parameters	**Static**	**Non-static**
Keyword	Static	-
Calls	Direct static to static	Only calls when object is created
Memory	Memory allocated for static variable is only once	Memory allocated for non-static variable is multiple times whenever object is created
Members calls	Only static	Static and non-static

Non static == non static and static
Static == static

```
Int a=10; //
Int b=20;
```

memory

10	20		
a	b		

```
Syso(a);
```

Cons.	method
It calls when object of a class is created	It calls with method name not call directly
Not return type	Return type
Same as class name	Give any name of method
It takes parameters	It takes parameters
We can only overload cons. Not override	We can overload and Override methods

Cons.

```
public demo2() {
		System.out.println("default cons. calling");
	}
	
	public demo2(int a) {
//		this();
		System.out.println("int para cons. calling");
	}
	
	public demo2(String a){
		this();
		System.out.println("string para cons. calling");
	}
	
	
	
	public static void main(String[] args) {
		
		demo2 b=new demo2("java");
		System.out.println("===========");
		demo2 c=new demo2(20);
```

Access modifiers:

All are keywords use in oops that is set the accessibility of classes, methods and other members.

1. Public ==anywhere
2. Default == within package
3. Private== within class
4. Protected== within package but with the help of inheritance we can access outside a package.

So how we can access private members in other class?
➔with the help of getters and setters.

Public private static String p= “Taran”;

```
public static String getter() {
    return p;
}

public static void setter(String q) {
    p=q;
}
```

What is OOPS?

-In java we can write and execute a code with the help of objects and classes, without object and class we can write and run a program that’s why is called oops.

- Oops real world problems solve.

Major Pillars of oops.

Inheritance, abstraction, polymorphism, encapsulation, interface.

Super, parent, base ➔the class get inherited to another class.

Child, sub, derived➔the class inherited to another class.

Parent to parent ➔only parent class prop.
Parent to child ➔Only in parent class prop.
Child to parent ➔Invalid.
Child to child➔ both the classes.

If one method presents in both classes then priority goes to local class or child class.

Why multiple inheritance not support in java?
➔In java not give support of multiple inheritance Because two parent class of one child class is not possible.

It results diamond ambiguity problem occur.

This and super

```
public class classa{
	int a=10;
	public void m1() {
		int a=20;
		System.out.        (a);

		System.out.        (this.a);
	}

	public static void main(String[] args) {
		classa b=new       ();
		b.  ();
	}
}
============================================================
public class classc extends classa {
	int a=40;

	public void m3() {
		int a=50;
		System.out.       (a); //50
		System.out.       (this.a); //40
		System.out.       (super.a); // 10
	}


	public static void main(String[] args) {
		classc ob=new       ();
		ob.  ();
	}
}
```

Polymorphism ➔Ability of an object to take many forms. Or
              One word having diff. meaning.

There are two types of poly.
1.compile time(overloading)
2.Runtime poly(overriding)

1.in this, code bind to its definition during compile time.

Declaring multiple methods in same class with same name but
  diff arg. Or parameters are called overloading.
E.g.= test(), test(int a), test(String b)

2. In this code bind to its definition during runtime or execution time.

-It done dynamically so also called dynamic binding.
-It acquires superclass prop. Into base class with the help of inheritance concept.

Static method we can overload but we can’t override static method why?

➔Method overriding is based on dynamic binding at runtime and the static method are based on static binding at compile time. So, we can’t override static method.

3.Abstaction (data hiding)

-In this process hiding implementation details and showing only functionality to end user.
We can’t make body of abs. method.
-Only show req. info to user.
Eg. car aahe fakt chalav bhanbhan karu nako aat kay kas start hot.

We can’t make object of an abstract class we can only take reference of it.

1. concrete class (Normal class)
2. Abstract class (at least one abstract method)

Interface: It is a pure abstract in nature.
All methods are by default public and abstract other modifiers we can’t use.

Interface is inherited by using “implements” keyword.

```
public interface WebDriver {

     public void getTitle();
     public void getisDisplayed();
     public void getisEnabled();
     public void getAtribute();
     public void manage();
     public void quit();

}
public class chromeDriver implements WebDriver{
@Override
     public void getTitle() {
     System.out.println("get page title");
     }
@Override
     public void getisDisplayed() {
     }
@Override
     public void getisEnabled() {
     }
@Override
     public void getAtribute() {
     }
@Override
     public void manage() {
     }
@Override
     public void quit() {

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

          WebDriver driver=new chromeDriver(); //upcating
          driver.getTitle();

          chromeDriver d= new chromeDriver();
```

```
        //If we make object of child class then we can
access all method other than interface class.

  }
```

Abstract class	Interface
It contains abstract and concreate class.	It contains only abs. class
Abstract keyword is used declared an abs. class.	Interface keyword is used as class name.
Extends is used to inherited	Implements keyword is used to inherited
Class members are private, protected and public also default.	In this only public is used by default.
It doesn’t support multiple inheritance.	In this multiple inheritance is support.

```
WebDriver driver=new chromeDriver(); //upcasting
    driver.getTitle();

we can access only interface class methods.

If we make object of a child class then we can access all
  methods which are extended but not in Interface class.
```

Java is not fully object oriented because it supports primitive data type like it, byte, long etc., which are not objects. Because in JAVA we use data types like int, float, double etc. which are not object oriented, and of course is what opposite of OOP is.

Wrapper Class: Wrapper class provides the mechanism to convert primitive into object and object into primitive. In Java, you can use Integer, Float etc. instead of int, float etc. We can communicate with objects without calling their methods. ex. using arithmetic operators.

Primitive ➔	boolean	char	byte	short	int
Wrapper class ➔	Boolean	Character	Byte	Short	Integer

e.g.   int a=10;

Integer b=a;

Even using Wrapper classes does not make Java a pure OOP language, as internally it will use the operations like Unboxing (convert primitive datatype info into wrapper type info) and Autoboxing (convert wrapper datatype info into primitive type info). So, if you create instead of int Integer and do any mathematical operation on it, under the hoods Java is going to use primitive type int only.

Exception ➔It is unwanted / unexpected situation or event occurred during runtime of program.

-It occurred during runtime.

Exception handling➔ why we need to handle exception because

1. To maintain flow of programme.
2. Exception break the code.

Definition

Why

Types

We can handle exception by using below methods.

1. Try
2. Catch
3. Throw
4. Throws
5. Finally

try and catch➔In our code is any chances of occurred excp.
Then we use try and catch block.
-risky code we write in try block, in catch block we handle the exception which are throws by try block. we can show or print what exception is occurred and maintain flow of execution.

Only one block will execute.

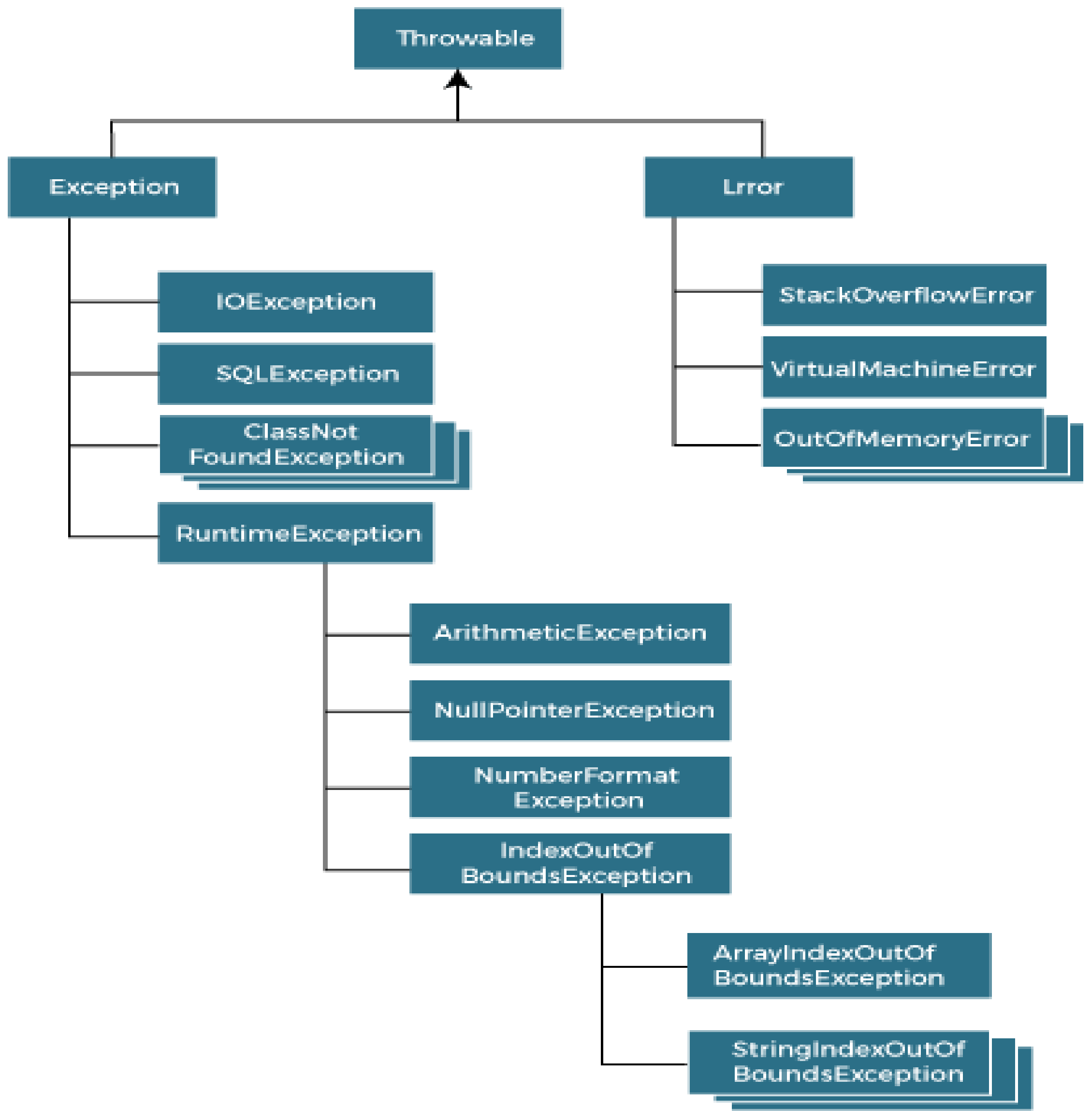
Without catch block we can handle exception?
➔ No. why, alternative option is finally block used.

finally➔ It is a method which have a body.
this block is executed compulsory. If excp. Is occurred or not.
Why this is used == if catch block not mention code can't maintain flow of execution. Then we write remaining code in finally block.
In this we write imp code.
Eg. File close(), security code().

Throw and throws➔ in throw user can create custom exception & In throws user not take responsible to handle exception. This exception. Handle By JVM.

Throws are not recommended to use it not handle all types of exceptions

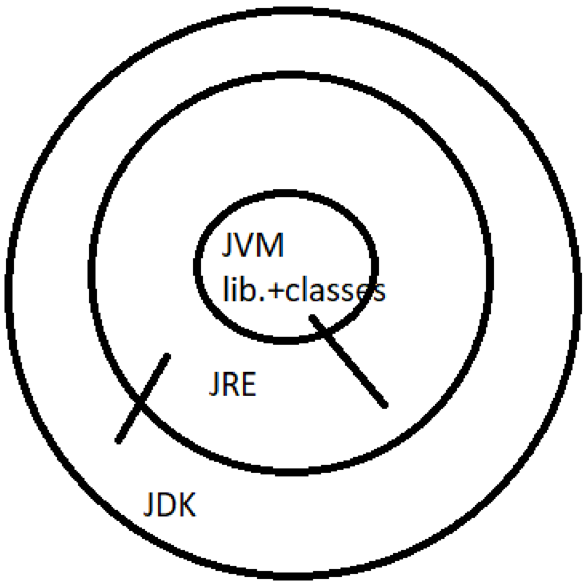
Always go with try catch block.



Checked Exception	**Unchecked Exception**
These exceptions are occurred at compile time.	These exceptions are occurred at run-time
These are checked by the compiler.	These are not checked by the compiler.
Examples Class not Found Exception File not Found Exception	Examples Null Pointer Exception Array Index Out of Bounds Exception

Final	**Finally**	**finalize()**
It is keyword or access modifier.	It is block. (Used with try-catch)	It is method of object class.
It is applicable for variable, method &classes.	It is always executed weather the exception is handled or not	It is used to deallocate the resources which are allocated by unused object
Once it declares it becomes constant and can’t modify. - Also, it can’t override by subclass. - Also, can’t inherited.	In this block important code will write which we want to execute anyhow	it performs cleaning activities which respect to the object before its destruction

JDK (java development kit)	**JRE (java runtime environment)**	**JVM (Java virtual machine)**
It is used to develop java application.	It provides the environment to execute java code.	It is responsible to execute java code
Contains no. of development tools, compiler and debuggers etc.	It is internally containing JVM which are responsible to run java code	It is software written is "c" lang.
It contains inbuild JRE and JVM		Classes + lib.



JVM == classes and lib.
JRE == JVM(classes and lib.)
JDK==JRE + JVM

Collection ➔ It is set of predefined classes and interface in java, that helps to done different types of data structure operations like sorting, searching, storing, insertion, deletion by efficient manner.

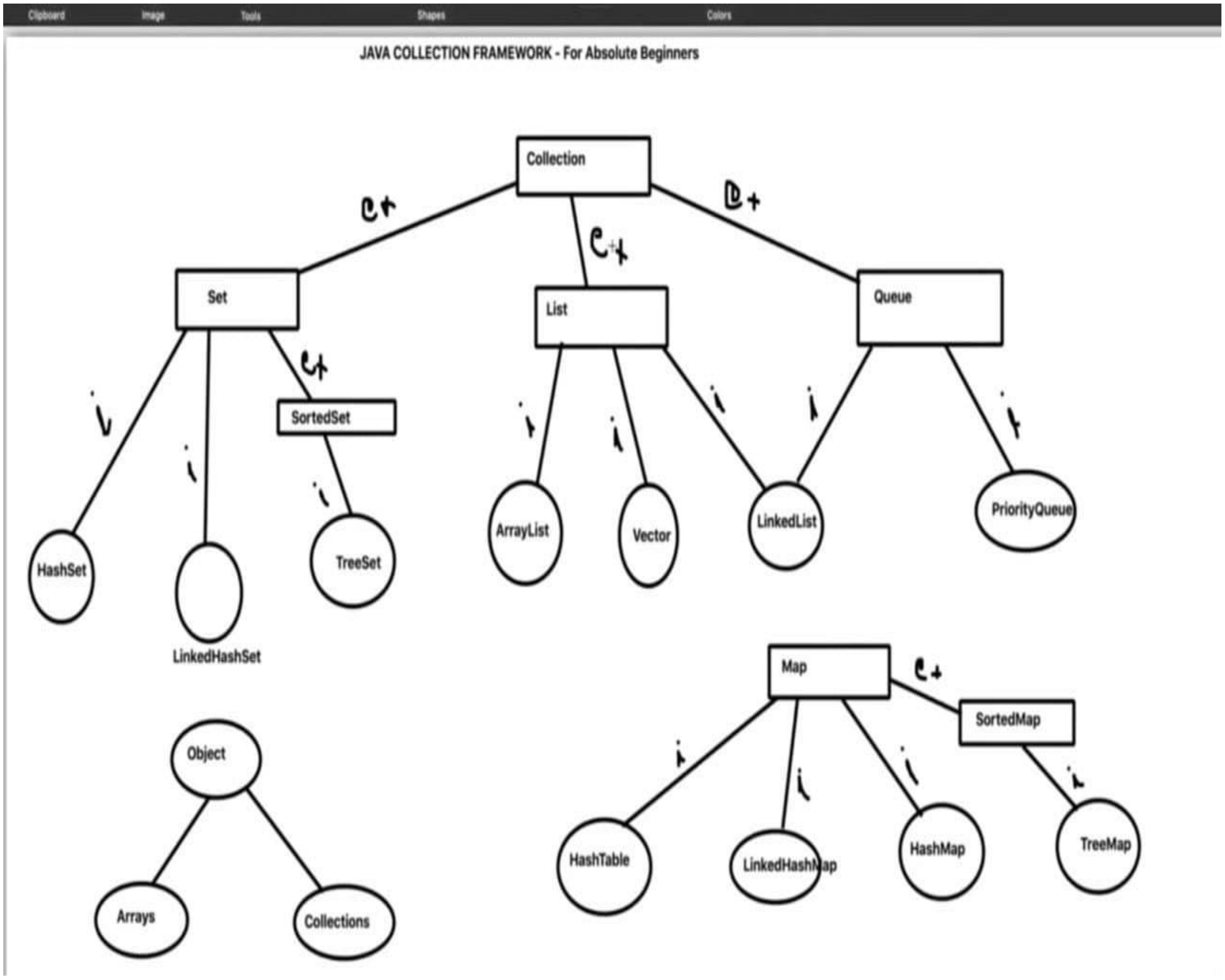
- Java.util package is parent class of collection.

Why collection is used?

Array size is fixed. Size cannot be increased dynamically.

Insertion or deletion of element in the array is also costly in terms of performance.

1. Adding object in the collection dynamically.
2. Removing the object from collection
3. Retrieving specific object.
4. Searching specific object from collection.
5. Deleting particular object.



List	**Set**	**Map**
List is an interface allows duplicate elements	Set does not allow duplicate elements	The map not allow duplicate elements.
It maintains insertion order	Set does not maintain any insertion order	The map also does not maintain any insertion order
We can add multiple null values.	Inset only one null value we can store	The map allows a single null key at most and multiple number of null values
if we need to access the element frequently by using the index then list is use.	If we want create a collection of unique elements then we use set.	If we want to store the data in the form of key/value pair then we use map.
It provides get () method to get the element at a specific index.	It does not provide get method.	Also, it does not provides get method.
List implementation classes are ArrayList, LinkedList and vector.	Set implementation classes are hashset, Linkedhashset and treeset.	Map implementation classes are Hashtable, treemap and Linkedhashmap.

Hashtable	**Hashmap**
It is synchronized.	It is not synchronized.
It doesn’t allow any null key or null value.	It allows one null key and multiple null value.
It is slow	It is fast
Hashtable is internally synchronised so it can’t be unsynchronised	We can make hashmap synchronised by using Map m=collections.synchronisedmap(hasmap).

ArrayList	**LinkedList**
It is internally uses a dynamic array to store the elements.	It internally uses a doubly linked list to store elements.
Its slow, its internally uses an array, if we remove any element so other values shifted in memory.	Its faster than arraylist because it uses doubly LinkedList so no shifting is required in memory if any element is removed.
It implements only list.	It implements list and queue.
It is faster in storing and accessing data.	It is faster in manipulation data.

String buffer class

```
/*
          normal string is immutable (not changeable)

          string buffer is predefined class present in java.
          it is same like string but it have some more
features than normal string like.
          when we assign a value to string buffer it take
more memory or extra space than total value.

          insert(at perticular index), reverse, delete(from
perticular index), replace, findcapacity.

         */

         StringBuffer sb=new StringBuffer("Shashank");

         sb.append("meka.");             //insert from last index.

         System.out.println("capacity of string "+sb.capacity());
         System.out.println("length of string "+sb.length());

         System.out.println(sb);

         System.out.println(sb.insert(8, " sanjay "));

         sb.insert(0, "Name : ");
         System.out.println("capacity of string "+sb.capacity());
         System.out.println("length of string "+sb.length());

         System.out.println(sb);
         System.out.println(sb.reverse());
```

➔
```
 capacity of string 24
length of string 13
Shashankmeka.
Shashank sanjay meka.
capacity of string 50
length of string 28
Name : Shashank sanjay meka.
.akem yajnas knahsahS : emaN
```