we have StringBuffer class.

→ StringBuffer class is given as synchronized or threadSafe object. Due to this we will get less performance in single thread model & method local operations because locking & unlocking object is not necessary.

→ So StringBuilder class is given to have non-ThreadSafe mutable string.

When Should we use String & StringBuffer.

String	StringBuffer
If we don't want to modify String data or if String data modified result want to store in new object, then we must use String object to store String data.	If you want to modify String data in same object we must use ~~String~~ StringBuffer.

When should we use StringBuffer & StringBuilder

→

StringBuffer	StringBuilder
for storing & modifying string data in multithread environment we must use stringBuffer.	for storing & modifying string data in single thread model app & in method local we must use StringBuilder.

What is the diff b/w Immutable & Mutable object.

Immutable object	Mutable object
→ An object whose data is not modifyable or modifyable but result is stored in new object is called Immutable.	An object whose data can be modifyable in the same object is called Mutable object.
→ for e.g. String, all wrapper classes, java.io.File classes are immutable means once data is stored in this object we cannot modify it.	for e.g StringBuffer, StringBuilder, all collection objects are mutable. By default every variable is also Mutable.

What is the diff. b/w threadsafe & non-threadsafe objects.

threadSafe	Non-threadsafe.
An object which is not available to multiple threads at a time to modify its data concurrently is called threadsafe.	An object which is available to multiple thread at a time to modify its data concurrently is called Non-threadSafe Objects. If we use Non-threadSafe object in multithread environment we will get wrong Results. This object is recommended to use in single thread environment & in method local operations.
OR	OR
An object which is available to multiple threads for modifying its data concurrently, but one thread modified result is not affecting to another thread, means result	If we create class as mutable object class without declaring its method as synchronized then it is non-threadsafe object.

is not stored in same object. We can develop a threadsafe object in two way 1. Declare all mutator methods as synchronized. 2. Create class as immutable Object class. for e.g. StringBuffer, Vector, Hashtable classes or threadsafe object classes.	By default every class is mutable non-threadsafe object classes. for e.g StringBuilder is non-threadsafe object.

—x—

~~Imp~~

Is the String object is threadsafe object?

→ Yes, it is threadsafe object because of Immutable.

→ All wrapper classes, java.io.file classes are also threadsafe. Because they are also Immutable.

Imp

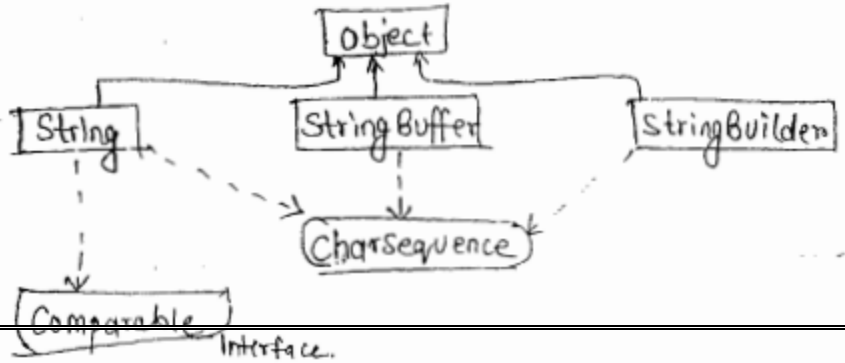
What is String, StringBuffer, StringBuilder?

→ String is immutable sequence of characters. It is a predefined final class.

→ StringBuffer is an mutable, threadsafe sequence of characters. It is also predefined final class.

→ StringBuilder is an mutable, non-threadsafe sequence of characters. It is also predefined final class.

What is the relation b/w above 3 classes.



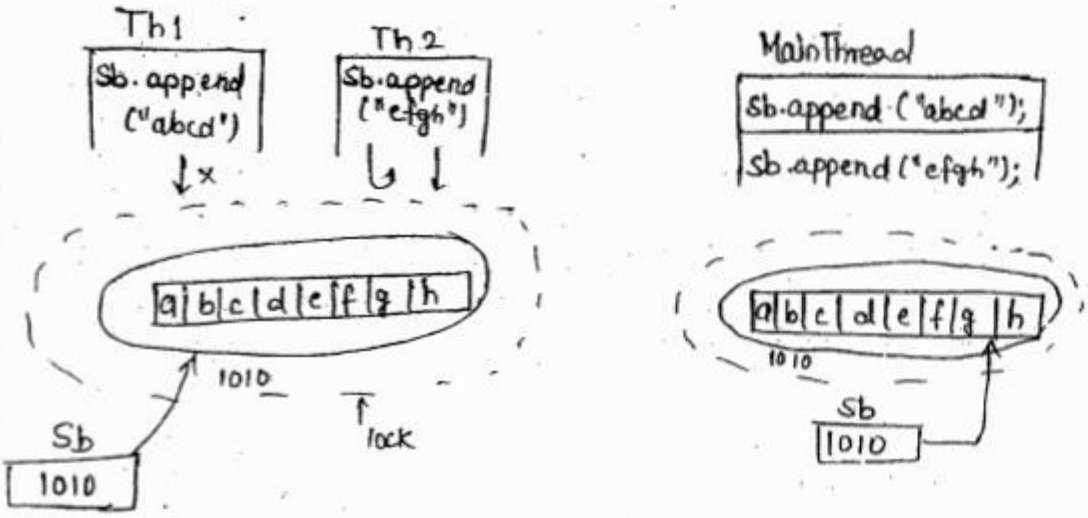
→ These 3 objects are siblings, they ~~did~~ d'nt have inheritance relation because behaviour is different.

→ These 3 classes are subclasses of java.lang.Object class & java.lang.charsequence interface.

→ String is also subclass of comparable Interface so we can store its objects inside Treeset & TreeMap Collections.

→ We cannot store StringBuffer & StringBuilder Objects in TreeSet & TreeMap Collections because they are not Comparable type objects.

Note:-



String Object Creation :-

→ We can create string objects in two ways

1. Using String Literal
2. Using new Keyword Constructor.

for e.g.

```
String S1 = "abc";
String S2 = new String("abc");
```

Note: Any data we place in double quotes is a string class object for the double quote JVM will create string object, stores these literal characters inside this object automatically.

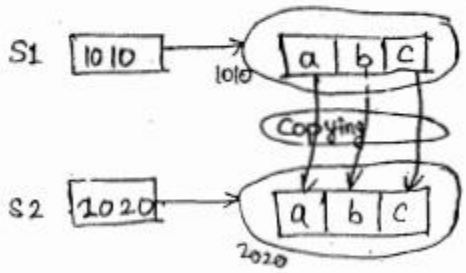
→ The actual way of String object creation approach is String Literal approach. Using new Keyword & constructor we cannot create the very first object.

→ We must new Keyword & Constructors approach only for copying operation that means we have already one String has existed, we want to create new String object with the data available in previous String object. Then we must use use Keyword constructor approach.

for e.g.

```
String S1 = "abc";
String S2 = new String(S1);
```

assignment, assignment, copy



What are the diff b/w these two approaches.

→ We have two differences in creating String object in the above two approaches.

1. No. of Objects creation.
2. String Pooling.

Let us understand first one.

→ In Literal approach only 1 object is created.

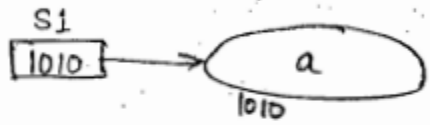
→ In new Keyword approach 2 objects are created.

for Eg.

Approach #1 :

```
String s1 = "a";
```

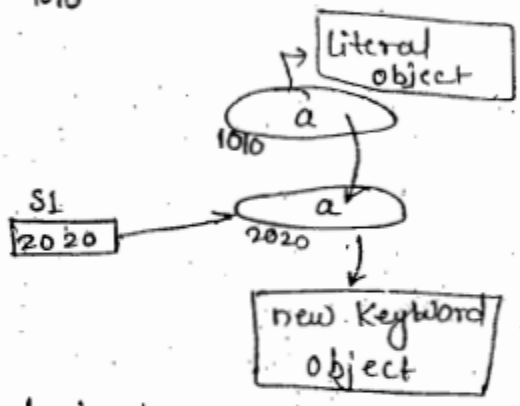
1 object



Approach #2 :

```
String s1 = new String("a");
```

2nd object 1st object



find out how many objects are created in below statements.

	No. of objects
→ String s1 = "a"; ①	1
String s2 = "a";	0
String s3 = new ② String("a");	1
String s4 = new ③ String("a");	1
String s5 = new ⑤ String("b"); ④	2
String s6 = new ⑥ String("b");	1
	6

String Pooling

→ String Pooling means "grouping String Literal Objects, reusing them in the next line of the code without recreating new objects".

→ Advantage of String Pooling :-

- We can avoid creating new String Literal object with same content so that memory & time will be saved, execution is fast.

Note: String Pooling is applied to only literal object, it is not applied to new Keyword object.

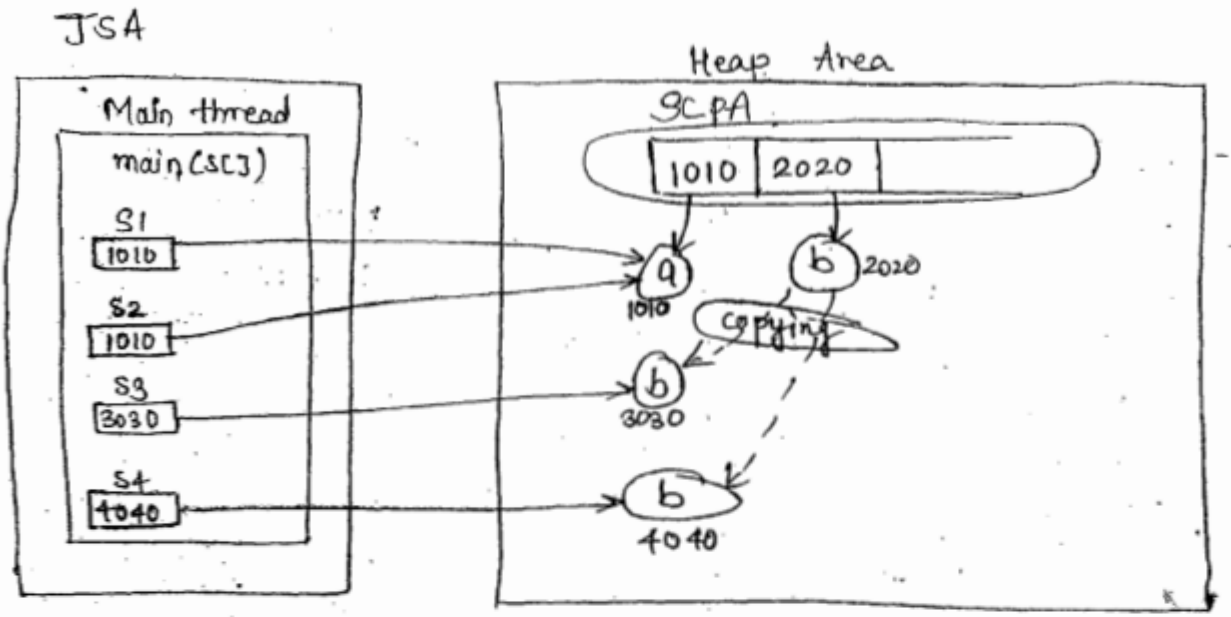
String Constant Pooled Area :-

→ for implementing String pooling, Inside JVM one special object is used. It is called as String Constant pooled Area (SCPA).

→ This object is nothing but Collection Object Vector/Hashtable (I cannot prove it, it is simple common sense. That means as many String Literal objects are creating in the program those many String Literal objects should be stored in SCPA without size limitation. The only object that can store other object without size limitation is collection. The collection available in 1.0 is vector.)

→ Below diagram shows String object creation in two ways, storing them in ~~pot~~ pooled Area.

```
String s1 = "a";
String s2 = "a";
String s3 = new String("b");
String s4 = new String("b");
Sopln(s1==s2);  ⟶ True
Sopln(s3==s4);  ⟶ false
```



→ SCPA is created one per JVM. Then the String Literal Objects created from different classes loaded in this JVM are created in this same SCPA object this JVM.

→ If a String Literal is repeated in different classes, one object created in pool, shared to all these classes.

Below program will prove SCPA is created 1 per JVM.

A.java

```
class A {
    static String s1 = "a";
}
```

B.java

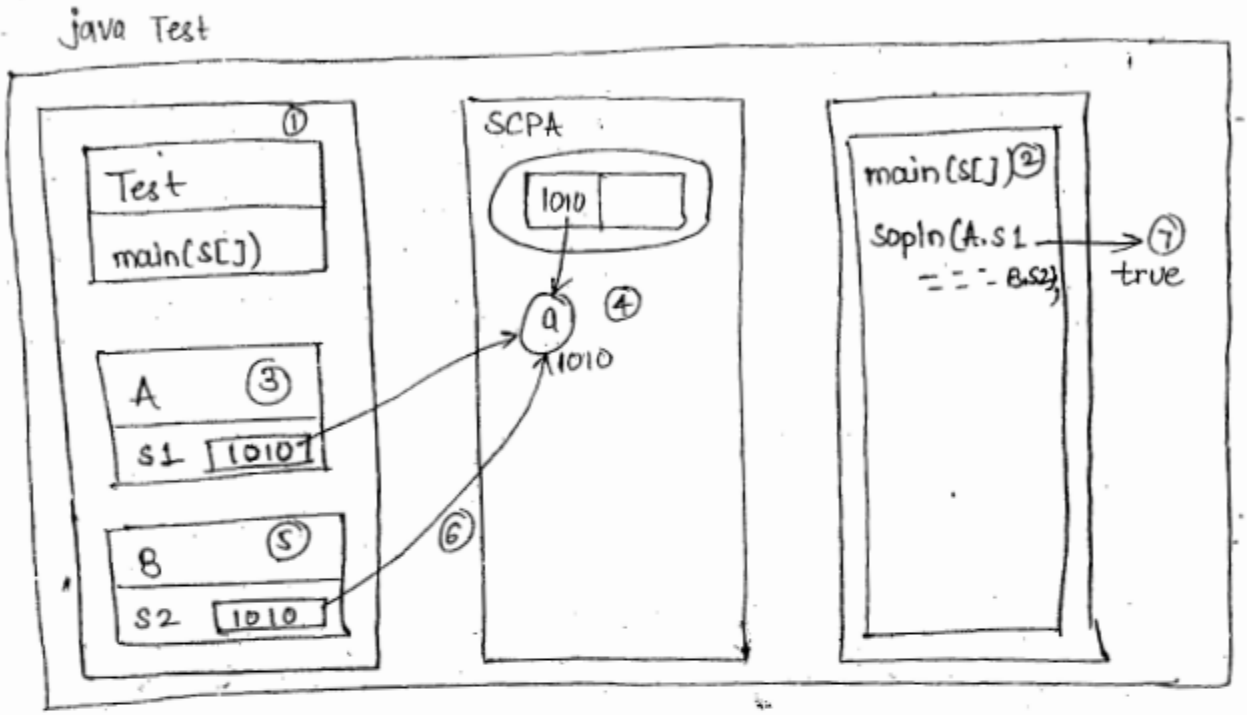
```
class B {
    static String s2 = "a";
}
```

Test.java

```
class Test {
    P S V m(S[] args) {
        Sop ln(A.s1 == B.s2);
    }
}
```

O/p: true

s1 & s2 variables from A & B classes are pointing to same string literal object creating in pool as shown in below diagram.

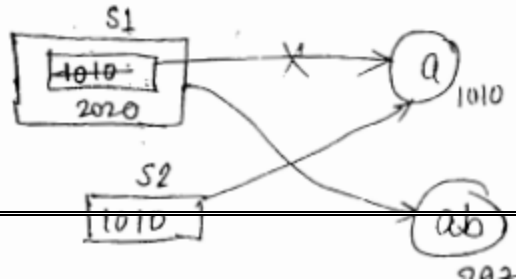


Because of string Pooling String object is pointing by multiple variables then d'nt we have problem when we modify String object using one variable?

→ No, we d'nt have problem because string is immutable object, result is stored in new string object then with another referenced variable we will get original value.

for e.g

```
String S1 = "a";
String S2 = "a";
S1 = S1 + "b";
Sopln(S1); ——→ ab
Sopln(S2); ——→ a
```



Why String is Immutable?

→ Because for storing String object as a key in Map Collection

Why pooling is implemented only to String Object?

→ Because String is immutable, the modified result is stored in different object, so their will not be problem in implementing pooling in String object.

→ If pooling is implemented for StringBuffer or StringBuilder we will have problem as they are mutable objects.

→ In pooling a object is pointing by multiple variables then if modified result is stored in same object other variables will get modified result not original values. So pooling should be implemented only for immutable objects.

Garbage Collection in String Objects?

→ Literal String objects are not eligible for Garbage Collection because it is referencing from SCPA always.

→ String object created using new Keyword & constructor it is eligible for garbage Collection. when it is unreferenced, because it is not pointing by SCPA.

→ Below program proves literal object is not eligible for GC. In the below prg we have displayed reference of String literal object before & after GC. The same value is displayed before & after GC this means String literal is not eligible for GC.

```
//TestGC.java

class TestGC{
    p s v m(S[] args){
        String s1="a";
        Sopln(System.identityHashCode(s1));
        s1=null;
```

→ Unreferencing String Literal object from prg.

System.gc(); → Requesting JVM to start GC

```
try {
   Thread.sleep(10000);
}
Catch(InterruptedException e){}
```

Passing main thread to allow GC to execute.

```
String s2 = "a";
```

∵ Literal is not GC same the previous object reference is assigned, same reference is displayed.

```
Sopln(System.identityHashCode(s2));
}
}
```

O/P: javac TestGC.java

java TestGC

21559264 → Reference Value

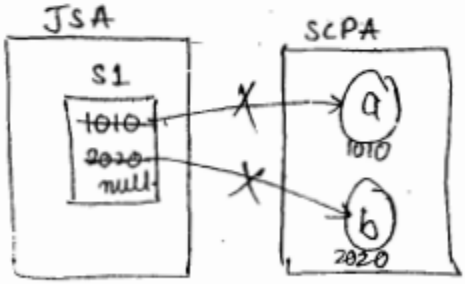
21559264

Is String reference Variable Immutable?

→ No, String referenced variable is mutable that means we can assign another string object reference or null.

for e.g

```
String s1 = "a";
       s2 = "b";
       s1 = null;
```

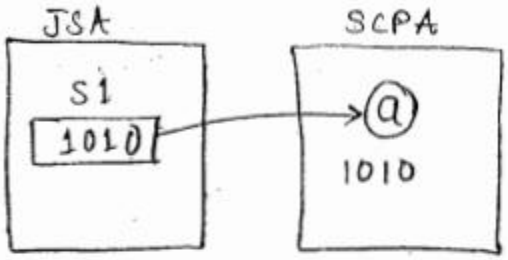


How can we make String Variable also Immutable?

→ By declaring it as final

Ex:

```
final String s1 = "a";
    // s2 = "b"; //CE
    // s1 = null; //CE
```



→ String is Immutable so the modifications we are doing on string data result is not stored in current object for every modification result is stored in new object.

→ String class has a several method for verifying string data, retrieving characters from string object, modifying characters in string object.

→ We have four methods for modifying string data these four methods internally creates new string object with a result string data return this new string object reference.

→ This methods will never modify data in current object.

Those methods are

1. For Concatenation

   public String concate (String s)

2. For changing case

   public string toLowerCase()

   public string toUpperCase()

3. For replacing char/substring

replace all occurances
- public string replace (char oldchar, char newchar)
- public string replace (CharSequence oldString, CharSequence newString)
- public string replaceAll (String oldString, String newString)

replacing only first occurance
- public string replaceFirst (String os, String ns)

4. For removing begin & end spaces

   (public String trim()

Q:- What is immutable and mutable object?

→ A object which doesn't allow modifications on its data and if it allows modification but result is stored in different new object not in the current object memory is called immutable object.

→ A mutable object allows modifications to be stored in current object memory itself. ~~The predefined immuta~~

→ The predefined immutable objects are string and all wrapper classes.

→ String allows modifications but stores result in new object.

→ Wrapper classes doesn't allow modification.

→ StringBuffer and StringBuilder are mutable objects.

Q:- Can we develop custom immutable object?

→ Yes, we can develop.

Procedure:

i) Define a class by declaring all its variable as private.

ii) Define a parameterized constructor for initializing object variable at least once.

iii) Don't define setter method, so data will not be modified. If we want to allow modification define setter method but store ~~obj~~ result in new object.

iv) Define getter method to read values.

Below class shows developing immutable object that doesn't allow modifications.

```
class Example {                    class Test {
    Private int a;
```

```
public Example (int x) {
        this.x = x;
    }
    public int getX() {
        return x;
    }
}
```

x 50

1010

e 1010

```
Example e = new Example(50);
e.x = 7 ; XCE: variable has private access
System.out.println(e.getX());
}
}
```

Below class shows developing Immutable object by allowing modifications but result is stored in new object.

~~class E~~

```
class Sample {
    private int x;
    public sample () { }
    public sample(int x) {
        this.x = x;
    }
    public Sample SetX(int x) {
        Sample s = new Sample();
        s.x = x;
        return s;
    }
    public int getX() {
        return x;
    }
}
```

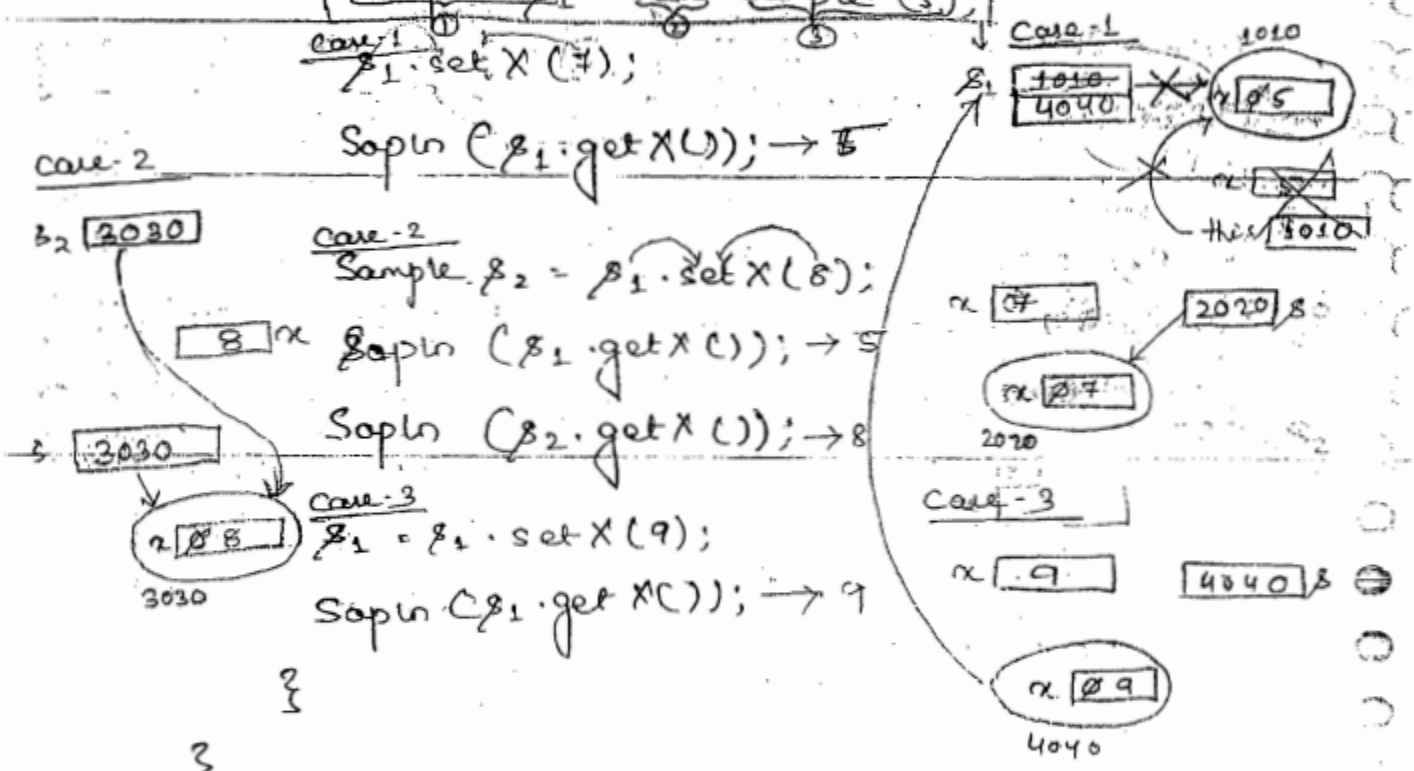
```
class Test {
    p s v main (String[] args)
        Sample s1 = new Sample(50)
        Sample s2 = s1.getX(7);
        System.out.println(s1.getX());
        System.out.println(s2.getX());
    }
}
```

→ For comparision all wrapper classes are developed by Example class (No setter method)

→ Like Sample class String class also contains no-arg constructor to create empty string object.

Q:- What is the output we get from below lines of code?

```
class Test {
    public static void main (String [] args {
        Sample s1 = new Sample (5);
        s1.setX (7);
        Sopln (s1.getX());
        Sample s2 = s1.setX(8);
        Sopln (s1.getX());
        Sopln (s2.getX());
        s1 = s1.setX(9);
        Sopln (s1.getX());
    }
}
```



→ In above program 5 methods are executing.

i) main ()
ii) Sample (5)
iii) setX ()
iv) Sample ()
v) getX ()

→ So 5 stack frames are created.

s1.setX(7) call :

→ Here setX(7) method stack frame is created.

→ Parameter 'x' variable created with arguement 7.

→ Local object created using no-arg constructor. As part of

object is created with 'x' variable with default value.

→ Storing arguement value in this local object.

→ Returning local object reference to main() method and s_1.setX is replaced with this reference 2020.

→ Since we are not storing this reference in any destination var the object is lost. Finally conclusion is current object s_1 is not modified.

→ So s_1.getX() method return 5.

<u>Conclusion</u> :-

→ If we hold the returned object we can have original valu and also modified value.

→ In this case we assign returned object reference in the current pointing object referenced variable.

→ Now it is pointing to new object and old object is unreferenc hence we lost old data.

String Handling methods

13/8/15

String

→ for storing string object in JVM we must create string type referenced variable

→ String class internally uses char[] object for storing string literal characters.

→ So, the characters available in string literal will contain index starts zero.

String Handling

→ Below are the operations we can perform on string data & their appropriate methods.

String class methods

01. Find String is empty or not

```
public boolean isEmpty()
```

- This method returns true if string doesn't have characters
- Return: false if atleast 1 character available in this string object.

02. Finding string length

```
public int lenght()
```

- It returns no. of characters available in this string.
- returns zero if no character available.

03. Displaying String Characters (content/data)

```
public String toString()
```

- This method is overriden is string class for returning current string object's content to display on console.

04. Retrieving hashcode of a String Object

```
public int hashCode()
```

- This method is overriden in string class for generating hashCode based on ~~its~~ string content.

05. Comparing two string objects for equality

`public boolean equals(Object o)`

- It also overriden in string class for comparing two string objects with their content.
- It compares characters in both string objects by considering their case.

`public boolean equalsIgnoreCase(String s)`

- This method specially defined in string class for comparing two string objects with their content, without considering case.

06. Comparing two string objects lexicographically
- means comparing two string objects & returning their difference in no.

`public int compareTo(String s)`

- It is an implementing method of comparable interface.
- It compares two string content by considering case. returns ~~the~~ a no. representing diff. b/w the two string.

`public int compareToIgnoreCase(String s)`

- It is string class specially defined method for comparing content of two string objects ~~content~~ without considering case.

07. Retrieving a character from the given index

`public char charAt(int index)`

- This method returns a character from the given index
- If the given index not found, this method throws StringIndexOutOfBoundsException.

08. Finding a index of a given character / substring

```
public int indexOf(char ch)
public int indexOf(String s)
```

→ These methods return the first occurance of given of char / string.

```
public int LastIndexOf(char ch)
public int LastIndexOf(String s)
```

→ These methods will return last occurance of given char / String.

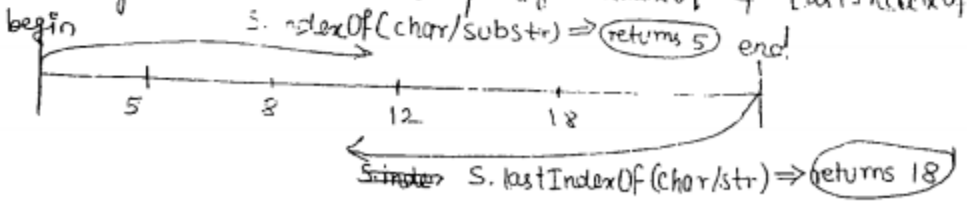
```
public int ~~LastIndex~~Index Of(char ch, int fromIndex)
public int indexOf(String s, int fromIndex)
```

```
public int LastIndexOf(char ch, int fromIndex)
public int LastIndexOf(String s, int fromIndex)
```

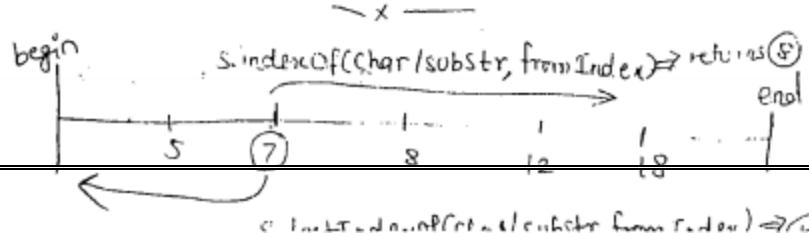
→ These two methods will return the first occurance of given char or substring after the given from index inclusive.

→ These two methods will return index of given char or substring before the given from index first occurance.

Working ~~Index~~ Functionality of IndexOf & LastIndexOf()



— x —



09. Searching character / Substring in this given String in same sequence.

```
public boolean contains(String s)
```

10. Verifying ~~given~~ this string startsWith given substring

```
public boolean startsWith(String s)
```

11. Verifying this String endsWith given substring

```
public boolean endsWith(String s)
```

12. Retrieving substring from this String

```
public String substring(int fromIndex)
public String substring(int fromIndex, int endIndex)
```

→ In the returning substring fromIndex character is included endIndex character is not included.

Rule:- The given Index should be b/w 0, s.length()-1 & also fromIndex should be less than or equals to endIndex

Rule: If fromIndex & endIndex are equal than empty string will be returned bcoz endIndex character is exclusive.

13. Converting primitive value or object into String

```
public static String valueOf(xxx values)
```

int double char[] Object ...

Note:- If you pass null directly to this method it matched to char[] parameter method, then it internally throws ~~NE~~ NPE.

Sopln(String.valueOf(null)); ✓ NOCE ✗ RE:NPE

If we pass null with other referenced type cast operator it is matched with Object parameter method, it doesn't throws NPE, directly null is displayed

for e.g.

Sopln(String.valueOf((String)null)); $\xrightarrow{op}$ null

Sopln(String.valueOf((Integer)null)); ⟶ null

Eg:-

```
class Student {
    int sno;
    public String toString() {
        return String.valueOf(eno);
    }
}

class Test {
    p s v m(S[] args) {
        Student s1 = new Student();
        Sopln(s1);
    }
}
```

② ③ ④ "101" ⑤ "101"

14. Converting String to char[] (charArray)

```
public char[] toCharArray()
```

15. Converting String to byteArray

```
public byte[] getBytes()
```

Example for charArray

```
interface A {
    char[] getFUS(int uid);
}

class B implements A {
    public char[] getFUS(int uid) {
        String s = ____;
        return (s.toCharArray());
    }
}
```

NIT

uid = 1

UID = 1

NIT

Uid	Uname	Club
1	K	NIT
2	BB	Acting

→ So far we have learnt methods for either retriving characters or for Comparing characters.

— x —

→ Below methods are meant for modifying string data.

16. Concatenating two string characters

```
public String concat(String s)
```

17. Changing characters case in of this String.

```
public String toUpperCase()
public String toLowerCase()
```

18. Replacing char/substring with new char/substring

```
public String replace(char oldChar, char newChar)
public String replace(CharSequence oldstr, CharSequence newstr)
public String replaceAll(String oldstr, String newstr)
public String replaceFirst(String oldstr, String newstr)
```

19. Removing began & end spaces (trim)

```
public String trim()
```

String Operations performed by using String class Method:-

→ Sun Microsystem defined multiple methods in String class to perform different operations on String data.

Q:- Find how many characters available in the below String or what is the output from the below program:

```
class StringLength{
    public static void main(String [] args){
        String s1 = args[0];
        System.out.println(s1.length());
```

> javac StringLength.java

> java StringLength TC HariKrishna

1. 4
2. 11 → Complete string is read by args[0] as there is
3. 12 no space in middle
4. 7

> java StringLength TC Hari Krishna

1. 4
2. 11
3. 12
4. 7

Q:- If we use ~~BufferReaders~~ br.ReadLine() to reade String data from keyboard, what is the output in the above two cases?

→ Case-1:- O/p - 11 Because br.readLine() reads complete line

→ Case-2:- O/p - 12

Q:- If we use scn.nextLine() to read string data from keyboard, what is the output in the above two cases?

→ Case-1:- o/p - 11

Case-2:- o/p - 12

Because scn.nextLine() read complete line

Q:- If we use scn.next() to read data from keyboard, what is the output in the above two cases?

→ Case-1:- O/p - 11

Case-2:- O/p - 4

→ If we call scn.next() method again in the same JVM, reads next tocken(word) in the same string. If all words are read, then it prompts cursor for entering new string.

Q:- What is the difference between length property and length paranthesis method?

→ Length property is used for finding length of the Array Object means no. of elements stored in array.

→ Length paranthesis method is used for finding Length of the String Object means no. of characters available in the String

→ We cannot use them vice-versa. It leads to CE: cfs.

Q: Identify CE in the below lines?

```
String s2 = "abc";
System.out.println(s2.length()); O/P - 3
System.out.println(s2.length); × CE: cfs
String[] sa = new String[10];
System.out.println(sa.length()); × CE: cfs
System.out.println(sa.length); O/P - 10
```

Limitations Of Array Object:-

→ String Limitation - String is immutable. So for every modificatn of current object, a new string object is created with result and returned.

→ String class has 4 mutator operation methods.

i) concat()

ii) toLowerCase()

→ So as a result of above methods if current String Object data is modified, result is not stored in the current object, ~~a~~ new String Object is created and returned with the result. If the current String Object data is not modified, current String Object reference is returned.

Q:- Assume we called a mutable method on an immutable object and then it definitely return a new object with result. Then what is the output we get in below cases:

- Case # 1: Returned object is not assigned to any variable
- Case # 2: Returned object is assigned to new referenced variable.
- Case # 3: Returned object is assigned to same current object variable

→ Case # 1: Result object is lost, old object is still pointing to current object (old).

→ Case # 2: Result object is hold in the program. So we can use original value and new value both.

→ Case # 3: We can use only new object and old object is lost.

Concatination:-

→ Placing arguement String object characters at end of current String Object characters and storing result in new String Object and returning it's reference is called concatination

→ We can perform concatination operation in 2 ways.

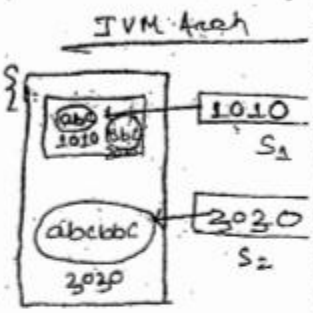
i) using '+' operator

ii) Concation using concat():—

prototype - public String concat (String s)

Write a test program to concat "bbc" with "abc".

```
class TestConcat {
    public static void main (String [] args) {
        String s1 = "abc";
        String s2 = s1.concat ("bbc");
        System.out.println (s1); → o/p - abc
        System.out.println (s2); → o/p - abcbbc
    }
}
```

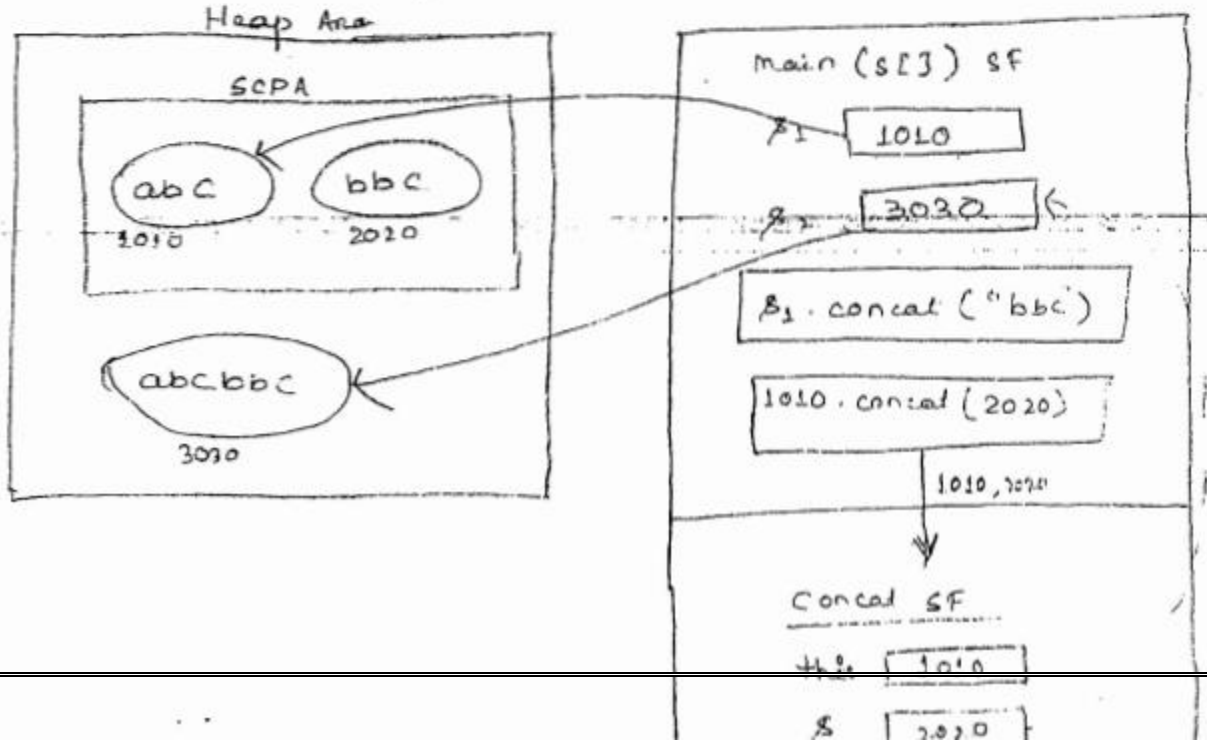


* In the above program 3 objects are created.

i) "abc" in p.SCPA.

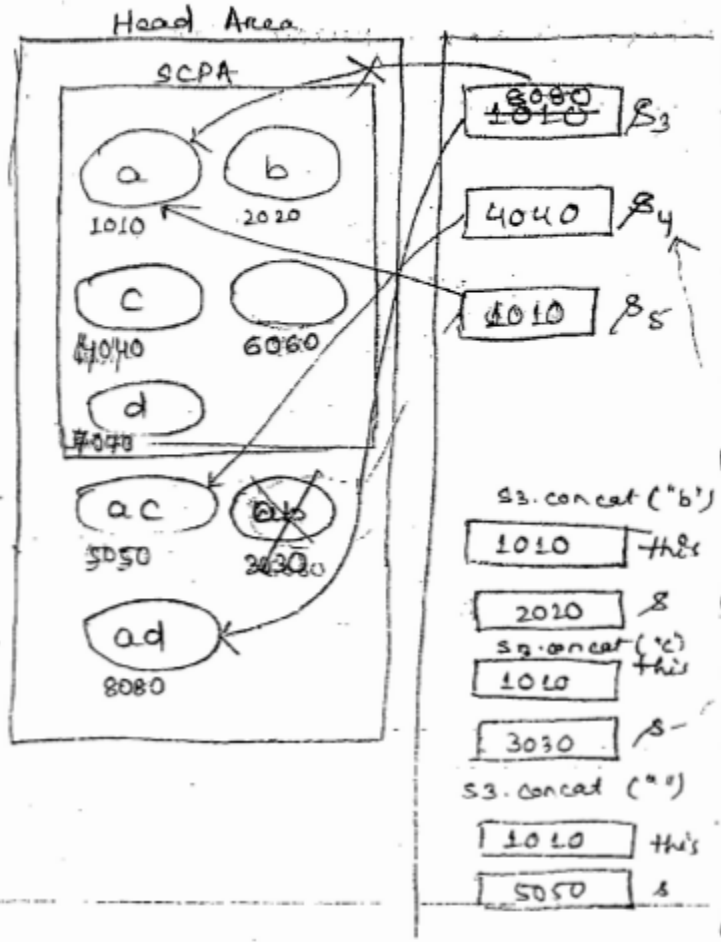
ii) arguement String object "bbc" in SCPA.

iii) Result String object "abcbbc" in head area.



Written Test Questions

```
String s3 = "a";
s3.concat("b");
Sopln(s3); → a
String s4 = s3.concat("c");
Sopln(s3); → a
Sopln(s4); → ac
Sopln(s3 == s4); // false
String s5 = s3.concat("");
Sopln(s3); → a
Sopln(s5); → a
Sopln(s3 == s5); → // true
s3 = s3.concat("d");
Sopln(s3); → ad
```



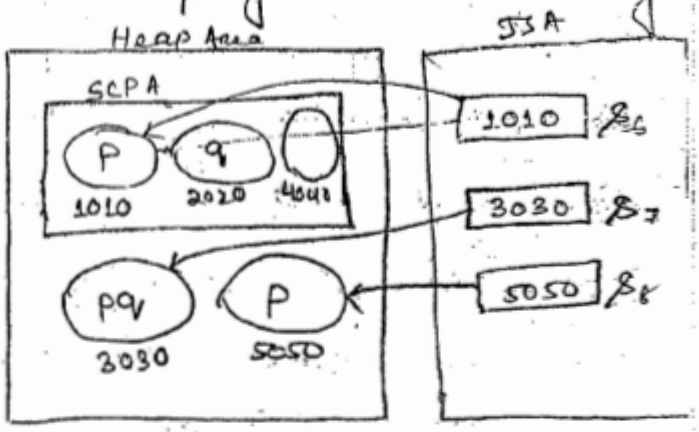
* Total 8 objects are created in the above program.
  - Here 2 objects created b and ab.
  - Here 2 objects created c and ac.
  - Here 1 object created "".
  - Here 2 objects created d and ad

Concatination with '+' operator :=

→ Unlike concat() method, '+' operator always returns new object irrespective of current object is modified or not. Because we don't have current String object and arguement String Object in '+' operator to no length calculations.

What is the output from the below program and how many objects are created?

```
String s6 = "p" ; →
String s7 = s6 + "q" ;
String s8 = s6 + " " ;
Sopln (s6);
Sopln (s7);
Sopln (s8);
Sopln (s6 == s7);
Sopln (s6 == s8);
```



→ If we want to concat only two strings we must use concat

→ If we want to concat more than two string in single line it is recommended to use '+' operator, using concat() method is not readable.

⇒ For example:

```
String s9 = s1 + s2 + s3 + s4 ;
String s10 = s1.concat(s2).concat(s3).concat(s4);
Sopln (s9);
Sopln (s10);
```

→ Method call chaining

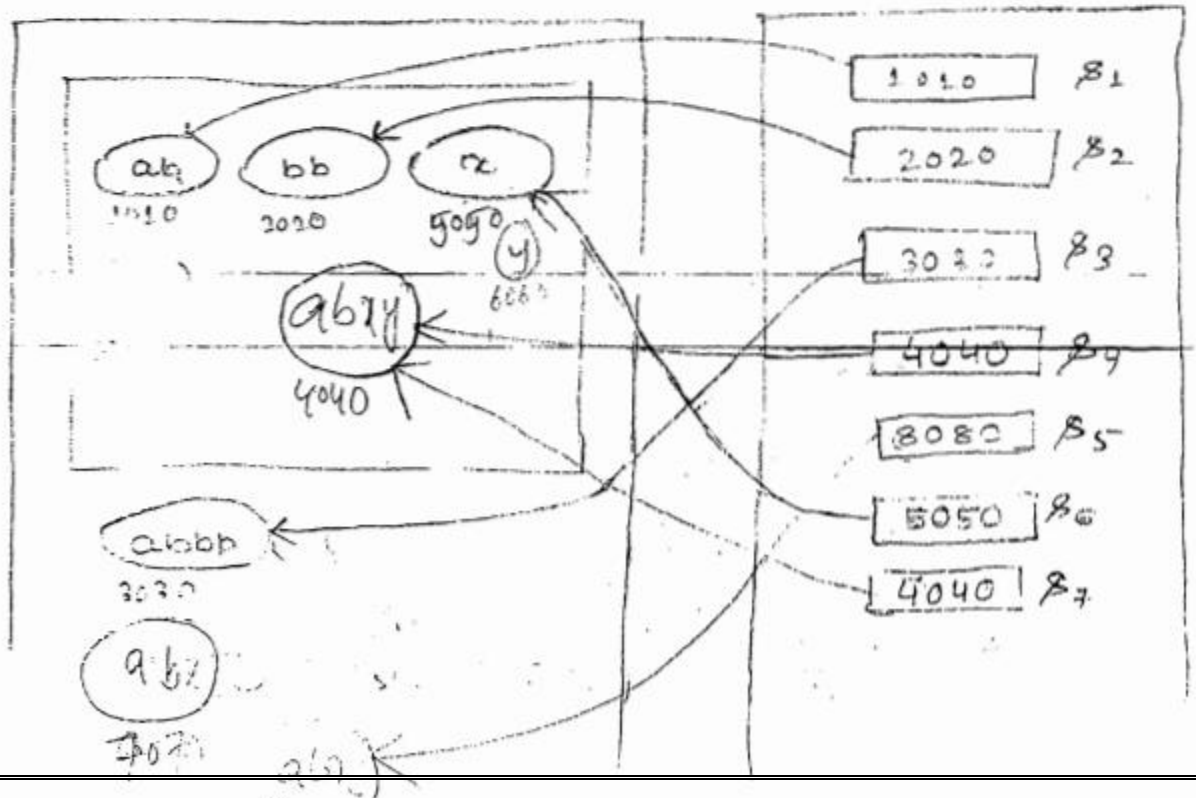
⇒ When we call a method with a method we should search for the called method in the first method return type class. And it is executed with first method returned object data.

⇒ In concatination operation using '+' operator if an expression contains literals only then it is evaluated by compiler and replace expression with the result string. Then JVM creates only one string object with the result data.

→ If an expression contains variable then that expression is calculated by JVM and it creates every string literal as object and then it evaluates.

How many objects are created in the below program?

```
class StringLiteralExprTest {
    public static void main (String [] args){
        String s1 = "ab";          ⟶   1
        String s2 = "bb";          ⟶   1
        String s3 = s1 + s2;       ⟶   1
        String s4 = "ab" + 'x' + 'y';  ⟶   1
        String s5 = s1 + 'x' + 'y';    ⟶   3
        String s6 = "x";           ⟶   0
        String s7 = "abxy";        ⟶   1
```



Write a program to read first name and last name of the customer and print his fullname on the console with wish "Hi".

Enter firstname : Hari

Enter Last name : Krishna

O/P - Hi HariKrishna

Changing case Of the String

// public String toLowerCase ()

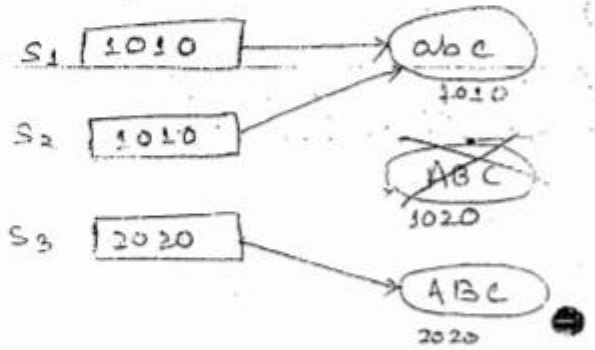
// public String toUpperCase ()

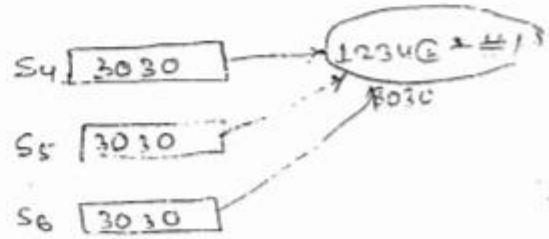
→ Above methods changing the case of alphabets in the current String.

→ If it has only numbers or special characters there is no effect in calling these methods i.e. no change in the current String. As a result of these method calls if at least one character case is changed result String is returned with new object else return same object.

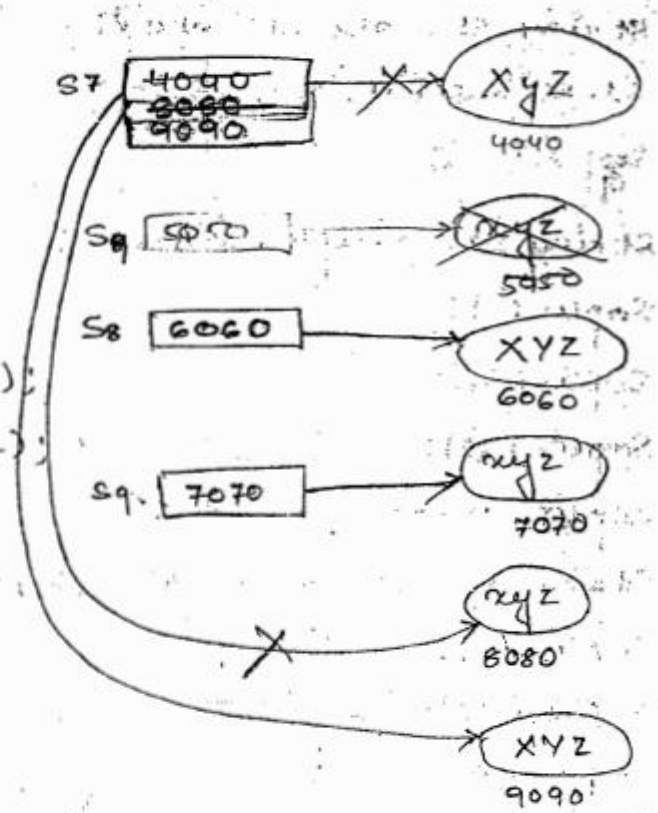
→ Below program shows the change in case of the String

```
String S1 = "abc";
S1.toUpperCase();
Sopln(S1); → abc

String S2 = S1.toLowerCase();
String S3 = S1.toUpperCase();
Sopln();
Sopln(S1); → abc
Sopln(S2); → abc
Sopln(S3); → ABC
Sopln(S1 == S2); // true
Sopln(S1 == S3); // false

String S4 = "1234@*#!";
String S5 = S4.toLowerCase();
String S6 = S4.toUpperCase();
Sopln();
```





```
Sopln(s5 == s6);   // true
String s7 = "XyZ";
Sopln();
Sopln(s7.toLowerCase());
Sopln(s7);  → XyZ
String s8 = s7.toUpperCase();
String s9 = s7.toLowerCase();
Sopln(s7);  → XyZ
Sopln(s8);  → XYZ
Sopln(s9);  → xyz
s7 = s7.toLowerCase();
Sopln(s7);  → xyz
Sopln(s7 = s7.toUpperCase());  → XYZ
Sopln(s7);  → XYZ
```



→ Here s7 pointing objects all characters are changed to UpperCase and stores its reference in s7 variable and then printed. So in the next line we got same output.

Replacing Character or Substring in Current String :—

→ If current String is modified all below methods returns new String object with result.

Prototype - public String replace(char oldChar, char newChar)
public String replace(CharSequence oldString, CharSequence newString)
public String replaceAll(String regex, String replacement)
public String replaceFirst(String regex, String replacement)

```
String s1 = "abc abc abc";
s1.replace('a','b');
Sopln(s1); → abc abc abc
String s2 = s1.replace('A','b');
Sopln();
Sopln(s1); → abc abc abc
Sopln(s2); → abc abc abc
sopln(s1==s2); // true
String s3 = s1.replace("ab","cb");
Sopln();
Sopln(s1); → abc abc abc
Sopln(s3); → cbc cbc cbc
Sopln(s1==s3); // false
String s4 = s3.replaceFirst("cb","bb");
Sopln();
Sopln(s4); → bbc bbc bbc
Sopln(s3); → cbc cbc cbc
```

(doesn't replace 'a' in s1 pointing object)

s1 [1010] → abc abc abc (1010)

~~bbc bbc bbc (2020)~~

s2 [1010]

(java is case sensitive. So a ≠ A)

s3 [3030] → cbc cbc cbc (3030)

Replaces all the occurences of "ab"

s4 [4040] → bbc cbc cbc (4040)

Replaces only the first occurence of "cb".

Q:- Create a program with string object "Java". Replace its characters in below order and print result.

"Java" → Kava → Kaka → Keka

```
→ class ReplaceCharacterTest
    public static void main (String[] args){
        String s1 = "Java"
        String s2 = s1.replace('J', 'K');
        System.out.println(s2);
```

```
        String s3 = s2.replace('v', 'k');
        System.out.println(s3);
        String s4 = s3.replaceFirst("a", "e");
        System.out.println(s4);
    }
}
```

Trim() :—

→ It is used for removing leading and trailing spaces. (beg and end spaces).

prototype :— public string trim().

```
String s1 = "  a b c  a b c  a b c  ";
Sopln(s1.length()); → 16
```

s1 [1010] → " a b c a b c a b c " 1010

```
String s2 = s1.trim();
Sopln();
Sopln(s1.length()); → 16
Sopln(s2.length(); → 11
```

s2 [2020] → "a b c a b c a b c" 2020

```
String s3 = "abc  a b c  ";
s3.trim();
Sopln(s3.length()); → 9
Sopln();
String s4 = s3.trim();
Sopln(s3.length()); → 9
Sopln(s4.length()); → 7
Sopln(s3==s4); // false
```

s3 [3030] → "a b c a b c " 3030

~~a b c a b c~~ 4040

s4 [5050] → "a b c a b c" 5050

s5 [5050]

```
String s5 = s4.trim();
Sopln (s4.length); → 7
Sopln (s5.length); → 7
Sopln (s4 == s5); // true
```

→ Trim() removes only begin and end space.

So how can we remove the middle spaces?

⇒ We must use replace() method to remove middle spaces.

```
String s3 = s1.replace(" ", "");
String s4 = s1.replace("  ", " ");
Sopln();
Sopln (s1.length()); → 16
Sopln (s3.length()); → 9
Sopln (s4.length()); → 12
```

s_3 [6060] → a b c a b c a b c

6060

Importance of trim() methods:

* In projects when we are reading the data from Html form text fields, we must read the data by removing leading and trailing spaces. Here user enter unknownly. And then further the result data is stored in database.

Ex:- Reg.html

first Name [Hari]
Last Name [Krishna]
User Name [gravaharikrishna]
password [Naresh&t]
email [harikrishna.hk@gmail.com]

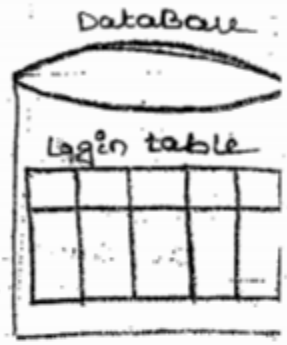
[Submit]

Reg

Servlet

```
String fname = req.getParameter("fn").trim();
String lname = req.getParameter("ln").trim();
String usn   = req.getParameter("usn").trim();
String pwd   = req.getParameter("pwd").trim();
String email = req.getParameter("email").trim();

JDBC
```

text fields name given in Reg.html file



Equals and EqualsIgnoreCase :=

→ In projects username are compared without case and password are compared with case.

→ So we must use equalsIgnoreCase() method to compare username and equals() method to compare passwords.

CompareTo() Algorithm :=

→ compareTo() compares between two string by subtracting each character in both string and returns the ASCII number difference.

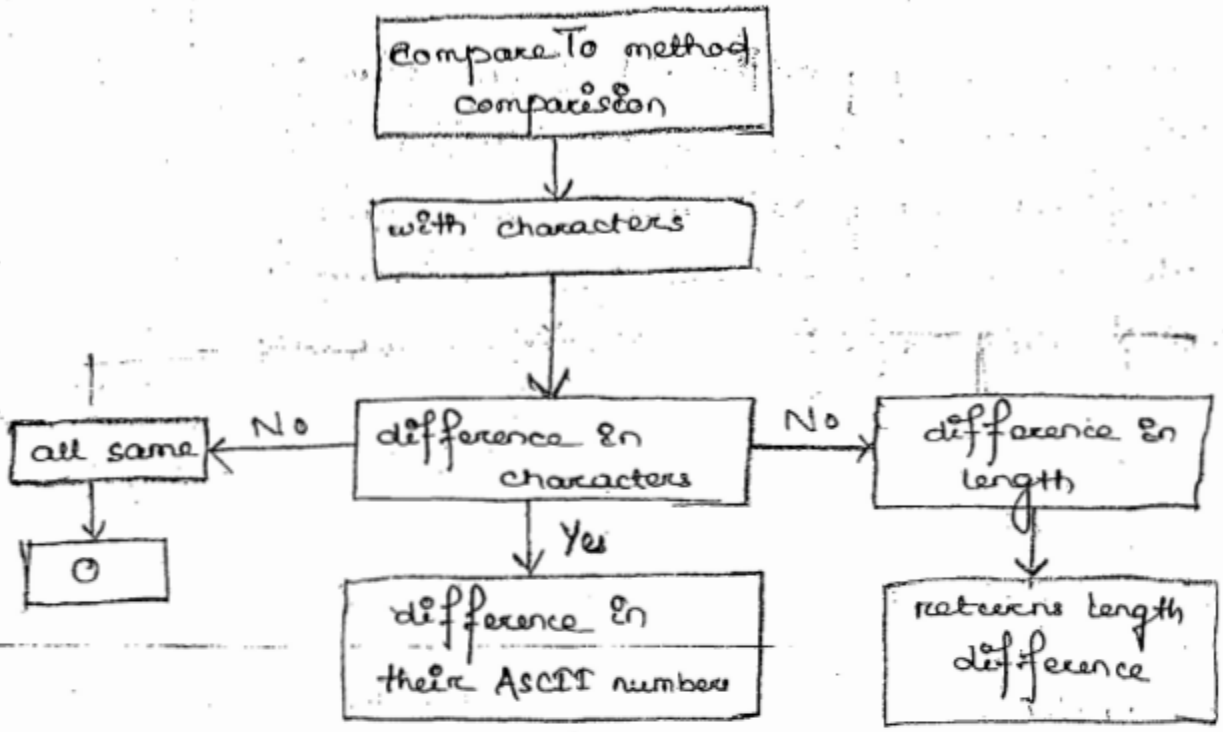
→ It continues this ~~subtraction~~ comparision till non-zero ~~not~~ number occured.

→ If no. of characters in both string are

→ All characters in case equal but no. of characters is different it returns, current string object.length − argument String object length

→ Equals() method internal logic uses compareTo() method.

→ In projects equals() method is used for checking purpose, given input is right or wrong.



Char charAt (int index) :—

prototype : public string char charAt (int index);

```
String s1 = "abcd";
Sopln(s1.charAt(0));
Sopln (s1.charAt(2));
Sopln (s1.charAt (4));
```

Q:- Write a program to read a string from the keyboard and print each character with its position as i/p "abc".

o/p - character 1 : a
2 : b
3 : c

→ import java.util.*;

```
class CharacterReader {
    public static void main (string [] args) {..
        Scanner scn = new Scanner ( System.in);
```

```
Sop(" Enter String: ");
String data = scn.nextLine();
int no of chars = data.length();
for (int i=0 ; i< no of chars ; i++)
{
Sop("character" + (i+1) + " is : " + data.charAt[i]);
}
}
```

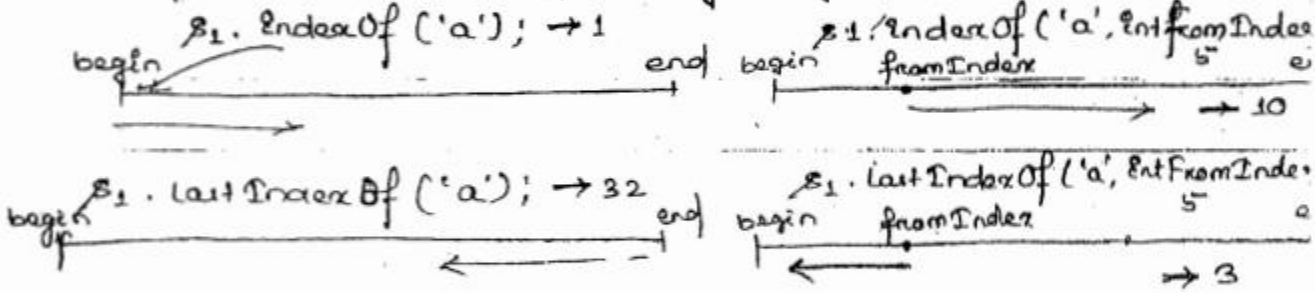
IndexOf and LastIndexOf () methods :—

→ IndexOf () is used for finding the position of given character of substring from begin or 0 index () and returns 1st occurance.

→ LastIndexOf () returns the last occurance of the given character or substring

→ Also we have overloaded forms of these two methods to find the position of character or substring from begin to end index.

```
String s1 = "Java programming language";
```



prototypes :

```
public int IndexOf (int ch)
public int IndexOf (String s)
public int LastIndexOf (int ch)
public int LastIndexOf (String s)
```

public int lastIndexOf (int ch, int fromIndex)
public int lastIndexOf (String s, int fromIndex)

→ Above methods return '-1' if the given char or string is not found in the given string.

→ Note that both methods searching for given char or substring in the given case.

→ If we want to search for the string or character without case we must convert given string all characters to Lower case and then we have to search in the modified string.

Ex :- String s2 = "javaHari";
Sop ln (s2.indexOf("hari"));
String s3 = s2.toLowerCase
Sop ln (s3.indexOf("hari"));

→ Above methods return '-1' if it doesn't find any given character or string what we should do if we want to boolean value true/false.

*→ We should write our own code as shown below :-

Q:- Write a program to read string find "Hari" substring available in the string, if it is available true else returns false. In main method print message ("Hari is available") if it returns true, else print message "Hari is not available".

```
import java.util.*;
class TestIndexOf {
    static boolean searchIn (String s) {
        if (s.indexOf("Hari") != -1) {
            return true;
        }
        else {
            return false;
        }
```

```
    return (s.indexOf("Hari") != -1);
  }
public static void main (String [] args){
    Scanner scn = new Scanner (System.in);
    Sopln ("Enter String : ");
    String s1 = scn.nextLine();
    boolean bo = searchIn(s1);
    if (bo == true){
        System.out.println("Hari is available");
    }
    else {
        System.out.println("Hari is not available);
    }
}
}
```

Case-1

i/p - JavaHari

o/p - Hari is available

Case-2

i/p - Javahari

o/p - Hari is not available

Q:- How can we get is available in case-2?

→ We must change the given string to Uppercase in searchIn method as below.

```
static boolean searchIn(String s){
    s = s.toUpperCase();
    return(s.indexOf("Hari") != -1);
}
```

→ In Java 5 SunMicroSystem given a special method called contains() in String class for finding substring or character in

prototype:

public boolean contains (CharSequence s)

→ In our searchIn method arguement is the actual String and substring we hard coded in this method.

→ In our contains method arguement is the substring for which we have to search and actual

Rule:

→ ~~contains()~~ We cannot call this method by passing char in ' ' because it's parameter is charSequence. So we must pass character with " ".

Ex: String s1 = "Java Programming Language";

Sopln (s1.contains('a')); // false

Sopln (s1.contains("a")); // true

StartsWith() and endsWith():—

Q:- Write a program to count how many text files available in a given array?

→
```
class TestEndsWith {
    public static void main(String[] args){
    String[] filesArray = {"1.txt","2.doc","3.txt","4.pdf",
    "5.xls","6.txt"};
    for(int i=0; i<filesArray.length; i++){
        if(filesArray[i].endsWith(".txt")){
            count++;
        }
    }
    System.out.println(count+" text files are available");
}
```

prototype : public boolean endsWith(String s) public boolean startsWith(String s)

Ex:—
```
String s1 = "abc bbc cbc";
Sopln(s1.startsWith("abc")); // true
Sopln(s1.endsWith("cbc")); // true
Sopln(s1.endsWith("bbc cbc")); // true
Sopln(s1.startsWith("abc bbc cbc")); // true
Sopln(s1.endsWith("abc bbc cbc")); // true
Sopln(s1.startsWith("bbc cbc")); // false
```

Difference between equals(), contains(), startsWith() and endsWith():—

→ equals() method compares current String character and arguement string character completely. It returns true if all are equal.

→ contains() method search for given String as substrin

in current String. It returns true if its characters are available in sequence in the current String at any position.

→ startsWith() and endsWith() methods are also searching given String as subString.

→ startsWith() returns true if this given String characters are available in the begin index.

→ endsWith() returns true if the given String characters are available at the end.

Substring():—

prototype : public String substring(int index) public String substring(int start, int end)

Ex:—

```
String s1 = "Java Programming Language"
Sopln(s1.substring(5)); // Program
Sopln(s1.substring(5,12)); // Program
Sopln(s1.substring(5,11)); // Progra
Sopln(s1.substring(5,5)); // " "
Sopln(s1.substring(12,5)); // CE:
String s2 = "Java Object-oriented Programming Language";
int startIndex = s2.indexOf("Program");
int endIndex = startIndex + 7;
```

Q:— Write a program to check 'Hari' substring is available in given String.

If available print "Hari" word in the given case with

→ ~~class~~

```
import java.util.*;
class TestSubString {
    public static void main (String[] args) {
        Scanner scn = new Scanner(System.in);
        while (true) {
            System.out.print("\n Enter String : ");
            String s1 = scn.nextLine();
            String s2 = s1.toLowerCase();
            boolean available = s2.contains("Hari");
            if (available) {
                int startIndex = s2.indexOf("Hari");
                int endIndex = startIndex + 4;
                String Hari = s1.substring(startIndex, endIndex);
                System.out.println(Hari + " is available");
            }
            else {
                System.out.println("Hari is not available");
            }
        }
    }
}
```

Enter String : JavaHariKrishna
Hari is available.
Enter String : JavaHariKrishna
Hari is available
Enter String : Hari
Hari is available
Enter String : HaRiKrishna
HaRi is available

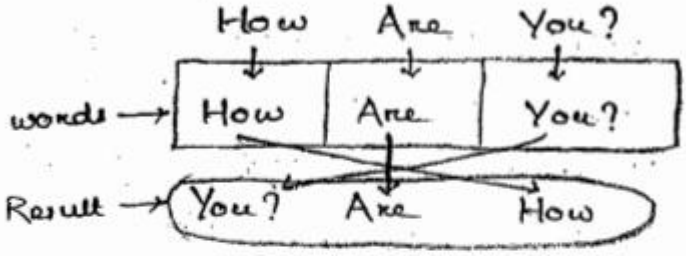
split() :—

Q:— Write a program to reverse words in given String.

i/p - How Are You?

o/p - You? Are How

A Procedure:




```
class StringReverser {
    static String reverseWords (String s){
        String [] words = s.split (" ");
        String result = " ";
        for (int i = (words.length)-1) ; i>=0; i--){
            result = result + words [i] + " ";
        }
    return result.trim ();
}

class TestSubString {
    public static void main (String [] args){
        Scanner scn = new Scanner (System.in);
        System.out.println ("/n Enter String:");
        String s1 = scn.nextLine ();
        System.out.println ("You entered:" + s1);
        String result = StringReverse.reverseWords (s1);
        System.out.println ("Result:" + result);
    }
}
```

valueOf():-

→ It is a overloaded method used for converting any Java data type value or object to String type.

→ This method is used internally by compiler and JVM.

→ If we concat any other data type value with the string using '+' operator.

Prototype: public static String valueOf(xxx type)

(Any Java data type is allowed)

→ It is a overloaded method has 9 forms with parameter int, long, float, double, char, char[], boolean, (char[], int, int), object.

→ Byte and short parameter methods are not there because there is no literals subtype byte and short.

→ Ex:-

```
Sopln("data:" + 10);
```

10 is converted as string by compiler using valueOf() method as shown below.

```
Sopln("data:" + String.valueOf(10));
```

→ Sopln("data:" + "10");

→ Sopln("data: 10");

→ data: 10

String Handling using StringBuffer Object:-

→ A buffer means precast memory that means it creates some default capacity memory prior to store

→ StringBuffer object is internally maintains a char[] object by default 16 location.

→ So we can say StringBuffer object default capacity is 16 buffers (i.e. 16 locations).

→ It's capacity is incremented after when it's size is crossing it's capacity.

→ Here size means no. of characters we ~~can~~ have stored (let us say 4 characters).

→ Here capacity means no. of characters we can store (16 by default).

→ StringBuffer class has below 2 methods to find capacity and length.

```
public int capacity()
public int length()
```

For details read String Handling chapter in Vol-2

WAP to read name from the Keyboard. If entered name is empty display error message, prompt message 'Enter Name Again'. If name is entered correctly display message 'Hi name' 'Welcome to NareshIT'

14/7/15

If ~~spaces~~ only space entered also throw Error.

→ To develop this project we must use isEmpty() method & also trim() method from String class.

```
//isEmptyTest.java
import java.util.*;

public class IsEmptyTest{
    p s v m(S[] args){

        Scanner scn = new Scanner(System.in);
        while(true){
            Sopln("\n Enter name");
            String name = scn.nextLine();
            if(name.isEmpty()){
                Sopln("Name is required");
                continue;
            }
            else{
                String resName = name.trim();
                if(resName.isEmpty(){
                    Sopln("Name is required");
                    continue;
                }
                Sopln("Hi"+resName);
                Sopln("Welcome to Naresh IT");
                break;
            }//else
        }//while
```

→ Real time example verifying textfield is empty or not

Project-2

Read password from the Keyboard. Verify its length is b/w 8 to 16 if yes display message 'Registration successful' else throw error 'password length should be b/w '8-16'

```
→ //LengthPasswordTest.java
import java.util.*;

public class LengthPasswordTest {
    p s v m(S[] args){
        Scanner scn = new Scanner(System.in);
        while(true){
        Sopln("Enter Password:");
        String pwd = scn.nextLine();
        if(pwd.isEmpty()){
            Sopln("Password should not be empty");
            Continue;
        }
        else {
            int len = pwd.length();
            if(len<8 || len>16){
                Sopln("Password length should be 8 to 16");
                Continue;
            }
            else{
                Sopln("Registration Completed successfully");
                break;
            }
        }
    }
}
```

Project 1:-

WAP to validate mobile number length. If 10 digits not existed throw error message. In this project we must implement totally 4 validations. 1. Empty or not. 2. All are digits are not

3. Ten digits exists or not 4. Is it really mobile number or not (send OTP) sms

```
// MobileValidator.java
import java.util.*;
public class MobileValidator{
    P S V m(S[] args){
        Scanner scn = new Scanner(System.in);
        while(true){
         Sopln("Enter Number:");
         String mobile = scn.nextline();
         if(mobile.IsEmpty()){
             Sopln("Mobile no. is Mandatory");
             continue;
         }
         else{
             ~~long mn = Long.parseLong(mob~~
             try{
                long mn = Long.parseLong(mobile);
                if(mobile.length()!=10){
                    Sopln("Mobile no. should be 10 digits");
                    continue;
                }
                else{
                    Sopln("Activation Key has send to your no.");
                    Sopln("Enter Activation key:");
                    break;
                }
             Catch(NumberFormatException nfe){
                Sopln("Mobile ~~st~~ no. should contain only digits");
                continue;
             }
         }
        }
    }
```

19/8

Develop a prg to find no. of vowels & consonents available in the given string Read string from Keyboard.

→ // CheckVC.java

Mine Code

```
import java.util.*;
public class CheckVC {

    P S V m(S[] args){
        String S1

    P V checkVC {String S1,
        {
            this.S1 = S1;


        }
        public toString(){
            return
```

```
import java.util.*;
public class VowCons {
    P S V m (S[] arge){
    Scanner Scn = new Scanner (System.in)
        Sopln("Enter a string");
        String s1 = scn.nextline();
        String s2 = s1.tolowerCase();
        int vCount = 0;
        int cCount = 0;
        for(int i=0; i<s2.length(); i++) {
            Char ch = s2.charAt(i);
```

```
if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
        vCount++;
    }
    else {
        cCount++;
    }
}

Sopln ("Available vowels are: " vCount);
Sopln ("Available Consonents are: " cCount);
}
```

Develop a prg to register a user Account in website. In this project you develop code for validating password criteria

1. password should contain atleast 1 uppercase Alphabet.
2. password should contain 1 digit.
3. password should contain 1 special character.
4. password length should be betn 8, 16.

PasswordPatternValidator (Sir)

Mine Code

```
// LoginPage.java
import java.util.*;
public class LoginPage {
    P S V m(S[] args){
        Scanner scn = new Scanner(System.in);
        Sopln (" Enter Username: ");
        while (true){
        String usn = scn.nextLine();

        Sopln(" Enter Password: ");
        String pwd = scn.nextLine();
        if (usn.isEmpty()){
            Sopln(
```

```
if (pwd. isEmpty()) {
    Sopln (
    Continue;
}
else {
if (char ch >65 && ch<90
```

```
//Password PatternValidator.java
import java.util.Scanner;
public class PasswordPatternValidator {
    P s v m(S[] args) {
        Scanner scn = new Scanner(System.in);
        boolean upperfound = false;
        boolean digitfound = false;
        boolean spCharfound = false;

        while(true){
            Sopln("Password:");
            String pwd = scn.nextLine();
            int len = pwd.length();

            if(len <8 || len >16){
                Sopln(" Password should contain 8-16 characters");
            }
            else {
                for(int i=0; i<len; i++){
                    char ch= pwd. charAt(i);
                if(Character. isLetter(ch)){
                    if(Character. isUpperCase(ch)) {
                        upperfound= true;
                }
```

```
// (or) if(Character.isLetter(ch)&&
        Character.isUpperCase(ch)){
```

```
        else {
            spCharfound = true;
        }
    }// for close

    if(upperfound && digitfound && spCharfound) {
            Sopln("Registration success");
            break;
    }
    else {
      Sopln("Password should contain ");
      Sopln("1.one upper case alphabet");
      Sopln(" 2. one digit");
      Sopln(" 3. one special character");
    }
  } // else close
} // while
}// main
}// class
```

~~Index~~

WAP to read a string from the keyboard. If this string contains substring "hari" then display 'Substring hari available' else display 'Substring hari is not available'.

~~WAP~~

Rewrite above program to display ~~the~~ substring hari is available even though characters ~~ha~~ h, a, r, i are available at diff. places in given string.

Rewrite above program substring hari is available even though characters h, a, ri available in diff. places but the character sequence should be 'hari'

Validate the given email is a email or not.

Validation: This email should contain @ & '.' characters, '.' should come after @.

—x—

WAP to read a String from the Keyboard. If this String contains substring "hari" then display 'substring hari available' else display 'substring hari is not available'

```
// SearchString
import java.util.Scanner;
public class SearchString {
    P S V m (S[] args){
      Scanner scn = new Scanner(System.in);
      Sopln("Enter String");
      String s1 = scn.nextline();
      String s2 = "Hari";
      for(int i=0; i<S1.length(); i++){
          char ch = S1.~~length();~~
                      CharAt(i);
      public int indexOf (String s2){
          sopln("substring Hari available");
      }
    } else {
          Sopln("substring Hari not available");
    }
}
```

Rewrite above program to display substring hari is available even though characters h,a,r,i are available at diff. places in given string.

→

```
// SubString
import Java.util.Scanner.*;
public class SubString {
    P S V m (S[] args) {
        Scanner scn= new Scanner(System.in);
        Sopln("Enter a String:");
        String s1= scn.nextline();
        for(int i=0; i<s1.length(); i++) {
            char ch = s1.charAt(i);
            if (ch=='h' && ch=='a' && ch=='r' && ch=='i') {
                Sopln("Available");
            }
            else {
                Sopln("Not Available");
            }
        }
}
```

if (s1.indexOf('hari')!=-1
s1.indexOf('h')!=-1

Rewrite above program substring hari is available even though character h,a,r,i available in diff places but the character sequence should be 'hari'

→ //

```
import java.util.Scanner.*;
public class
    P  S  V  m(S[] args){
```

Email Validator programe :- (@ in back page)

```
// EmailValidator.java
import java.util.*;
public class EmailValidator {
    p s v m (S[] args) {
        Scanner scn = new Scanner(System.in);

        Sopln("Email:");
        String email = scn.nextline();
        int atIndex;
        if((( atIndex = email.indexOf('@')) != -1) &&
            (email.indexOf('.', atIndex) != -1)) {
          Sopln();
        }
        else {
          Sopln("Invalid email");
    }
```

registered

9: Ant

What is the right parameter for taking String data as argument, 26/7/15
returing string data from a method?

→ Parameter type & return type of the method should be java.lang.String.

→ If we take String as a parameter we can pass string literal directly without using new keyword & constructor. Even we can read string data from keyboard.

→ If we take StringBuffer or StringBuilder as parameter for passing String data we must create object using new ~~keyboard~~ keyword constructor explicitly. It increases more lines of code burden to method call.

→ Below code shows creating a method with string & StringBuilder as the parameter & calling them.

```
Void m1(String s){
    ========
}
```

```
m1("Hari");
```

```
Void m1(StringBuilder sb){
    ========
}
```

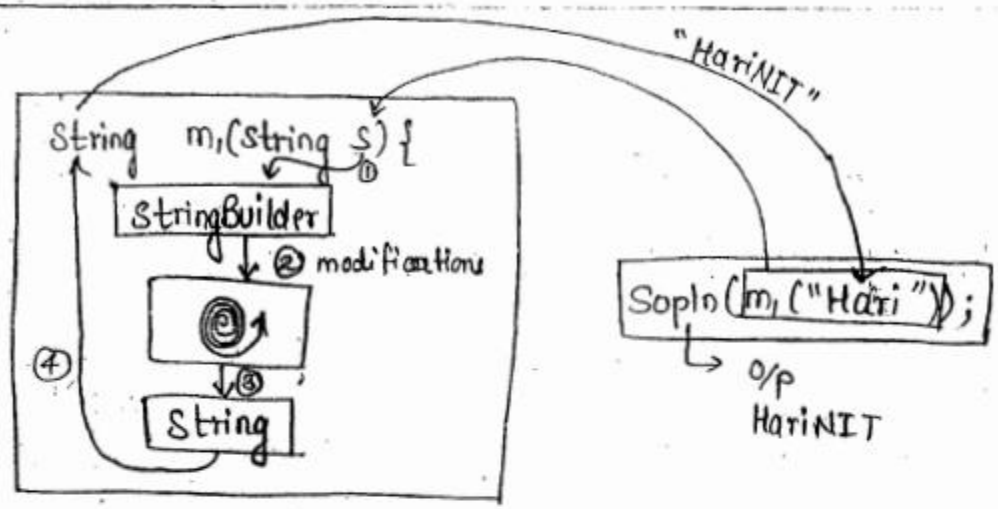
```
m1(new StringBuilder("Hari"));
```

additional work

→ If we take a string as parameter, if you modify string data stored in the string object for every modification new string object is created with result. How should we solve this performance issue. Inside the method convert string object to StringBuilder object.

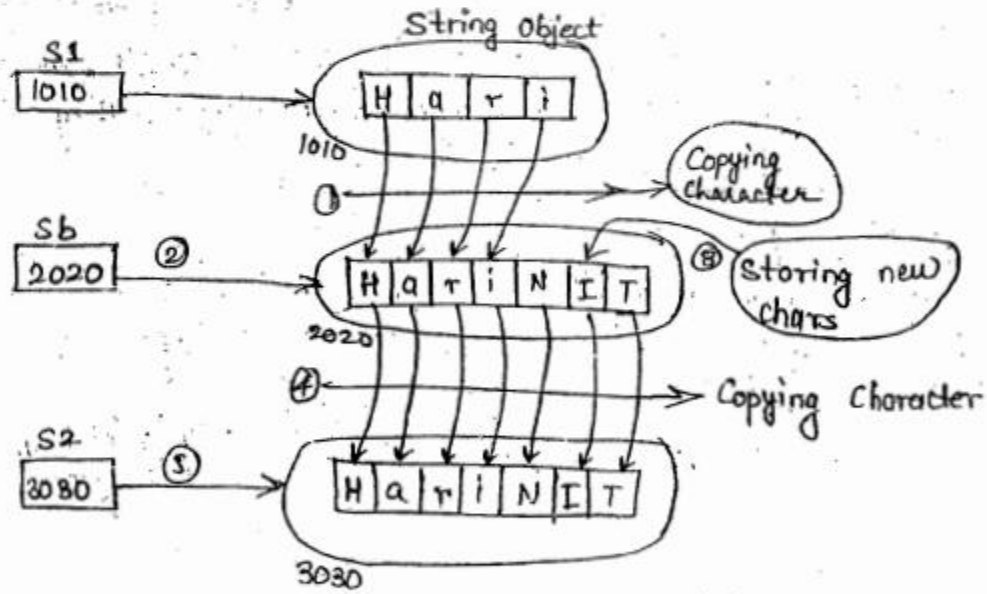
→ Do reqd. modification on StringBuilder object.

→ Convert the result StringBuilder object to String object, return this string object to method caller as shown below →



String & StringBuilder objects are incompatible then how can we convert String to StringBuilder & StringBuilder to String?

→ Here converting means not casting string to StringBuilder rather copying characters from string object to StringBuilder & StringBuilder to string object as shown below.



What is the Api we must must use to convert String to StringBuilder & viceversa?

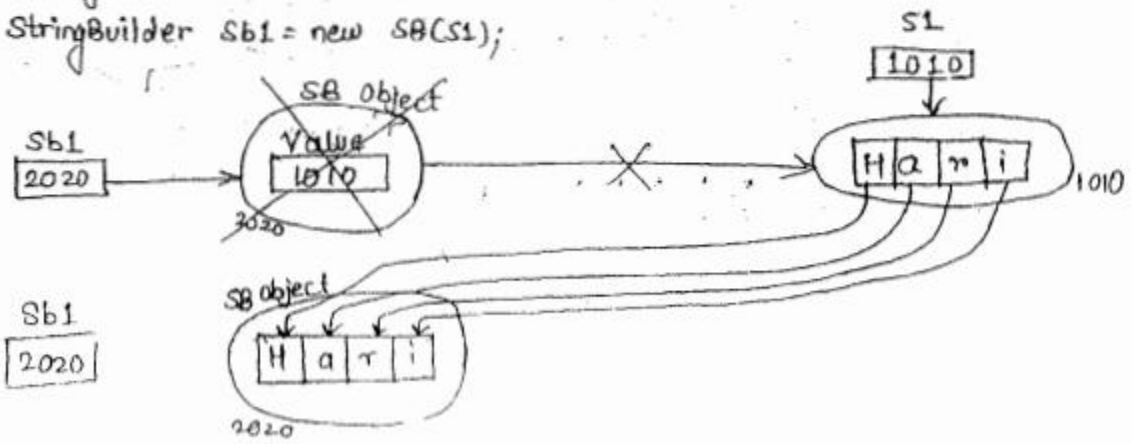
→ In String class & StringBuilder class we have Constructor for copying characters from string object to stringBuilder object & StringBuilder object to String object.

```
class String {
    String (StringBuilder sb) { }
    String (StringBuffer sb) { }
    ==========
}
```

```
class StringBuilder {
    StringBuilder (String s) { }
}
```

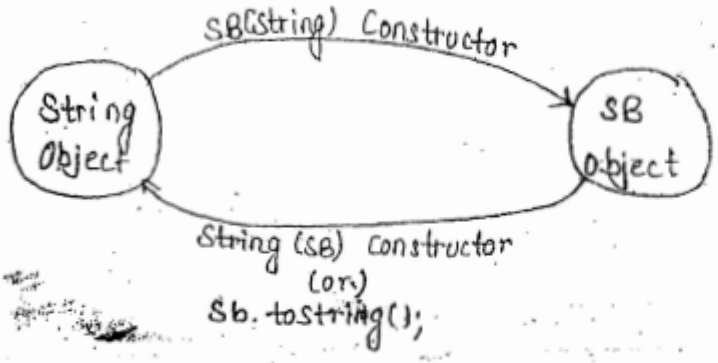
→ The diff betn normal class constructors to String & StringBuilder constructors in normal class constructors stores given argument data in current object whereas String, StringBuilder class constructors will copy given argument object characters into current object. They will not store arguments as shown below.

```
String s1 = "Hari";
StringBuilder sb1 = new SB(s1);
```



We can convert StringBuffer to String Object in two ways

1. Using Constructors
2. Using tostring method



Develop a program to take String data as argument say "Hari", append NareshIT characters each character individually to the argument String data, return result String.

→ Class StoSBConversion {

```
    P s v m(S[] args) {
        String S1 = "Hari";
        String S2 = modify(S1);
        Sopln("S1 : " + S1);   // → Hari
        Sopln("S2 : " + S2);   // → HariNareshIT
    }
    //Converting String to StringBuilder
    static String modify (String s1) {
        StringBuilder sb = new StringBuilder(s1);   ① S to SB Conversion
        String S2 = "NareshIT";
        for(int i=0; i<S2.length(); i++) {
            sb.append(S2.charAt(i));
        }   ② Modifications in StringBuilder object
        return Sb.toString();   ③ Modifications in String Builder object
                                ④ Converting SB to String
    } ④ Returning String
```

COLLECTIONS
FRAMEWORK

BY

Mr.HARI KRISHNA SIR

NARESH technologies

An ISO 9001 : 2008 Certified Company

AMEERPET HYDERABAD

sri raghavendra Xerox

All software language materials available

beside sathyam theatre line balkampetroad ameerpet Hyderabad

cell :9951596199

04/09/15

Collections Framework

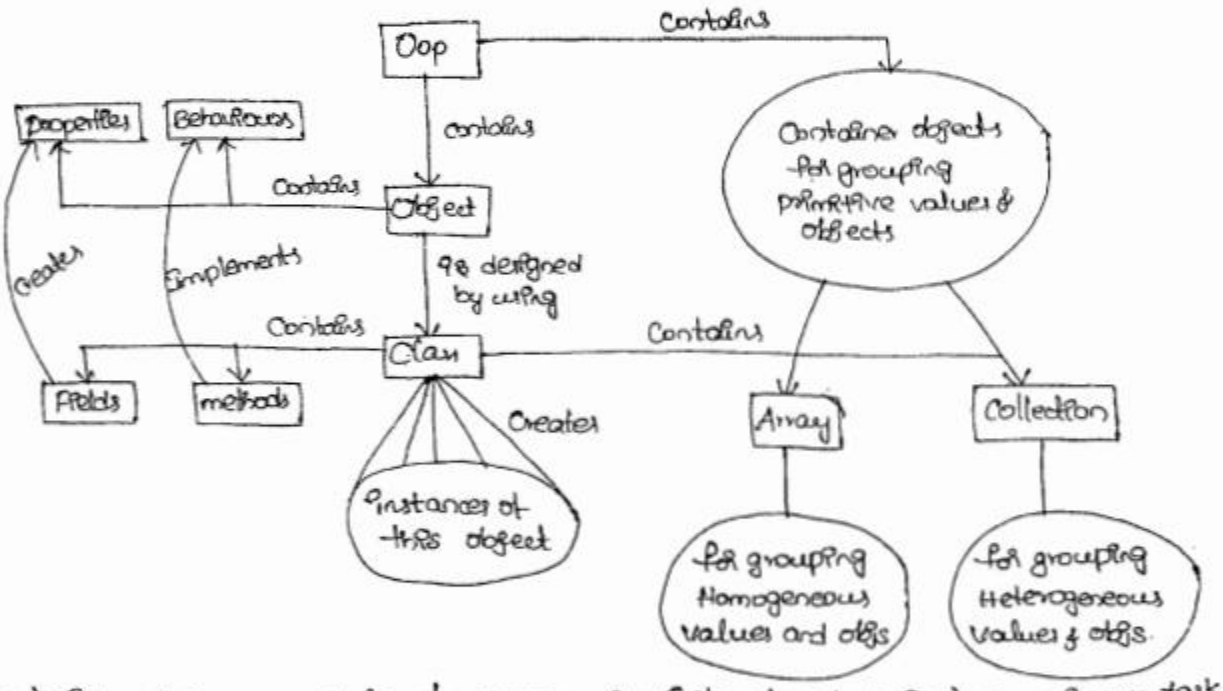
1) Oop terminology
2) Collection terminology
3) List operations
4) Set operations
5) Map Operations
6) Queue Operations
7) Cursor objects
8) utility objects

→ This chapter is meant for storing multiple objs as one group using one container obj that is collection.

1) Oop terminology:-

→ Object oriented programming concepts are invented for representing real world object in programming world by achieving security to data and dynamic method dispatching in automating business operations.

→ Below diagram shows the basics of Oop.



→ By using any style of programming (Structured or Oop) our primary task is storing data in program.

→ In Java we can store data in 4 ways.

1) Variable
2) Array obj
3) Class obj
4) Collection obj

Problem:-

→ using variable we can store only one value.

Sol:- Array.

→ Array can store fixed no. of similar type of values.

→ class obj can store fixed no. of different type of values.

Collection Definition:-

→ Collection is a Container obj it is used for storing multiple homogeneous and heterogeneous, unique and duplicate objs without size limitation and further used for sending all these objs at a time from one class to another class as method argument and return value.

→ Collection is called container obj becoz it can contain other objs for storing and transferring to other class methods.

→ we must use variable for storing one value of an object of an operation.

→ we must use class for storing multiple values of an object (real world obj).

→ Array for storing multiple values of same type of an obj property like courses, mobiles nos, emails.

→ we must choose Collection for storing multiple objs of a single class or different classes.

Note:- Using array we can also store objs but of same class.
Using Collection we can also store primitive values but as objs.

Q) When we have array for storing objs why do we need collection for storing objs.

Ans:- Array has 3 problems.

1) Homogeneous objs are allowed only.
2) fixed in length, can't grow
3) Doesn't have inbuilt methods for performing different operations on array.

→ To solve above 3 problems collection API (classes) are invented.

→ To solve array first problem we can simply choose java.lang.Object[] object.

→ Since it is superclass of all classes, we can store all types of java objects in this array.

→ But size problem and inbuilt operation methods problem can't be solved automatically we must define our own class with Object[] for storing objs, with methods for performing several operations on this array elements.

Algorithm for storing multiple objs without size limitation:-

1) create a class with Object[] instance with required initial capacity, say 10.
2) store elements in this array object.

3) before storing element we must check whether size reached its capacity or not.

4) If reached,

a) create new array with more required capacity

b) Copy elements from old array to this new array.

c) Assign this new array obj reference to old referenced variable.

5) Then store new element at end of all elements.

Sample code:-

program:-

Array object creation

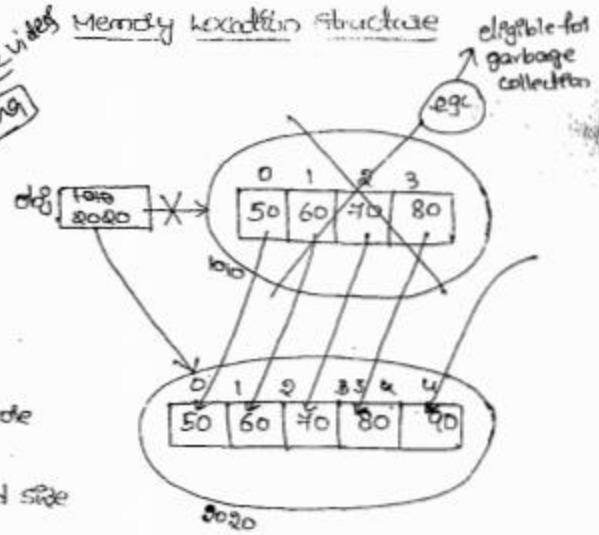
```
Object obj = new Object[4];
```

For complete programmatic explanation watch youtube video Collection by Hari Krishna

Storing elements

```
obj[0] = "50";
obj[1] = "60";
obj[2] = "70";
obj[3] = "80";
```

Memory location structure



Size reached its capacity so further we can't store any more elements,

1) So creating new array object with required size

```
Object[] tempObj = new Object[10];
```

2) Copying old array elements to new array

```
obj.length;
int i=0;
for (i; i<obj.length; i++)
{
    tempObj[i] = obj[i];
}
```

3) Assigning new array obj reference to old array referenced variable

```
obj = tempObj;
```

4) Storing new element at end of new array obj.

```
obj[i] = "90";
```

→ this algorithm is technically called growable array (or) resizable array.

→ this code is tumare code becoz it is not reusable by following this algorithm we must develop a full-fledged class with Obj[] for storing elements and with several methods to perform operations.

08/09/15

→ Below program shows implementing our own collection like sun given collection.

```
// NITCollection.java
class NITCollection {
  //meant for storing/collecting objects.
  private Object[] objArray = new Object[10];

  // maintains array index and also size
  private int index=0;

  //meant for adding object in collection
  public void add(Object ele) {
    if(size()==capacity()) {
      incrementCapacity();
    }
    objArray[index] = ele;
    index++;
  }

  public int size() {
    return index;
  }
```

→ Size means no. of objs we have stored in collection this number is maintain by index variable so we return it.

```
  public int capacity() {
    return objArray.length;
  }
```

→ Capacity means total no. of objs we (can have) stored in collection. This no. maint is nothing but length of Array.

```
  private void incrementCapacity() {
    Object[] tempArray = new Object[capacity()*2];
    for (int i=0; i< objArray.length; i++) {
      tempArray[i] = objArray[i];
    }
    objArray = tempArray;
  }
```

For youtube video explanation search Growable array algorithm by Harikrishna

Above 4 methods are meant for solving Array size problem this method will be called when size = capacity to increase current capacity to double.

```
  public Object get(int i) {
    return objectArray[i];
  }
```

→ meant for retrieving & returning given index object.

```
  public void replace(int i, Object ele) {
    objArray[i] = ele;
  }
```

→ this method replaces given index obj with new given obj in this collection.

For youtube [illegible]

search removing element [illegible] by [illegible]

```
public void remove(int i){
  for(i<size()-1;i++){
    objArray[i] = objArray[i+1];
  }
  objArray[i] = null;
  index--;
}
```

→ removing an element meaning not dropping in a location, instead dropping that element from this array for this purpose we must replace current element with next elements until end of all elements finally in last location we must add null, should decrease size by one.

search inserting element by [illegible]

```
public void insert(int i, object ele){
  if(size()==capacity()){
    incrementCapacity();
  }
  for(s=s
  for(int j=size()-1;j>=i;j--){
    objArray[j+1] = objArray[j];
  }
    objArray[i] = ele;
    index++;
  }
```

→ Insert algorithm is not inserting a location in the middle of collection.

→ we must move elements to their right locations from current index till end of elements then we must add/insert this new element in the given location.

@override

```
public String toString(){
  StringBuilder sb = new StringBuilder();
  sb.append("[");
  for(int i=0; i<size();i++){
    sb.append(objArray[i]);
    sb.append(",");
  }
  int start=sb.lastIndexOf(",");
  if(start!=-1){
    sb.delete(start,start+2);
  }
  sb.append("]");

  return sb.toString();
}
}
```

→ This method meant for displaying the objs those are added to collection.

→ deleting last, (comma) and space

→ we have completed our own development with collection operations adding, counting, retrieving, removing and inserting.

→ so this class has solved Object[] two problems.

1) size and
2) operations

→ let us develop a test app for creating collection obj, adding objs without size limitation then further to perform replace, remove & insert operations.

```
//NITCollectionTest.java
class NITCollectionTest {
  public static void main(String[] args) {
    NITCollection col=new NITCollection();
    Sop.ln("capacity:"+col.capacity()); →10
    Sop.ln("size:"+col.size()); →0
    Sop.ln("elements:"+col); — []
    col.add("a");
    col.add("b");
    col.add("c");
    col.add("a");
    col.add(5);
    col.add(6);
    col.add(4);
    col.add(3);
    col.add(d);
    col.add(10);
    Sop.ln(col);
    Sop.ln("capacity:"+col.capacity()); →10
    Sop.ln("size:"+col.size()); →10
    Sop.ln("elements:"+col);
    col.add(11);
    ---
    ---
    ---
```

*1 ← (capacity, size, elements)

} → same *1

```
//retrieving 3rd index object from collection
Object obj=col.get(2);
Sop.ln("\n3rd index element:"+obj);
---
```

Same as *1 ←

```
//inserting new element at 4th index
col.insert(4,true);
Sop.ln("\n The element true is inserted");
-----
-----
```

Same as *1 ←

// removing 0 index element

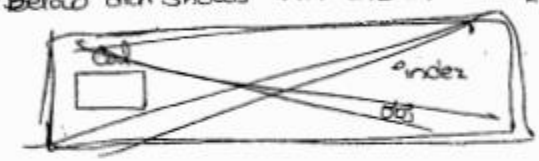
```
Col.remove(0);
System.out.println("In the 0 index element is removed")
- - - - - - - } → same as #1
```

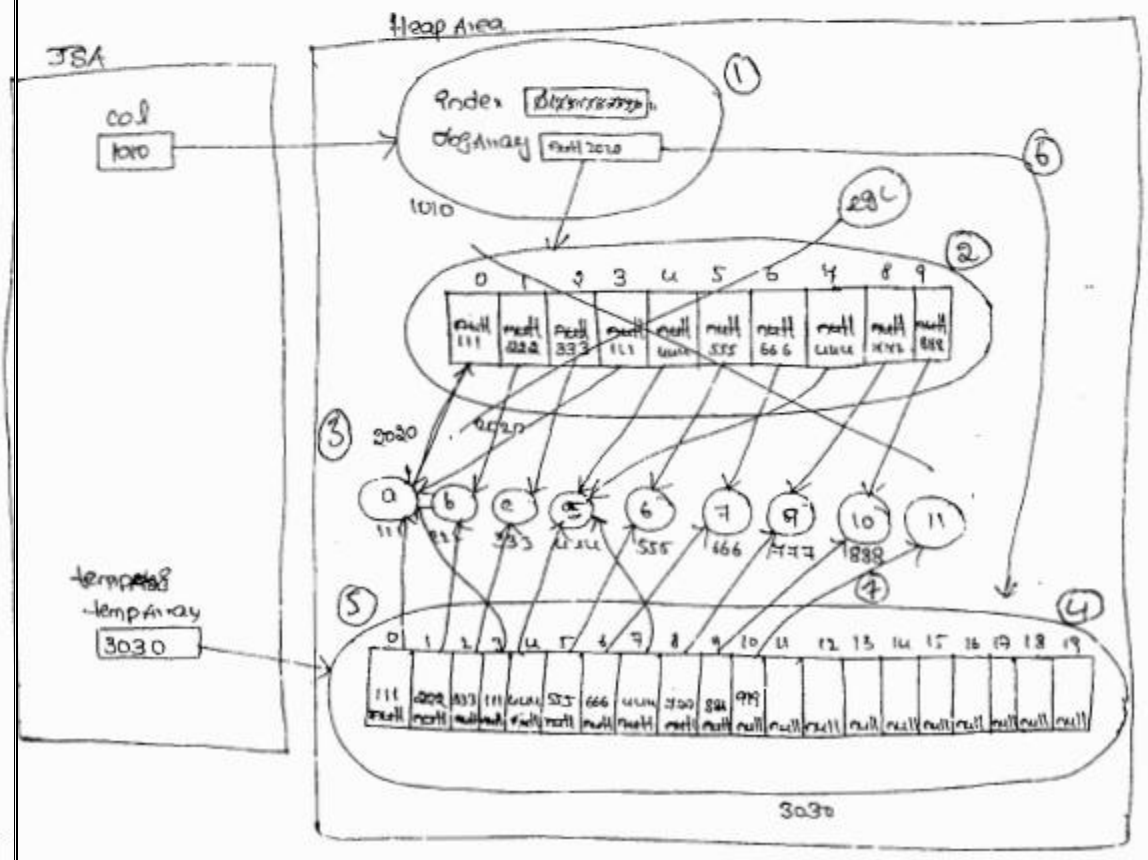
→ Converting primitive datatype into its wrapped object is called autoboxing. and its reverse process is called autounboxing.

// replacing 5th index obj with new object 6.9

```
Col.replace(5, 6.9);
            primitive double
System.out.println("\n the 5th index obj is replaced");
- - - - - - - } → same as #1
```

→ Below diagram shows JVM architecture (with above obj memory) with collection obj memory diagram.





Different operations we perform using collection object:-

→ Using collection obj we will perform below seven operations.

1) Adding objs
2) counting objs
3) searching objs
4) retrieving "
5) removing "
6) replacing "
7) Inserting "

for video explanation search in youtube

Sun given collection classes:-

→ To perform above 4 operations Sun defined several collection classes.

→ All these classes are available in the package java.util.

x[→ 2 formats]x

Two formats for collecting objects:-

→ (Collection API has)

→ we can store/collect objs in 2 different formats.

1) Array format
2) key,value pair format

→ In Array format object doesn't have identity in key, value pair format objs will have identity. Here key is identity, value is the actual objs. For example to collect an employee data,

In Array format, it looks like as below.

0	1	2	3
4249	HariKrishna	NareshIT	Java

In key, value pair format, it looks like as below.

key	value
eno	4249
ename	HariKrishna
Institute	NareshIT
Teaches	Java

→ In 1st approach data doesn't have identity then end user may wrongly understand this data.

→ In 2nd approach data has identity so end user understands this data correctly. It doesn't mean that first approach is worst approach. When we want send data without identity we must follow 1st approach.

- for example if we want to send all employee objs as a group we will follow first approach becoz employee object doesn't need any identity.

Note :- our class NITCollection collects objs in array format. for collecting objs in key, value pair format, we must use two array objs in NITCollection class. One is for storing keys and second is for storing for values.

Create a new class called NITMap with the below Code change in NITCollection:-

1) add() method must have two parameters to send (key, value) objs
2) IncrementCapacity() method must increment both key & value array objects capacity equally.
3) then we must store
   first(key, value)
      in 0 indexed array objs
   second (key, value),
      in 1st index of array objs
   etc...

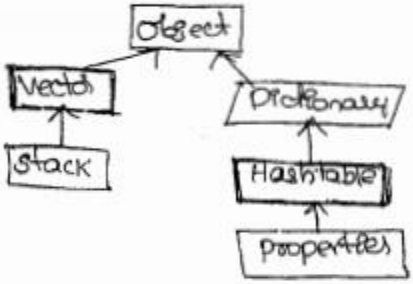
Java 1.0	Java 1.2	Java 1.4	Java 1.5	Java 1.6	Java 1.7
Vector → 1) Synchronized → 2) Gives low performance in insert & remove operations → 3) Can't stop duplicates → 4) Can not sort objects	→ Array List → Linked List → Hash Set → cannot store objs in insertion order. → Tree Set	→ Linked Hash Set	Queue type collection Generics Autoboxing & unboxing Concurrency package	→ Navigable set	→ Type Inference Java- → factory methods to create immutable list set for video explanation refer collections API by Harikrishna
Hashtable → 1) Synchronized → 2) Can not sort entries	→ Hash Map → Can't store entries in insertion order. → Tree Map Weak Hash Map	→ Linked Hash Map Identity Hash Map		→ Navigable Map	Java 1.8 ① Default methods ② Static methods ③ Functional interface ④ Lambda expression ⑤ method reference ⑥ Functional API ⑦ stream
Enumeration → method names are large → Can not remove elements while iterating	→ Iterator → List Iterator		Iterable (i) for-each loop		

SUN given Collection classes :- in java 1.0

→ For storing elements in array and (key, value) pair formats SUN defined below 2 classes in java.util package.

1) Vector → for storing elements in array format
2) Hashtable → for storing elements in key:value pair format.

→ These 2 classes are available from java 1.0 version itself. These two classes are created with below hierarchy.



Collections framework classes :- given in java 1.2

→ In java 1.2 version java.util package is updated by adding many new classes & interfaces for adding more functionality in storing objs in diff. formats & orders.

Diagram :-

Below dia. shows all classes added in java 1.2, 1.4, 1.5 & 1.6 refer dia. in previous page.

→ From java 1.2 onwards java.util packages are called Collections framework classes. because all these classes are working for solving a particular problem that collecting objs without size limitation.

> A Framework is a semideveloped app that contains set of classes and interfaces for solving a particular problem.

→ We can interchange one Collection obj with another collection obj because collection API is a runtime polymorphic API.

→ To achieve Runtime polymorphic nature all collection classes are created under 2 superclasses

Collections framework hierarchies :-

→ Collections framework classes are grouped into 2 hierarchies.

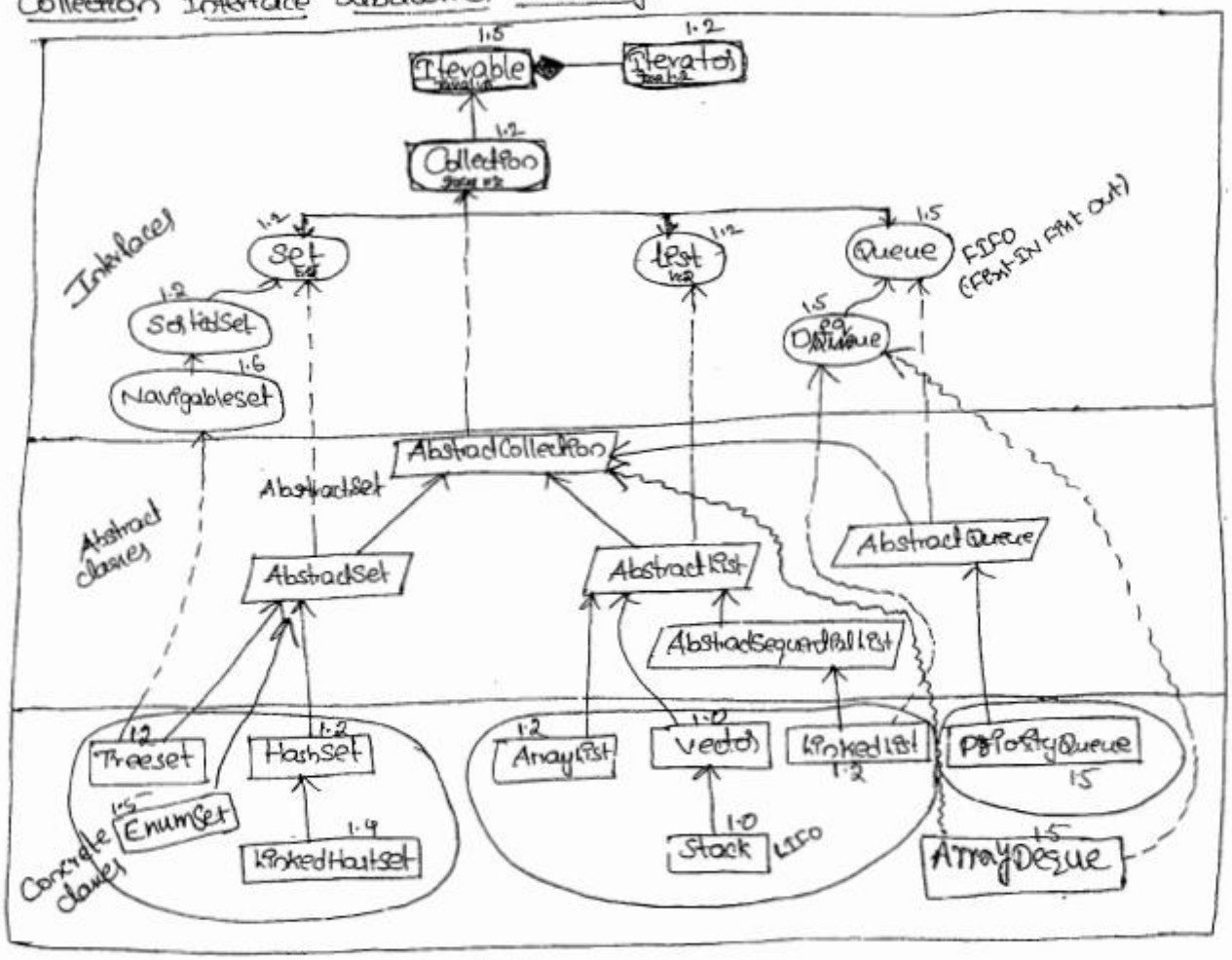
1) Collection hierarchy
2) Map hierarchy

→ Collection & map are 2 root interfaces of collection framework classes.

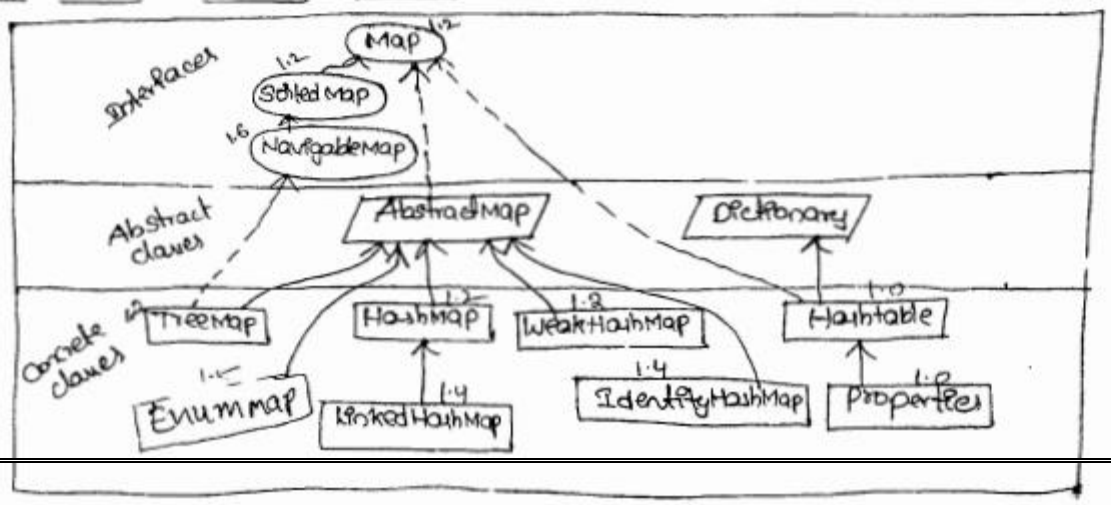
→ Collection Interface subclasses are meant for collecting objs in array format.

→ Map interface subclasses are meant for collecting objs in key, value pair format.

Collection Interface subclasses hierarchy:-



Map Interface subclasses hierarchy:-



Q) What classes are called legacy classes, collection framework classes and generic framework classes?

A:- legacy means old version classes

→ The collection classes those are given in Java 1.0 are called legacy collection classes. They are:

1) Vector
2) Hashtable
3) Enumeration (I)

→ The collection classes given newly from Java 1.2 onwards are called collection framework classes

Collection, Map interfaces subclasses are (added to) called collection framework as classes.

Note:- In Java 1.2 Vector & Hashtable classes are added to collection framework as subclasses of List & Map interfaces.

→ From Java 5 onwards collection framework classes are also called Generic collection classes. In Java 5 version collection classes are rewritten with generic syntax for acting only homogeneous objs.

```
class ArrayList{}
class HashMap{}
```
→ are collection framework classes

```
class ArrayList<E>{}
class HashMap<K,V>{}
```
→ are collection framework classes with generic syntax.

→ From Java 5 onwards we will have collection classes only with Generic syntax but we have chance to use collection classes with Generic syntax or without generic syntax, it is our wish.

```
ArrayList al = new ArrayList();
```
→ collection obj without generic syntax

```
ArrayList<String> al = new ArrayList<String>()
```
→ collection obj with generic syntax.

→ The term legacy is only applicable to collection classes, because Vector & Hashtable classes got alternative collection classes.

Collection programming:

→ To implement code using collection API we must follow below 3 steps.

1) add "import java.util.*;" statement
2) Create appropriate collection obj based on storing and retrieving data.
3) Call appropriate methods to perform any one of the 4 operations.

→ for creating collection obj we must know the available constructors in all collection classes.

→ In every collection class we will have minimum 2 constructors.
In collection subclasses we have:
1) Non-parameterized constructor
→ for creating empty collection object.
2) Collection parameter constructor
→ for creating a collection with given collection elements (copied).

Note :- Stack has only noparam / zero-param constructor.

→ In Map subclasses we have:
1) Non-parameterized constructor
→ for creating empty map object
2) Map parameter constructor
→ for creating map objs with given map entries.
→ for more details check API documentation of each collection class.

Sample code:-

```
class ArrayList {
    ArrayList() {}
    ArrayList(Collection c) {}
}
```

```
class HashMap {
    HashMap() {}
    HashMap(Map m) {}
}
```

Q) Is Map subinterface collection?

A:- No, because Map chars are different from collection. Collections can be stored in the form of Array format while Map can be stored in the key, value pair format.

→ These are many classes working together to perform a particular task is called "Framework".

11/12/15

Collection Interface methods:-(16+5)

1) Collect

1) public boolean isEmpty()
2) public boolean add(object obj)
3) public boolean addAll(Collection c)
4) public boolean remove(object obj)
5) public boolean removeAll(Collection c)
6) public void clear()
7) public boolean contains(object obj)
8) public boolean containsAll(Collection c)
9) public boolean retainAll(Collection c)
10) public int size() → for counting
11) public Iterator iterator() → used for retrieving elements from Collection.
12) public int hashCode()
13) public boolean equals(object obj)
14) public object[] toArray()
15) public object[] toArray(object[] obj)

Java 8v added new method

16
17
18
19
20

Refer collectionInterface API Documentation

+

II Common interview questions asking on all type of Collections:-

1) What is the use of this Collection?
2) What is the implemented datastructure?
3) Default Capacity, Increment Capacity, load factor
4) What type of elements are allowed? (Homogeneous, heterogeneous like that)
5) Is null allowed how many?
6) What is the storing order?
7) What is the retrieving order?
8) Is it synchronized collection?
9) Is it ordered collection?
10) Internally calling method?
11) Performance

→ These 4 questions we must learn for collections.

Working with ArrayList:-

Q1) - A) for storing multiple objs with index multip including duplicates without thread safety or synchronization we must use Array list.

Q2 - A) Growable array or resizable array is the implemented datastructure of Arraylist.

Q3 - A) Default capacity - 10

Incremental capacity $= \left(\frac{\text{current capacity} * 3}{2}\right) + 1 \Rightarrow \left(\frac{10*3}{2}\right) + 1$

$\Rightarrow 16$

load factor not applicable here

Qu-A) All 4 types of elements are allowed are homogeneous, heterogeneous, unique and duplicate objs.

Q5-A) null insertion is possible, more than one null allowed becoz duplicates are allowed.

Q6-A) Insertion order with index. Insertion order means the order in which add() methods are called.

Q7-A) Random access of course sequential access is also possible.

Q) Below program shows all above points with (a program) arraylist.

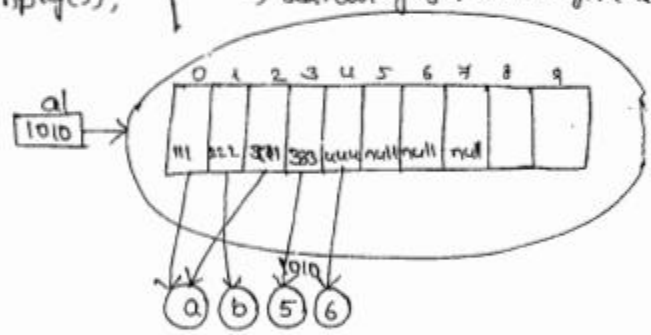
```
// ArrayListDemo.java
import java.util.*;
  Class ArrayListDemo {
    public static void main(String[] args) {
      ArrayList al = new ArrayList();
        System.out.println("isEmpty:"+al.isEmpty());
    // adding all 4 types of objs
        al.add("a");
        al.add("b");
        al.add("a");
        al.add(5);
        al.add(6);
        al.add(null);
        al.add(null);
        al.add(null);
        System.out.println("size:"+al.size());
        System.out.println("elements:"+al);
    //retrieving objs randomly
        System.out.println("3rd index object:"+al.get(3));
      }
    }
```

Q8-A) not synchronized

Q9-A) ordered collection

Q10-A) equals() method in searching & removing operation

Q11-A) gives high performance in adding & retrieving
gives low portance in searching & removing & inserting



Compilation & execution:-

```
> javac ArrayListDemo.java
> java ArrayListDemo
        isEmpty : true
        size : 8
        elements : [a, b, a, 5, 6, null, null, null]
        3rd index object : 5
```

Working with Vector:-

→ Vector and ArrayList functionalities are same the only differences are there in

i) ArrayList not synchronized and vector is synchronized.

ii) ArrayList incremental capacity half+1 ($\frac{1}{2}$+1) whereas vector incremental capacity is double.

Q1-A) Vector is used for storing homogeneous and heterogeneous unique and duplicate objs in insertion order with index in multithreading environment with thread safety & synchronization.

Q2-A) Growable array

Q3-A) default capacity-10
Incremental capacity - double

Q4-A) All types of elements are allowed.

Q5-A) null insertion is possible, more than one.

Q6-A) insertion order with index

Q7-A) Random access

Q8-A) it is synchronized collection

Q9-A) it is ordered collection

Q10-A) equals() method in searching & removing

Q11-A) slow performance compared to AL
→ adding & retrieving gives high performance
→ searching & removing & iterate

Q) Rewrite above program by replacing ArrayList obj with vector for testing above points.

Ans:-

```
//VectorDemo.java
import java.util.*;
Class VectorDemo{
  public static void main(String[] args){
    Vector v = new Vector();
    Sopln("IsEmpty:"+v.isEmpty());
    v.add("a");
    v.add("b");
    v.add("a");
    v.add(5);
    v.add(6);
    v.add(null);
    v.add(null);
    v.add(null);
    Sopln("size:"+v.size());
    Sopln("elements:"+v);
    Sopln("3rd index object:"+v.get(3));
  }
}
```

o/p:-

```
IsEmpty: false
size: 8
elements: [a, b, a, 5, 6, null, null, null]
3rd index object: 6
```

12/09/15

Working with stack:-

Q1-A) → We must use stack for storing multiple objs and further for retrieving them in "Last in First out manner (LIFO)."

→ Stack is a class in Java which is a subclass of vector. All methods of vector are inherited to stack. but we shouldn't use vector class methods on stack object becos we can't retrieve objs in LIFO manner.

→ To treat vector as stack obj stack class has its own special methods.

1) push :-

```
public E push (E item)
```

This method will store obj in stack

2) public obj pop:-

```
public E pop()
```

→ Removes the obj at the top of this stack and returns that top object to our program.

Note :- top object means last added obj.

3) public obj peek:-

```
public E peek()
```

→ This method returns the top obj from this stack without removing.

Rule :- Above 2 method we shouldn't call on empty collection both methods will throw java.util.EmptyStackException. It is unchecked exception optional to handle.

4) public int search(Object o):-

```
public int search(Object o)
```

→ It returns obj position inside stack the position (searching index) start with 1 from top to bottom top is last element and bottom is first element.

→ If obj not found this method returns -1

Q2-A) :- stack is implemented datastructure (Growable array backed by vector obj).

Q3-A) :- default capacity - 10
Incremental capacity - double

remaining points are same as vector

Q4-A) :- all u types of objs

Q5-A) :- null allowed, more than one.

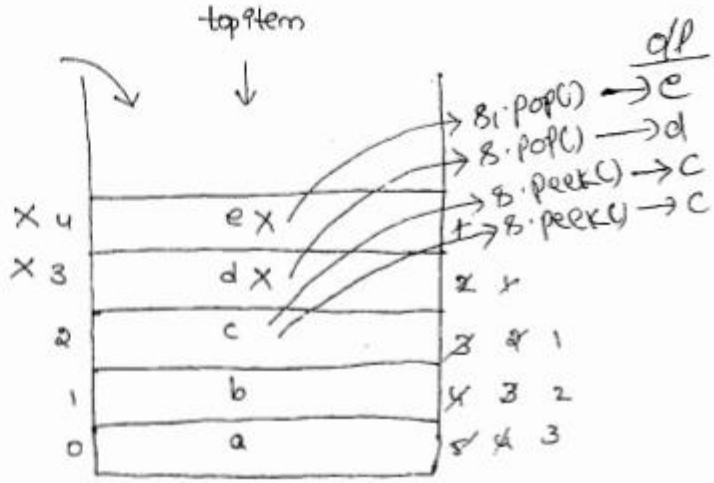
Q6-A) :- insertion order

Q7-A) : LIFO order.

Q) Below prg shows all above points and working with stack.

```
import java.util.*;
class StackDemo{
    public static void main(String[] args){
        Stack s=new Stack();
        s.push("a");
        s.push("b");
        s.push("c");
        s.push("d");
        s.push("e");
        Sopln(s);
    // retrieving & removing top item
        Sopln(s.pop());
        Sopln(s);
        Sopln(s.pop());
        Sopln(s);
        Sopln();
    // only retrieving top item
        Sopln(s.peek());
        Sopln(s);
        Sopln(s.peek());
        Sopln(s);
        Sopln();
    // searching for an item
        Sopln(s.search("b"));
        Sopln(s.search("e"));  -> -1
    }
}
```

O/P:- [a, b, c, d, e]
e
[a, b, c, d]
d
[a, b, c]
c
[a, b, c]
c
[a, b, c]
2
-1



Q) What is the diff. b/w pop & peek methods?

A:- → pop() method will remove & return top item. peek() method will only retrieve or and return will not remove top item.

→ Everytime when we call pop() method different object will return. Everytime when we call peek() method it returns same o/p.

Working with LinkedList:-

Q1-A) → If our operations are more insert and remove elements in the middle of the collection then we must use LinkedList. It gives better performance compared to other collections for insert and remove elements in the middle of collection.

Q2-A) Implemented datastructure is Doubly-LinkedList

Q3-A) Default capacity-0
Increment capacity- 1

Q4-A) All u elements

Q5-A) null allowed, multiple

Q6-A) Insertion

Q7-A) only sequential because no index

Q8-A) it is not synchronized

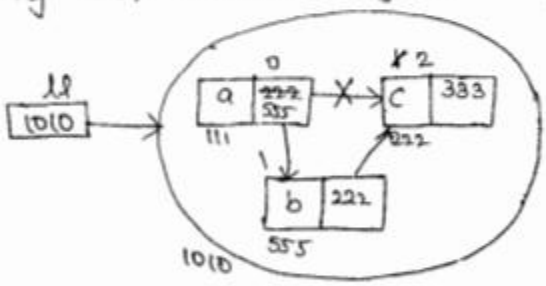
Q9-A) It is ordered collection, insertion order

Q10-A) It internally calls equals() method in searching & removing operations

Q11-A) gives high performance in insert & remove operations in the middle of collection (by object)

Below program shows working with LinkedList object.

```
LL ll=new LL();
ll.add("a");
ll.add("c");
ll.add(1,"b");
Sop(ll);
```



→ How LinkedList will give high performance in insert and remove operation compared to arraylist.

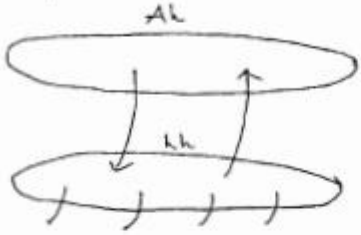
A:- In Arraylist elements will shift to right locations in insert operations, elements will shift to left locations in remove operation. So in Arraylist more copy operations are performed from one location to another location.

Assume we have 1000 elements in AL, want to insert 50 elements in beg-first / begining location then 1000 elements 50 times should copy to next/Right locations which is performance issue.

→ In this case if we use LinkedList existing 1000 elements no need to shift right side so 50000 copy operations only 100 copy operations need to perform for changing the links. So LinkedList execution is fast.

→ <u>Algorithm for performing insertion operation using LL:-</u>

1) Create AL object
2) Add objs into AL
3) For inserting objs in the middle
   i) Create LL obj
   ii) Copy obj from AL to LL
   iii) Insert missing obj in LL
   iv) Clear AL (remove all objs from AL)
   v) Copy results obj from LL to AL.



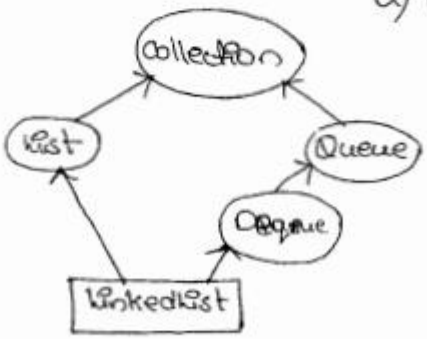
<u>Note</u>:- We shouldn't use LinkedList for adding objs and also shouldn't be used in LinkedList obj for inserting one or two elements it will gives less performance becos of the below reasons.

1) It consume more memory compared to ArrayList
2) If we directly insert ArrayList the no. of copy operations are less than the insert operation through LinkedList. So we must use LL only for more objs insert and remove operations.

→ In real world (Java world) we can't use multiple inheritance with classes becos (classes contains implemented) of reusability is not present in classes. we use multiple inheritances with interfaces.

Q) What is the collection class exist in collections implementing multiple inheritance?

A:- LinkedList class. It is the subclass of List Interface and also subclass of Queue Interface.



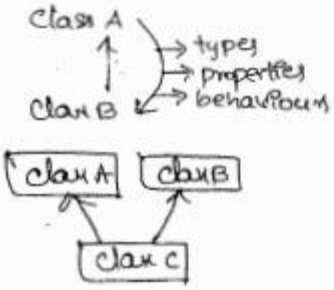
Q) How can we use LinkedList as List type collection and Queue type collection and Deque type collection?

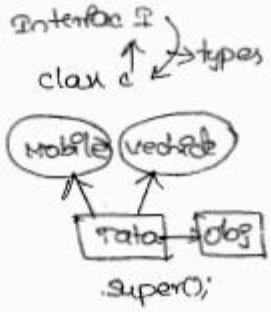
```
List l = new LinkedList();
Deque d = new LinkedList();
```

→ In first line LinkedList will act as a list type collection here we can only access List() methods.

→ In second statement LinkedList will act as Queue type here we can access only Queue type methods.

→ Interface is used to provide only forcebility and class is used to provide only reusability and abstract class have these two forcebility & reusability.





10/09/15

Q) Write a program for adding String object to ArrayList, display them in uppercase on console.

```
import java.util.*;
class ArrayList {
    p s v main(String[] args) {
        ArrayList al=new ArrayList();
        al.add("a");
        al.add("b");
        al.add("c");
        Sop(al);
        for(int i=0; i<=al.size(); i++){
        al.get(i);
Stmt1 → Object dog= al.get(i);
Stmt2 → String str=(String) dog;
        Sop(str.toUpperCase());
        }
        Sop();
        Sop(al);
    }
}
```

→ understanding Retrieving objects from collection using get(index) method

→ get method CE & RE, solutions

O/P:- [a, b, c]

A
B
C

[a, b, c]

Q) In above program why did u cast string obj to string type in str1?

Ans:- The obj comming out from the collection will have the type java.lang.Object, becoz get method return type is Object, becoz Collection is heterogeneous for returning any type of obj method return type should be java.lang.object.

→ We can'it perform the current return obj specific operations when it is stored in Object type variable so we must convert its type from Object type to its own type for this purpose we must use cast operator.

Short Ans:-

→ Get method returns every element from Collection as Object type, we must convert its element to its own type for calling its own specific operation methods so we must do cast operation.

Upcasting & downcasting:-

→ Storing subclass Object reference in superclass variable is called upcasting.

for eg:- `Object obj = al.get(i);`

↓

Here returning object is subclass

→ Converting subclass object type from superclass type to (subclass) its own type is called downcasting.

```
String str = (String) obj;
```

→ We must do downcasting only when we have to subclass specific functionality methods.

→ In above program we have casted object to string type for calling String class methods to uppercase

```
Sopln (obj.toUpperCase());  → X CE:
Sopln (str.toUpperCase());  → ✓
```

Q) Write a program for adding 3 employees salary and name. Display all these 3 employees details in table format with 2 columns name and sal. display names in uppercase.

Sample project on adding, retrieving & modifying data in collection

Ans:-

```
import java.util.*;
class DisplayALHeterogeneOUS?
    P s v main (String[] args) ?
        ArrayList al = new ArrayList();
```

```
al.add("Hari");
al.add(10000);

al.add("Balayya");
al.add(80000);
al.add("Nani");
al.add(30000);
sopln(al);
for(int i=0;i<al.size();i++){
sopln();

Sopln("NAME|ESAL");
Sopln("==========");

for(int i=0;i<al.size();i++){
  Object ele=al.get(i);
  if(ele instanceof String){
    String name=(String)ele;
    Sopln(name.toUpperCase()+"|+");
  }else{
    Sopln(ele);
  }
}
}
}
```

for avoiding CCE

Checking the ele type if ele is string type converting it to string

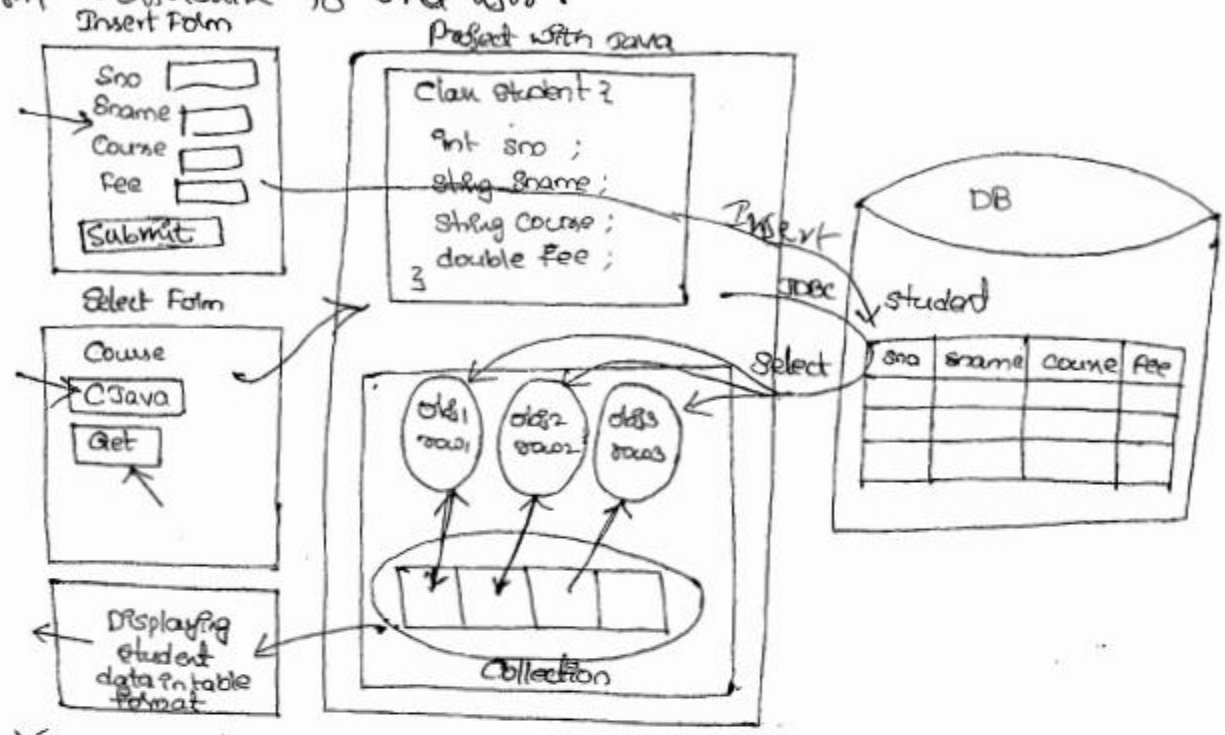
→ If Collection contains different class objects, while retrieving these objs from collection we must (do casting) cast these objs to their own type inside instanceof operator if condition as shown in above program otherwise we will get Class Cast Exception, program terminated in the middle of execution.

→ In above program comment if & else lines u will get CCE

1) Adding → c.add(obj); (C)
2) counting → c.size(); (Collection)
3) Retrieving → c.get(index); (L)
4) Searching → l.contains(obj) (C)
   l.indexOf(obj) (L)
5) removing → l.remove(obj) (C)
   l.remove(index) (L)
6) replacing l.set(index, obj) (List)
7) inserting l.add(index, obj) (L)

~~8) Develop a project for transferring student data from database to excel sheet based on given id.~~

Q) Develop a project for transferring student data ~~from~~ from database to end-user.



15/09/15

Java DB mapping :-

DB	Java
table	class
columns	variables
rows	instances

Object data

n objects → n rows in database
→ n instances in Java Heap.

Java bean :-

→ As many tables we have in database those many bean classes we develop in Java.

→ bean is also a class created with the tablename.

→ A class that contains only private variables and public setter, getter methods for storing and retrieving an object is called bean class.

→ bean class is implemented from java.lang.io.SerializableInterface. Then only this class obj data will travel through n/w.

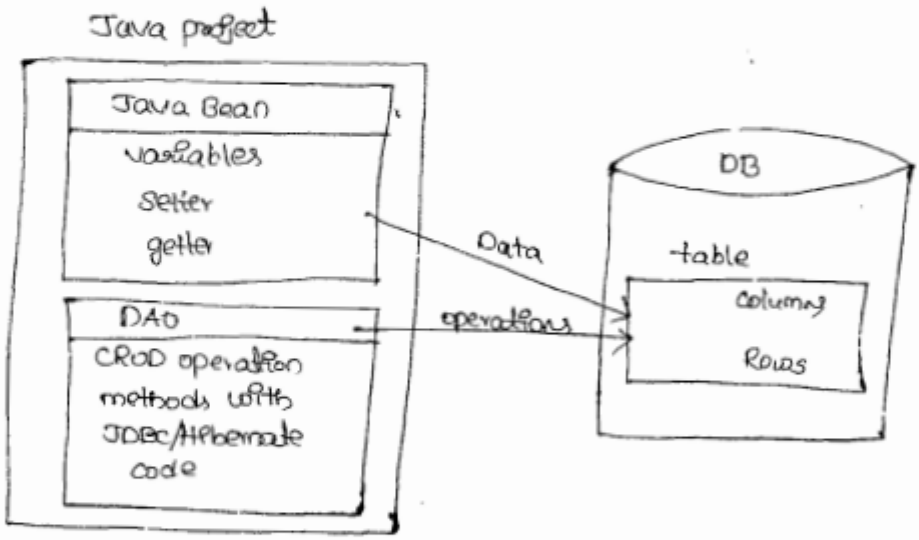
→ If this class object data is using only within the JVM in single computer we no need to implement this class from Serializable interface.

→ Inside bean class we must create variable as many columns as we have in table. Generally variable names will be same as column names. We must create instances from this class one per row available in this table.

DAO:-

→ A class that contains methods to perform CRUD operations in DB table is called DAO.

→ We will create DAO class one per table so as many tables we have those many Java bean and DAO classes we will develop.



→ In order to develop Naresh IT Students information system. we must develop below components.

1) One new user SIMS|SIMS
2) One table - Student (sno, sname, course, fee)
3) Insert sample rows with some student data (H, B, N)
4) One Bean class - [Student.java] StudentBean.java
5) One DAO class - StudentDAO.java
6) Main method class - FrontOffice.java

User creation command:-

1) Login to system / manager user
2) Run below commands at SQL prompt
   > create user sims identified by sims;
   > grant DBA to sims;
   > connect sims/sims;

Table creation command:-

```
> Create table Student(
      Sno number(5) primary key,
      Sname varchar2(20),
      Course varchar2(10),
      fee number(4,2)
);
```

total fractional part

```
> Insert into student (Sno, sname, course, fee)
  values (101, 'Hari', 'Corejava', 1000);
  Insert into Student (Sno, Sname, Course, fee)
  values (102, 'Balayya', 'Corejava', 1000);
                  :
> commit;
> Select * from Student;
```

SNO	SNAME	COURSE	FEE
101	Hari	Corejava	1000
102	Balayya	Corejava	1000
:	:	:	:

Java side coding:-

```
// StudentBean.java
package com.naresh.bean;
import java.io.Serializable;
  public class StudentBean implements Serializable {

      private int sno;
      private String sname;
      private String Course;
      private double fee;
// generate getter and setter methods using eclipse
//rightclick -> source -> generate getters and setters
    public int getSno() {
      return sno;
    }
    public void setSno(int sno) {
    this.
```

DAO Class design:-

1) JDBC objects variables declaration
2) A constructor for creating con and Pstmt objects
3) A select methods for executing select query
   i) obtain RS object
   ii) Create one bean instance for each row
   iii) read row values and store in the bean
   iv) store bean instance reference in Collection objects
   v) repeat above 2,3,4 steps for every row available in RS object
   vi) return this Collection object

```
package com.naresh.dao;
package
import java.sql.*;
import com.naresh.bean.*;
  public class StudentDAO {

    private Connection          con;
    private PreparedStatement   pstmt;
    private ResultSet           rs;

      public StudentDAO() {
        try {
          Class.forName("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:
                                                      1521:xe", "sims", "sims");

        String squery = "select * from student" +
                        where course = ?";

        Pstmt = con.prepareStatement(query);
      } catch (ClassNotFoundException e) {
        e.printStackTrace();
      } catch (SQLException e) {
        e.printStackTrace();
      }
    } //constructor close
// method for retrieving students information from DB
  public ArrayList getStudents(String course) {
    ArrayList studentsList = new ArrayList();
```

```
try {
  pstmt.setString(1, course);
  rs = pstmt.executeQuery();
    while(rs.next()) {
        StudentBean bean = new StudentBean(); - Create this line outside of the while loop and test
        bean.setSno(rs.getInt(1));
        bean.setSname(rs.getString(2));
        bean.setCourse(rs.getString(3));
        bean.setFee(rs.getDouble(4));

        Studentslist.add(bean); -> Adding this bean obj to collection.
    }
  } Catch (SQLException e) {
      e.printStackTrace();
  }
  finally {
      try {  -> if(rs!=null) {
if(rs!=null)  rs.close();
      } Catch(SQLException e) {
          e.printStackTrace();
      }}
      return studentslist;
  }
// @Override
    public void closeObjects() {
      try {
        if(pstmt!=null) { pstmt.close(); }
        if(con!=null) { con.close(); }
      } catch (SQL Exception e)
          e.printStackTrace();
      }
    }
// FrontOffice.java
package com.nareshit.user;
  import java.util.*;
  import com.nareshit.bean.*;
  import com.nareshit.dao.*;
    public class FrontOffice {
      public static void main(String[] a1) {
    x [Scanner sc = new Scanner(System.in); ]x
```

```
StudentDAO dao = new StudentDAO();
Scanner scn = new Scanner(System.in);
while(true) {
  Sop.ln("Enter Course:");
  String course = scn.readLine();
  ArrayList StudentsList = dao.getStudents(course);

    Sop.ln("Sno\t\tSname\t\t\t course\t\t\t fee");
    Sop.ln("_____________________________________");
    for(int i=0; i<StudentsList.size(); i++){
      StudentBean bean = (StudentBean) StudentsList.get(i);
        Sop.ln System.out.print(bean.getSno()+"\t");
              System.out.print(bean.getSname()+"\t");
              System.out.print(bean.getCourse()+"\t");
              System.out.print(bean.getFee+"\t");
            }
          }
        }
      }
```

Note :- To run this project we must add ojdbc6.jar file to eclipse buildpath. Oracle driver classes are available in this jar file

Rightclick on project → build path → configure build path → libraries → Add external jars. ⇒ End of project

(3) Searching an object in Collection:-

→ For searching an object in collection, we must use contains method.

* prototype :- public boolean contains(Object obj)

→ contains method will search an obj in the collection with the given object.
→ argument

→ For this purpose inside contains method logic, given argument object will be compared with objects available in the Collection. by using java.lang.Object class equals() method

→ If match found, it returns "true" else returns "false".

→ for comparing objs, inside contains method, Equals() method is used.

→ Equals() method compares '2' objects, using either "reference" or using "data" based on its implementation.

→ If equals method not overridden in added objects class, the two objects are compared using reference.

→ If it is overridden in added (using) object class, then objs are compared using their data.

Note :- In string & in all "wrapper classes" equals method has overridden so, their objects are compared using data.

→ In StringBuffer & StringBuilder classes "equals method" not overridden, so their objs are compared using reference.

Another Ex:- Searching StringBuilder object in AL :-

```
AL al=new AL();
al.add(new StringBuilder("a"));
al.add(new StringBuilder("b"));
StringBuilder sb3 = new StringBuilder("c");
al.add(sb3);
Sopln(al.contains(new StringBuilder("a")); false
Sopln(al.contains(sb3)); true
```

→ Since equals() method not OVR in StringBuilder class only same referenced SB objects are found in collection.

→ In above program in first contains() method called we have passed new referenced StringBuilder object so it returns false becoz obj not found with this reference.

→ In 2nd contains method called we passed sb3, same referenced obj so contains method returns true becoz obj found in Collection.

→ Below is the searching algorithm

```
Sopln(al.contains(new StringBuilder("a"));
    sb1.equals(sb1); → executed from SB class
        false          → not OVR
    sb1.equals(sb2);   → so executed from object class
        false            → two SB objs are compared
                           with reference
    sb1.equals(sb3);     → returns false
        false
```

```
Sopln(al.contains(sb3));
    sb3.equals(sb1);
        false
    sb3.equals(sb2);
        false
    sb3.equals(sb3);
        true
```

Sample project1 on searching operation
→ Adding and searching custom objs in
↓
Employee
Student
BankAccount etc. are custom objs

→ Develop a project to add Employee object List type collection.
Those added employee objects must be found by using new referenced objects with same ueta.

(P.T.O)

```
//AddingSearchingCOEmp.java
import java.util.*;
class AddingSearchingCOEmp {
    public static void main(String[] args) {

        ArrayList al=new ArrayList();
        al.add(new Emp(101, "HB", 10000, "Java"));
        al.add(new Emp(102, "BB", 20000, "Java"));
        al.add(new Emp(103, "MB", 30000, ".NET"));
        Sopln(al.contains(new Emp(102, "BB", 20000, "Java")));

            //equals() not OVR        OVR
            // e1.equals(e1); -> false   false
            // e1.equals(e2); -> false   true
            // e1.equals(e3); -> false
        Sopln(al.contains(new Emp(102, "PB", 50000, "Java")));
    }
}
```

matched with this second object hence returns true

```
//Emp.java
class Emp {
    int eno;
    String ename;
    double sal;
    String dept;
    Emp(int eno, String ename, double sal, String dept) {
        this.eno=eno;
        this.ename=ename;
        this.sal=sal;
        this.dept=dept;
    }
    @Override
    public boolean equals(Object obj) {
        if(obj instanceof Emp) {
            Emp e=(Emp)obj;
            return (this.eno==e.eno) && (this.dept.equals(e.dept));
        }
        return false;
    }
}
```

Run above program:- with the below test cases

Case(1):- remove equals method from Emp class:-

O/P :- false
false

Ans:- Becoz equals method will execute from Object class so Emp objs are compared with reference.

Case(2):- override equals method in Empclass by comparing eno & dept data in both objs

O/P :- true
true

Ans :- Employee objs are compared with data.

Case(3):- Add (Condition) an expression for comparing ename in equals method.

O/P :- true
false

Summary:- on contains, endsWith & remove methods of WTE:

1) When we see contains method in exam, called on any of the list implemented classes [vector, Stack, ArrayList, LinkedList], we must check below 2 things.

i) equals method overridden or not in this searching obj class if not overriden compare with reference, [Case(1)]. If overriden compare objs with data [Case(2)].

ii) we must compare only the variables data which is used in equals method comparision [Case(3)].

Q) What will happen if we don't override equals method in our subclass?
Ans:- No Compile-time error, No Runtime error but the problem is the obj added to collection will not find in Collection with new referenced object.
→ It is mandatory to override equals method in a class whose objs will add to collection.

Sample project on searching operation

Q) Write a pm pgm for adding BankAccount objs to ArrayList, these objs must find by their account no. and branch.

```
import java.util.*;
class AddingnSearchingCoBankAccount{
    BankAccount{
        p s v m(String[] args){
            ArrayList al=new ArrayList();
            al.add(new BankAccount(382501, "HB", 40000, "Ammerpet"));
            al.add(new BankAccount(382502, "BB", 20000, "Ammerpet"));
            al.add(new BankAccount(382503, "MB", 30000, "Reddiguden"));
            Sopln(al.contains(new BankAccount(382503, "MB", 30000, "Reddiguden")));
            //eu.equals(e1);
            //eu.equals(e2);
            //eu.equals(e3);
            internal calling order
            Sopln(al.contains(new BankAccount(382503, "LB", 50000, "Reddiguden")));
        }
    }
```

```
// BankAccount.java
class BankAccount{
    long accno;
    String accHoldername;
    double balance;
    String branch;
BankAccount(long accno, String accHoldername, double balance, String branch){
    this.accno = accno;
    this.accHoldername = accHoldername;
    this.balance = balance;
    this.branch = branch;
}
    public boolean equals(Object obj){
        if (obj instanceof BankAccount){
            BankAccount b=(BankAccount) obj;
            return (this.accno = b.accno) && (this.branch.equals(b.branch));
        }
        return false;
    }
}
```

23/09/15

Q) Difference b/w contains and index of() in searching an object :-

```
public boolean contains(Object obj)
public int     indexOf(Object obj)
```

→ Contains method will return boolean value to tell us given object found or not. where as index of() will return the index of given object in collection if found.

→ If not found it returns -1.

→ So using contains method we can only find obj in collection, further we can't retrieve it.

→ whereas using indexof method we can retrieve the obj from collection with the help of get method by passing returned index.

for example :- we have an ArrayList with BankAccount objs. A customer want to know account exist in this branch or not. In this case we must use contains method.

→ customer wanted to know account available or not, if available he wanted to get balance information and last transaction information etc. from this account. In this situation indexof() and get() method refer project no. 2 in fb.com\Javaetutorial.

→ literal concept (pooling) is applicable only these 3 classes are String, wrapper class & Java.lang.class apart from these classes the obj can be created only by using new keyword.

Eg :- al.add("a");

al.add(new A());

→ when we display ArrayList object it internally calls the toString() method.

→ for searching we can use equals() method.

→ If we can't override toString() method then it can displays classname @ hashcode.

(4) Working with remove method :-

→ Remove() method is used for removing object from the collection. we have two overloaded remove methods in List collection.

→ they are

1) public boolean remove(Object obj)

2) public ~~int~~ boolean removeAt(int index)
object

→ remove(object) parameter method will remove given objs matched objs from arraylist.

→ remove method also will use equals method for finding matched objs in collection for the given objs.

→ If obj found, it removes the obj from collection, returns true.

→ If obj not found it doesn't remove any obj, returns false.

→ we must overridde equals method in the class whose objs we added to the collection.

→ If we don't overridde equals method obj not found, obj not removed eventhough it is available in collection.

→ remove of index method will remove the given index obj from this collection, and returns this removed obj reference.

Rule:- the passed index value must be in the range of 0 to al.size()-1 if it is lesthan 0 or greater than size()-1 we will get exception IndexOutOfBoundsException

Q) what is the difference b/w al.remove(5) and al.remove(new Integer(5))?

Ans:- al.remove(5) will remove 5th index obj becos the given argument 5 is matched with remove of int parameter method so 5th index obj removed.

→ al.remove(new Integer(5)) will remove integer obj with data 5, by comparing objs in collection from 0 index until (object) this integer obj found using Integer class equals method.

Q) what is the o/p from below program?

(Same)

```
Al al = al new Al();
al.add("a");
al.add("b");
al.add("a");
al.add("c");
sopln(al);
al.remove(new String("a"));
sopln(al);
```

o/p:- [a, b, a, c]
[b, a, c]

Here equals method executed from string class, data compared, first string obj is removed from collection becos their data is same.

al.remove(new String("a"));
↳ sele (searchingelement)
→ sele.equals(e);
↳ executed from string class
↳ return true, e in removed from al.

Case (ii):- Removing Integer objs from ArrayList

```
Al al=new Al();
al.add("5");  - 0
al.add(5);    - 1
al.add(6);    - 2
al.add(7);    - 3
al.add(5);    - 4
al.add(8);    - 5
al.add(9);    - 6
al.add(10);   - 7
Sopln(al);
al.remove(5); → 5th index obj (8) is removed
// al.remove(10); → Here 10th index obj should remove, since 10th index
                    not available IOBE raised
Sopln(al);
al.remove(new Integer(5)); → Integer obj with data 5 is removed,
                             only 1st occurance.
Sopln(al);
al.remove(new Integer(10)); → Integer obj with data 10 is removed.
Sopln(al);
```

Index

O/P:- [5, 5, 6, 7, 5, 8, 9, 10] (Its string obj 5)

[5, 5, 6, 7, 5, 9, 10]

[5, 6, 7, 5, 9, 10]

[5, 6, 7, 5, 9]

Case (iii):- Removing character objs

```
Al al = new Al();
al.add('a');
al.add('b');
al.add('c');
Sopln(al);
// al.remove('b'); → 98 index obj should be removed becoz this method
                     call will be matched with remove of index method
Sopln(al);           for removing character obj ('b') we must create character
al.remove(new Character('b'));  class obj explicitly as shown next line
Sopln(al);
```

O/P:- [a, b, c]

Exception:

[a, c]

Note:- The given argument matching with remove of index only for the types byte, short, char, int. Becoz these type arguments are matched with remove of int method.

→ So for calling remove method, for removing Byte, Short, Character, Integer objs from Collection we must create their objs explicitly as shown in above case(2) & case(3).

→ For other datatypes long, float, double, boolean we no need to create explicit obj for remove method call becoz these types are not matched with remove of int, they are matched with remove of obj.

Case(iv) :- removing long, double, float, boolean objs.

```
Ah al = new AL();
al.add(5);
al.add(5L);
al.add(7.0);
al.add(8.0f);
al.add(true);
al.add(false);
al.add('a');
al.add(true);
sopln(al);
al.remove(5); → remove of 5 5th index obj removed.
sopln(al);
al.remove(5L); → remove of long obj with data 5L is removed.
sopln(al);
// al.remove('a'); → X RE: IOOBE becoz 97 index not existed.
// sopln(al);
al.remove("a"); → string 'a' not available, no element removed.
sopln(al);
al.remove(8.0); → double type 8.0 not available so no element removed.
sopln(al);
al.remove(7.0); → double 7.0 is removed.
sopln(al);
al.remove(true); → first true is removed.
sopln(al);
```

O/P :- [5, 5, 7.0, 8.0, true, false, a, true] (5th index obj)
[5, 5, 7.0, 8.0, true, a, true]
[5, 7.0, 8.0, true, a, true]
Exception: IOOBE
[5, 7.0, 8.0, true, a, true]
[5, 7.0, 8.0, true, a, true]
[5, 8.0, true, a, true]
[5, 8.0, a, true]

Case(v) :- Removing StringBuilder objs

```
Ah al = new AL();
al.add(new StringBuilder("a"));
al.add(new StringBuilder("b"));
StringBuilder sb = new StringBuilder("c");
al.add(sb3);
sopln(al);
al.remove(new StringBuilder("a"));
sopln(al);
al.remove(sb3); → sb3("c") is removed becoz same reference of obj.
sopln(al);
```

O/P :- [a, b, c]
[a, b, c]
[a, b]

→ no obj removed from collection becoz equals method not overridden in StringBuilder class so references are comparing.

Case(vi):- removing userdefined objs (A class objs).

```
Ah al= new Ah();
al.add(new A(5));
al.add(new A(6));
al.add(new A(7));
Sopln(al);
al.remove(new A(7));
Sopln(al);
```

O/P:-

20/09/15

//case(1): toString() and equals() not overridden

```
class A {
  int x;
  A(int x) {
  this.x=x;
  }
}
```

→ with this 'A' class 'A' obj not removed from collection and classname@Hashcode is displayed.

//case(2): toString() overridden

```
class A {
  int x;
  A(int x) {
  this.x=x;
  }
  public String toString() {
  return "A("+x+")";
  }
}
```

→ with this changed A class also A obj is not removed from collection. obj data is displayed as below [A(5), A(6), A(7)].

//case(3): toString() and equals() method overridden

```
class A{
    int x;
    A(int x){
    this.x=x;
    }

    public boolean equals(Object obj){
//  System.out.println(this+" and "+obj+" compared");
    if (obj instanceof A){
        A a=(A)obj;
        return(this.x==a.x);
    }
    return false;
    }
    public String toString(){
    return "A("+x+")";
    }
}
```

→ with this changed code obj is removed from collection

//case(4): equals() is overridden by returning true literal directly.

```
class A{
    int x;
    A(int x){
    this.x=x;
    }
    public boolean equals(Object obj){
    return true;
    }
}
```

→ with this equals() code 1st obj in the collection will be removed not the exact object

→ In above program A(5) is removed with A(7)

(5) Replacing an object in ArrayList:-

→ In list implemented classes V, S, AL, LL we can replace one obj with new object. for this purpose we have below method.

public Object set(int index, Object obj)

It replaces given index object with the given new object in collection, returns old object reference.

Q) Write a prg with Arraylist with 3 string objects replace 2nd object with its upper case.

Prg:-

```
//ReplaceTest.java
import java.util.*;
class ReplaceTest {
  public static void main(String[] args) {
    Arraylist al=new Arraylist();
    al.add("a");
    al.add("b");
    al.add("c");
    System.out.println("Original Elem: "+al);
    String s1=(String) al.get(1);
    String s2=s1.toUpperCase();
    al.set(1,s2);
    System.out.println("Changed Elem: "+al);
  }
}
```

O/P:-

Original Elem: [a, b, c]

Changed Elem: [a, B, c]

(6) Inserting new objects in AL:-

→ We can insert new obj in list implemented collections V, S, AL, LL using below method.

public void add(int index, Object obj)

Q) Write a prg with (collection) string objs "A", "B", "C", "D". After adding these string objs insert string obj "B" before "C".

Prg:-

```
//InsertTest.java
import java.util.*;
class InsertTest {
  P S V m(S[] args) {
    Arraylist al=new Arraylist();
    al.add("a");
    al.add("c");
    al.add("d");
    System.out.println("Original Elem: "+al);
    al.add(1,"b");
    System.out.println("Changed Elem: "+al);
  }
}
```

O/P:-

Original Elem: [a, c, d]

Changed Elem: [a, b, c, d].

al.add(size, "a") will append this element means stored end of collection.

Rule:- Insertion index value must be in b/w 0, size else it leads to IndexOutOfBoundsException.

```
import java.util.*;
class InsertTestCase {
    p s v m(S[] args) {
    Al al=new Al();
    Sopln(al);
    al.add(0,"b");  --> appended
    al.add(1,"c");  --> appended.
    Sopln(al);
    al.add(6,"d");  --> X RE:
    Sopln(al);
    }
}
```

What is the output from the below program:

```
//RemoveTestCase.java
import java.util.*;
class RemoveTest {
  P S V main(S[] args){
    AL al = new AL();
    for(int i=1; i<10; i++){
      Sopln(i*10);
    }
    sopln(al);
    al.remove(1);
    al.remove(2);
    Sopln(al);
  }
}
```

→ All 4 operations of collection are performed on list implemented classes.

Operations	List methods	Internal method
1) adding	l.add(o); (c)	Not Applicable (NA)
2) counting	l.size() (c)	NA
3) retrieving	l.get(index) (l)	NA
4) searching	l.contains(obj) (c)	obj.equals(ele)
	l.indexOf(obj) (l)	obj.equals(ele)
5) removing	l.remove(obj) (c)	obj.equals(ele)
	l.remove(index) (l)	NA
6) replacing	l.set(index, obj) (l)	NA
7) inserting	l.add(index, obj) (l)	NA

What is the o/p from below prg.

```
//RemoveTestCase.java
import java.util.*;
class RemoveTestCase {
    P S V m(S[] args) {
    Al al = new AL();
    for (int i=1; i<=10; i++) {
      al.add(i*10);
    }
```

10, 20, 30, 40, 50, 60, 70, 80, 90, 100

10, 20, 30

```
Sopln(al);
al.remove(1);
al.remove(2);
Sopln(al);
}
}
```

O/P :- [10, 20, 30, 40, 50, 60, 40, 80, 90, 100]

[10, 30, 40, 50, 60, 40, 80, 90, 100]

[10, 30, 50, 60, 40, 80, 90, 100]

Set Implemented classes :-

If we want to store only unique elements in collection we must use set implemented classes HashSet, LinkedHashSet, TreeSet.

→ HashSet doesn't maintain insertion order it will store objs in hashcode based order it wont maintain insertion order.

→ If we want to store objs in insertion order we must use LinkedHashSet class.

→ If we want to store objs in sorting order we must use TreeSet.

```
import java.util.*;
class HSLHSTS {
  p s v m(S[] args) {
    HashSet hs = new HashSet();
    LinkedHashSet lhs = new LinkedHashSet();
    TreeSet ts = new TreeSet();
    hs.add("a");
    hs.add("c");
    hs.add("b");
    hs.add("s");
    hs.add("d");
    Sopln(hs); → [a, b, c, d, s] → not insertion order
    lhs.add("a");
    lhs.add("c");
    lhs.add("b");
    lhs.add("s");
    lhs.add("d");
    Sopln(lhs); → [a, c, b, s, d] → insertion order
    ts.add("a");
    ts.add("c");
    ts.add("b");
    ts.add("s");
    ts.add("d");
    Sopln(ts); → [a, b, c, d, s] → sorting order
  }
}
```

Because collections are Runtime polymorphic classes, we can call add() common to all collections as shown in this store() method

```
store(hs);
Sopln(hs);
store(lhs);
Sopln(lhs);
store(ts);
Sopln(ts);
}

static void store(Set s) {
  s.add("a");
  s.add("e");
  s.add("b");
  s.add("s");
  s.add("d");
}
```

→ Below table shows the 4 collection operations and appropriate method to perform these operations on HashSet & LinkedHashSet.

Operations	HashSet, LinkedHashSet	Internal method
1) adding	hs.add(obj); (C)	→ obj.hashCode(), → == operator, → obj.equals(ele)
2) counting	hs.size() (C)	—
3) retrieving	hs.iterator() (C)	—
4) searching	hs.contains(obj) (C)	→ obj.hashcode(), → == operator, → obj.equals(ele)
5) removing	hs.remove(obj)	→ obj.hashCode(), → == operator, → obj.equals(ele)
6) replacing, 7) inserting	are not supported because Set donot have index	

HashSet Interview Questions:- (refer questions in previous pages before list).

Q1-A):- Hashset is used for storing only unique elements without considering storing order.

Q2-A):- Hashtable backed by HashMap instance

when we create HashSet obj, internally HashMap obj is created, elements are stored in this HashMap

Q3-A):- Default capacity - 16

Incremental capacity - double

load factor - 0.75

load factor is a measurement that specifies when hashtable capacity should increment. that measurement is 75% i.e after adding 75% entories (elements) its capacity will be increased.

Q4-A):- Homogeneous, heterogeneous and only unique elements.

Q5-A):- null allowed, only one. Because duplicate not allowed.

Q6-A):- ~~obj~~ Hashcode order

Q7-A):- Only sequential

Q8-A): Not synchronized collection

Q9-A): Not ordered collection

Q10-A): hashCode(), == operator, equals() in adding searching & removing operations

Linked Hashset FQs :-

Q1-A) :- When we want to store only unique objs in insertion order we must use LinkedHashSet.

Q2-A) :- A Doubly-linked list & hashtable backed by LinkedHashMap instance

Q3-A) :- Same as Hashset

Q4-A) :- Same as Hashset

Q5-A) :- Same as Hashset

Q6-A) :- Insertion order

Q7-A) :- Sequential order.

Q8-A) Same as HS

Q9-A) Same as HS

Q10-A) Same as HS

Q) How Hashset & LinkedHashSet can stop duplicate objs ?

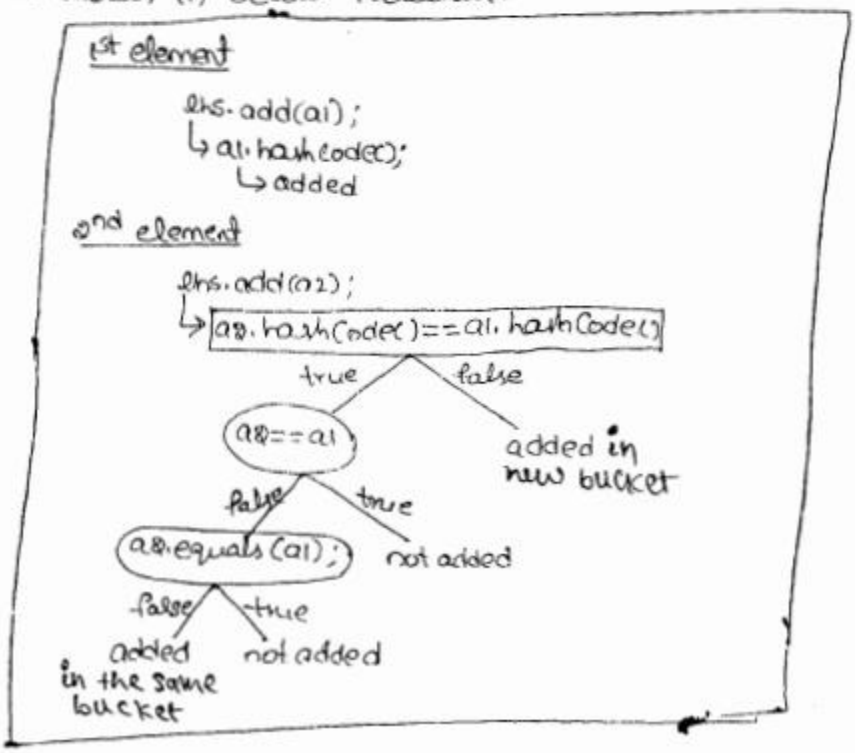
Ans :- By using HashCode, == operator and equals methods

→ The classes those contains a word hash will use HashCode, == operator & equals methods for stoping duplicate objs and also used for searching and removing correct element from the collection.

The procedure is :-

1) First hashCode() method on the current adding obj is called
2) Next "==" operator is used
3) Finally equals() method is called

as shown in below flowchart.



25/09/15

→ hashCode() method is used for storing same hashcode objs as one group

→ "==" operator is used for comparing the adding objs with its hashcode objs using reference.

→ equals() methods is used for comparing the adding objs with its hashcode objs using data.

Q) What will happen if we don't override hashCode() and equals() methods in adding objs class?

Ans:- Database duplicate objs are added, and also these objs will not found and remove further from collection with new referenced objs.

→ If we don't override hashCode() methods in subclass they will executed from Object class. then hashcode is generated using reference and objects are compared using reference. then only same referenced duplicate objs are not added.

→ So we must override hashCode() and equals() methods to adding object class for stopping data wise duplicate objs.

Right procedure for overriding hashCode() & equals() method?

→ From hashCode() method we must return the property, using which one group of objs are differentiated from another group of objs of same class.

→ In equals() method we must compare the properties, using which one obj is differentiated from another obj in the same hashCode group.

For example:

In Student class

→ From hashCode() method we must return "course"

→ In equals() method we must compare "course & rollnum".

In Employee class

→ From hashCode() method we must return "dept".

→ In equals() method we must compare "dept & eno".

In BankAccount class

→ From hashCode() method we must return "accType".

→ In equals() method we must compare "accType & accNum".

→ Explain HM/HT Internals:

→ add(), put(), contains(), remove() methods algorithm in HS/LHS/HM/LHM/HT collection objs :-

1) All above 5 objs internally uses "hashtable" data structure for storing objs.

2) "hash table" is internally uses another data structure called "bucket".

3) bucket is a collection of elements there have same hashCode. this the bucket is also a collection obj.

4) bucket is used for storing same hashCode objs as one group, so that comparing & searching will be fast because currently adding objs will be compared only with this objs hashcode group of elements.

5) new bucket will be created for every new hashcode obj. then this obj, will add to this bucket directly without any comparisions.

6) If bucket is already available with this currently adding obj's hasCode, then add() method will check whether this obj is unique or duplicate by using "==" operator and equals() method.

7) If "==" operator returns "true", this obj is reference wise duplicate obj, not added to collection.

8) If "==" operator returns false, adding obj is unique based on reference. but it may be duplicate based on state.

Then add() method will call equals method for comparing objs using its content.

9) If equals() method also return false, obj is added, unique on data wise.

10) If equals() method return true, obj is not added, duplicate on data wise.

Note :- Read all above 10 points with respect to previous page flowchart

Q) below prg shows add() method functionality and hashmap memory structure with string & integer objs.

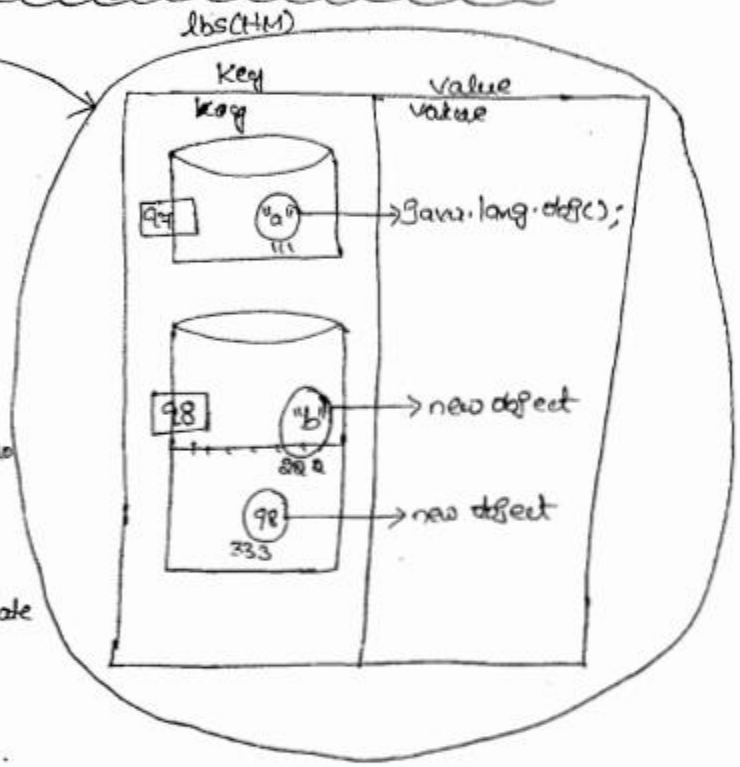
Ans:-

```
//LHSAddingTestCases.java

import java.util.*;

class LHSAddingTestCases {

    p s v m(String[] args) {
```

//case(1): adding string & Integer objs.

```
LinkedHashSet lhs1 = new LinkedHashSet();
lhs1.add("a");                  ← Added in new bucket
lhs1.add("a");                  ← Not added, duplicate (ref)
lhs1.add(new String("a"));      ← Not added, duplicate (data)
lhs1.add("b");                  ← added in new bucket
lhs1.add(98);                   ← added in "b" bucket
System.out.println(lhs1); → [a, b, 98]
```

Execution flow as per the 10 points explained in previous page



lhs1.add("a");
↳ s1.hashCode();
↳ 97 → bucket not found
↳ new bucket created
"a" added in this obj. bucket

lhs1.add("a");
↳ s2.hashCode();
↳ 97 → bucket found
↳ comparision done with objs exist in this bucket
↳ s2 == s1 true, not added, same reference obj, duplicate

lhs1.add(new String("a"))
↳ s3.hashCode()
↳ 97 → bucket found.
↳ s3 == s1 false, unique, reference wise
↳ s3.equals(s1) true, duplicate data wise

lhs1.add("b");
↳ s4.hashCode()
↳ 98 → bucket not found.
↳ new bucket created

lhs1.add(98);
↳ 98.hashCode(); → 98
bucket found.
↳ 98 == s4 → false, unique reference wise
↳ 98.equals(s4) → false, unique type wise

// Case(2): adding StringBuilder objs [In SB class hc() & eqs() methods are not overridden]

```
LinkedHashSet lhs2 = new LinkedHashSet()
StringBuilder sb1 = new StringBuilder("a");
      "       sb2 = "       "       ("a");
      "       sb3 = "       "       ("b");
      "       sb4 = "       "       ("c");
StringBuilder sb5 = sb4;
lhs2.add(sb1); <- [added in new bucket]
    // -> sb1.hashCode() -> 31168322 -> new object bucket
lhs2.add(sb2); <- [added in another newBucket]
    // -> sb2.hashCode() -> 14225372 -> new bucket
lhs2.add(sb3); <- [added in another new Bucket]
    // -> sb3.hashCode() -> 5433634 -> new bucket
lhs2.add(sb4); <- [added in another new bucket]
    // -> sb4.hashCode() -> 2430287 -> new bucket
lhs2.add(sb5) <- [not added be cause same ref obj]
    // -> sb5.hashCode() -> 2430287 -> bucket found
    // -> sb5==sb4 -> true, not added (same ref. obj).
sopln(lhs2);
```

// Case(3): adding custom objs to set.

→ For adding custom objs [our own class objs], we must override hashcode & equals() methods in our class.

→ Then only data wise duplicate objs are not added, further added obj will found in collection & removed from collection with new referenced obj.

→ we have 6 cases in adding & removing obj with equals() & hashcode() methods.

From the Below Test cases findout

1) How many objs are added
2) " buckets are created
3) " objs are removed.

Given

```
import java.util.*;
  class ExCollection {
      p s v m(S[] args) {
      LinkedHashSet lhs = new LinkedHashSet();
       lhs.add(new Ex(5,6));
       lhs.add(new Ex(5,6));

       Ex e3 = new Ex(5,6);
       Ex e4 = new Ex(5,6);
       Ex e5 = new Ex(6,5);
       Ex e6 = new Ex(7,8);
       Ex e7 = e6;

       lhs.add(e3);
       lhs.add(e4);
       lhs.add(e5);
       lhs.add(e6);
       lhs.add(e7);
       Sopln("No.of objs added:" + lhs.size());
       sopln(lhs);
       sopln();
       Sopln("It is removed:" + lhs.remove(new Ex(6,5)));
       Sopln("It is removed:" + lhs.remove(e7));
       Sopln("No of objs after remove:" + lhs.size());
       Sopln(lhs);
  }
}
```

verify hashCode() and equals() methods overridden(or) not

```
// below Ex class
class Ex {
    int x,y
      Ex(int x, int y) {
        this.x=x;
        this.y=y;
      }
    public String toString() {
        return "Ex("+x+", "+y+")";
    }
}
```

Find o/p in below cases:-

Case(i):- Both hashcode() & equals() methods are not overriden in Ex class.

Case(ii):- Only Equals() hashcode() method is overriden by returning 5;

Case(iii):- Only Equals() method is overriden by either true/false;

Case(iv):- Hashcode() method is overriden by returning 5; also equals() method is overriden by returning true;

Case(v):- In above case return false from equals() method.

Case(vi):- hashcode() method is overriden by returning x+y & and equals() method is overriden by comparing data.

Case(i)-A):- 6 buckets are created (all objs have different references)
6 objs are added in different buckets
1 obj is removed i.e; e6.

Case(ii)-A):- 1 bucket is created (hash code() returning same val 5)
6 objs are added in the same bucket
1 obj is removed i.e, e6

Case(iii)-A):- 6 buckets are created (different objs)
6 objs are added in different buckets
1 obj is removed i.e, e6

Case(iv)-A):- 1 bucket is created (hashcode returning same value).
1 object is added i.e; first added object
1st object is removed irrespective of reference & state.

Case v)-A) 1 bucket is created
6 objs are added in the same bucket
1 obj is remove i.e; e6.

Case vi)-A):- 2 buckets are created
Ex(5,6) Ex(6,5) Ex(7,8) are added
2 objs are removed, i.e: Ex(6,5) Ex(7,8)

Contract in Overriding hashCode() & Equals() method:-

→ we must override hashcode() & Equals() method in subclass to satisfy below Contract.

distinct case:-

hc is same ⇒ Equals() returns t/f

* hc is diff ⇒ Equals() should return false

continuation point from previous page

Eq() return true:-

* Eq() compare '2' objs return true ⇒ hc should be same.

Eq() compare 2 objs return false ⇒ hc() should be either same or different.

→ It is rule should be override Equals() & hashcode() methods in every class.

Ans:- No, we must override hashcode() & Equals() methods in the classes whose objs were adding to Collection.

For high performance:-

x(generate different hashcode per each object.)x

generate different hashcode per each object, so no comparisions, object is directly added in different new bucket.

But According to business:-

we must generate same hashcode for related objs, and different hashcode for unrelated objs of the same class. so that searching element is fast minimum comparisions.

So According to business:-

Develop a project for adding student objs to lhs Collections and further search and remove the given student obj from L.H.S.

Note:- you must built student class, for storing its objects in the pattern is per real time school, classroom based arrangement. -that means, student obj in LHS should group on course wise, should not allow two students objs those have same course & rollnum.

So from hashcode() method we must return course & in Equals() method we must compare course & rollnum;

Coding:-

Number of class: 3

1. student class
2. School class
3. Course Map class.

In student class: we must

1) create fields for storing students data
2) create a param constructor for initializing
3) create HashMap obj for storing course name and pts num
   (this program will be in course map class)
4) Override hashcode() & equals() methods.
5) From hashcode() method we must return course num
6) In equals() method we must compare rollnum & course.

In school class:- we must

1) Define main method
2) In main method create LHS object
3) create student objs with unique & duplicate content.
4) Add student objs to LHS.

In course map class:- we must

1) create HashMap obj with private static ref variable.
2) Define static {} Inside static {} initialized HashMap obj with coursename & pts number entries.
3) Define get CourseNum() method for retrieving given course number.

project :-

```
package com.naresh.it.helper;

public class CourseMap {
    private static HashMap courseMap = new HashMap();

// static Block
static {
    courseMap.put("CRT",1);
    courseMap.put("C",2);
    courseMap.put("CORE JAVA",3);
    courseMap.put("Adv JAVA",4);
    courseMap.put("ORACLE",5);
}

public int getCourseNum(String course) {
    return (Integer) courseMap.get(course.toUpperCase());
}
}
```

```
package com.naveshit.bean;
  public class Student {
    private int rollnum;
    private String sname;
    private String course;
    private double fee;

  public Student(int sno, String sname, String course, double fee) {
      this.sno = sno;
      this.sname = sname;
      this.course = course;
      this.fee = fee;
  }
  @override
  public int hashCode() {
    return CourseMap.getCourseNum(course);
  }
```

check previous page for logic

```
  ~~public static void main(String[] args) {~~
    ~~logger~~

  @override
  public boolean equals(Object obj) {
    if (obj instanceof student) {
      student s = (Student) obj;
      return (this.course.equals(s.course)) && (this.rollnum == s.rollnum);
    }
    return false;
  }
  @override
  public String toString() {
    return "InStudent("+rollnum+", "+sname+", "+course+","+fee+");
  }
}
package com.naveshit.user;
  public class School {
      p s v m(S[] args) {
      LinkedHashSet lhs = new LinkedHashSet();
```

```
lhs.add(new Student(101,"Hari", "corejava",1000);
lhs.add(new Student(102,"Balayya","corejava",1000);
        :
lhs.add(new Student(101,"Hari","oracle",1000);
lhs.add( "      "      "      "      "      " );
lhs.add(new Student(101,"Balayya","oracle",1000);
lhs.add(new student(102,"Balayya","corejava",1000);
}
  Sopln(lhs);
  lhs.remove(new Student(102,"Balayya","core java",1000);
  sopln(lhs);
}
```

2nd project:-

project title:- using collection object as mini database

```
//AccountBean.java
package com.naresh.bean;
public class AccountBean{
 private long accnum;
 private string accHName;
 private double balance;
 private string accType;
 public long getAccNum(){
  return accNum;
 }
  public void setAccNum(long accNum){
   this.accNum=accNum;
  }
   public string getAccHName(){
    return accHName;
   }
   public void setAccHName(String accHName){
    this.accHName = accHName;
   }
    public double getBalance(){
     return balance;
    }
```

```
public void setBalance (double balance) {
    this.balance = balance;
  }
    public string getAccType() {
     return accType;
    }
     public void setAccType(string accType) {
      this.accType =accType;
     }
     @override
     public boolean equals (Object obj) {
     if(obj instanceof AccountBean) {
      AccountBean bean = (AccountBean) obj;
       return (this.accNum == bean.accNum)
                    &&
              (this.accType.equals (bean.accType));
     }
       return false;
      }
  }
 //CollectionDB.java
package com.nareshit.db;
import java.util.ArrayList;
import class Collection
import com.nareshit.bean.AccountBean;
public class CollectionDB {
private ArrayList accList=new ArrayList();
public void addAccount (AccountBean bean) {
 accsList.add(bean);
}

 public AccountBean getAccount(long AccNum. string accType) {
 AccountBean resAccBean=null;
 AccountBean searchBean = new AccountBean();
 searchBean. setAccNum(accNum);
 Search Bean.setAccType (accType);
 int index = accsList.indexOf(searchBean);
  if(index!=-1) {
  resAccBean = (AccountBean) accsList.get(index);
 }
  return resAccBean;
}
}
```

```
// clerk.java
package com.nareshit.user;
import java.util.Scanner;
import com.nareshit.bean.AccountBean;
import com.nareshit.db.CollectionDB;
public class clerk {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  CollectionDB db = new CollectionDB();
  while(true){
   Sop ln("\n Choose option");
   Sop ln("1. create Account");
   Sop ln("2. Get Balance");
   Sop ln("3. Exit");
   Sop ln("Enter option");
   int option = sc.nextInt();
   sc.nextLine();
   switch (option) {
Case(1):-
    AccountBean bean = new AccountBean();
    Sop ln("accNum:");
    bean.setAccNum(sc.nextLong());
    sc.nextLine();
    Sop ln("accHName");
    bean.setAccHName(sc.nextLine());
    Sop("balance");
    bean.setBalance(sc.nextdouble());
    sc.nextline();
    Sop("acctype");
    bean.setAccType(sc.nextLine());
    db.addAccount(bean);
    Sop ln("Account created");
    break;
```

Case(2):-

```
        Sop("accNum");
        long accNum = Scn.nextLong();
        Scn.nextLine();
        Sop("accType");
        String accType = Scn.nextLine();
        bean = db.getAccount(accNum, accType);
        if(bean != null){
         sopln("Current Balance:" + bean.getBalance());
        } else{
          Sopln("Account Number is wrong");
        }
          break;
```

Case(3):-

```
        break loop;
        default:
          Sopln("Invalid option");
        }
      } //while(true)
      Sopln("\n***** Thank you, visit again *****");
      }
    }
```

28/09/15

TreeSet; Comparable, Comparator :-

→ TreeSet is used for storing elements in sorting order either in ascending or descending order based on the adding objs

→ TreeSet doesn't have any sorting order this sorting order should be define by the class whose objs are adding to treeset.

→ The default sorting order of TreeSet is that objs those are adding to the default sorting order of object those are adding to treeset.

→ For storing objs in sorting order treeset internally uses either of the below two interfaces

1) Comparable (or)
2) Comparator.

→ Treeset uses these 2 interfaces because the objs sorting logic will be changed from one type of obj to another type of obj so the sorting order logic should be supplied by adding object class programmer. Through interface we can force subclass programmer to implement a method logic.

→ TS.add() method internally calls these interfaces methods. Based on the value returned from these methods, TS will sort objs.

→ add() method will sort objs based on return value as below.

- If return number is 0,
  ↳ element is not added.
- If return number is -ve
  ↳ element is added LEFT to tsele.
- If return number is +ve
  ↳ element is added RIGHT to tsele.

If there is already element existed RIGHT to tsele, comparision is repeated with the next element & then decides the position of currently adding element.

Adding custom objects to Treeset with Comparable Interface:-

→ Comparable interface has a method CompareTo(object) for defining / implementing objs comparision logic

```
public int compareTo(Object obj)
```

→ For adding our class objs to TS, we must also derive our class from Comparable interface and should implement CompareTo() method with sorting order logic.

→ Inside TS.add() method given obj will be cast to Comparable interface.

→ Algorithm / Procedure for adding custom objs to TS:

1) we must derive class from Comparable interface.

2) we must implement CompareTo() method.

Inside CompareTo() method sorting logic we must use "-" operator for comparing objs to generate a number.

3) When we add objs to TS class, inside add() method

i) Given obj will be cast to Comparable type

ii) Invokes CompareTo() method using this obj by passing already added objs.

iii) Then this adding obj will be stored in TS per the value returned from CompareTo() method as shown above.

→ Below diagram shows sample code of TS.add() method, and compareTo() method logic for storing Emp objs in emp wise.

```
class TS {
public boolean add(object obj){
Comparable cmp=(Comparable)obj;
if(this.isEmpty()){
  print obj added
} else {
int pos=cmp.compareTo(ele);
if(pos<0){
  obj added LEFT to ele
  break;
} else if(pos>0){
  obj added RIGHT to ele
}
```

```
class Employee implements Comparable {
  int eno;
  String ename;
  String dept;
  double sal;
  double exp;
public int compareTo(object o){
  Emp e=(Emp)o;
  double ed=this.exp-e.exp;
  if(ed==0){
    return this.eno-e.eno;
  }else if(ed<0){
    return -5;
  }else {
    return 5;
  }
}
```

ep, el

0/+ve/-ve

Assignment

Q) Date & Time working functionalities in java

29/09/15

Note :- String class, all wrapper classes are subclasses of Comparable interface so we can add these classes objs in treeset directly. In these classes compareTo() method has overriden to store their objs in ascending order [logic will be current obj - Argument obj].

→ String buffer and String builder classes are not subclasses of Comparable interface so we can't add these class objs in treeset it leads to CCE

Below program shows adding string objs to treeset.

```
class TSwithStringobj {
  P S V main(String[] args){
    TS ts=new TS();
    ts.add("a");
    ts.add("c");
```

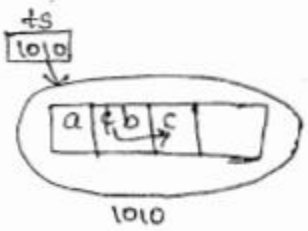
→ "c".compareTo("a");

"c"-"a"

99-97

+ve

```
ts.add("b");  ← +ve
    → "b".compareTo("a") ; ⇒ +ve ⇒ RIGHT
    → "b".compareTo("c") ; ⇒ -ve ⇒ LEFT
-ve
System.out.println(ts); [a,b,c]
    }
}
```



Adding custom objs to treeset :-

Step(1) :- Implement class dealing from Comparable interface

Step(2) :- Implement compareTo() method by returning int value 0 to -ve number & +ve number.

→ we have 5 test cases in implementing compareTo() method

Case(i) :- return ZERO directly

Only first element is stored

Case(ii) :- return +ve number directly

All objs are stored in insertion order including duplicates

Case(iii) :- return -ve number directly

All elements are added in reverse insertion order including duplicates.

case(iv) :- return (this.field - arg.field)

Only unique elements are added in ascending order.

case(v) :- return (arg.field - this.field)

Only unique elements are added in descending order.

Rule in implementing compareTo() method :-

→ we shouldn't use "instanceof" operator in condition for casting argument obj to current class type. becoz treeset shouldn't allow heterogeneous object.

→ Below Example shows adding custom obj class to TS find o/p for all above 5 cases.

```
//TSwithCustObj.java
    import java.util.*;
    class TSwithCustObj {
        p s v main(String[] args) {
```

```
TreeSet ts = new TreeSet();
    ts.add(new Sa(4));
    ts.add(new Sa(3));
    ts.add(new Sa(5));
    ts.add(new Sa(2));
    ts.add(new Sa(2));
    Sopln(ts);
  }
}
//Sa.java
class Sa implements Comparable{
    int x;
    Sa(int x){
      this.x=x;
    }
    public String toString(){
      return "Sa("+x+")";
    }
    public int compareTo(Object obj){
      //insert code here
    }
}
```

Q) from the below cases identify how many objs are added to TS, and in which order?

Case(1) :- return directly ZERO, i.e; "return 0".

Ans:- [Sa(4)]

Case(2) :- return directly 5, i.e; "return 5;"

o/p:- [Sa(4), Sa(3), Sa(5), Sa(2), Sa(2)]

Case(3) :- return directly -5, i.e; "return -5;"

o/p :- [Sa(2), Sa(2), Sa(5), Sa(3), Sa(4)]

Case(4) :- ~~or~~ AO-CO state is returned i.e; return this.x-s.x;

o/p:- [Sa(3), Sa(3), Sa(4), Sa(5)]

Case(5) :- AO-CO state is returned i.e; return s.x-this.x;

o/p :- [Sa(5), Sa(4), Sa(3), Sa(2)]

Note :- Case (4) & Case (5) we will implement in project.

→ Case (1), (2), (3) - they are written test cases we don't implement in project.

Understanding Comparator interface :-

→ Comparator is used for providing custom sorting order logic of an existing class objects.

→ Using comparator we can provide three logics

i) - for storing comparable objs in their reverse natural sorting order

ii) - for storing comparable objs on different property

iii) - for storing non-comparable objs.

→ Assume Employee class has sorting order logic for storing its objs on name based ascending order. This sorting logic is called natural sorting order for providing natural sorting order Employee class should be subclass of Comparable interface then Employee class is called Comparable obj.

→ If you want to store Employee obj name wise descending order or want to store different property wise let us say on exp/sal/joining date, we must develop custom comparator using java.util.Comparator interface.

→ The subclass of Comparator interface is called CustomComparator. This CC checks should pass to TS obj. Then Employee objs are stored as per CC sorting order.

Rule :- So, for adding an object to treeset

1) either it should be a subclass of Comparable interface (or)

2) It should be supplied with custom Comparator.

else TS throws CCE.

→ If we want to store string & all wrapper class obj in descending order we must develop Custom Comparator.

Storing String objs in descending order in TS using customComparator :-

Procedure :-

Step(1) :- Define a class from Comparator interface.

Step(2) :- Implement compare(o1, o2) method in this method cast parameters to the class whose objs we want to compare. Then return result value. this case cast to String class.

Step(3) :- pass this customComparator obj as argument to TS object Constructor. Then add method will compare adding object with treeset using compare method of this CC.

Compare() will be called internally as below

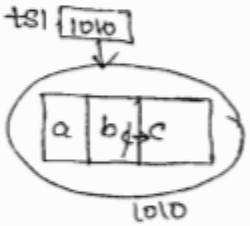
CustomComparator.Compare(addingobject, tsele)

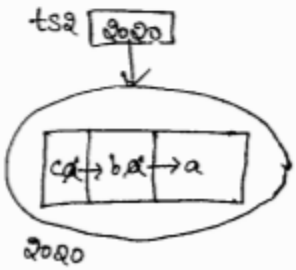
Code:- //StringComparator.java

```
class StringComparator implements java.util.Comparator{
    public int compare(Object o1, Object o2){
        String s1 = (String)o1;
        String s2 = (String)o2;
        return s2.compareTo(s1);
    }
}
```

→ Below program shows adding string objs with NSO (Natural sorting order), CSO (custom sorting order).

```
// TSwithStringObjs.java
import java.util.*;
class TSwithStringObjs {
    p s v main(String[] args) {
        Treeset ts1 = new Treeset();
        ts1.add("a");
        ts1.add("c");
            // "c".compareTo("a");
        ts1.add("b");   // executed from String class
            // "b".compareTo("a");
            // "b".compareTo("c");
        sopln(ts1);   // [a, b, c]
    }
    Treeset ts2 = new Treeset(new StringComparator());
        ts2.add("a");
        ts2.add("c");
            // sc.compare("c","a");  a-c => 97-99.
                // executed from SC class
        ts2.add("b");
            // sc.compare("b","c");  c-b => 99-98
            // sc.compare("b","a");  a-b => 97-98
        sopln(ts2);   // [c, b, a]
    }
}
```





→ Below prg shows defining custom comparator for storing non-comparable objs for example StringBuilder.

```
//SBComparator.java
class SBComparator implements java.util.Comparator {
    public int compare(Object o1, Object o2) {
        StringBuilder sb1 = (StringBuilder)o1;
        StringBuilder sb2 = (StringBuilder)o2;
        String s1 = sb1.toString();
        String s2 = sb2.toString();
        return s1.compareTo(s2);
    }
}
```

→ Comparable method is not available in StringBuilder class. Comparing StringBuilder data is equal to comparing string data so converted to string

```
//TSwithSBobjs.java
import java.util.*;
class TSwithSBobjs {
    P s v m(S[] args) {
    // TreeSet ts = new TreeSet();  → this obj causes CCE
    TreeSet ts = new TreeSet(new SBComparator());
    ts.add(new StringBuilder("a"));
    ts.add(new StringBuilder("c"));
        //sbc.compare(sb2, sb1);
    ts.add(new StringBuilder("b"));
        //sbc.compare(sb3, sb1);
        //sbc.compare(sb3, sb2);
        Sopln(ts);  → [a, b, c]
    }
}
```

30/09/15

Difference b/w Comparable, comparator interface :-

Comparable	Comparator
1) It is used for developing natural sorting comparision logic.	1) It is used for developing custom sorting comparision logic.
2) It has compareTo(object) method.	2) It has compare(object, object) and equals(object) methods.

3) It is available in java.lang package

4) TreeSet use Comparable.compareTo() method if its object is created using non-parameterized constructor.

5) TS calls compareTo() method using adding object by passing already we added object in TS.

Ex:- e2.compareTo(e1);

3) It is available in java.util package

4) TS uses Comparator.compare() method if its object is created using Comparator parameter constructor by passing custom Comparator object.

5) TS calls compare() method using the registered Comparator obj by passing current adding obj as first argument and already added element as second argument.

Ex:- Cmpr.compare(e2,e1);

Q) Why compareTo() has one parameter and compare() has two parameters?

Ans:- CompareTo() method will be implemented in the same adding obj class, so among two objs one obj is send as current obj, another obj is sent as argument object. Hence compareTo() method has only one parameter.

→ compare() method will be implemented in the other class, not in adding object class. So the two objs must be passed as arguments. Hence Compare() method has two parameters.

Q) Develop a project for storing student object in sorting order class wise, within the class sort students in rollno wise if student rollnumber of class is equal to an existing student rollno class don't add finally display all students details.

→ In this project we must develop 3 classes.

1) CourseMap.java
2) Student.java
3) School.java

```
//CourseMap.java
  Collect code from previous project.

//Student.java
package com.naresh.bean;
import com.naresh.helper.CourseMap;
  Public class Student implements Comparable {
    private int rollnum;
    private String sname;
    private string course;
    private double fee;
    private double height;
```

```
public Student(int rollnum, String sname, String course, double fee, double height){
    this.rollnum=roll.num;
    this.sname =sname;
    this.course =course;
    this.fee    = fee;
    this.height = height;
  }
   @override
     public int compareTo(Object o){
      Student s= (Student)o;
      int foCourseNum= CourseMap.getCourseNum(this.course);
      int soCourseNum = CourseMap.getCourseNum(s.course);
      int courseDiff= foCourseNum - soCourseNum;
      if( courseDiff!=0) {
         return courseDiff;
       }else {
         return this.rollnum- s.rollnum;
       }
     }
   @override
     public String toString(){
      return "\nStudent("+rollnum+","+name+","+course+","+fee+","+height+")";
     }
  }

  //School.java
package com.naveshit.user;
import java.util.TreeSet;
import com.naveshit.bean.Student;
  public class School{
    public static void main(String[] args){
      TreeSet ts=new TreeSet();
       ts.add(new Student(101, "Hari", "CoreJava", 1000, 6));
       ts.add(new Student(101, "Hari", "Oracle", 1000, 9));
       ts.add(new Student(101, "Hari", "CoreJava", 1000,6));
       ts.add(new Student(102, "sampu", "CoreJava", 1000, 5.9));
       ts.add(new Student(103, "[illegible]", "CoreJava", 1000, 5.9));
       ts.add(new Student(108, "Hari", "Oracle", 1000, 5);
       ts.add(new Student(103, "Mahesh", "CoreJava", 1000, 6.4));
       ts.add(new Student(101, "Balayya", "Oracle", 1000, 8));
```

```
Sopln(ts);
Sopln(); → ts.remove(new student(101,"Hasi",crack"coreJava",1000,6));
          Sopln(ts);
```

→ Create a custom Comparator for storing student objs in the reverse of above natural sorting order.

```
Study
package com.naresh.custom.cmpr;
import java.util.Comparator;
                                  Reverse Natural Sorting order
public class StudentRNSOComparator implements Comparator {
@override
public int compare(object o1, object o2) {
    Student s1=(Student)o1;
    Student s2=(Student)o2;
return s2.CompareTo(s1);
}
}
```

→ for testing this comparator in School.java pass this class obj to TreeSet Constructor as shown below.

```
TreeSet ts = new TreeSet(new StudentRNSOComparator());
```

→ Create CustomComparator to store student objs on height wise. 2 students can have same height if their height is same sort them on rollnum wise

```
package com.naresh.customcmpr;
import java.util.Comparator;
import com.naresh.bean.Student;
import com.naresh.helper.CourseMap;
public class StudentHeightComparator implements Comparator {
@override
public int compare(object o1, object o2) {
    Student s1=(Student)o1;
    Student s2=(Student)o2;
    int foCourseNum = CourseMap.getCourseNum(s1.getCourse());
    int soCourseNum = CourseMap.getCourseNum(s2.getCourse());
```

```
int courseDiff = foCourseNum - soCourseNum;
 if(courseDiff!=0){
  return courseDiff;
 }else{
   int rollnumDiff=s1.getRollnum()-s2.getRollnum();
   if(rollnumDiff!=0){
     return 0;
     }else{
      double heightDiff=s1.getHeight()- s2.getHeight();
       if(heightDiff!=0){
         if(heightDiff<0){
           return -5;
          }else{
           return 5;
          }
       }else{
         return rollnumDiff;
       }
     }
   }
 }
}
```

→ In school class replace TreeSet obj by passing this obj Comparator obj as shown below.

```
TreeSet ts = new TreeSet(new StudentHeightComparator());
```

→ Below table shows ts operations & mapping methods.

Operations	TreeSet	Internal method
1) adding	ts.add(obj);	→ obj.compareTo(ele) → Cmpr.compare(obj, ele)
2) Counting	ts.size()	—
3) retrieving	ts.iterator()	—
4) Searching	ts.contains(obj)	→ obj.compareTo(ele) → Cmpr.compare(obj, ele)
5) removing	ts.remove(obj)	→ obj.compareTo(ele) → Cmpr.compare(obj, ele)
6) replacing		
7) Inserting		

TS FAQS :-

Q1-A) :- for storing objs in sorting order Red-

Q2 - A) :- Red-black-tree

Q3-A) :- default capacity = 1
increment " = 1

Q4-A) :- homogeneous, unique, comparable type objs allowed and homogeneous, unique, non-comparable objs with Custom Comparator.

Q5-A) :- null is not allowed, it leads to NPE.
solo null is allowed only till Java 6.

Q6-A) :- Given objs sorting order.

Q7-A) :- sequential.

Q-8A) NOT Synchronized

Q-9A) ordered collection, sorting order

Q-10A) Comparable.compareTo(-)
(or)
Comparator.compare(-,-)

01/10/15

Q) What are the methods we must override in subclass for adding its objs in anyone of the collection and further searching & removing from collection?

Ans :- 3 methods.

1) equals()
2) Hashcode()
3) CompareTo()

→ Hashcode() & equals() method for hashset & linkedhashset and CompareTo() for TreeSet list all u subclasses use only equals methods for searching & removing elements.

→ In previous project in Student class we have overriden in only CompareTo() method, copy Hashcode() & equals() methods from project-3 into this class. Then we can say it's fullfledged implemented class for adding, searching & removing its objs from any one of the collection including map collection also.

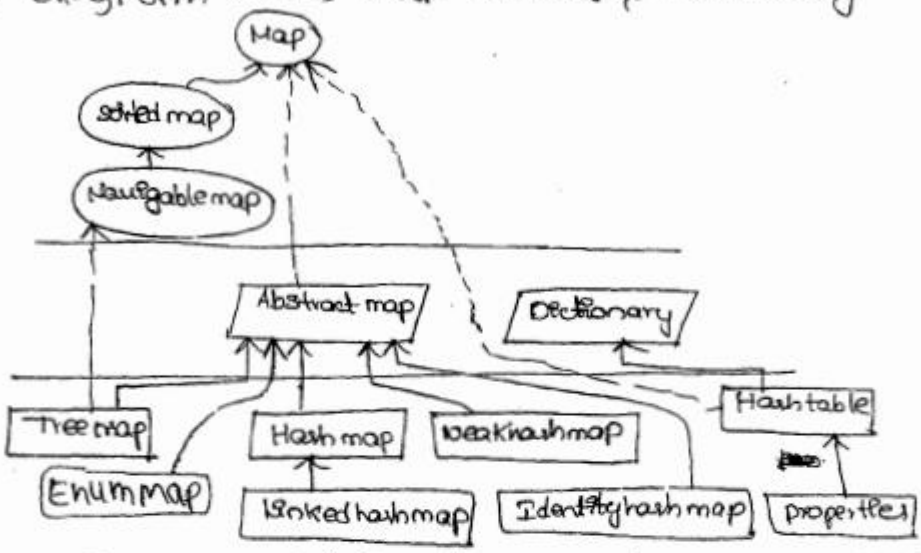
Note :- In String and all wrapper classes equals(), hashcode(), CompareTo() all these 3 methods are overriden so we can add these classes objs in any of the collection including map.

→ In StringBuffer and in StringBuilder classes hashCode(), equals() & compareTo() all three methods are not overridden. Hence we can not add and can not find StringBuffer and StringBuilder objects in Set, Map, SortedSet and SortedMap collections.

Working with Map class :-

→ When we want store objs in key, value pair format we must choose map type collection.

→ Below diagram shows Map subclasses hierarchy



→ Interface map is the root interface of all map type collection. It is not sub interface of Collection beco3 both interfaces functionalities are different.

→ Map contains only methods related to general operations adding, searching, removing, retrieving, counting.

→ The methods required to store and retrieve objs on sorting order are given in sortedmap & navigable map.

Map methods :-

1) boolean isEmpty()
2) object put(object key, object value)
   ↓
   Why return type
3) void putAll(Map m)
4) object remove(object key)
5) void clear()
6) boolean containsKey(object key)
7) Boolean containsValue(object value)
8) int size()
9) object get(object key)
10) Collection values()
11) Set keySet()
12) Set entrySet()
13) boolean equals(object o)
14) int hashCode()

Q) Why put() method return-type is object?

Ans:- If key already existed in this map it replaces existed value object with this new value obj, returns old value object.

→ All above 14 methods are common to all subclasses of map. These methods are implemented in subclasses as per they needs.

Q) Write a program to add number & name as key, values in (array) LHM collection?

Ans:-

```
//LHMDemo.java
import java.util.*;
class LHMDemo {
    p s v main(String[] args) {

    LinkedHashMap lhm = new LinkedHashMap();
                    {5, "MB"});
     Sopln(lhm.put(1, "BB")); → null
     Sopln(lhm.put(4, "PB")); → null
     Sopln(lhm.put(3, "MB")); → null
     "     "     "  (1, "CB")); → BB
     "     "     "  (2, "BB")); → null
/*   Object vObj = lhm.put(1, "CB"); → Sopln(lhm.put(5, "MB"));
      Sopln(vObj);

      vObj = lhm.put(2, vObj);

      Sopln(vObj);  */

      Sopln(lhm); → {1=CB, 4=PB, 3=MB, 2=BB, 5=MB}
    }
}
```

Map object memory structure:-

Key	Value
1	~~BB~~ CB
4	PB
3	MB ←
2	BB
5	MB ←

Note:- In a map object we can't store duplicate keys but we can store duplicate values. If we try to duplicate key old value will be replaced with new value of this key.

Q) What is the use of Collections and Arrays?

Ans:- These two classes are called utility classes.

→ These two classes contains only static methods

→ Collections class contains several static methods to perform operations common to all Collection & Map objs.

→ Arrays class contains static methods to perform several operations on Array objs.

→ Open API documentation of these classes (DVD/03 SUN API Docs / Java SE 1.6 docs (01)/ Api/java/util/

/Collections.html

/Arrays.html

then find several methods available in these classes, call them by passing Collection objs, observe o/p on Console.

05/10/15 Functionality of addAll(), containsAll(), removeAll(), retainAll(), addAll() method:- addAll() method:-

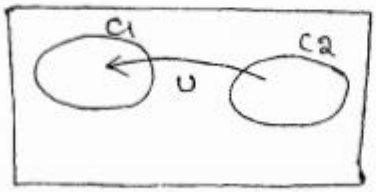
AddAll will copy given Collection element into current collection

for example

C1.addAll(C2);

All elements of C2 will copy into C1. then the elements of C2 will have references from C2 and also from C1.

Note:- If C1 is list type Collection, duplicate objs can copy from C2 to C1. If C1 is set type Collection, if C2 elements already available in C1, elements are not copied.



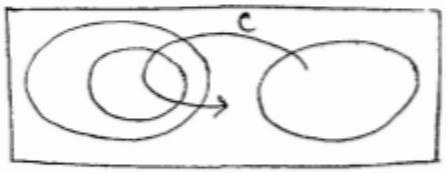
As per set theory addAll() method perform union (∪).

```
import java.util.*;
class addAllTest{
  public static void main(String[] args){
    AL al= new AL();
      al.add("a");
      al.add("b");
```

```
LHS lhs1 = new LHS();
lhs1.add("a");
lhs1.add("b");
AL al2 = new AL();
al2.add("d");
al2.add("e");
al2.add("a");
al2.add("b");
al1.addAll(al2);
lhs1.addAll(al2);
Sopln(al1); ---> [a,b,d,e,a,b]
Sopln(lhs1); --> [a,b,d,e]
}
}
```

ContainsAll():-

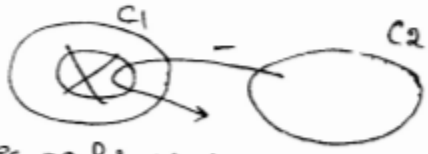
It is used for checking all elements of given collection are available in this collection or not. If available it returns true else returns false. Atleast one element missing also returns false.



ContainsAll() method perform subset(c) operation. When we pass empty Collection contains All() method returns true.

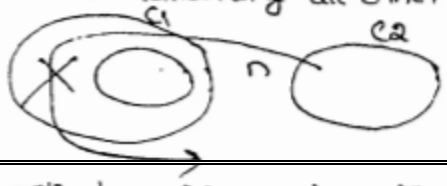
removeAll():-

It is used for removing given Collection elements from current Collection all occurances.



This performs '-' operation of set theory

retainAll():- This method will retain given Collection elements in current Collection, removes remaining all other objs in current collection.



This method performs intersection operation.

Retrieving objects from Collection & Map

List, Set (Collection) :-

→ There are 8 ways to retrieve objects from Collections (List, Set).

1) using index based → l.get(index)
2) using Enumeration → Collections.enumeration(c)
3) using Iterator → c.iterator()
4) using ListIterator → l.listIterator()
5) using for-each loop → for(:)
6) forEach(-) method
7) stream API
8) splIterator

available from 1.8v

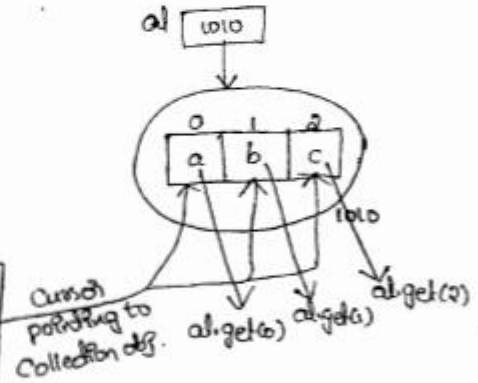
Map :-

→ retrieving objects from map we have 4 ways for retrieving objs from Collection & map.

1) using get(key)
2) using m.values()
3) using m.keySet()
4) using m.entrySet()

Retrieving objects from List using index :-

```
import java.util.*;
class RetrievingUsingIndex {
    p s v main(String[] args) {
        ArrayList al = new ArrayList();
        al.add("a");
        al.add("b");
        al.add("c");
        Sopln(al);
        for (int i=0; i< size(al.size()); i++) {
            Object obj = al.get(i);
            Sopln(obj);
        }
    }
}
```



Retrieving objs from List using Enumeration :-

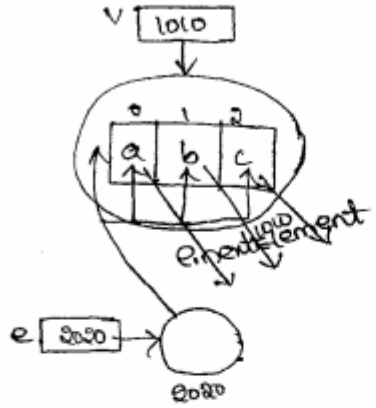
→ Enumeration is an interface because e.hasMoreElements() and e.nextElement() methods are having abstract class.

→ Enumeration is a legacy interface given in java 1.0. It is used for retrieving objs from vector and hashtable.

→ For checking element available or not & for retrieving element from Collection it has below 2 methods.

Program:-

```
import java.util.*;
class RetrievingUsingEnumeration {
    p s v m(S[] args) {
        Vector v=new Vector();
        v.add("a");
        v.add("b");
        v.add("c");
        System.out.println(v);
        Enumeration e=v.elements();
        while(e.hasMoreElement()) {
            Object obj=e.nextElement();
            System.out.println(obj);
        }
    }
}
```



i) public boolean hasMoreElements()

It verifies whether element available in the next location or not if available returns true else returns false.

ii) public Object nextElement()

this method meant for retrieving objs from collection when we call this method

→ cursor will move to the next location

→ returns element from this location.

Rule:- If element not available in this location this method will throw NoSuchElementException. So we must not call this method on empty location or an empty collection.

→ Enumeration is an Interface its subclasses are defined by sun inside Collection classes has inner class. To obtain this subclass obj we have a factory method elements defined in vector class.

public Enumeration elements

→ for retrieving elements from vector obj using Enumeration we must call 3 methods

1) v.elements() for obtaining enumeration obj
2) e.hasMoreElements() for checking element available or not
3) e.nextElement() for retrieving obj.

Q) How can we obtain Enumeration obj on Collection framework Collections like ArrayList, HashSet.

Ans:- Factory method given in Collections class.

we have 3 words

collections, Collection, Collections

Collections:→ It represents one group of objs (In English word)

Collection:- → It is the root interface of all types of collections. It supplies methods for performing collection operation.

Collections:- → It is a utility class, Contains methods to perform operations common to all Collection and map objs. such as obtaining Enumeration, Sorting, Searching etc..

The Method for obtaining Enumeration:-

```
public static Enumeration enumeration(Collection c)
```

→ we can call this method by passing list or set or queue objs.

Q) Write a program for retrieving elements from ArrayList using Enumeration.

Ans:-

```
import java.util.*;
class RetrievingEnumerationFromAL {
  P S V main(String[] args) {
    ArrayList al = new ArrayList();
    al.add("a");
    al.add("b");
    al.add("c");
    Sopln(al);
    Enumeration e = Collections.enumeration(al);
    while (e.hasMoreElements()) {
      Object obj = e.nextElement();
      Sopln(obj);
    }
    Sopln(al);
  }
}
```

→ memory diagram copy from previous page.

Working with Iterator:- → It is replace of Enumeration from Java 1.2 onwards.

```
import java.util.*;
class RetrievingUsingIterator {
    p.s v main(String[] args) {
        ArrayList al = new ArrayList();
            al.add("a");
            al.add("b");
            al.add("c");
            sopln(al);
            Iterator itr = al.iterator();
              while(itr.hasNext()) {
                Object obj = itr.next();
                sopln(obj); → itr.remove();
            }
                sopln(al);
            }
        }
```

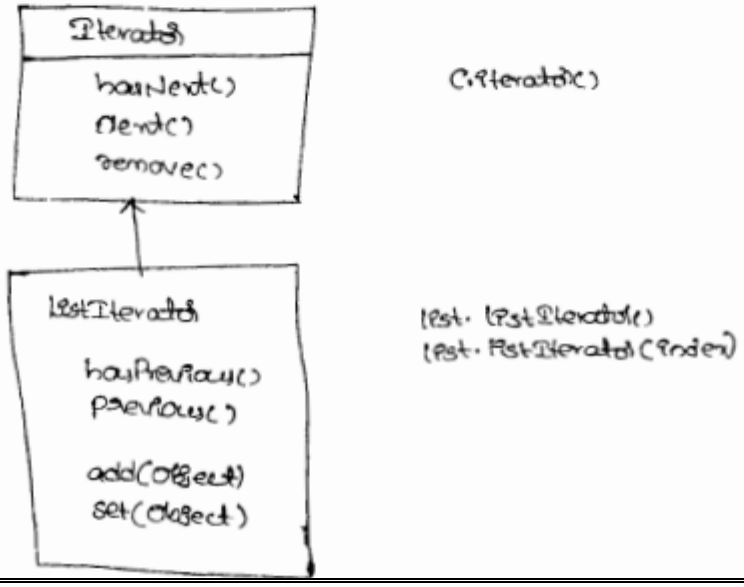
Obj/Ques

→ Enumeration is a Legacy Interface used for only retrieving elements from collection.

→ Iterator is a Collection framework interface used for retrieving and also remove the elements from Collection. [for retrieving get method is used only for list objs.]

→ Iterator is unidirectional (forward only) cursor.

→ ListIterator is a subclass of Iterator it is bidirectional cursor.



Q) Write a prg for creating ArrayList with 3 Integer String objs with 3 Integer objs add string & Integer objs alternatively then while iterating and insert Integer 20 after 1st Integer value, replace all strings with their uppercase, display result elements in reverse order.

Ans:-

```
import java.io.*;
class ListIteratorDemo {
  p s v main(String[] args) {
    ArrayList al = new ArrayList();
    al.add("a");
    al.add(1);
    al.add("b");
    al.add(2);
    al.add("c");
    al.add(3);
    Sopln(al);
    ListIterator litr = al.listIterator();
    int count=1;
    while(litr.hasNext()) {
      Object obj = litr.next();
      if(obj instanceof String) {
        litr.set((String)obj).toUpperCase());
      }
      else if(obj instanceof Integer) {
        if(count==1) {
          litr.add(20);
          count++;
        }
      }
    }
    Sopln(al);
    while(litr.hasPrevious()) {
      Sopln(litr.previous());
    }
  }
}
```

O/P:- [a, 1, b, 2, c, 3]

[A, 1, B, 2, C, 3]

[A, 1, B, 20, C, 3]

3
C
20
B
A

For-each loop :- It can be used from java 5 onwards in place of using Iterator

```
for (Object obj : al) {
    Sop.ln(obj);
}
```

Enhanced for loop :-

→ for loop syntax is enhanced for retrieving objs from array and Collection objs without using regular for loop and iterators.

→ Enhanced for loop is also called for-each loop because it iterates for each obj available in array or Collection its syntax

```
for (variable declaration : obj) {
    ---
    ---
}
```

← should be either array obj or Iterable interface subclass obj.

→ Collection interface is subclass of Iterable interface so for-each loop is allowed to apply on Collection type objs.

→ Map is not subclass of Iterable so we can't apply/use for-each loop on Map type objs.

→ **Why Enhanced for loop** :-

→ To reduce no. of lines of code development in retrieving objs from array and collection enhanced for loop is given.

→ For example for retrieving objs from any Collection same code used Iterator. To eliminate this code enhanced for loop given. Below diagram shows retrieving objs from ArrayList using Iterator (lengthy code) its equivalent for-each loop (short code).

```
Iterator itr = al.iterator();
while(itr.hasNext()) {
    Object obj = itr.next();
    Sop.ln(obj);
}
```

← Common code eliminated using for-each loop

```
for (Object obj : al) {
    Sop.ln(obj);
}
```

3 rules on for-each loop:-

1) object type should be array obj or Iterable type.

i) String s1="a";
for(Object obj : s1) {} → XCE: String is not Iterable type

ii) String[] sa={"a","b"};
for(String s:sa){} → ✓

iii) LHS lhs=new LHS();
for(Object obj: lhs){} → ✓

iv) LHM lhm=new LHM();
for(Object obj: lhm){} → ✓ XCE: (map is not subclass of Iterable)

v) AL al=new AL();
Iterator itr=al.iterator();
for(Object obj;itr){} → XCE: (because for & itr are both cursors can't be open one in another)

2) variable should declare inside parantheis

Object obj;
for(obj:al){} → XCE:

3) variable type should be same type or superclass type of the objs retrieving from Collection.

AL al=new AL();
al.add("a");
al.add("b");
for(String s:al){} → XCE:
for(Object obj:al){} → ✓
for(String s:(String) obj) {} → XCE:
al

for(Object obj : al){
String s=(String) obj; → ✓
}

→ get() method will'nt throw Exception if an object is not available while Iterator throw an Exception.

→ ("a" = 1) It is → Map.Entry type object
key value

Q) Develop a program for storing employee number and its complete obj as key and value. for the employee 7280 increase salary by 1000 Rs. then display all employees key and values.

06/10/15 **Generics :-**

→ This concept is introduced in Java 5.0.

→ To avoid 'Type casting' and further to avoid CCE by providing compile time type safety.

→ It's nothing but, by using generics we are informing to compiler the type of objs are used and must use at execution time.

→ Mainly generics concept was introduced for collection framework objs for creating homogeneous collections, means to store only same type objs.

Eg :- ArrayList al = new ArrayList();

→ It is a heterogeneous collection, allows us to store different type of objs. So at retrieving we must use cast operator and if (instance of) condition for converting type & avoid CE.

```
ArrayList<String> al = new ArrayList<String>();
```

It is generic collection, homogeneous collection

```
ArrayList<Integer> al = new ArrayList<Integer>();
ArrayList<Employee> al = new ArrayList<Employee>();
```

Q) Below prog shows adding & retrieving elements from ArrayList with & without generics.

Ans :-

```
import java.util.*;
Class ALWithOutGenerics {
  P S V m(S[] args) {
    AL al=new AL();
    al.add("a");
    al.add("b");
    al.add(1);
    Sop.ln(al);
  Iterator itr=al.iterator();
  while(itr.hasNext()) {
  Object obj = itr.next();
  if (obj instanceof String) {
    String s = (String) obj;
    Sop.ln(s.toUpperCase());
  }
}

// Retrieving using for:each loop
  for(String s:al) {
    Sopln(s.toUpperCase());
  }
```

```
import java.util.*;
Class ALWithGenerics {
  P S V m(S[] args) {
  AL<String> al=new AL<String>();
    al.add("a");
    al.add("b");
    Sopln(al);
Iterator<String> itr=al.iterator();
while(itr.hasNext()) {
  String s=itr.next();
    Sopln(s.toUpperCase());
  }
}
```

REFLECTION API
REGULAR EXPRESSIONS

BY

Mr. HARI KRISHNA SIR

NARESH technologies

An ISO 9001 : 2008 Certified Company

AMEERPET HYDERABAD

sri raghavendra Xerox

All software language materials ava'lable

beside sathyam theatre line balkampetroad ameerpet Hyderabad

cell :9951596199

Reflection API

Topics List

Reflection API

→ Need of Reflection API

→ Important classes used in Reflection API - with JVM Architecture.

→ Accessing and printing a class's (super interfaces)

- super interfaces
- super class
- constructors and their modifiers, parameters, exceptions, logic execution
- variable's (field) name, modifiers, datatype, value
- methods and their modifiers, return type, name, parameters, exceptions, logic execution

→ Important methods

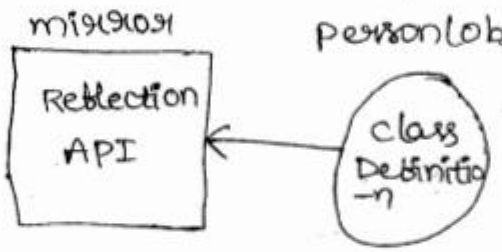
- to read and set values of variables.
- to execute logic of no-arg and parameterized methods
- to create object with no-arg and parameterized constructors.

- Projects
  - JDBC API Structure
  - Servlet API Structure
  - Mobile project development

Need of Reflection API:-

→ It is used to get information of runtime class (the class that is passed at runtime while program is executed) that its no' of variables, constructors, methods. And also its used to call a class members (variables, constructors, methods) without having that class's .java/.class files physically at development and compilation time

→ Reflection API provides mirror operations, it means it shows the class information dynamically.



But the class's .class file must be existed at the time of program execution, else reflection API throws exception "java.lang.ClassNotFound Exception"

→ As shown in the above diagram, the class (mirror) with Reflection API can load and get the class information ~~automatically~~ dynamically

Reflection API classes

— The main class/start of reflection API is "java.lang.Class"

— The remaining all classes are defined in "java.lang.reflect" package they are:

1. Field – it stores variable information
2. Constructor – it stores constructor information
3. Method – it stores method information.
4. Modifier – The Modifier class provides static methods and constants to decode class and member access modifiers. It means Field, and Method classes returns the variable and method modifiers as int value.

This class has methods to convert this int value to string form.

Procedure to develop as class with Reflection API

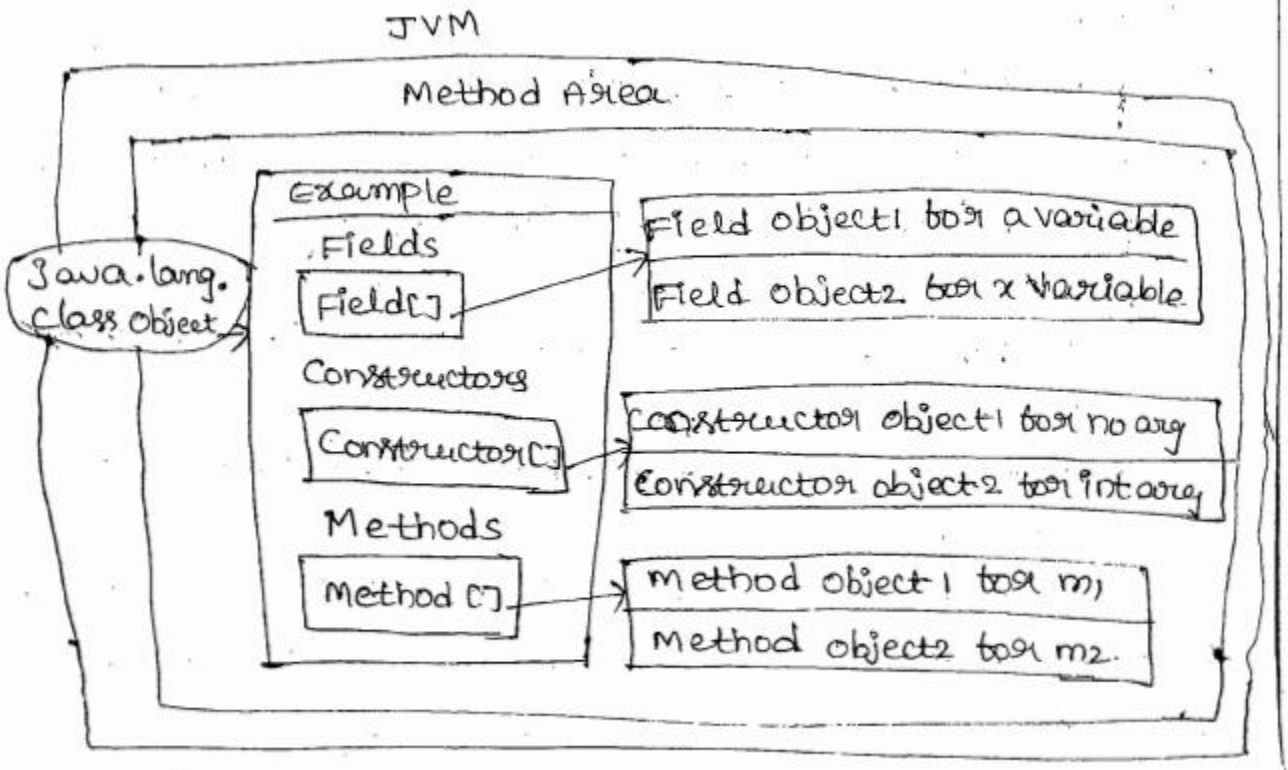
- when a class is loaded into JVM, JVM stores that class bytecodes by using java.lang.Class object.
- Inside java.lang.Class object
  - ~~Field~~ variables are stored by creating "Field" class objects
  - Constructors are stored by creating "Constructor" class objects.
  - methods are stored by creating "Method" class objects
- All Field objects are stored by using a "Field[]".
- All Constructor objects are stored by using a "Constructor[]" object.
- All Method objects are stored by using a "Method[]" object.

~~In java.lang.Class we have below~~

JVM Architecture with Reflection API Objects

```
class Example {
    static int a=20;
           int x=20;
    Example() {
      Sopln("no-arg");
    }
    Example(int x){
      Sopln("int-arg");
      this.x=x;
    }
    static void m1(){
      Sopln("m1");
    }
    void m2() {
      Sopln("m2");
    }
}
```

>Java Example



Methods:- java.lang.Class Methods:-

In java.lang.Class we have below methods to get array objects.

1. $\boxed{\text{Public Field[] getDeclaredFields()}}$

- It returns Field objects of every variable declared in the class, include private variables.

2. $\boxed{\text{Public Field[] getFields()}}$

- It returns Field objects of every public variables declared in that class

For example // Example.java

```
class Example{
        private int a;
                int b;
        protected int c;
        public    int d;
}
```

```
//Test.java
import java.lang.reflect.*;
Public class Test {
    P s v main(S[] args) throws ClassNotFoundException {
/*To load Example class dynamically into JVM in java.lang.
Class we have below static method:
```

public static Class forName(String name) throws ClassNotFoundException.

```
this method creates java.lang.Class object and stores the
argument class's bytecodes in this Class object, and finally
returns this Class object reference.*/
    Class cls = Class.forName("Example");
    //retrieving Field[]
    Field[] decFields = cls.getDeclaredFields();
    S.o.pln("No.of fields declared in Example class:"+
            decFields.length);
    Field[] fields = cls.getFields();
    S.o.pln("No.of public fields declared in Example
            class:"+fields.length);
  }
}
```

O/P:- 4

Note:- Example.java file compiled explicitly before executing Test.java. Here Test is mirror and Example is object.

2. public Constructor[] getDeclaredConstructors()
   - returns all declared constructors objects including private

4. public Constructor[] getConstructors();

→ It returns only public constructors objects

For example

```
// Example.java
class Example{
  private Example(){}
          Example(int a){}
  protected Example(float f){}
  public Example(String s){}
  public Example(Double d){}
}

// Test.java - reflectionAPI class
import java.lang.reflect.*;
class Test{
    p s v main(S[] args) throws CNFE{
      // Loading class
      Class cls = Class.forName("Example");
      // retrieving all declared constructors
      Constructor[] declCons = cls.getDeclaredConstruct-ors();
      // retrieving only public constructors.
      Constructor[] cons = cls.getConstructors();
      S.o.pln("Declared Constructors:"+declCons.length);
      S.o.pln("public Constructors:"+cons.length);
    }
}
```

o/p:- Declared constructors: 5

public constructors: 2

5. `public Method[] getDeclaredMethods();`

→ It returns all declared method's objects, including private methods. It doesn't include inherited methods.

6. `public Method[] getMethods();`

→ It returns only public method's objects. Also including public inherited methods. Always the length of method array is 9 + current class public methods list.

For example

```
//Example.java
class Example{
    private void m1(){}
    void m2(){}
    void m2(int a){}
    public void m2(float a){}
    protected void m3(){}
    private void m3(int a){}
    public int m4(){return 30;}
    public int m3(float f){return 30;}
    public static String m3(String s){return s;}
}
```

→ 9 public methods are inheriting from java.lang.Object class. It has totally 11 methods, two of them are "protected" they are "clone" and "finalize".

```
//Test.java - reflection API class
import java.lang.reflect.*;
class Test{
    P S V main(S[] args) throws CNFE{
        Class cls = Class.forName("Example");
        //retrieving all declared methods.
        Method[] decMethods = cls.getDeclaredMethods();
        //retrieving only public methods.
        Method[] methods = cls.getMethods();
        S.o.pln("declared methods:"+decMethods.length);
        S.o.pln("public methods:"+methods.length);
    }
}
```

O/P:- declared methods:-9
Public methods: 13

9 methods are declared in Example class among them 4 are public. so 9+4=13

All above three methods return all Fields, Constructors, Methods objects of a loaded.

Q) How can we get a particular variable, constructor, method objects?.

A) java.lang.Class has below specific three methods for this purpose.

7. public Field getField(String VariableName) throws NoSuchFieldException, SecurityException.

8. public Constructor getConstructor(Class... ParameterTypes) throws NoSuchMethodException, Security Exception

9. public Method getMethod(String methodName, Class... parameterTypes) throws SecurityException, NoSuchMethodException

Rule:- The above three method returns only public field, constructor, method. If we call it on non-public members it throws exception "NoSuch... Exception".

Q) Then how can we non-public specific members?

A) call 10. getDeclaredField(-),

11. getDeclaredConstructor(-),

12. getDeclaredMethod(-)

check below program to use above methods:

// Example.java

```
class Example{
    public int x=10;
    int y=20;
    public void m1(){}
    public void m2(Integer i1,Integer i2){}
    //Rule:- To call method using ReflectionAPI getMethod(), its
      parameters should be referenced types, not primitives.
    void m4(){}
}
//Test.java
import java.lang.reflect.*;
class Test{
    P S V main(S[] args) throws CNFE, NoSuchFieldException,
      NoSuchMethodException, SecurityException {
    Class cls=Class.forName("Example");
    Field f1=cls.getField("x");
    S.o.pln("f1:"+f1);
    //Non-parameterized method's Method Objects
    Method m1=cls.getMethod("m1");
    //parameterized method's Method Object
    Method m2=cls.getMethod("m2",Integer.class,
                                 Integer.class);
    S.o.pln("m1:"+m1);
    S.o.pln("m2:"+m2);
    }
}
```

Q) How can we call below method?

```
void m1(String s, Class... cls){}
```

we must pass one String object and zero to 'n' number of java.lang.Class objects

Q) How can we create java.lang.Class object?

A) Its object can be created in two ways.

1. with ".class" syntax
2. By using "forName()" method.

Ex:-

```
Class cls1 = String.class
Class cls2 = Class.forName("String");
```

In above two cases we create class object with String class byte code.

```
~~m1(String ss)~~
    m1("abc", String.class);
    m1("abc", Integer.class);
    m1("abc", Integer.class, String.class);
```

Notes:-

→ class referenced variable stores a datatype (reference datatype byte codes)

→ Object referenced variable stores datatype value (all objects)

Q) For below method what arguments must be passed?

```
void m1(Object obj){
}
void m2(Class cls){
}
```

A) → To call m1() method we must pass object of any class

→ To call m2 method we must pass Class object with any referenced datatype as shown below.

```
m1(new String("abc"));
m2(String.class);
```

```
m1(new Integer(30));
m1(Integer.class);
```

1.5
```
m1(10);
m1(Integer.class);
```

Q) Find out the datatype of below values?

1. 10 → int / Integer
2. 10.0 → double / Double
3. 'a' → char / Character
4. "a" → String / CharSequence / Object
5. new Example() → Example / Object
6. new StringBuffer() → StringBuffer / CharSequence / Object
7. String.class
   StringBuffer.class
   Integer.class
   Example.class
   Sample.class

   } Class

→ W.A.P to create Constructor object of int parameter Constructor of loaded class

```
class Test{
    P S V main(S[] args){
    Class cls=Class.forName(args[0]);
    Constructor cons1=cls.getConstructor(Integer.class);
    //creating String parameter Constructor object
    Constructor cons2=cls.getConstructor(String.class);
    //creating String and int parameter Constructor object
    Constructor cons3=cls.getConstructor(String.class,
                                         Integer.class);
```

// creating no-arg Constructor object

```
Constructor cons4=cls.getConstructor();
```

}

}

Q) How can we create object of a loaded class dynamically?

A) we must use below method from Class.

```
public Object newInstance()
        throws InstiantiationException, IllegalAccessException
```

Creates a new instance of the class represented by this Class object. The class is instantiated as if by a "new" expression with an empty argument list. The class is initialized if it has not already has been initialized.

This method throws IllegalAccessException

- if the class or its nullary Constructor (no-arg) is not accessible.

It throws InstiatiationException

- If the passed String represents the passed String to forName() represents an abstract class, an interface, an array class, a primitive type or void or if the class does not have no arg Constructor or if the instiantiation fails due to Exception raised in constructor.

→ c.A.p to load and Instiantiated runtime class.

```
class Test{
    P S V main(S[] args){throws Exception{
        Class cls=Class.forName(args[0]);
        // only class is loaded.
```

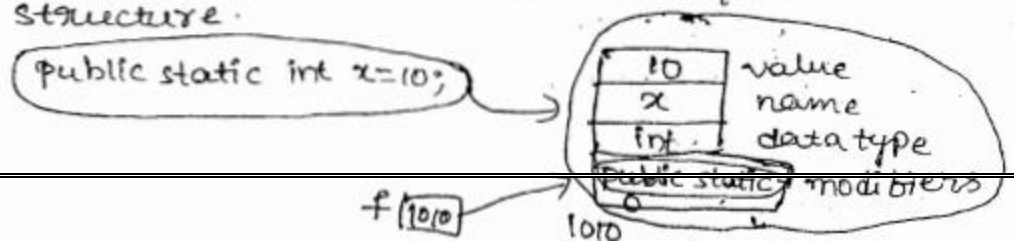
```
    //creating its object
    Object obj=cls.newInstance();
  }
}
// A.java
class A {
    static {
      System.out.println("A class is loaded");
    }
    A() {
      System.out.println("A class object is created");
    }
}
class B {
    static {
      System.out.println("B class is loaded");
    }
    B() {
      System.out.println("B class object is created");
    }
}
```

- forName() method creates object of java.lang.Class with loaded class byteCodes.
- newInstance() method creates object of loaded class
- This is the reason forName() method return type is class and newInstance() method return type is "Object".

<u>Field Class Methods:-</u>

- Field object memory structure.
- for the variable

public static int x=10; below is the Field class object's memory structure.



Methods:-

1. public int getModifiers()

→ This method returns modifiers of the current variable in "int" form.

→ We must convert it into String form to get original modifiers name by using "java.lang.reflect.Modifiers" class method toString().

Public static String toString(int n)

2. public Class getType()

→ Returns datatype of the variable in Class object form.

3. public String getName()

→ Return variable name

4. public Object get(Object loaded ClassObj)

→ It returns value of the variable in Object form.

→ To call this method we must pass current loaded class object.

Note:- If the variable is private or not accessible it throws java.lang.IllegalAccessException.

- But we can get other details of private variable.

→ W.A.P to print the given class variables details including value. If it is a private variable do not access its value.

```
// P6PrintVariablesDetails.java
import java.util.*; import java.lang.reflect.*;
class P6PrintVariablesDetails {
    P S V main(SC[] args) {
        Scanner sc = new Scanner(System.in);
```

```
S.o.pln("Enter class name:");
String name=scn.nextLine();
try{
  Class cls = Class.forName(name);
  Object obj = cls.newInstance();
  Field[] declFields=cls.getDeclaredFields();
  for(Field f:declFields){
    int m=f.getModifiers();
    // converting number to string form
    String modifiers= Modifier.toString(m);
    S.o.pln("modifiers:"+modifiers);
    Class dt=f.getType();
    ~~String~~
    // getting name of the data type
    String dtname=dt.getName();
    S.o.pln("datatype:"+dtname);
    String varName=f.getName();
    S.o.pln("Name:"+varName);
    if(modifiers.contains("private")){
      ~~int value~~
      Object value=f.get(obj);
      S.o.pln("value:"+value);
    } // if
  } // for
  S.o.pln("----------");
} // try
catch(ClassNotFoundException e){
  S.o.pln(name+"class is not found");
}
catch(InstantiationException e){
  S.o.pln("object creation is failed, no arg
          constructor ~~is~~ might not ~~are~~ have been
          available");
}
catch(IllegalAccessException e){
```

```
        S.O.println("No-arg constructor might have been
                    declared as private");
    }
}
}

// A.java
class A {
    private int a=10;
    static double b=20;
    public final String s="abc";
    protected static final char ch='a';
    float f=10.34f;
}
}
```

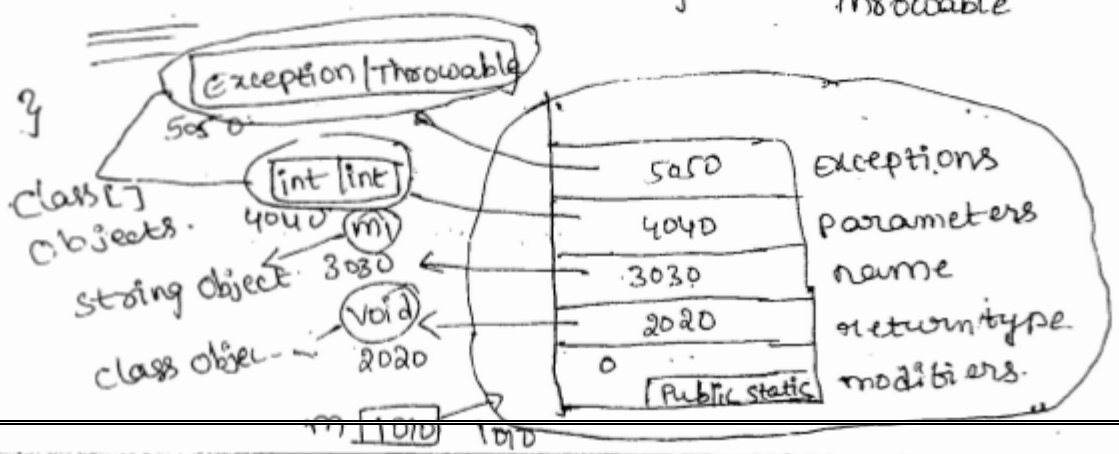
Compilation:-

```
>javac P6PrintVariablesDetails.java
>javac A.java
>java P6PrintVariablesDetails
```

Method class methods:-

- Method class object memory structure for the below method

```
public static void m1(int x, int y) throws Exception {
}
```



1. public int getModifiers()
2. public Class getReturnType()
3. public String getName()
4. public Class[] getParameterTypes()
5. public Class[] getExceptionTypes()
6. public Object invoke(Object loadedClassObj, Object... methodArgs) throws IllegalAccessException, IllegalArgumentException, InvocationTargetException

Note:- Programming Problem

- We cannot call invoke() method common for all methods of a class.
- Because the parameters of the methods are changed.
- If all methods are no-arg methods we can call this method common for all methods.
- The first five methods can call commonly on all types of methods of loaded class.

→ W.A.P to print all methods details of a loaded class.

```
// PrintMethodDetails.java
import java.lang.reflect.*;
class PrintMethodDetails{
    P S V main(S[] args) throws Exception{
        Class cls=Class.forName("B");
        Method[] methods=cls.getDeclaredMethods();
        for(Method m: methods){
            String name=m.getName();
            S.O.pln("----"+name+"method details");
            String modifiers=Modifier.toString(
                                m.getModifiers());
```

```
S.o.pln("modifiers:"+modifiers);
S.o.pln("return type:"+m.getReturnType()+getName());
Class[] parameters=m.getParameterTypes();
S.o.pln("parameters list:");
for(Class parameter: parameters){
    S.o.pln(parameter.getName()+",");
}

S.o.pln();
Class[] exceptions=m.getExceptionTypes();
S.o.p("exceptions list:");
for(Class exception: exceptions){
    S.o.p(exception.getName()+",");
}
S.o.pln("\n");
}
}
}

// B.java
class B{
    static void m1() throws ClassNotFoundException{}
    void m2(int x, String s){}
    public static int m3(A a, String s) throws
        ArithmeticException, NullPointerException{
        return 10;}
    P S V main(S[] args){}
}
```

cal
I
in

//I.ja

//E:

→ C.O.P to call a specific method from the loaded class.

→ To call a method we must know that method's prototype.

Call below method from different subclasses of ~~Animal~~ interface I.

```
~~interface Animal{~~
    ~~void eat(String food){ }~~
//I.java
interface I {
    ~~void m1(int a, int b)~~ void m1(Integer a, Integer b);
    int m2();
}
```

```
//ExecuteSpecificMethod.java
import java.util.*;
import java.lang.reflect.*;
class ExecuteSpecificMethod{
    P S V main(S[] args) throws Exception{
        Scanner scn=new Scanner(System.in);
        S.O.pln("Enter I subclass name:");
        String name=scn.nextLine();
        Class cls=Class.forName(name);
        Object obj=cls.newInstance();
        Method mt1=cls.getMethod("m1",Integer.class,
                                      Integer.class);
        Method mt2=cls.getMethod("m2");
        Object mt1Rv=mt1.invoke(obj,20,30);
        Object mt2Rv=mt2.invoke(obj);
        S.O.pln("mt1 returnvalue:"+mt1Rv);
        S.O.pln("mt2 returnvalue:"+mt2Rv);
    }
}
```

```
//P.java
class P implements I {
    public void m1(Integer i1, Integer i2) {
        S.o.pln("P m1:"+(i1+i2));
    }

    public int m2() {
        S.o.pln("P m2:");
        return 40;
    }
}

//Q.java
class Q implements I {
    public void m1(Integer i1, Integer i2) {
        S.o.pln("Q m1:"+(i1+i2));
    }
    public int m2() {
        S.o.pln("Q m2");
        return 50;
    }
}
```

JVM architecture with Reflection API object

Example.java

```
class Example{
static int a=4;
int x=10;
Example(){
x=5;
}
void m1(){
sopln(x);
}
psvmain(String[] args){
sopln("main");
}
}
```

Reflection API code responsibility:-

⇒ Reflection API code has 3 levels of responsibility.

1. Loading the class
2. Instantiating the loaded class
3. Executing variables, constructors & Methods from loaded class.

Loading class and instantiating:-

⇒ class class contains methods to load & instantiate a class (any) dynamically.

Method for loading:-

```
public static Class forName(String name)
                throws ClassNotFoundException
```

→ It is only responsible to load and store class it doesn't instantiate (or) also doesn't create object of this class

→ It creates class class object with field, Constructor and method objects and returns the class object reference, i.e lolo in above diagram.

⇒ Since it is a static and class return type method we can call it by using its class name as it is static & by assigning a referenced variable as show below:

```
Class cls = Class.forName("Lenovo");
```

⇒ Note that in projects we should not hard code class name as shown above. Because it becomes T.C. Instead we must ~~use~~ read it from keyboard as show in class NameReader.java.
(we developed this program in previous pages)

⇒ Method for instantiation:-

```
public object newInstance()
        throws InstantiationException
               IllegalAccessException
```

⇒ It creates loaded class object

⇒ It creates loaded class object by using no arg constructor.

⇒ If loaded class doesn't have no arg constructor it terminates program execution by throwing instantiation Exception.

⇒ If the loaded class is interface, also it throws instantiation Exception because it can't have constructor. (Interface can't be instantiated).

⇒ If in the loaded class no arg constructor is declared as a private it throws IllegalAccessException.

Assignment IV:-

① what we learn so far in Reflection API, advantages of JVM architecture.

comparision b/w normal programming and Reflection API class programming in loading, instantiating & calling methods.

Given:

```
class A{
Eg. static {
REQ.
  sopln("A class is loaded");
  }
Eg. A(){
Eg.
  sopln("A class object is created");
REQ.
  } REQ.
Eg. void m1(){
Eg. REQ. REQ.
  sopln("A class m1 method is executed");
  }

}
```

It is a static block executed when class is loaded into JVM automatically by JVM

It is a no-arg constructor executed when class object is created by calling this constructor

It is a non-static method executed only if we call, If we dont call JVM doesnot execute it automatically like static block & constructor

```
class NormalProgram{
E1. static{
    sopln("NP class is loaded");
}
psvmain(S[]args){
E5. sopln("NP main class main method");
E2. A a1 = new A();
E6. a1.m1();
E3. A a2 = new A();
E7. a2.m1();
}
```

Here A class is loaded first and then instantiated and its object is also created

Here only A class object is created doesn't, loaded again becauseed class is loaded only once in one JVM

q/p:- Java c A.Java

Javac normal program.Java

Java normal program

NP class is loaded

NP class main method

A class is loaded

A class object is created

A class M1() method is executed.

```
class Reflection API code program{
RE1. static{
    sopln("R API class is loaded");
}
psvmain(S[]args){
    throws CNFE, IE, IAE{
RE2. sopln("R API P class main method");
RE3. 1. Class cls1 = Class.forName("A");
RE5. 2. object obj1 = cls1.newInstance();
     // obj1.m1(); CE: can't find the symbol m1 method in Object class.
RE7. 3. A a1 = (A)obj1;
RE9. 4. a1.m1();
     sopln();
RE10. 5. Class cls2 = Class.forName("A");
RE11. 6. object obj2 = cls2.newInstance();
RE13. 7. A a2 = (A) obj2;
RE14. 8. a2.m1();
```

Here at line no.1 class is loaded & at line no.2 object is created. ~~At~~

At line 3 we calling casting A object to A type & calling m1 method.

A class object is created

A class M1() method is executed

problem in this approach:-

⇒ This process is T.C with A class means it always uses only A class functionality. It can't use other class functionality without recompiling changing the class name & recompiling it.

Solution:- we must use Reflection API as shown in the beside column.

class Reflection API code

Here at line no 5 class is not loaded again, instant it returns the same Class class object reference is return which is created at line 1 and stores in cls2 variable. so both cls1 & cls2 pointing to same object. Hence it is useless like we can use cls1.new instance to create new object. To prove this point use below statement

sopln (cls1 == cls2);

It returns true.

At line no 6 new A class object is created and return.

⇒ there is no change in o/p. But this program takes more time to execute compare to previous program as we can see code is increased and also there is a difference in execution steps.

summary:-

Q) when to use normal programming in creating object with new() keyword/constructor, with Reflection API code?

class.forName("A").newInstance();

⇒ Reflection API code we must use only to develop a class for using operations (methods) of an object from its different classes dynamically to develop this user class as i.e runtime polymorphic applications. ~~Else~~

⇒ Else we must develop class with direct object creation statement so that it takes less execution time so that it gives high performance.

Test cases:-

difference in compiling & executing direct object creation/ Reflection API based object creation statement.

case 1:

Given:

```
class NPTestcases
    PSVmain(S[] args)
        A a = new A();
    }
}
```

Here compiler search for A.class file herein current folder. If not available it throws compile time error. can't find symbol class A.

```
class BAPTestcase{
public static void main(String[] args)
throws Exceptions
Class.forName("A").newInstance();
}
}
```

⇒ Here compiler don't consider A as class Name it is just string for compiler so it doesn't check for A.class file. Given A.class file is not available program is compiled.

⇒ But JVM consider it as a class name searches for A.class, if available it is loaded into JVM. If not available it throw exception class not found exception

9) what is the o/p for below cases?

Delete take A.java Backup (ctrl c, ctrl v)

Delete A.java and A.class

compile above two classes

Javac NPTestcase.java ↵

can't find symbol class A

Javac BAPTestcase.java ↵

compiled successfully & .class file is generated

Java BAPTestcase ↵ it leads to Exception

Exception in thread "main" java.lang.classNotFoundException: A

Test case No: 2 :

Rename A.java backup file to A.java by pressing F2.
(we have to select A.java backup file)

⇒ Then compile & execute above two classes again.

Javac NPTC.java

op
A class is loaded
A class object is created
program compiled &
executed successfully

Javac RAPI Test case.java

delete A.class then
compile.

⇒ No compile time error
but leads to exception
because in this case
compiler can't find
A.java file because
it is not using as
class name directly

Summery:-

In normal programming when we compile A class all its internal class.java files are automatically compiled by compiler. where as in Reflection API based program compiler dont compiler internal classes.java file. Because in RAPI we giving classname as string. so this is mandatory compiler individual all classes by us (developer) hence at runtime it leads to CNFE.

Test case3: Remove no arg constructor in A.java then compile & execute above classes.

no arg constructor
default "
parameterised "

⇒ No compile time error & runtime error because compiler places no-arg constructor automatically in every class if class has zero constructor. Read page no (236,37,38 in volume-IA)

Test case4: press ctrl+z on A.java editor file no arg constructor is placed back declared it as private compile & execute

⇒ For first NPTestcase1.java leads to compile time error.
A() has private access in A.

⇒ Javac RAPTestcaseJava (A class is loaded).
compiled successfully in Execution it leads to Exception
IAE. A class is loaded (because of class.forname "A" statement execution)
Exception in thread "main" java.lang. IllegalAccessException.

⇒ write bulletpoints on your understanding.

⇒ In Testcase2 A.java file is automatically compiled NPTest case.java file. where as in RAP Test case.java we should compile it expressively

⇒ compiler algorithm for searching classes to compile them automatically.

Read package chapter page No 150 in volume 1A.

Test cases :-

⇒ In A.java only define parameterised constructor. As per above case in A.java file remove private & add int a in constructor () as below

```
A(int a){
    System.out.println("A class object is created");
}
```

then compile A.java then further compile both classes.

⇒ NPTest case.java compilation failed where as RAPI Test case.java compiled successfully but execution is failed. As there is no noarg constructor

⇒ In first case compiler identify this error since we called no arg constructor directly.

⇒ In second case cls.newInstance() method throw identifies A class doesn't have no arg constructor and throws IE.

Note that class is loaded.

LCRP architecture :-

⇒ LCRP architecture is the standard architecture for developing projects with objects communication in loosly coupled way.

→ where as MVC architecture is standard architecture for designing projects for separating development roles, to develop project modules in parallel for completing project development in short time.

⇒ LC stands for loose coupling.

⇒ RP stands Runtime polymorphism.

⇒ Both the LC & RP are developed in user application (class) that means a class i.e calling other object (class) methods.

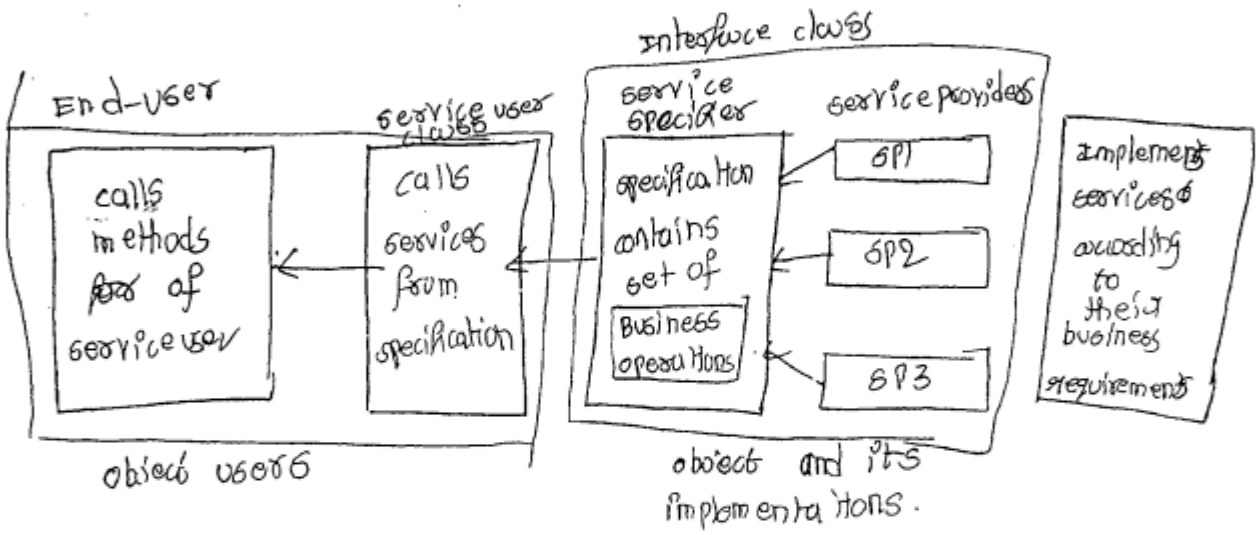
⇒ Loose coupling ~~across~~ is implemented in project we have advantage to receive and store different subclasses objects and further to call and execute method from different subclasses of an object without changing & recompiling user-class.

⇒ so loose coupling should be implemented by declaring a reference variable with super class type then only different subclass objects can be stored, further we must call

methods using this superclass reference variable.

⇒ Runtime polymorphism is the process of executing a called method from different subclasses of an object based on object stored in the superclass referenced variable.

LCRP architecture diagram :-



Here service means
- operation according to real life business
- Method " " oo programming

Note
⇒ All above 3 classes (user class, specification class & implementation class) are developed by different developers either in the same ~~needs~~ team (or) in different ~~needs~~ team, either in same (or) different companies anywhere in the world.

Q) Then how user class can use method operations implemented by different team at runtime dynamically?

(LC)

ans:- Becaused methods are called in user class using super class (specification).

Q) How implementation class (subclass) object is passed into user class at runtime for executing called methods in subclass?

Ans:- its name must be passed either from keyboard (or) from properties files (or) from XML files and further we must load & instantiate subclass byte code using Reflection API.

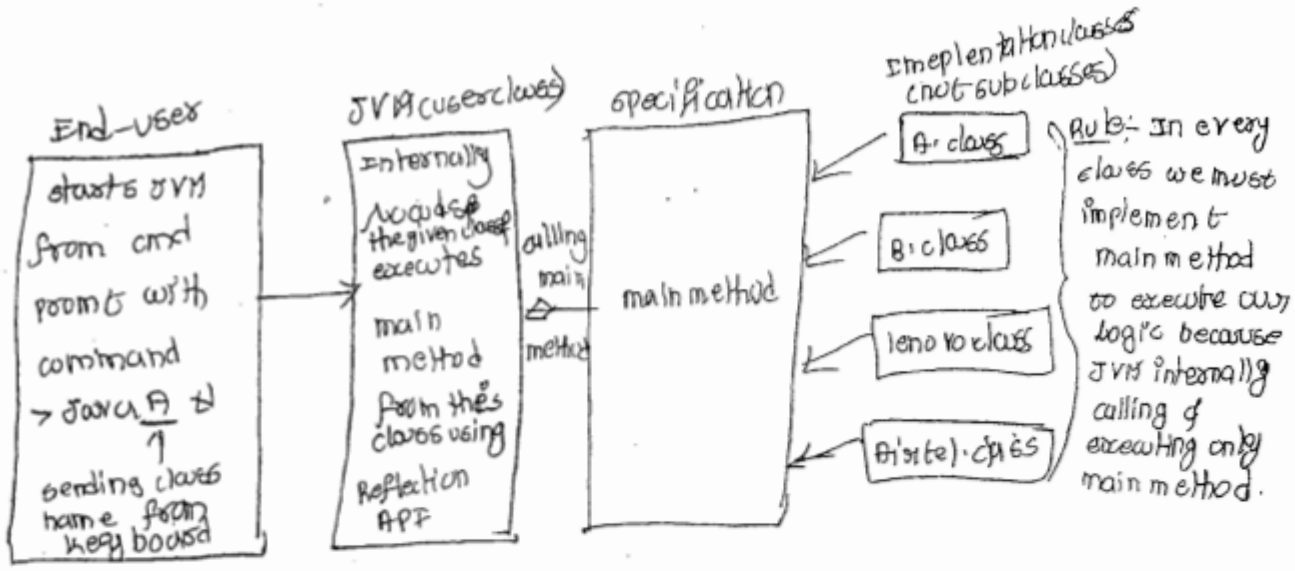
Real time examples for LCRP:-

case1

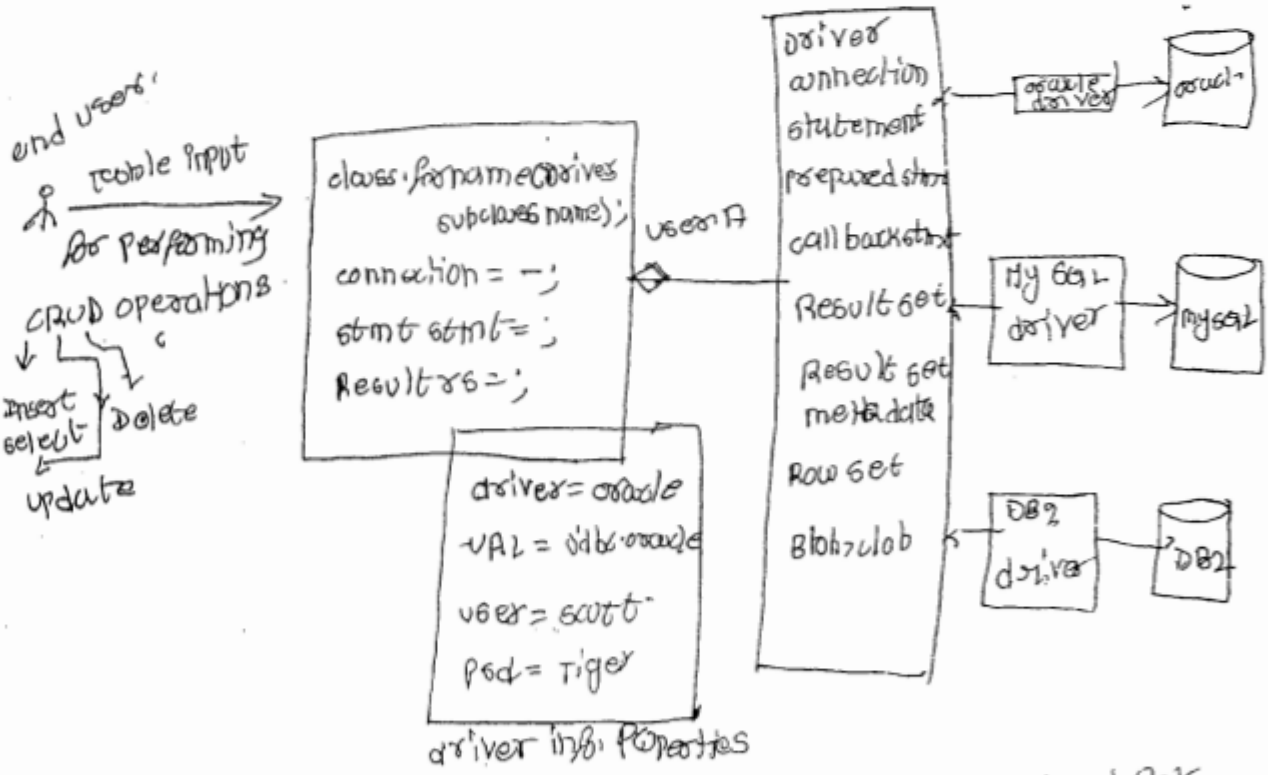
⇒ In core Java. JVM software is already develop to execute future coming classes logic.

⇒ Here JVM is user class and our program (class) this implementation class & main method is specification.

⇒ we sending the class name to the JVM from the keyboard as java command argument (Java A) from command prompt.



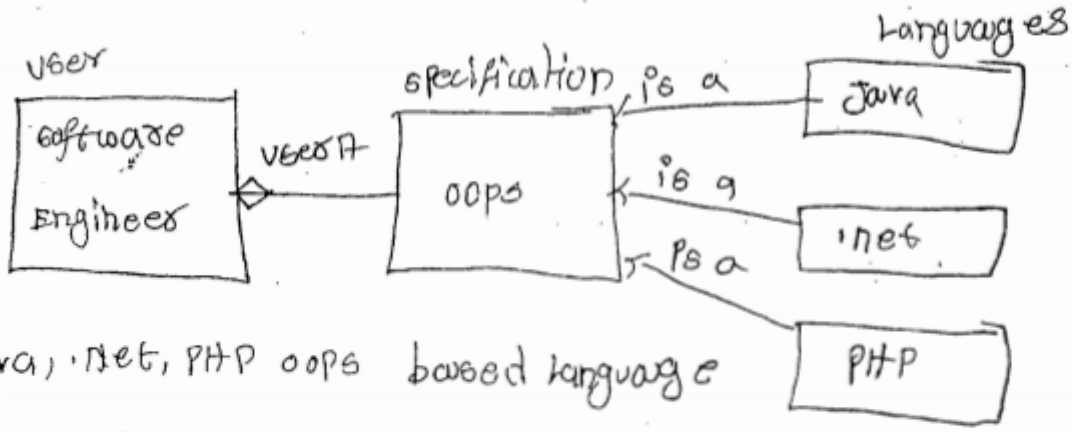
JDBC LCRP



→ In JDBC programming our role is easier prgm developer to develop a loosely coupled way to connect with any database we must a load driver subclass given by

renders dynamically by usin RAPI by receiving class name from properties file or XML file.

OOPS & CRP :-



Java, .Net, PHP OOPS based language

they implemented OOPS base concept using their own syntax so we must learn OOPS to develop the project any one language.

End of project design Architecture:-

compiler → file not found
error → not found
error → not found
TreeIf →
JVM execution error
setclasspath

Execution error:- Environment variable classpath not found

compile time error:- file not found.

errors (semi colon)
; → exceptions
" → unclosed string literal
" → unclosed string literal

for Mobile & SIM project check DOPS chapter last page in vol-18

REGULAR EXPRESSIONS

9/9/08

Regular Expressions

→ The Regular Expressions is a special text string for describing search pattern.

→ you think Regular Expression as wild cards

→ The important terminology or terms are Pattern & matcher

→ pattern is a Regular Expression describing a certain amount of text to be found in the specified or passed text

→ Matcher are match is a piece of text is a sequence of bytes or characters that pattern was found to corresponding to by the Regular expression processing software. Matcher is a string passed to found for a piece of text

Eg:- String str = "abcd4567efgh" ← Matcher
A Macher passed to found 4567

string str = "4567"; ← Regular Expression (or) pattern
A piece of text to be found in the Macher

In shortform RegularExpression are called regex, reg exp

→ To Develop the Regular Expression or to create Matcher or pattern we must import package java.util.regex

This package contains 2 import classes
1. pattern 2. Matchers.

Regular Expression or pattern :- is created or represented using the class or pattern

Public final class pattern implement Serializable

It is a compiled representation.
To create pattern class object it does't have any visible constructor, instead it has been given static method.

Public static pattern compile (string regex)

Compiles given regex into a pattern and return pattern object reference. This method throws java.util.regex.

Pattern Syntax Exception :- unchecked Exception. If the regular expression syntax is wrong

→ Matcher class object is also does't have visible constructor to create its object instead it must be created from pattern class nonstatic method.

Public Matcher match(CharSequence sq)

CharSequence is an interface implemented by String, StringBuffer, String Builder. we can pass any of the above 3 objects

→ Create a Matcher that will match the given input against this pattern

Matcher class Methods

Public final class matcher extends Objects implements MathResults

If the pattern is found in the matcher then we can retrieve the start index, end index and the pattern using the below Matcher class methods.

→ Public boolean find()

It return true if the pattern found in the matcher

→ Public int start( )

It returns the start index of the previous match start( ) method must be called after find( ) method. violation leads to RE: IllegalStateException.

→ Public int End( )

Return the offset (index) after the last character matched. It also throws IllegalStateException if we call it before find( )

→ Public int start (int group)

→ Public int end (int group)

Returns the string index after parsed index. These method also throws IllegalState Exception

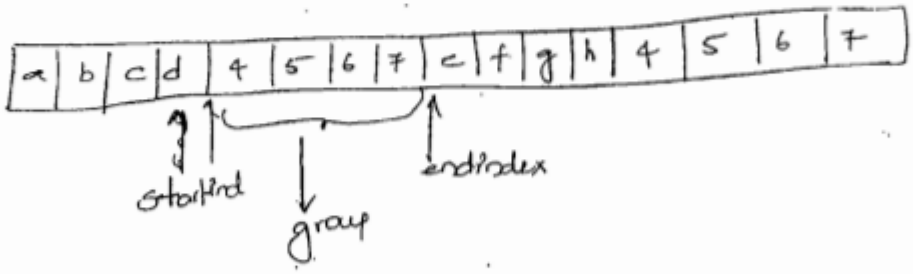
→ Public string group( )

Return the matched string. It also must be called after find( ) method else leads to IllegalStateException.

```
import java.util.regex.*;
Public class RegExDemo
{
  p s v.m(SL] arg)
   {
    //pattern/regex creation
    Pattern p = Pattern.compile("4567");
      // Match creater
    Matcher m = p.matter("abcd 4567 efgh 4567");
          while(m.find());
          {
          sop(m.start()+" " + m.end()+"..."+m.group[0]);
  }
 }}
```

o/p 4 . . . 8 . . . 4567

13 . . . . 17 . . . 4567



creating Regular Expression

The Below characters must be used to create Regular Expression.

Character classes : [abc]

a or b or c

Simple class

2. [^abc]:- Any characters except a, b, c.

3. [a-z A-z]:- All character in between range of a to z & A to z inclusive range.

4. [a-d [m-p]]:- a through d or m through p
It means [a-d m-p]

5. [a-z && [def]] :- Search for d, e or f

6. [a-z && ^bc] :-
It means a through z except b & c

7. [ad-z]

8. [a-z && [^m-p]])
[a-l q-z]

9. [0-9] 0 through 9

10. Note abc
Search for the word abc not for character

Predefined character classes

1. . (any character)
It is also called meta character

2. \d :- A digit range 0 to 9

3. \D:- [^0-9]

4. \s;- is white space character [\t \n \b \f \r]

5. \S;- Not white space character [^\s]

6. \w;- A word character [a-z A-Z 0-9]

7. \w;- Non word character [^\w]

Note :- As per java syntax \ represent the escape sequence characters in the string. The above said characters are not valid escape sequence character. Hence we must prefix one more \ before the above character.

write a program to return only integer character index

```
Pattern p = pattern.compile("[0-9]")
Pattern p = pattern.compile("\d"); X CE: Illegal escape charac
Pattern p = pattern.compile("\\d");
Matcher m = p.matcher("abcd 4567 abcd 4567");
while(m.find())
{
System.out.(m.start()+"..."+"m.end()"+"..."+m.group());
```

Metacharacters

Metacharacters supported by java are ([{ | ^ - $ }]) * +

The characters ! @ # % ~ are not metacharacter but mean does have special meaning

There are 2 ways to force a metacharacter to be treated as an ordinary character

→ Precede a metacharacter with \ or enclosing it within \Q which starts the code (and)

→ When using this technique. \Q \E can be placed at any location with in the expression \Q must come first.

Quantifiers

Quantifiers allow you to specify the no. of occurrency to match against. In some language espicially In xml They are called occurence specifiers.

There are 3 types of Quantifiers are there

1. Greed>.
2. Reluntant-
3. possive

The special character are used to specify occurence specifier.

sign	Regular Expression	Occurence	
1. no-sign →	a	exactly one	1
2. + →	a+	atleast one	1...x
3. * →	a*	optional	0...x
4. ? →	a?	atmost one	0 or 1

? represents one or not at all (atmost 1)

* represent zero or more times

+ represent one or more time. (atleast one)

no-sign represent exactly one

a :- exactly once

a(n) :- a exactly n times

a(n,) :- a atleast n times

a(n, m) :- atleast n times, atmost m times

split :-

split method of string class is used to split by taking regular expression. The same tokenizing & split operation can be done using stringTokenizer also

Public string[] split(string regex)

split string around matches of the given regular Expression.

Public string[] split(string regex,

regex is used as delimiter

The above 2 method throws pattern syntax exception if pattern syntax is wrong.

Note split method will not return delimiter

Ex:-
```
string str = "www.activenetIndia.com";
string[] s1 = str.split(regex);
    for (int i=0; i<s1.length; i++)
        {
            Sop(s1[i]);
    }
```

If regex = India

o/p www.activenet

if regex = .ww.

.activeindia.com

If regex = .

no output It has special meaning represent all character. Hence each character in the

in the passed string will be considered as delimiter. Since delimiters are not returned by split() will not get only o/p

```
www
activeindia
com x
```

If we want to use . as delimiter in the regular Expression. It must preceded with \ or it should written in betn \Q x \E

```
str.split("\.");
str.split("\Q.\E");
```

o/p
```
www
activeIndia
com
```

Ex:—

pattern syntax error

```
str.split("\.\E") x
```

RE: pattern syntax Error

\E must be after \Q only

```
str.split("\Q.\E");
```

pattern syntax Error

```
str.split("\Q.\E");
```