1. What is Java?

Java is a programming language platform, which is based on OOP concept and widely used for internet software application, which is developed in 1995 by James gosling.

What is Data types?

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java -:

1. How many types of primitive and non-primitive data types?

* Primitive (Pre-defined)
* Int
* Byte
* Short
* Long
* Float
* Double
* Boolean
* Char

Why they are primitive data types?

A variable of primitive data type holds the actual value or stored in the memory that is allocated to it. Variable with primitive data type does not store references of object.

int age = 22;

int age; means not initialized the value of variable :> int age; by default, JVM will assign 0 zero.

int age=0;

* Non-primitive (User defined)
* Array
* String
* Class
* Object
* Reference variables

Why they are non-primitive data types?

A variable with non-primitive data type stores the reference of object in a memory and does not holds the actual value directly. Actually, they are **objects**.

String str=**new** String ("");

String str; means not initialized the value of variable :> String str; by default, JVM will assign null.

String str=null;

**NOTE**-:

* 1 byte =8 bit
* // double forward slash for comment the line in java class, commented line will be not execute.
* /\*

for comment the line for java class.

\*/

1. If initialize char ‘65’ what will be output?

Based on the ASCII table output will be given

* Start from 65-A to 90-Z
* Start from 97-a to 122-z



What is casting?

Casting is a type of conversion process from one data type to another data type.

1. How many types of casting?

* **Up casting = Bigger data types <=Smaller data types**

**Byte1-🡪short2🡪 // Compiler is responsible for up casting**

**Int4🡪 Float4🡪long8🡪double8**

**Char2-----🡪**

* **Down casting= Bigger data types =>Smaller data types**

**// User is responsible for down casting**

**Byte1<--short2<--**

**Int4<--Float4<--long8<--double8**

**Char2<--**

Long l;

Int i= (int)l;

1. How to convert int to long and as well other?

By up casting.

1. How to convert long to int and as well other?

By down casting.

1. What are the class elements in Java?

**1.** Variables **int a = 10;**

**2.** Methods **void add () {business logic}**

**3.** Constructors **Test () {business logic}**

**4.** Instance blocks **{business logic}**

**5.** Static blocks **static {business logic}**

1. What is variables and use?

* Variables are used to store the constant values by using these values we are achieving project requirements.
* Variables are piece of memory that can contain data value or literal.
* Variables are also known as **fields** or **properties** of a class.
* All variables must have a type. You can use primitive types such as int, float, Boolean, etc. either you can use non-primitive or reference types, such as strings, arrays, or objects.
* Variable declaration is composed of three components in order
* Zero or more modifiers.
* The variable type.
* The variable name.
* Example: public final int x=100;

1. How many types of variables?

* Local variable:
* Declaration: local variable declared inside the method, constructor or blocks only.
* Scope: Local variable can be access with in method, constructor or blocks only not outside of them.
* Memory: Local variable always store inside the stack memory.
* Allocation: *For the local variables memory allocated when method starts, and memory released when method completed.*
* Access specifier (Modifier): cannot be used.
* Public void M1() {

Int a=10;

Int b=20;

}

* Instance variable (Non-static):
* Declaration: Non-static variable declared inside the class but outside of the method, constructor and blocks.
* Scope: Non-static variable can be access with in the non-static method directly, constructor or blocks with in class but not inside the static method directly.
* Memory: Non-static variable always stores inside the heap memory.
* Allocation: When object is created non-static variable memory is allocated.
* Access specifier: can be used.

NOTE:

1. Non-static method can call non-static variable directly.
2. Static method cannot call non-static variable directly, so need to create object of class and call them too to access non-static variable.

* Public class A {

Int a=10;

Int b=20;

Public void M1 () {

}

* Static variable:
* Declaration: Static variable declared with static keyword inside the class and outside of the method, constructor and blocks.
* Scope: Static variable can be access inside the static and non-static method, constructor or blocks with in class by directly, by class name and reference of object
* Memory: static variable always stores inside the non-heap or static memory.
* Allocation: When we run program and .class file is loaded.
* Access specifier: can be used.

NOTE:

1. Static method can call static variable directly.
2. Non-static method can call also static variable by using three methods:

* Directly
* By using class name
* By using object of class (NOT RECOMMENDED)
* Public class A {

Static Int a=10;

Static Int b=20;

Public void M1 () {

}

1. How many types we can call instance variable?

* By using reference of object, we can call instance variable.

1. How can we call instance variable in static method?

* Using reference of object, we can call instance variable.

1. How many types we can call static variable?

* By using class name, we can call static variable.
* directly we can call static variable.
* By using reference of object, we can call static variable.

1. How can we call static variable in instance method?

* By using class name, we can call static method.
* directly we can call static method.
* By using reference of object, we can call static method.

NOTE:

***Instance vs. Static variables: -***

 *For the instance variables the JVM will create separate memory for every object it means separate instance variable value for every object.*

 *For the static variables irrespective of object creation per class single memory is allocated, here all objects of that class using single copy.*

***Example: -***

*class Test*

*{ int a=10;* ***//instance variable***

*static int b=20;* ***//static variable***

*public static void main(String[] args)*

*{ Test t = new Test();*

*System.out.println(t.a);* ***//10***

*System.out.println(t.b);* ***//20***

*t.a=111; t.b=222;*

*System.out.println(t.a);* ***//111***

*System.out.println(t.b);* ***//222***

*Test t1 = new Test();*

*System.out.println(t1.a);* ***//10***

*System.out.println(t1.b);* ***//222***

*t1.b=444;*

*Test t2 = new Test();* ***//10***

*System.out.println(t2.b);* ***//444***

What is a class?

Class is an entity where we have to define multiple properties in the form of variables and methods like: no of variables and methods.

What is object?

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc.

Object is real world entity that has state, behaviors and identity which is physically available.

* State – Well defined condition of item (instance variables, properties and fields)
* Behaviors – Effect on an item(methods/functions)
* Identity – identification of an item (hash code)

An object has three characteristics:

* **State:** represents the data (value) of an object.
* **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
* **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

What is OOP?

OOP language is that programming language where everything represented in the form of object.

* Object means real world entity such as pen, table, chair and that everything which is physically

Available.

What is class?

Class is nothing but blueprint or template for creating the different object which defines its states and behaviors.

What is object?

* **state:** represents data (value) of an object.
* **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
* **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Example :-

Object : Car

State : gear,speed,color…etc

Behavior : current speed,current gear,Accelarate…etc

Identity : car number

Object : house

State : location

Behavior : doors open/close.

Identity : house no

1. What is method?

Inside the classes directly writing the business logics are not allowed in java hence inside the class declare the method inside that method writes the business logics of the application.

* Methods are used to write the business logics for the project.
* Coding convention: - method name starts with lower case letter and every inner word starts with uppercase letter(**mixed case**).
* Example:- post() , char**A**t() , to**U**pper**C**ase() , compare**T**o**I**gnore**C**ase()……etc

Advantage:

* 1. Method or functions can be reused by using inheritance concept means no need to write the same method again and again.

1. How many types of method?

* **Instance method (Non static): -**

Void m1 () **//instance method**

{

//body //**instance area**

}

Note: - for the instance member’s(Variables+Method) memory is allocated during object creation hence access the instance members by using object-name (reference-variable).

**Method calling Syntax: -**

Void m1 () {

} //instance method

**Reference of object.instanceMethodName (); //calling instance method**

Test **t** = new Test ();

t.m1 ();

* **static method: -**

Static void m1 () **//static method**

{

//body **//static area**

}

Note: - for the static member’s (Variables Method) memory allocated during **.class** file loading hence access the static members by using class-name.

**Method calling syntax: -**

Static void m2 () {

} //static method

**Classname.staticmethod (**

***Test.m2 ();***

***Method Syntax or method parameter: -***

***[modifiers-list] return-Type Method-name (parameters list) throws Exception***

***Modifiers-list*** *------**represent access permissions*. *----* ***[optional]***

***Return-type*** *------**functionality return value ----* ***[mandatory]***

***Method name*** *------**functionality name -----* ***[mandatory]***

***Parameter-list*** *------**input to functionality ----* ***[optional]***

***Throws Exception*** *------**representing exception handling ---* ***[optional]***

***Method Signature: -*** *Method-name & parameters or input arguments list is called method signature.*

**Syntax: - *Method-name(parameter-list)***

***Example: - m1(int a), m1(int a, int b)***

NOTE: When will you call the method from one method to another method only one method is allowed at one time means M1 method -> M2, we cannot call both method at one time M1 method -> M2, M2 method -> M1.(We cannot interchange the method at one time) if you did then you will get error “Exception in thread "main" java.lang.StackOverflowError”

1. How many types we can call instance method?

* By using reference of object, we can call instance method.

1. How can we call instance method in static method?

* Using object creation of class, we can call instance method.

1. How many types we can call static method?

* By using class name, we can call static method.
* directly we can call static method.
* By using reference of object, we can call static method.

1. How can we call static method in instance method?

* By using class name, we can call static method.
* directly we can call static method.
* By using reference of object, we can call static method.

1. How can we call static variable and static methods & instance variable and instance method from any other class?

* By using class name and reference of object, (not directly in other class) we can call static variable and method.
* By using reference of object, we can call instance variable and method.

1. Can we create inner class in java?

Yes

1. Can we create inner method in java?

* Java doesn’t support inner method concept.

***Polymorphism: -***

*The ability to appear in more forms is called polymorphism.*

*One thing can exhibit more than one form is called polymorphism.*

*One functionality with different actions is called polymorphism.*

*One person with different behaviors is called polymorphism.*

*Polymorphism is a Greek word poly means many and morphism means forms.*

There are two types of polymorphism in java,

*1) Compile time polymorphism / static binding / early binding*

***Example: - method overloading.***

*2) Runtime polymorphism /dynamic binding /late binding.*

Example: - method overriding.

***Compile time polymorphism [Method Overloading]: -***

1) Java class allows more than one method in one class with same name but different number of arguments or same number of arguments but different data types those methods are called overloaded methods.

2) To achieve overloading concept one java class sufficient.

3) It is possible to overload any number of methods in single java class.

EX:

1. If both methods name are same but arguments or signatures are different, so they are overloaded method.
2. If one method is having one argument and second method is having same no of argument but different data types so they are overloaded method.
3. If one method is having two argument and second method is having different no of argument or more than two with same data types so they are overloaded method.
4. If one method is having two arguments or more than two and second, method is having two or more than two with same data types, but all arguments are in different place instead of one method means replace the place of argument in second method.

1. Same method name but different number of arguments.

**public** **void** sum() {

}

**public** **void** sum(**int** i) {

}

2. Same method name & same number of arguments but different data types.

**public** **void** sum(**int** i) {

}

**public** **void** sum(**long** l) {

}

3. Same method name but different number of arguments with same data types.

**public** **void** sum(**long** l) {

}

**public** **void** sum(**long** l, **long** j) {

}

4. If one method is having two arguments or more than two and second, method is having two or more than two with same data types but all arguments are in different place instead of one method means replace the place of argument in second method.

**public** **void** sum(**long** l,**int** j) {

}

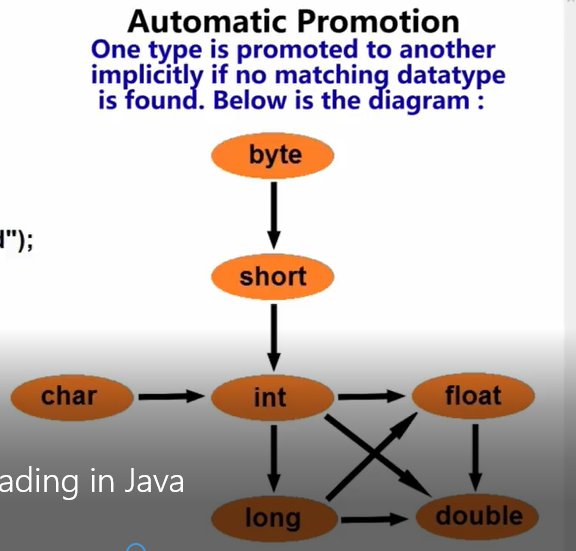
**public** **void** sum(**int** i,**long** k) {

}

NOTE:

1. Can we achieve method overloading changing the return type of method only?

No, in java we cannot achieve method overloading changing the return type of method because of ambiguity (we will get compile time error)



* Can we overload main method?

Yes, we can but with different no of arguments or signatures.

***Operator overloading: -***

One operator with different behaviors is called Operator overloading.

Java is not supporting operator overloading but only one overloaded in java language is ‘+’.

 If both operands are integer then “**+”** performs addition.

 If at least one operand is String then “**+**” perform concatenation.

**Runtime polymorphism [Method Overriding]: -**

* When same method is present in parent class and as well as child class with same method name and same no of arguments that is called method overriding.

*To achieve method overloading one java class sufficient but to achieve method overriding we required two java classes with parent and child relationship.*

*The method implementations already present in parent class,*

*a. If the child class required that implementation then access those implementations.*

*b. If the child class not required, parent class method implementations then override parent class method in child class to provide child specific implementations.*

*The sub class overrides super class method to provide sub class method implementations.*

*In overriding parent class method is called ===>* ***overridden method***

*Child class method is called ===>* ***overriding method***

***While overriding methods must fallow these rules: -***

*1) Overridden method signature & overriding method signatures must be same.*

*2) The return types of overridden method & overriding method must be same (at primitive level).*

*3) While overriding it is possible to change return type by using co-variant return types concept.*

*4) Final methods can’t override.*

*5) Static method can’t override but method hiding possible.*

*6) Private methods can’t override.*

*7) While overriding it is not possible to maintain same level permission or increasing order but not decreasing.*

*8) Overriding with exception handling rules.*

***Note:- Parent class reference variable is able to hold child class object but Child class reference variable is unable to hold parent class object.***

*class Parent*

*{ };*

*class Child extends Parent*

*{ };*

*Parent p = new Child();* ***//valid***

***Example-2:-***

In java parent class reference variable is able to hold Child class object but Child class reference variable unable to hold Parent class object.

*class Parent*

*{ };*

class Child extends Parent

*{ };*

*Child c = new Parent();* ***//invalid***

*class Parent*

*{ void m1(){System.out.println("parent m1 method");}* ***//overridden method***

*}*

*class Child extends Parent*

*{ void m1(){System.out.println("child m1 method");}* ***//override method***

*void m2(){System.out.println("child m2 method");}* ***//direct method of child class***

*public static void main(String[] args)*

*{*

***//parent class is able to hold child class object***

*Parent p1 = new Child();* ***//creates object of Child class***

*p1.m1();* ***//child m1() will be executed***

***//p1.m2(); Compilation error we are unable to call m2() method***

*};*

*In above example parent class is able to hold child class object but when you call* ***p.m1();*** *method compiler is checking m1() method in parent class at compilation time. But at runtime child object is created hence Child method will be executed.*

*Based on above point decide in above method execution decided at runtime hence it is a runtime polymorphism.*

*When you call* ***p.m2 (); compiler*** *is checking m2 () method in parent class since not there so compiler generate error message. Finally it is not possible to call child class m2 () by using parent reference variable even thought child object is created.*

*Based on above point we can say by using parent reference it is possible to call only overriding methods (****m1 ()*** *) of child class but it is not possible to call direct method(****m2()*** *) of child class.*

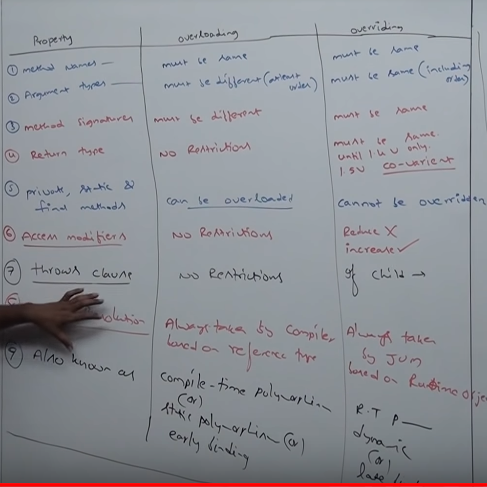
*To overcome above limitation to call child class method perform typecasting.*

*Child c1 =(Child) new Parent();* ***//type casting (Down casting) child class reference variable to parent class object but will get* \*ClassCastException\***

*c1.m1();*

*c1.m2();*

*}*



* In overloading one class is required but overriding more than one class is required along with parent child (IS-A) relationship.

What is inheritance?

* The process of acquiring the properties from one class to another class that is called inheritance.
* We are acquiring the properties from another class by extends keyword.
* In Inheritance, parent class is giving the properties and child class is acquiring the properties.

NOTE -

* in java if we are extending the class then it will be parent class, if we are not extending the class then **object** class will become parent class by default.
* Advantage:
* code reusability
* We can achieve run time polymorphism or method overriding
* Code redundancy
* Cost cutting
* The root class of all java classes is “**object**” class.

***Object creation of parent & child classes: -***

In java it is possible to create objects for both parent and child classes,

If we are creating object for parent class it is possible to access only parent specific methods.

***A a=new A();*** *a.m1(); a.m2();*

if we are creating object of child class it is possible to access both parent & child specific methods.

***B b=new B();*** *b.m1(); b.m2(); b.m3(); b.m4();*

* ***C c=new C();*** *c.m1(); c.m2(); c.m3(); c.m4(); c.m5(); c.m6();*

How many types of inheritance?

In java five types of inheritance

* Single level inheritance

If one child class extending the properties from direct parent class that is called single level.

Inheritance.



* Multilevel inheritance –

If one child class extending the parent class, then it will become parent class of next extended class that is called multilevel inheritance.

  
Class B extends A ===> class B acquiring properties of A class

Class C extends B ===> class C acquiring properties of B class

[indirectly class C using properties of A]

* Hierarchical inheritance –

More than one child class extending to only one parent class that is called Hierarchical inheritance



Class B extends A ===> class B acquiring properties of A class

Class C extends A ===> class C acquiring properties of A class

Class D extends A ===> class D acquiring properties of A class

* **Multiple inheritances -**

 One child class is extending more than one parent class is called Multiple inheritance and java does not supporting multiple inheritance because it is creating ambiguity problems (confusion state).

 Java not supporting multiple inheritances hence in java one class able to extends only one class at a time but it is not possible to extends more than one class.

**Class A extends B ===>valid**

**Class A extends B ,C ===>invalid**



Ex –

class Parent1

{ void money() {System.out.println("parent1 money");}

};

class Parent2

{ void money() {System.out.println("parent2 money");}

};

class Child extends Parent1,Parent2

{ public static void main(String[] args)

{ Child c = new Child();

c.money(); **//ambiguity problems**

}

* **Hybrid inheritance -**

 Hybrid is combination of hierarchical & multiple inheritance.

 Java is not supporting hybrid inheritance because multiple inheritance (not supported by java) is included in hybrid inheritance.



**Preventing inheritance: -**

 You can prevent by sub class creation by using final modifier.

 If a parent class declared as final we can’t inherited of parent class.

final class Parent

{ }

class Child extends Parent

{ }

**compilation error:- cannot inherit from final Parent**

*Types of Relationship between two classes:*

*1.Inheritance (IS-A Relationship): If we are using inheritance between two class that is called IS-A relationship by using extends keyword (tightly coupled relationship, blood relationship)*

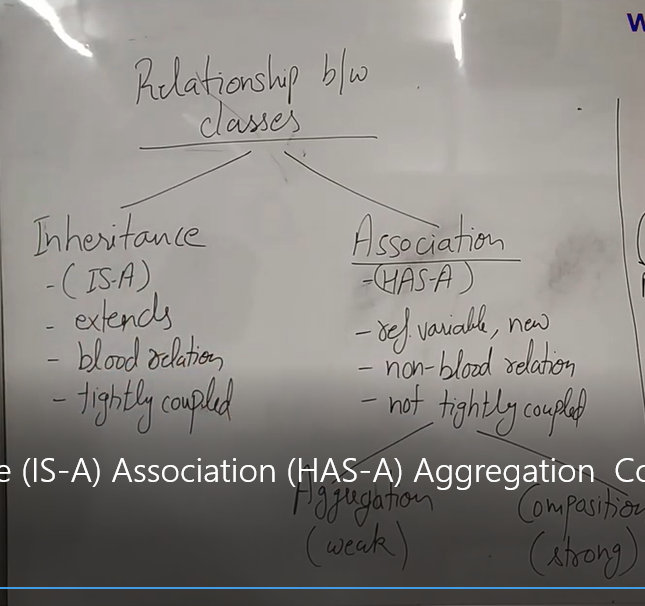
* *If we are using this concept changing the properties of parent class, then changes will be reflecting in those all places where we have used by extends keyword.*

*2.Association (HAS-A): If we are creating the object of class or new keyword in any other class that is called HAS-A relationship by using object of class or new keyword (Not tightly coupled relationship, Non-blood relationship)*

* *If we are using this concept changing the properties of parent class, then changes will be reflecting in those places where we have been creating the object of class or by new keyword.*

*Types of Association (HAS-A):*

1. *Aggregation*
2. *Composition*



***Abstraction: - Abstraction means hiding the implementation of method or functions or without any body that is called abstraction.***

* ***Abstraction can be achieved by interface which is 100% achieving***
* ***Abstraction can be achieved by abstract class which is 0 to 100% because if we declared abstract method as well as non-abstract method then cannot achieve 100% therefore in abstract class we can declared abstract and non-abstract method but in interface we can declared only abstract method.***

1. If you declare abstract method in a class, then that class must be abstract as well or need to write abstract keyword
2. If you declare abstract class, then that class can have abstract or non-abstract method both.
3. We cannot create the object of abstract class.
4. If child class is extending parent class and that class is abstract, then parent class of all abstract method must be implemented or with method body in child class.
5. Abstract modifier is applicable for methods and classes but not for variables.
6. We cannot declared static method in abstract class.
7. Child class is extending the properties of parent class by extends keyword or IS-A relationship.

There are two types of methods in java

**a. Normal methods**

**b. Abstract methods**

***Normal methods: - (component method/concrete method)***

Normal method is a method, which contains method declaration as well as method implementation.

***Example: -***

*void m1()* ***--->method declaration***

*{body;* ***--->method implementation***

*}*

***Abstract methods: -***

1) The method, which is having only method declaration, but not method implementations such type of methods is called abstract Methods.

2) In java every abstract method must end with “ ; ”.

***Example : - abstract void m1 (); ----******method declaration***

**Based on above representation of methods the classes are divided into two types**

**1) Normal classes.**

**2) Abstract classes.**

***Normal classes:-***

Normal class is an ordinary java class it contains only normal methods if we are trying to declare at least one abstract method that class will become abstract class.

***Example:-***

*class Test* ***//normal class***

*{ void m1() { body ; }* ***//normal method***

*void m2() { body ; }* ***//normal method***

*void m3() { body ; }* ***//normal method***

*};*

***Abstract class: -***

Abstract class is a java class, which contains abstract or non-abstract method or only abstract method.

**If any abstract method inside the class, that class must be abstract.**

**Example 1: -**

*class Test* ***//abstract class***

*{*

*void m1 () { }* ***// abstract method***

*void m2() {} //* ***abstract method***

*void m3();* ***// normal method***

*};*

***Example-2: -***

*class Test* ***//abstract class***

*{*

*abstract void m1();* ***//abstract method***

*abstract void m2();* ***//abstract method***

*abstract void m3();* ***// abstract method***

*};*

***Abstract modifier: -***

Abstract modifier is applicable for methods and classes but not for variables.

To represent particular class is abstract class and particular method is abstract method to the compiler use abstract modifier.

The abstract class contains declaration of methods it says abstract class partially implement class hence for partially implemented classes object creation is not possible. If we are trying to

create object of abstract class compiler generate error message “class is abstract cannot be instantiated”

***Example -1:-***

*abstract class Test* ***//abstract class***

*{ abstract void m1();* ***//abstract method***

*abstract void m2();* ***//abstract method***

*abstract void m3();* ***//abstract method***

*void m4(){System.out.println("m4 method");}* ***//normal method***

*public static void main(String[] args)*

*{ Test t = new Test();*

*t.m4();*

*}*

*};*

*Compilation error:- Test is abstract; cannot be instantiated*

*Test t = new Test();*

***Example-2 :*-**

**Abstract class contains abstract methods for that abstract methods provide the implementation in child classes.**

**Provide the implementations is nothing but override the methods in child classes.**

**The abstract class contains declarations but for that declarations implementation is present in child classes.**

*abstract class Test*

*{ abstract void m1();*

*abstract void m2();*

*abstract void m3();*

*void m4(){System.out.println("m4 method");}*

*};*

*class Test1 extends Test*

*{ void m1(){System.out.println("m1 method");}*

*void m2(){System.out.println("m2 method");}*

*void m3(){System.out.println("m3 method");}*

*public static void main(String[] args)*

*{* ***Test1 t = new Test1();***

*t.m1(); t.m2(); t.m3(); t.m4();*

***Test t1 = new Test1(); //abstract class reference variable Child class object***

*t1.m1(); //compile : Test runtime : Test1*

*t1.m2(); //compile : Test runtime : Test1*

*t1.m3() ; //compile : Test runtime : Test1*

*t1.m4() ; //compile : Test runtime : Test1*

*}*

*};*

***Example -3 :-***

**Abstract class contains abstract methods for that abstract methods provide the implementation in child classes.**

**if the child class is unable to provide the implementation of all parent class abstract methods at that situation declare that class with abstract modifier then take one more child class to complete the implementation of remaining abstract methods.**

**It is possible to declare multiple child classes but at final complete the implementation of all methods.**

*abstract class Test*

*{ abstract void m1();*

*abstract void m2();*

*abstract void m3();*

*void m4(){System.out.println("m4 method");}*

*};*

*abstract class Test1 extends Test*

*{ void m1(){System.out.println("m1 method");}*

*};*

*abstract class Test2 extends Test1*

*{ void m2(){System.out.println("m2 method");}*

*};*

*class Test3 extends Test2*

*{ void m3(){System.out.println("m3 method");}*

*public static void main(String[] args)*

*{ Test3 t = new Test3();*

*t.m1();*

*t.m2();*

*t.m3();*

*t.m4();*

*}*

*};*

*What is interface?*

*Interface is blueprint of class where we can declared only abstract method.*

* *In interface must be declared only abstract methods.*
* *In interface we cannot declared main method.*
* *We cannot create the object of interface.*
* *If child class is extending the properties of parent class and that is interface, then child class implement all the abstract method of interface in child class.*
* *No static method means we cannot declared static method in interface.*
* *Child class is extending the properties of interface by using implements keyword.*
* *In interface multiple inheritance is possible.*
* *In interface all methods by default abstract in nature means no need to use abstract keyword.*

1. How can we assign parameter in method?
2. Can we use operator overloading in java?

* One operator with different behaviors is called Operator overloading.
* Java is not supporting operator overloading but only one overloaded in java language is ‘+’.
* If both operands are integer, then “**+”** performs addition.
* If at least one operand is String, then “**+**” perform concatenation.

What is concatenation?

When one operand is String and another one is same or int(numeric) that is called concatenation.

* We can achieve concatenation by using two ways
* +
* Concat method which is available in string class.
* ***Example: -***

*class Test*

*{ public static void main(String[] args)*

*{ int a=10;*

*int b=20;*

*System.out.println(a+b); //30 [addition]*

*System.out.println(a+"ratan"); //10Ratan [concatenation]*

System.***out***.println("ratan"+"anu"+2+2+"kids"+2+2); //ratananu22kids22

System.***out***.println(2+2+"kids"+"ratan"+2+2+"anu”) ;//4kidsratan22anu

1. What is string?

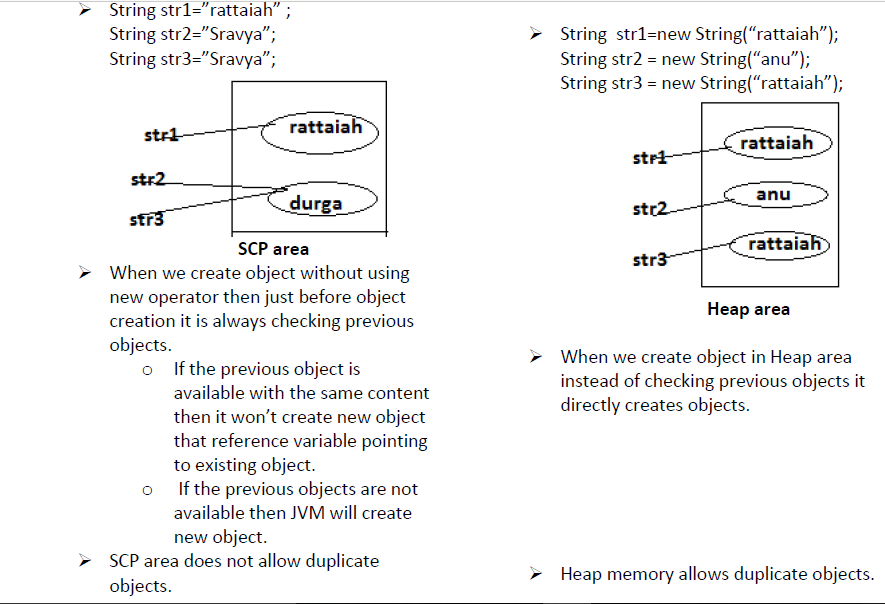
* String is used to represent the group of characters or character array enclosed with in the double quotes.

1. How many types we can create string object?

* By using new keyword
* String s=new String (“Test”);
* Without using new keyword
* String s=” Java”;

1. How many types we can create string buffer object?

* By using new keyword
* String Buffer s=new String Buffer (“Test”);
* ***Case 1:- String vs StringBuffer :*** *String & String Buffer both classes are final classes present in* ***java. lang*** *package.*



1. Where create string object without using new keyword?

String constant pool.

1. Where create string object using new keyword?

Heap memory.

1. Where create string buffer object using new operator?

Heap memory.

1. If string object contain same, then JVM will create object or not without using new keyword?

JVM will not create object because it will give reference of existing object.

1. If string object contain same, then JVM will create object or not by using new operator?

JVM will create object because JVM never check previous object.

1. Why we called string class object immutable by using new keyword and without new keyword?

* String is immutability class which means once we are creating String objects it is not possible to perform the modifications on existing object. (String object is fixed object)
* Example: - class Test {public static void main (String [] args)

{ //immutability class (modifications on existing content not allowed)

String str="ratan";

str.concat("soft");

System.out.println(str); //ratan

String s=**new** String("Aplha");

s.concat("beta");

System.***out***.println(s);

1. Why we called string buffer class object mutable by using new operator?

* StringBuffer is a mutability class it means once we are creating String Buffer objects on that existing object it is possible to perform modification
* //mutability class (modifications on existing content possible)
* StringBuffer sb = new StringBuffer("anu");
* sb.append("soft");
* System.out.println(sb); //anusoft

1. What string class equal method does?

In string class, we check only content of two object if contents are same then it will return true either false because in string class we don’t check reference variable.

* String str1 = new String("Sravya");
* String str2 = new String("Sravya"); //String class equals () method executed (content comparison)
* System.out.println(str1.equals(str2));

1. What string buffer class equal method does?

In string buffer class there is no equal method, but string buffer class use the object class of equals method because object is the parent class of string buffer class by default according to object class equal method we compare reference variable of two string object.

* StringBuffer sb1 = new StringBuffer("anu");

StringBuffer sb2 = new StringBuffer("anu");

StringBuffer class equals () executed (reference comparison) System.out.println(sb1.equals(sb2));

(==) operator--- In case of == if we are using this it always checks that two or more than two reference variables referring same object or not. if they are referring same object then it will return true otherwise it will return false.

String s="abc";

String s1="abc";

(s==s1)//true;

String s1="abc";

String s2="cd";

(s1==s2))//false;

oo

String s1=new String("ab");

String s2=new String("ab");

(s1==s2)//false;

StringBuffer sb=new StringBuffer("cd");

StringBuffer sb1=new StringBuffer("cd");

(sb1==sb2)//false;

Note:-In == operator case never check content of string.

equals():- s

1.If we are using equals() method in stirng class it will always check content .If content are same it will return true otherwise it will return false.equals() method of string class never check object it always check content.

String s1="ab";

String S2="ab";

(s1.equls(s2));//true;

String s3="cd";

(s2.equals(s3));//false;

String s=new Stirng("tt);

Stirng s1=new Stirng("tt");

(s.equals(s1));//true;

String s2=new Stirng("dd");

(s2.equals(s1));//false;

2.If we are using equals () method in String bufferclass as per java concept String buffer has no own equals method so it extends Object class equals method and object class equals () method always check two or more than two refernce variable are refering same object or not .if they are refering same object it will return true otheriwse it will return false.

StringBuffer sb=new StringBuffer("delhi");

StirngBuffer sb1=new StringBuffer("delhi");

(sb.equals(sb1));//false;

== operator in string class always check reference variable are referring same object or not.

== operator in string Buffer class always check reference variable are referring same object or not.

equals method in string class always check content.

equals method in stringBuffer class always check refrence variable are refering same object or not.

1. What are the string class method?

System.***out***.println(str.equals(str1)); System.***out***.println(str.equalsIgnoreCase(str1));

System.***out***.println(str1.length());

System.***out***.println(str1.charAt(29));

System.***out***.println(str1.startsWith("i"));

System.***out***.println(str1.endsWith("l"));

//System.***out***.println(str1.compareTo(str));

System.***out***.println(str1.trim());

System.***out***.println(str1.substring(8));

System.***out***.println(str1.substring(8, 12));

System.***out***.println(str1.concat("API"));

//System.out.println(str1.split("\_"));

System.***out***.println(str1.indexOf("a"));

System.***out***.println(str1.indexOf("a", 14));

System.***out***.println(str1.isEmpty());

//System.***out***.println(str1.contentEquals("q"));

//System.***out***.println(str1.contains(str));

NOTE: String class is final.

NOTE: Integer, double, long, float –Wrapper classes –immutable and final.

What is Array?

Array is the collection of homogenous or similar data type where we store similar data type of elements.

* In array values will be store in the form of index.
* In array value will be start from 0 index.
* Array of super class is object class.
* Array occupied in heap memory.
* Declare the size at the time of creation.
* If you won’t specify the size of array, then you will get compile time error.
* If you specify the size of array 0, then you won’t get compile time array but you will get run time array “java.lang.ArrayIndexOutOfBoundsException”
* If you specify the size of array in negative -1,-2,-3, then you won’t get compile time array but you will get run time array “java.lang.NegativeArraySizeException”
* If you specify the size of array and initialize more than size of array, then you won’t get compile time array but you will get run time array “java.lang.ArrayIndexOutOfBoundsException”
* If you declare the size but you won’t initialize full size of array, then by default ‘0’ value will be printed at console at that index position.
* Advantages:
* In array object we can store multiple values rather than to primitive data type.
* We can access any element randomly by using indexes.
* Disadvantages:
* Once you declare the size of array then you can not add remove the size of array that’s a reason array size is fixed.
* Wastage of memory in array
* Once you declare the data type of array then we can store that type of value only.
* If you add and remove in the array, then performance will be reduced.
* Types of array:

1. SingleDimensionalArray: One D Array

* Declaration: int [] a
* Creation: new int [3]
* We can declare and creation at one-line,

int [] a=new int [3];

* Initialization: Initialization means you need to assign the values in the form of indexes.

Int [0] =10;

Int [1] =20;

Int [2] =30;

* Retrieve: we can retrieve the value or print the value of array through for loop or for each loop.

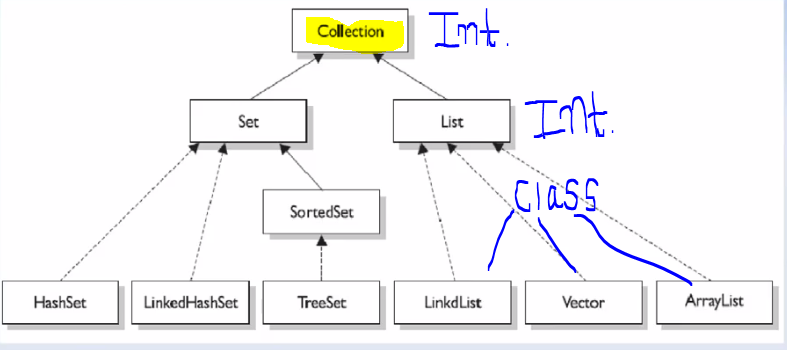
1. What is collection?

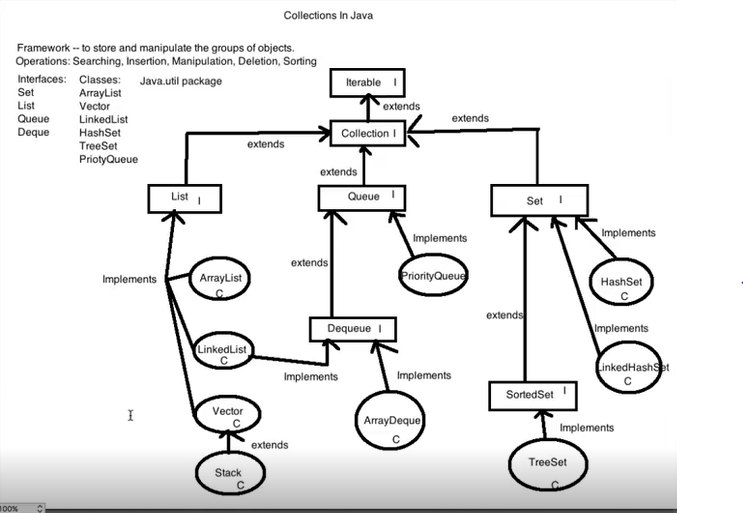
* Collection is an interface and dynamic array that provides an architecture to store and manipulate the group of objects, because in collection data will be store in the form of object not in the form of data types and all the operation you perform on data searching, sorting, insertion, manipulation etc. can be performed by java collection.

Why should use collection why we not use array?

Whenever you create the array, you have to declare the data type of object and as well you have to initialize the same data type of object in array. We cannot change the data type of object at the initialization time.

* Type – fixed type just suppose in student table has multiple columns or each data have different types. If you want to create student name then you have to create string array, or you want to create roll no. then you have to create int array etc, so we should not use array.
* Size – fixed size because whenever you create the array we have to declare the size of array, once you create the array of size you cannot change the array of size at the time initializing.





What is List?

List is an interface which is present in java. util package and it is extended by collection interface.

What is Array List with syntax?

Array list is the implemented class of List<I>, which uses dynamic array. To create an array list of java we need to create the object of array list class, which is present in java. Util package.

* Methods in Array list

#Methods in Array List

1.add

2.size

3.getindex(i)

4.remove(i)

5.Addall(collection)

6.IsEmpty(collection)

7.contains ()

8.retainsAll

Properties of Array List –

1. Array list is working based on dynamic array.

2. its allowed duplicate values.

3. its maintain insertion order.

4. its allows random access to fetch the particular element or object because it store on the basis of index.

5. non-synchronized.

6.It allow null value.

**NOTE** – Array List is a raw type, so you can store multiple types of object.

What is Linked List with syntax?

Linked list is implemented class of List<I>.Which use doubly linked list to store the element.It provides a linked-list data structure means pervious node and next structure.

Methods in linked list

#Methods in linked list

1.add

2.size

3.getindex(i)

4.remove(i)

5.Addall(collection)

6.IsEmpty(collection)

7.contains()

8.retainsAll

9.AddFirst()

10.Addlast();

11.removefirst()

12.removelast()

13.set(index,””)

Properties of linked List –

1. Linked list is working based on doubly linked list.

2. its allowed duplicate values.

3. its maintain insertion order.

4. its allows random access to fetch the specific element or object because it store on the basis of index.

5. non-synchronized.

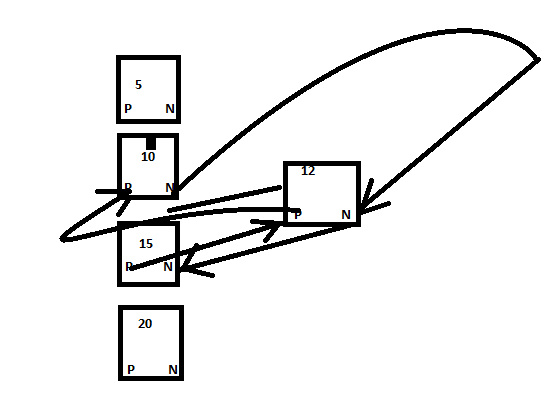
6.It allow null value

**NOTE** – linked List is a raw type, so you can store multiple types of object.

* Difference between array list and linked list
* Array list is working based on dynamic array but linked list is working based on double linked list means node.
* Major difference is performance in array list when you add or remove to

Element in between list then rest of the element of index will be up and down or shifting the place because its uses index to add or remove to element that’s y array list performance is slow but in linked list when you add or remove to Element then all element of node will not up and down or shifting the place only two element will be shifted in between

Means before one and after one element because its uses **Previous node and Next node** due to this reason performance is high.

**Linked list diagram**

Array List:

1. Array list is working based on dynamic array
2. Array list is slower rather than to linked list in manipulation only because it uses index but in searching faster rather than to linked list.

Linked List:

1. Linked list is working based on doubly linked list.
2. Linked list is faster rather than to array list in manipulation only because it uses previous node and next node but in searching slower rather than to Array list.

NOTE - If you want to add or remove to element then always go for linked list.

If you want to fetch the element, then always go for

Array list.

* Difference between array list and vector?
* Vector introduced in java 1.2 And Array list introduced in java 1.5
* In vector, vector has Capacity (); method but array list has not.
* If you exceed the limit or add the more element, then vector will increase the capacity of list by 100% means vector waste the memory but in array list will increase the capacity of list by 50% means array list save the memory.
* Array list is non-synchronized, but vector is a synchronized.

What is generic in collection?

Java new generic it will allows you to have only one type of object in collection, so

Now it is type safe so typecasting is not required at run time.

Write syntax of generic in collection?

Array List<**String**> obj=**new** Array List<**String**> ();

What is set?

Set is an interface, which is present in java. util package, and it is extended by collection interface.

What is Hash set?

Hash Set is implemented class of Set<I>.Its allow null value at only one time.

Properties of Hash Set –

1. Hash set is working based on doubly linked list.

2. It does not allow duplicate values.

3. Its maintains random order.

4. non-synchronized.

5. Can’t be accessed using index.

6.Its allow null value at only one time.

What is Linked Hash set?

Linked Hash set is implemented class of Set<I>.It allow null value at only one time.

Properties of Linked Hash set –

1. Linked Hash set is working based on doubly linked list.

2. It does not allow duplicate values.

3. Its maintain insertion order.

4. non-synchronized.

5. Can’t be accessed by sing index.

6.It allow null value at only one time.

What is Tree Set?

Tree Set is implemented class of Set<I>.In tree set does not allow null value because if insert null value it gives exception at run time **Null pointer Exception**.

Properties of Tree set –

1. Tree set is working based on doubly linked list.

2. It does not allow duplicate values.

3. Its maintain ascending order.

4. non-synchronized.

5. Can’t be accessed using index.

6. In tree set does not allow null value

How to convert set into List or Array List with syntax?

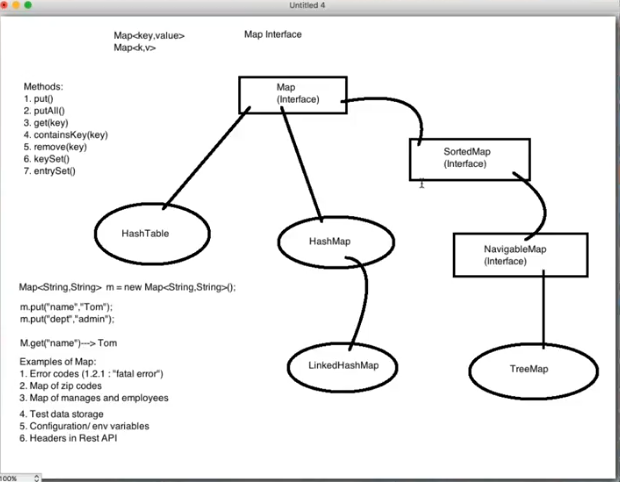
Set <string> obj=new set <string> (); //creation of set object

Array list <string> obj1=new Array list<string> (**obj**); // creation of list object

Here you have to pass the reference object of set in array list object then you can access the value by index in set.

What is Map?

Map is an interface present in java. util package and it is extended by collection framework.



What is Hash Map?

Hash map is an implemented class of map<I>, in hash map element will be store in the form of key and value.

Properties of Hash Map:

1. All element will be store in the form key and value.
2. It does not allow duplicate value. //if both of key is same then last of the key will be printed.
3. Its maintain random order, when you removed to element then rest of the element will not be shifted because it’s not uses index, its uses key and value form, so that removed key and value will be blank and not printed on console.
4. Hash map is non-synchronized that is a reason it is faster to hash table but not thread safe.
5. It may have one null key and multiple null values
6. Concurrent modification exception – fail –fast condition.

What is Hash Table?

Hash Table is an implemented class of map<I>, in hash map element will be store in the form key and value.

1. All element will be store in the form key and value.
2. It does not contain duplicate value. //if both of key is same and last of key will be printed.
3. It maintains random order, when you removed to element then rest of the element will not be shifted because it’s not uses index, its uses key and value form, so that removed key and value will be blank and not printed on console.
4. Hash table is synchronized that is a reason it is slower but thread safe.
5. It does not allow null key and value as well.

What is Hash code?

Hash code is no., which will be given by JVM or compiler, whenever create the object of class means whenever you create the object hash code generate the no. for object of class and this is a 32-bit int.

Difference between hash map VS hash table?

1. Hash map is non-synchronized that is reason it is faster but not thread safe, but Hash table is synchronized that is reason it is slower but thread safe.
2. Hash map allowed one null key and multiple null value, but hash table not allowed any null key and value as well.
3. In hash table, store the value based on hash code of key.

NOTE: If you want to fetch in insertion order in map always go for linked hash map.

What is this keyword?

this keyword refers invoke to current class instance and static variables.

* If instance and static variable and local variables name are same so we can use this keyword to access current class instance variable, then this keyword refers to current class instance variable.

What is super keyword?

super keyword refers invoke to parent class instance and static variables.

* If instance and static variables and local variables name are same in parent class and child class using extends keyword, so we can use super keyword to access parent class instance variable, then super keyword refers to parent class instance variable.

1. What is exception?

* An exception is an event that occurs during the execution of the program that disturb normal flow of the program instructions.

1. What is error?

Exceptions are caused to several reasons like developer mistakes, end user input mistakes, network problems.

But error is caused due to lack of system resources.

Heap memory full, Stack memory problem, AWT component problem etc.

Example: - StackOverFlowError, OutOfMemoryError, AssertionError…………etc

* We are handle the exceptions by using try-catch blocks or throws keyword, but through throws keyword it is not possible to handle the errors.

1. What is throwable?

Throwable is a class which extends error and exception both.

1. What is exception handling?

* Exception handling means just we are providing alternate code to continue the execution of remaining code & to get normal termination of the application.

1. How many types of exception?

1) Checked Exception

2) Unchecked Exception

1. What is checked exception?

Those Exceptions, which is checked by the compiler at the time of compilation, is called Checked Exceptions.

Examples: - IOException, SQLException, InterruptedException, ClassNotFoundException etc.

1. *What is UN checked exception?*

* Those exceptions which is not checked by the compiler at the time of compilation are called unchecked Exception.
* If the application contains un-checked Exception code is compiled but at runtime JVM (Default Exception handler) display exception message, then program terminated abnormally.

Examples: - Arithmetic Exception, ArrayIndexOutOfBoundsException, NumberFormatException, FileNotFoundException, Null pointer exception etc.

Difference between checked Vs unchecked exception?

* Checked exceptions are occurred at the time of compilation time while un checked exception are occurred at the time of run time.
* *For the checked exception when we write try-catch blocks or throws keyword then only code is compiled but for un-checked exceptions try-catch or throws keyword optional.*

1. Can we use multiple catch block with one try block?

Yes, we can if you are putting related code in try block.

1. Can we give try-catch block inside try block?

Yes, we can.

1. Can we give try-catch block inside catch block?

Yes, We can.

1. Can we write try-finally without catch block?

Yes, we can but after finally code will not be execute.

1. Can we give any statement immediate try block?

No, we cannot.

Ex:

try

{

Int x=10;

}

Int a=20; 🡪 it will not allow.

Catch ()

{

}

1. What is finally block?

Finally, blockis always executed whether exception is handle or not.

1. What happens if exception occur in try block, remains code of try block will execute or not (at runtime won’t execute)

No remaining code of try block will not executed. (Please follow Q. 40 and for program also in eclipse)

1. What order should give catch block if we are declaring multiple catch blocks?

Order should be Child to parent or should be only exception in only one catch.

1. What methods are used to print exception information and difference of them?

* *printStackTrace()) – this method prints name of exception, description of exception and line number where you occurred exception.*
* *toString() – this method prints name of exception and description of exception.*
* *getMessage() – this method prints only description of exception.*

NOTE – All exception will be stored in created object i.e ‘e’

**catch**(Exception e)

1. What happens if exception occur in try block, remains code of outside of try or catch block code will execute or not?

Code will be execute.

Difference between Throw and Throws?

1. Throw always is used to explicitly throw the exception or create own exception but throws keyword is always is used to declare the exception at method level.
2. You cannot throw multiple exception, but in throws you can declare multiple exception.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Final** | **finally,** | **finalize** |
| 1) | Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden, and final variable value can't be changed. | Finally, is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| 2) | Final is a keyword. | Finally, is a block. | Finalize is a method. |

1. If, else

**int** a=10;

**int** b=20;

**if**(a>b){

System.***out***.println("B is greater then a");

}

**else**

{

System.***out***.println("A is smaller then b");

}

1. if, else if, else

**int** a=10;

**int** b=10;

**if**(a>b){

System.***out***.println("B is greater then");

}

**else** **if**(a==b)

{

System.***out***.println("A is equal to b");

}

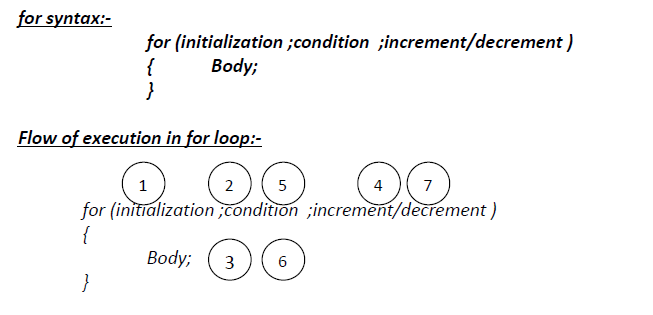
**else**

{

System.***out***.println("Both above condiion are failed");

}

}



1. while loop

**int** i=0;//initialization

**while**(i<=5)//conditional part

{

System.***out***.println(i+" avi");

i++;//incremental/detrimental part.

}

}

}

1. for loop 1. example

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

//int i=0=>intilization,i<20=>conditinoalpart,i++=>incremental/decremental

System.***out***.println(i+" =avi");

}

System.***out***.println("End");

}

1. for loop 2. Example

**for**(**int** i=10;i<20;i++) //int i=0=>intilization,i<20=>conditinoalpart,i++=>incremental/decremental

{

System.***out***.println(i+" => HCL");

}

System.***out***.println("End");

}

}

1. for loop 3. Example

**public** **static** **void** main(String[] args) {

**int** i =1;//Inside the for loop initialization part is optional.

**for**(;i<10;i++)

{

System.***out***.println(i+" => avi");

}

System.***out***.println("End");

}

}

1. for loop 4. Example

**public** **static** **void** main(String[] args) {

**int** i=1;//In the initialization part it is possible to take any number of System.out.println(“ratna”)

//statements and each and every statement is separated by comma(,) .

**for**(System.***out***.println("avi"),System.***out***.println("hcl");i<10;i++)

{

System.***out***.println(i);

}

System.***out***.println("End");

}

}

1. Compile time error =syntax error : compile time error occurs during writing the code means you are not following programming language concept.
2. Runtime error : runtime error occurs during the code execution.

What is constructor and why?

Constructor is block same as methods but not a method and having same name as that of class name.

\*Constructor are used to initialize the objects or to get unique values.

* Constructor doesn’t return anything.
* Access modifiers applicable for constructor only are public, protected, private and default.
* It executes automatically when we create the object.

How many types of constructor are available?

* Default constructor or no argument constructor: Default constructor is always creates by compiler even you didn’t create the constructor in the class and always creates no argument constructor and if you create constructor inside the class then compiler won’t create any default constructor.
* User defined constructor or no argument constructor: User defined constructor always creates by user with no argument and need to create the constructor to call them and called when we create the objects.

Constructor()

{

}

Constructor c4=new Constructor();

* Parametrized constructor: Parametrized constructor always creates by user with arguments and need to create the parametrized constructor to call them and called when we create the objects.
* Constructor(String EmpName,int EmpId)

{

}

* Constructor c4=new Constructor(“Tom”,101);
* Can we overload the constructor?

Yes, we can over load the constructor with same constructor name with different data type or same data type but different arguments.