INTRODUCTION TO JAVA

Java is by Sun micro systems AND Now it is acquired by Oracle
Father of Java is James Gosling

Java is developed from C++

Java is Platform independent.

In Java when compile a program it creates the .class file or it can also be called byte code-- This is done by Java compiler

Javac HelloWorld.java

Java HelloWorld
Or
Java -cp . className

JVM-> this reads the byte code or .class file and converts into .exe and executes it

Java is Platform independent

And JVM is platform dependent

Compilation:

Compilation is nothing but java checks for the syntactical errors in the program, like the below points

1. **Every statement in java should end with semi colon ; (except class and method starting line)**
2. **Keywords like class, public etc should be small letters**

Executing the Program:

1. **Executing the program is nothing but running the program at this point java checks for the main method. If there is any change to the main method like instead of String arr[], if we pass int a[] then system gives the error during running not during the compilation**

Questions:

- **When there is no main method in the program does the program compile ?**
- **When there is no main method does the program run ?**

- **Can I interchange the public and static words in main method ?**
- **Why the main method is getting executed without object ?**

What is Object oriented principles ?
Encapsulation
Polymorphism
Abstraction
Imheritance

What is class ?
What is object ?

- **Class – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.**

- **Object – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.**

Flower- class for flower

Rose object
Lilly object…
Rose,
lilly

Static
Final
Abstact

Methods

Data types:
1. **byte**
2. **char**
3. **int**
4. **long**
5. **float**
6. **double**

Static method
Final method
Abstract method

Object oriented principles

1. **Inheritance-- enabling the reusability**
2. **Polymorphism--exhibiting the different behaviors-- over riding and over loading**
   Compile time (Static)
   Run(Dynamic)
3. **Encapsulation --enables making the variables and methods into one Unit**
   Test
   Private String name;
   Public SetName();
   Public getName();
   Encapsulation allows data hiding
4. **Abstraction--hiding the things**
   Abstraction allows logic hiding

Variables:

c=a+b;-->

Long a=3232323232323232

Integer range 0 to 32567--4 bytes of memory

Long

Double

Float

Char--'a' or 'b'

String ="sdadasdassa454$$$d"

Integer or Long--Numbers
Int a;
Int b;
Int c= a+b;

Input as 5--a

What is primitive
Int,long, float,

Integer, Long, Float, Double, Character, String

Execution of Java program starts from Main method
Main methos syntax should be
public static void main(String[] a)

Object class is the super class for all classes

Example3 ->child ->parent->Object

First download Eclipse, Spring tool suite

OOP principles

Access Modifiers

a)

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Private -> Default -> Protected -> Public

Inheritance:

- Inheritance is possible with extends and implements keyword
- Private variables or private methods are not inherited
- Protected method can be accessed from sub class of other package but only in terms of inheritance only
- Difference between protected and Default is -> Default variables and methods can be accessed only in the same package whereas protected variable can be accessed in same package and the sub class of other package
- Constructor hierarchy is Object class -> Parent class->child class
- Super class constructor ->instance initializer block->child class constructor
- Base b= new Child()-->Parent p = new Child();
- We can access overridden methods of child class and method of super class. We cannot access the method only written in child class
- Static methods will not be overridden, if user tries to override the static method in child class then it is treated as redefining the method, because static methods and variables are per class
- Overriding is possible for instance method and instance variables only.
- Final variables and final methods are not inherited.

Encapsulation

- Encapsulation in Java is a *process of wrapping code and data together into a single unit*
  We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

- The Java Bean class is the example of a fully encapsulated class.

- It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
  It is a way to achieve data hiding in Java because other class will not be able to access the data through the

private data members

- If we want to achieve the immutability then encapsulation is the best.

Abstraction

- Abstraction is nothing but hiding the logic.
- It can be implemented by overriding principle.
- Refer to Overriding principles.

Polymorphism

- **In Java 2 types of polymorphism are present. 1. Compile time 2. Run time**
- **Compile Time (Static polymorphism)** ->compile time polymorphism or static polymorphism is implemented using over loading->
- Constructor over loading is possible.
- Whenever we create object to child class, based on the argument that we pass, corresponding constructor gets called and there will be a super() (super with no arg) to the base or super class hence no org constructor of super class gets executed and then child class parameter constructor gets executed.

- **Run Time polymorphism (Dynamic polymorphism)** -> this can be implemented using Over riding principle
- Constructor over riding is not possible, because in overriding we have to have the same name in child and super classes but when it comes to constructor class name is same as constructor name so there is no way that child class and super class will have same name. even if we write same that constructors are not treated as overriding.

Overriding Rules

- Method name and type or arguments and number of arguments in super and sub class should be same
- Will occur in terms of inheritance
- Access modifier of overridden method in child class should be the super type of access modifier to the super class. (in child class over ridden method should increase the visibility, it should not decrease the visibility)
- Return type of the overridden method in sub class should sub type of super class method return type--Co varient return type
- Exception level in overridden method should sub type of super class method exception level
- Private -> default-> protected->public

Overloading Rules

- Method name should be same
- Type of arguments or number of arguments should be different to implement overloading
- Occurs in same class only
- These overloaded methods are resolved at compile time.

Interfaces and Abstract Classes

18 June 2020 09:05

Example of Abstract class: void display();

One interface can extend another interface
One class can implement multiple interfaces
Multiple inheritance can be achieved using interfaces
Interface will have only abstract methods and final variables before java 1.8

What is abstract class ->if a class have at least 1 abstract method then the class becomes abstract class.
Abstract class can have non abstract methods
If I declare the class as abstract there is no need to have abstract method.
Why we cannot create the object for abstract because we don’t how much memory gets allocated for abstract method.
Because one can write 4 variable and other can write 10 variables.

Abstract Class:
Abstract class can have abstract methods and nonabstract methods
Abstract class cannot be instantiated, because since the abstract methods are present compiler cannot determine the memory allocation hence object creation or instantiation is not possible.

Difference between abstract class and interface

1. **Interface facilitates multiple inheritance**
2. **Whereas Abstract class facilitates multilevel inheritance**
3. **Interface can have only final variables whereas abstract final variables can be initialized in constructor hence we can have the final variable value per object**
4. **If 100 classes are implementing the interface then if there is any change or any addition method in interface then we have to override that method in all 100 classes. Which makes the code change in 100 classes**
5. **But if we use abstract class then no need of changing in the 100 classes just write non abstract method in abstract class since we are using it in terms of inheritance we can use this method**
6. **Abstract class can have the constructor but interface doesn't have the constructor**

Auto Boxing, Unboxing, widening, Variable arguements

19 June 2020 10:29

1. **Converting the primitive values into the corresponding wrapper class is nothing but auto boxing**
2. **Getting the primitive value from the wrapper class is nothing but un boxing**
3. **Widening: first compiler checks for the similar primitive method exist, if not then the control will be transferred to next higher primitive. If this is also not found then control will go to corresponding wrapper class of primitive (Auto boxing)**
4. **If we have primitive and there is no primitive method then control goes to the wrapper class of the primitive type**
   Int i=10
   Display(10);

   Public display(float d){
   For the above method call though we are sending the primitive int argument since we don't have any method with int argument the compiler will search for next higher primitive method argument. In this case the next level is long but we don't have long primitive method and next level is float hence float argument method gets executed.
   This is called Widening
   }

1. **Variable arguments:**
   This is introduced from java 1.5
   What is the difference between array and var args
   If you use array then the calling method should also

send array but using var args u can send primitives no need of creating the array

2. **Var arg should be the last parameter in the method**
3. **There should be only one var arg should present per method, because if we try to add multiple var args then compiler doesn't know how to divide the values between 2 var args.**

Wrapper Classes, String, User Defined Final Class

20 June 2020 08:35

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
	String

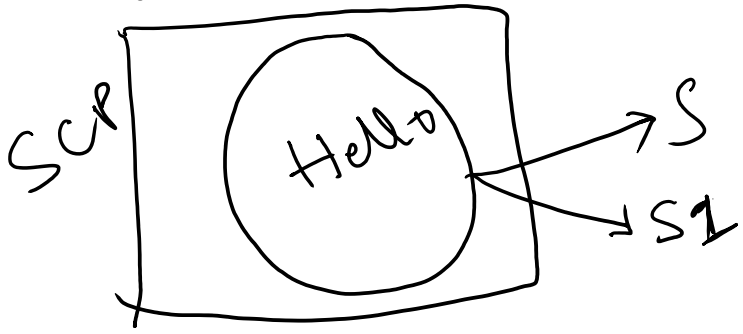
1. Wrapper classes are final
2. Because inheritance should not be done
3. Character wrapper class doesn't have the constructor with String argument and all other wrapper class are having constructor with corresponding primitive type and the string argument
4. Any operation that you perform on the string will return a new string object and it doesn't change the original string value
5. Why string is immutable.
   Whenever we create the String object using string literal
   String s="Hello";

   For the above statement memory gets occupied in String constant pool.

   String s1= "Hello"

   Now for this statement, since memory is already present for Hello in String Constant pool, no new memory will be created, both S and S1 will point to Hello like below.

both S and S1 will point to Hello like below.

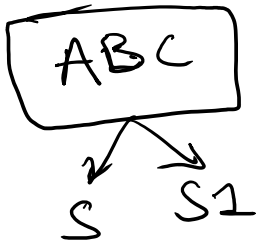


As long as the value is same new memory will not be allocated. Now since both S and S1 are referring to same location. If the content is modified then it gets reflected in both which is not desirable. Hence Strings are immutable.

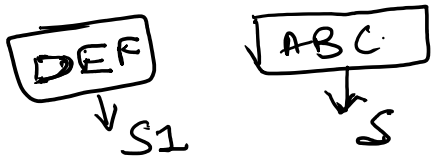
String s="ABC";
String s1="ABC";

s1="DEF";

In the above lines for s and s1 only one memory will be present like below



Later when s1="DEF" is encountered, instead of modifying the ABC to DEF,
JVM checks if DEF is present in String constant pool, if not present then it creates the memory for it and assign s1 to it and earlier s1 referent will be removed like below



String:

What is String in Java? String is a data type?
String is a Class in java and defined in java.lang package. It's not a primitive data type like int and long. String class represents character Strings.

What are different ways to create String Object?
We can create String object using new operator like any normal java class or we can use double quotes to create a String object. There are several constructors available in String class to get String from char array, byte array, StringBuffer and StringBuilder.

String str = new String("abc");
String str1 = "abc";
When we create a String using double quotes, JVM looks in the String pool to find if any other String is stored with the same value. If found, it just returns the reference to that String object else it creates a new String object with given value and stores it in the String pool.
When we use the new operator, JVM creates the String object but don't store it into the String Pool. We can use intern() method to store the String object into String pool or return the reference if there is already a String with equal value present in the pool.

All the wrapper classes are final that means immutable

The key benefits of keeping this class as immutable are caching, security, synchronization, and performance.

To Convert anything to String we use String.valueOf method
To convert String to integer we use Integer.parseInt
Similarly we use Long.parseLong to convert String to long.

All the wrapper classes have the parse method through which we can convert string to corresponding primitive value.

How to get Character from String.
We can use charAt() method and pass the position to get get the character at specific position in the String

We use toCharArray() to convert String to char array.

How to create user defined Immutable class.
Ex: Employee

1. **User defined class (Employee) should be final.**
2. **All the primitive variables in the class should be made as final and the initialization for these final variables will happen only in constructor.**
3. **If the user defined class is having the object reference (Department) then in order to achieve the immutability, we have to create the new object in Employee constructor and assign it to this variable. Or make private variables of department class as final**
4. **Disadvantage of String is it occupies memory every time. At the same time this is an advantage because if we are creating multiple strings object with same value then only once the memory will be allocated**
5. **Advantage, String or wrapper class will work as a best example for key in collection**

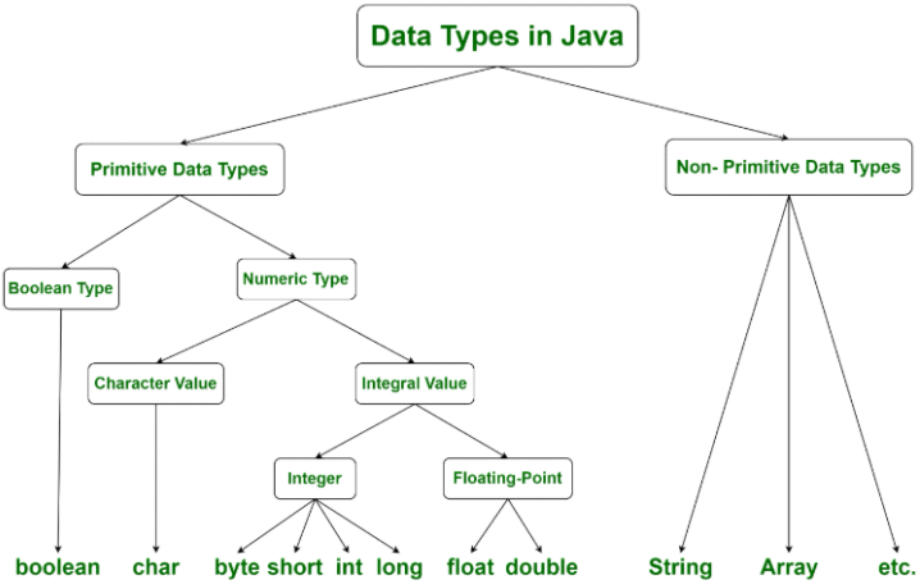
Work for the Day

1. **Write a program to reverse the given string**
   Using string class methods
   Without using string class methods
2. **How to find the length of string with string class method**
3. **How to find the length of string without using the string class methods**

No.	StringBuffer	StringBuilder
1)	StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.

Variables in Java

20 June 2020 14:45



Int, double, long, float -->Integer, Double, Float, Long
char c='Y';
String s="KARTHIK";

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'','\n',' β'	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

Different Methods

1. **Static method or static variable: static method can access only the static variables or another static method**
2. **Instance method: this can be accessed with object. Static method can access non static and static methods and**

variables

3. Private variable scope is till the class only. If it is declared inside method then scope till the method only
4. Inheritance is not possible for private
5. Inheritance is possible for default only if the classes are under same package
6. If it is under different package then we have to use protected or public

Abstract Class:
Class which have at least one abstract method
If a class is declared as abstract then there is no condition that it should have abstract method
We can't create object to abstract class because compiler don't how much memory to be allocated.

Interface
It has 100% abstract methods only. From 1.8 onwards default and static methods are introduced.
The variables declared in interface are final variable

We have to extend the class
And implement the interface

Adv of interface is -> multiple inheritance is possible- implement multiple interfaces

Abstract->multilevel inheritance

Constructors:
Constructor name should be same as class name
Difference between constructor and method is method has return type but constructor doesn’t have return type
Immediately after the object is created then constructor gets executed

Constructor is used to initialize the instance variable
Constructor and methods can be overloaded

Overloading:
Method name should be same and there should be change in the number of arguments or type of arguments.

Keywords in Java

10 August 2021 16:19

Keywords or Reserved words are the words in a language that are used for some internal process or represent some predefined actions. These words are therefore not allowed to use as a variable names or objects. Doing this will result into a compile time error.

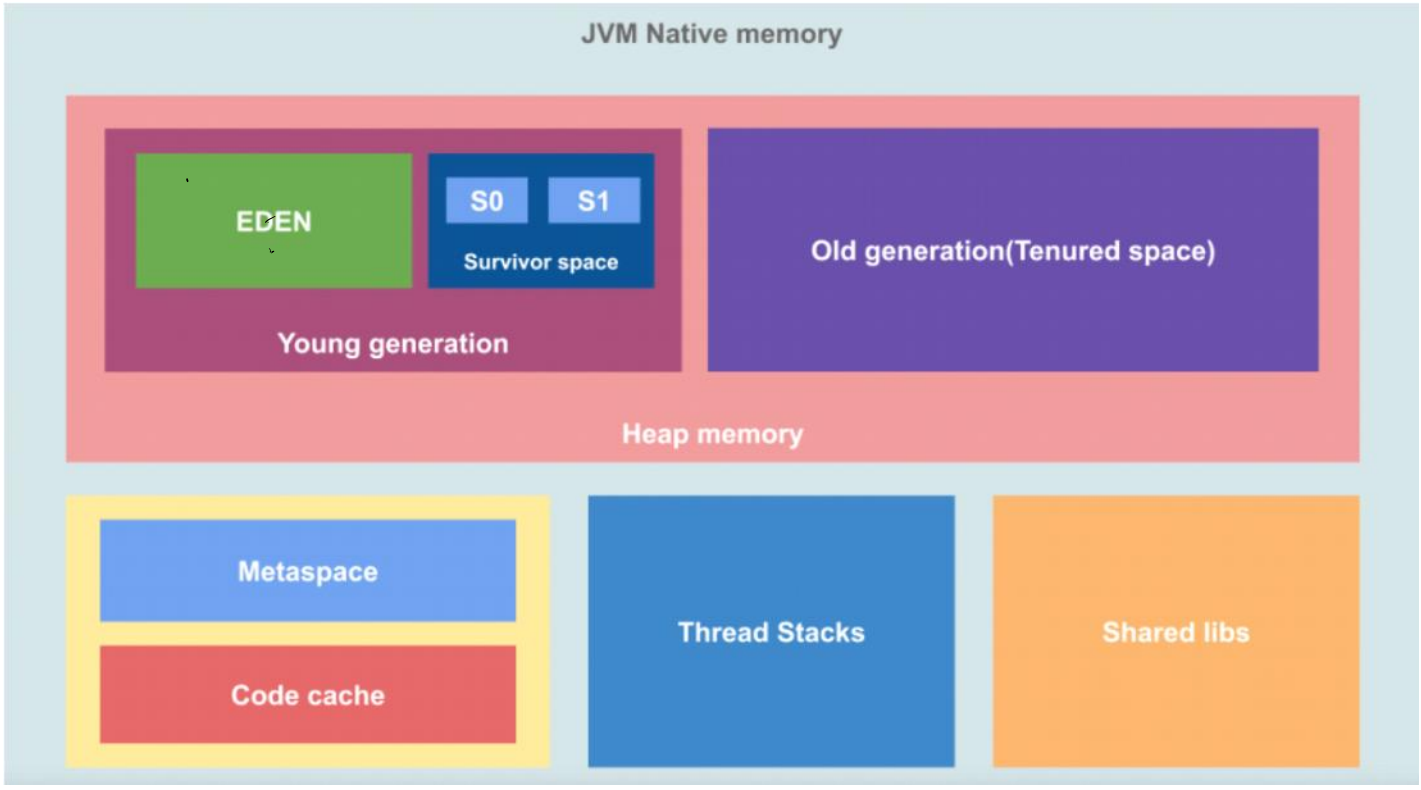
Java also contains a list of reserved words or keywords. These are:

abstract -Specifies that a class or method will be implemented later, in a subclass
assert -Assert describes a predicate (a true–false statement) placed in a Java program to indicate that the developer thinks that the predicate is always true at that place. If an assertion evaluates to false at run-time, an assertion failure results, which typically causes execution to abort.
boolean – A data type that can hold True and False values only
break – A control statement for breaking out of loops
byte – A data type that can hold 8-bit data values
case – Used in switch statements to mark blocks of text
catch – Catches exceptions generated by try statements
char – A data type that can hold unsigned 16-bit Unicode characters
class -Declares a new class
continue -Sends control back outside a loop
default -Specifies the default block of code in a switch statement
do -Starts a do-while loop
double – A data type that can hold 64-bit floating-point numbers
else – Indicates alternative branches in an if statement
enum – A Java keyword used to declare an enumerated type. Enumerations extend the base class.
extends -Indicates that a class is derived from another class or interface
final -Indicates that a variable holds a constant value or that a method will not be overridden
finally -Indicates a block of code in a try-catch structure that will always be executed
float -A data type that holds a 32-bit floating-point number
for -Used to start a for loop
if -Tests a true/false expression and branches accordingly
implements -Specifies that a class implements an interface
import -References other classes
instanceof -Indicates whether an object is an instance of a specific class or implements an interface
int – A data type that can hold a 32-bit signed integer

interface – Declares an interface
long – A data type that holds a 64-bit integer
native -Specifies that a method is implemented with native (platform-specific) code
new – Creates new objects
null -Indicates that a reference does not refer to anything
package – Declares a Java package
private -An access specifier indicating that a method or variable may be accessed only in the class it's declared in
protected – An access specifier indicating that a method or variable may only be accessed in the class it's declared in (or a subclass of the class it's declared in or other classes in the same package)
public – An access specifier used for classes, interfaces, methods, and variables indicating that an item is accessible throughout the application (or where the class that defines it is accessible)
return -Sends control and possibly a return value back from a called method
short – A data type that can hold a 16-bit integer
static -Indicates that a variable or method is a class method (rather than being limited to one particular object)
strictfp – A Java keyword used to restrict the precision and rounding of floating point calculations to ensure portability.
super – Refers to a class's base class (used in a method or class constructor)
switch -A statement that executes code based on a test value
synchronized -Specifies critical sections or methods in multithreaded code
this -Refers to the current object in a method or constructor
throw – Creates an exception
throws -Indicates what exceptions may be thrown by a method
transient -Specifies that a variable is not part of an object's persistent state
try -Starts a block of code that will be tested for exceptions
void -Specifies that a method does not have a return value
volatile -Indicates that a variable may change asynchronously
while -Starts a while loop

JVM

20 June 2020 14:48



Perm Gen Space:-xxPerGenSize:
What is JVM ?
JVM architecture
Different memories available in JVM - young generation, old generation,
Young generation- Eden and s0 and s1
Old generation
Meta space generation from 1.8 before this per Perm gen space
Java stacks
PC registers
Minor GC and Major GC
How to set the memory parameters.

-Xms for minimum heap
-Xmx for maximum heap ->-xmx1500m

Young Generation Maximum: -XX:MaxNewSize=-xmx/2
-XX:NewSize:
->Old generationSize=>-xmx/2

GC
What is garbage collector
How the object will be garbage collected
How many ways an object will be garbage collected
Algorithm - Mark, Sweep and Compact
3 programs example on finalize method
How many times finalize method gets called
Types of GC

1. **JVM over view**
2. **Different Types of memories in JVM**
3. **GC, types of GC**
4. **Programs-**

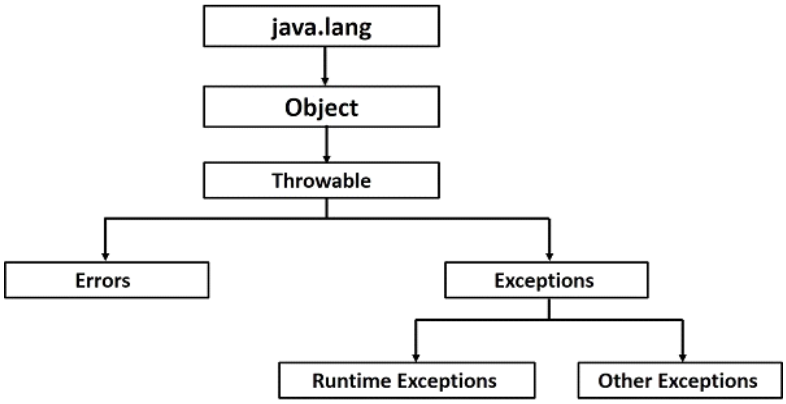
Exceptions

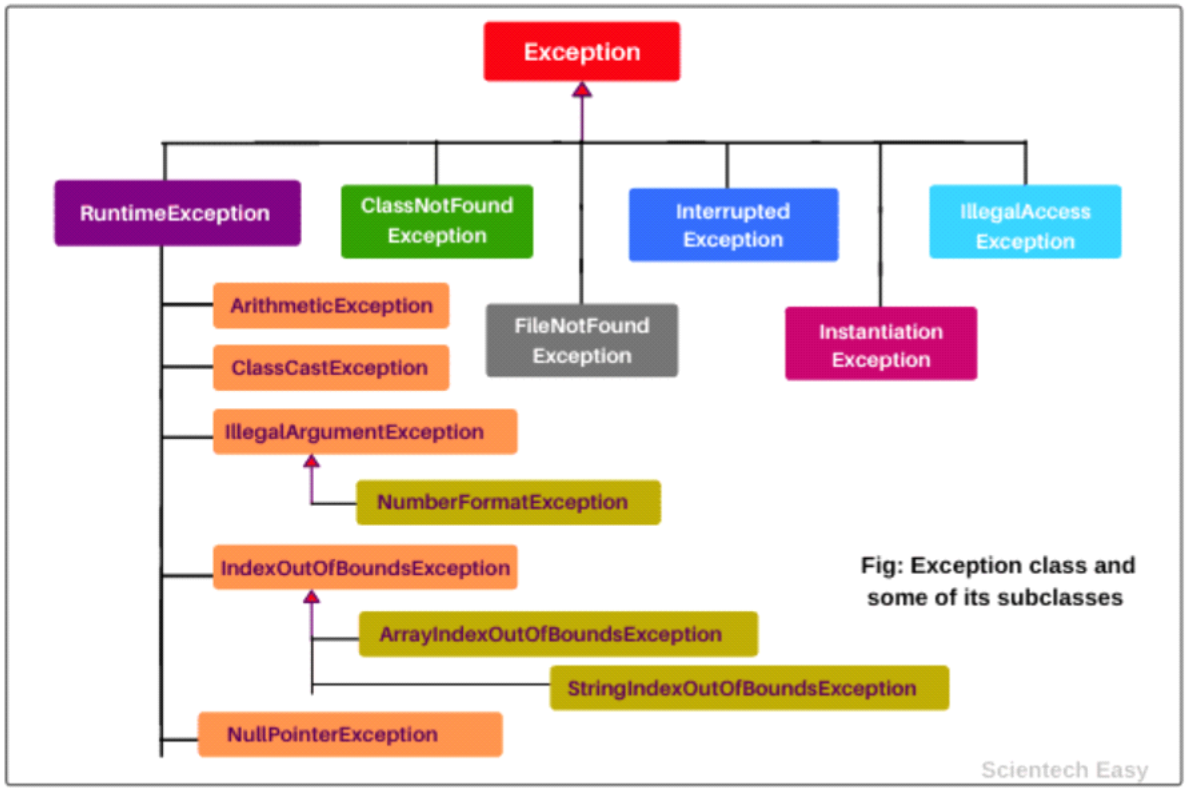
22 June 2020 08:55

1. **Exception : it is something which is stopping the normal flow of execution.**
   Error vs Exception
2. **Error: An Error indicates serious problem that a reasonable application should not try to catch.**
3. **Exception: Exception indicates conditions that a reasonable application might try to catch.**
4. **How JVM handle an Exception?**

   Default Exception Handling : Whenever inside a method, if an exception has occurred, the method creates an Object known as Exception Object and hands it off to the run-time system(JVM). The exception object contains name and description of the exception, and current state of the program where exception has occurred. Creating the Exception Object and handling it to the run-time system is called throwing an Exception. There might be the list of the methods that had been called to get to the method where exception was occurred. This ordered list of the methods is called Call Stack. Now the following procedure will happen.

   The run-time system searches the call stack to find the method that contains block of code that can handle the occurred exception. The block of the code is called Exception handler.
   The run-time system starts searching from the method in which exception occurred, proceeds through call stack in the reverse order in which methods were called.
   If it finds appropriate handler then it passes the occurred exception to it. Appropriate handler means the type of the exception object thrown matches the type of the exception object it can handle.
   If run-time system searches all the methods on call stack and couldn't have found the appropriate handler then run-time system handover the Exception Object to default exception handler , which is part of run-time system. This handler prints the exception information in the following format and terminates program abnormally





Type of Exception

Checked Exceptions

Checked are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

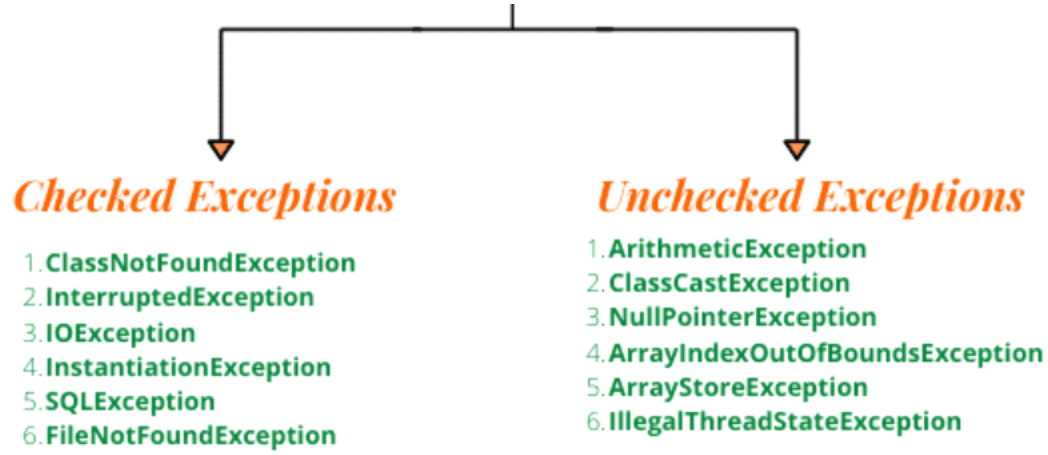
Unchecked Exceptions:

2) Unchecked are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.
In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked

If a client can reasonably be expected to recover from an exception, make it a checked exception. If

a client cannot do anything to recover from the exception, make it an unchecked exception

Frequently used exceptions



1. **Throwable is the super class for all the classes**
2. **In order to handle the exception we use try and catch block**
3. **One try can have multiple catch blocks and one finally block**
4. **Try can also have only finally without catch**
5. **There should not any statement between try, catch, finally**
6. **Finally gets executed every time (with exception or without exception)**
7. **While catching the exception we have to catch subclass first and super class next**

   Catch(ArithmeticException){
   }
   Catch(Exception){
   }

   Vice versa is not possible
8. **Throws is used for checked exceptions**

9. **Throw is used throwing the exception**
10. **Throw accepts new object like throw new Exception**
11. **How JVM handles the error. Create multiple methods and call those methods from one another**
12. **Design pattern that is used by JVM for exception handling is chain of responsibility design pattern**
13. **In try block if once exception occurred rest of the lines will not be executed, immediately control goes to catch or if catch is not present then goes to finally**
14. **How to create user defined exception**
15. **Create the class and extend it from Exception class and create the constructor with no arg and single argument.**
16. **If your project just to know whether are you using user defined exceptions, search for extends Exception word in your project**

List of Checked Exceptions in Java1. ClassNotFoundException: The ClassNotFoundException is a kind of checked exception that is thrown when we attempt to use a class that does not exist.

Checked exceptions are those exceptions that are checked by the Java compiler itself.

FileNotFoundException: The FileNotFoundException is a checked exception that is thrown when we attempt to access a non-existing file.

InterruptedException: InterruptedException is a checked exception that is thrown when a thread is in sleeping or waiting state and another thread attempt to interrupt it.

InstantiationException: This exception is also a checked exception that is thrown when we try to create an object of abstract class or interface. That is, InstantiationException exception occurs when an abstract class or interface is instantiated.

5. IllegalAccessException: The IllegalAccessException is a checked exception and it is thrown when a method is called in another method or class but the calling method or class does not have permission to access that method.

6. CloneNotSupportedException: This checked exception is thrown when we try to clone an object without implementing the cloneable interface.

7. NoSuchFieldException: This is a checked exception that is thrown when an unknown variable is used in a program.

8. NoSuchMethodException: This checked exception is thrown when the undefined method is used in a program.

Hope that this tutorial has covered almost all the

basic points related to the exception hierarchy in java. I hope that you will have understood the basic points of Throwable class and its subclasses: Exception and Error.

5.

Threads and Executors

23 June 2020 10:38

1. **What is Thread ? It is independent flow of execution with in a program**
2. **How many ways we can create thread object**
   3 ways(extending thread class, implementing runnable interface, implementing callable interface)
3. **Thread states (New, Running, runnable, dead)**
   New : This is the state when the thread object is created but start method is not invoked on it
   Runnable: after invoking the start method thread will go to runnable pool
   Running: from the runnable pool, JVM or scheduler will pick the thread based on the priority and may be some other factors and start the execution that means start executing the run method.
   Dead: after the completion of run method thread is considered as Dead
4. **When the thread flow of execution starts ? Immediately after calling the start method**
5. **Why to override run method (because Runnable is interface and it has abstract method called run. Hence when we implement the runnable interface we have to override run method)**
6. **Thread class internally implements runnable interface**
7. **What is the difference between start and run method**
   Start will create the thread execution but run will be treated as normal method execution
8. **What is synchronization**
9. **How synchronization is implemented**
10. **How the lock will be achieved**
11. **Try to create the thread object by implementing runnable interface**
12. **If we call start method on a thread which has completed the execution of run method what happens ? Illegal thread state exception will be thrown**

Refer to the programs shared on Telegram for practical implementation.

What is object Lock and what is class Lock ?
Object Lock: It is the mechanism when we want to synchronize the non-static method or non-static code block such that only one thread will be able to execute the code block on the given instance of the class

Assume on a bank account if you are performing different operations like ATM Withdrawal and Net banking deposit.

Here Karthik Account is Object
ATM is one Thread
Net Banking is another Thread
Methods are Deposit and Withdrawal

Like If ATM thread is working on Withdrawal method on Karthik Account then Lock will be acquired on Karthik Account So in order to make the data consistency or thread safe any other thread like Net banking should not be able to access any other methods like Deposit or Send Money to other threads on the same Karthik bank account. This is called Object lock

In object lock if one thread acquires the lock on object then any other thread cannot access any other synchronized methods but they can access non synchronized methods

So the Synchronized methods in object lock are instance methods

If the threads are working on different object of a class then lock acquires on different objects.
Whereas if we want to make the class level data means static methods thread safe then we should go for class lock

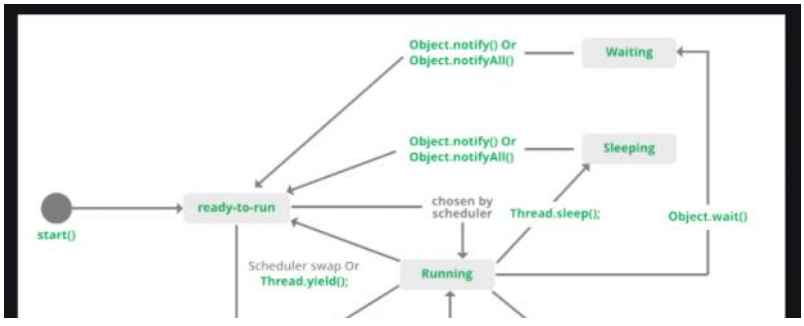
Difference Between Sleep and Wait

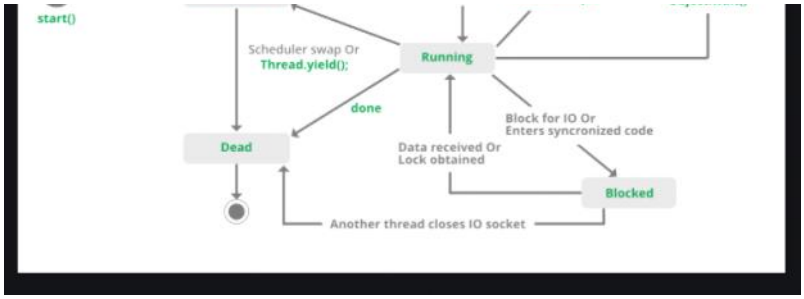
Sleep	Wait

This is Static method. Present in Thread class	This is instance method. Present in Object class
Sleep accepts milli seconds. The Thread will sleep for the specified time and after that starts the execution if system resource is not busy	Thread will be in waiting status unless until notify or notify all is called.
Sleep doesn't release the lock	Wait will release the lock
Sleep can be used with or with out synchronized block	Wait can be used only with synchronized block. If used in non synchronized block then it throws illegal monitor state exception
Thread is used to pause the thread execution for some time	Wait and notify or notify all are used for inter thread communication

Yield:

A yield() method is a static method of Thread class and it can stop the currently executing thread and will give a chance to other waiting threads of the same priority. If in case there are no waiting threads or if all the waiting threads have low priority then the same thread will continue its execution. The advantage of yield() method is to get a chance to execute other waiting threads so if our current thread takes more time to execute and allocate processor to other threads





Class Lock

==========

In Class lock if the thread acquires lock on a class reference then any other thread cannot access the static synchronized methods on the same class instance.

Thread Pool: It is mainly for re purpose of threads which are already created. Because on normal thread if you call start after completion on run method system throws illegal thread state exception

Callable

==========

If the thread needs to return something then there is no way when we implement runnable or override run method.
So in order to facilitate this we go for callable interface
It has the call method which returns some object whenever a thread is executing.

Thread Pools:
Executors.NewFixedThreadPool or cachedthreadpool or single thread pool
NewFixedThreadPool -- number of threads created will be fixed. Internally this thread pool uses queue concept to store the task. The queue that this thread pool uses is LinkedBlockingQueue
cachedthreadpool =it creates one thread and then it creates new thread as on when required.Internally this thread pool uses queue concept to store the task. The queue that this thread pool uses is SynchronousQueue queue.
single thread pool- only one thread. Since it is only one thread it executes sequentially.

To call the runnable interface and callable interface we can use submit method of executor service interface.

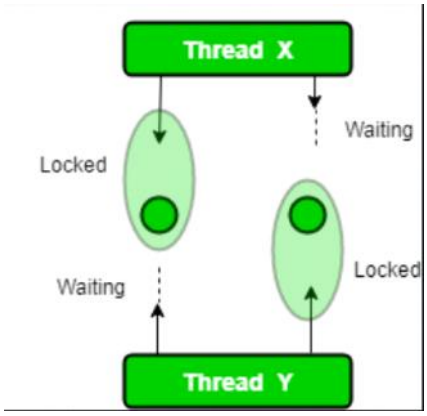
Difference between Execute and Submit method
Execute will take the runnable instance
Where as Submit will accept the runnable or callable instance.

Multi threaded programming

1. **Read data from DB**
2. **Create class which implements callable or runnable**
3. **Override the call or run methods**
4. **Write the logic that you want to perform for each thread in call or run method**
5. **Create a single arg constructor and the argument should be object on which you want to perform logic**
6. **Create the Thread Pool using Executors Class and mention the number of threads**
7. **From the main program, loop through the data that is retrived from DB and create the object for logic class of line number (2,3,4,5) and pass the required object**
8. **Submit callable statement or call the Submit method on Executor service.**

   Calculate the salary of all the employees in the organization for the month of August.

Dead Lock



Difference between Lock and Synchronization

Synchronization					Lock
1.No fair and Unfair					Fair and Unfair is present
We cannot acquire the lock across methods					We can acquire the lock across the methods
Try lock concept is not there					Try lock is present
Schedulr or JVM will take care of it no need of manual code for lock or un lock					Developer responsibility to provide unlock for each lock Better to write unlock in finally

Read-Write Lock-> multiple read operation threads can work at a time and once the write lock comes then all the read locks will go on hold.
Bus booking example-> multiple request or user searching for a bus for same route and same date then all the request or people will be able to see the same seats
But when one person tries to book the seat then all other request will be on hold.

Whenever the lock is called held count will be incremented and when unlock is called held count will be decremented

When the held count is zero that means there is no lock

```
if (lock.tryLock()) {
216:     *         try {
217:     *             // manipulate protected state
218:     *         } finally {
219:     *             lock.unlock();
220:     *         }
```

```
221:     *     } else {
222:     *         // perform alternative actions
223:     *     }
```

Serialization

25 June 2020 09:38

1. **Serialization is the process of converting the object into series of byte and send it over the network**
2. **In order to achieve the serialization we have to make use of serializable interface**
3. **Serializable is a marker interface**
4. **Marker interface is the interface in which there are no methods.**
5. **The work that marker interface to do is mentioned in JVM.**
6. **Generally in project pogo class or dto class or a class with setter and getters will be implementing the serializable interface**
7. **writeObject on object output stream performs the serialization**
8. **readObject on input stream performs the de serialization**
9. **Serialization is possible with Serial version id which we can mention manually or JVM computes it based on the class structure and class meta data using the SHA (Secure hash algorithm)**
10. **When ever we are doing the serialization the class that implements the serializable interface should have the default (no org) constructor if not deserialization is not found and class not found exception is thrown**
11. **Default constructor should be present because during the de serialization JVM does the reflection so in reflection default constructor will be called**
12. **Hence during de serialization we add throws class not found exception which is checked exception**
13. **Serialization is possible for object and not class**
14. **If we want not to serialize some part of the object or some variables then we have to mark that variable as Transient. Transient is the key word in java. Any variable which is marked as transient, then that variable will not be serialized that means after de serialization the transient variable value is default value of data type.**
15. **If we mark String variable as transient then after de serialization that string transient variable value is null**
16. **If we mark int variable as transient then after de serialization that int transient variable value is 0 (because default value of int data type is 0).**
17. **Since Serialization belongs to object, Static variables are not serialized, even if you declare a static variable and if static variable is not initialized then and try looking at the static variable value after deserialization then the value is null or default value of the data type.**
18. **If the static variable is initialized with the value like below**
    Static int i=10, even if you try modifying while serializing the object after de serialization you will get the value that is initialized at class level, that means serialization is not happening
19. **If we want to perform some part of the code as serialization then we go for Externizable**
20. **It has 2 abstarct methods**
21. **ReadExternal-> gets called from readObject**
22. **WriteExternal ->gets called from Write Object**
23. **Externizable interface extends serializable**
24. **In deserialization process, it is required that all the parent classes of instance should be Serializable; and if any super class in hierarchy is not Serializable then it must have a**

default constructor.

Singleton

28 June 2020 10:59

When we say single ton there should only be 1 object
How to know 1 object we have to compare the hashcode when ever you try to do

1. **Make the constructor as private-> we will not be able to create the object from outside of the class**
2. **Eager initialization. Creating the object at class level is called eager initialization. Drawback of this is even if we don't want the single ton object also during the class load object gets created**
3. **To solve this we for object creation with getInstance method**
4. **Why getInstance method is static. By making the constructor as private you will not be able to create the object from outside of the class but with getInstance we can get the object, inorder to access this method with class name we should make it static**
5. **How many times the singleton can be broken**
   ->Clonnable ->in clone method return the exception
   ->Serializable ->override the readResolve method and return the single ton instance
   ->creating object from outside of the class ->make the singleton class construtor as private
   ->Multi JVM-> If multiple people are executin the singleton in multiple JVM then it may give multiple instances. To resolve this what we do is we should create one table and store class name and the

status
At the beginning of the singleton getInstance is called make the status as STARTED and then when even next time if we start singleton class in other JBVM check if there is any record in DB with Started then don’t allow and throw exception. This way we can prevent multiple instances
->Reflection->

Arrays

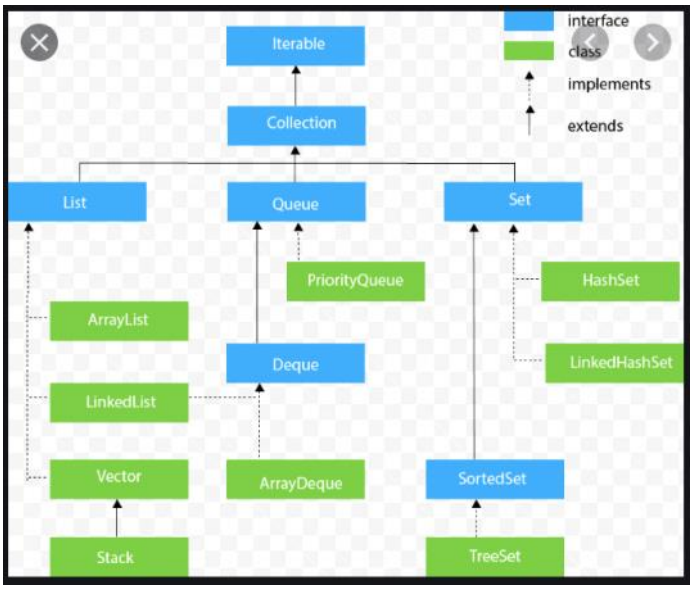
04 September 2021 07:48

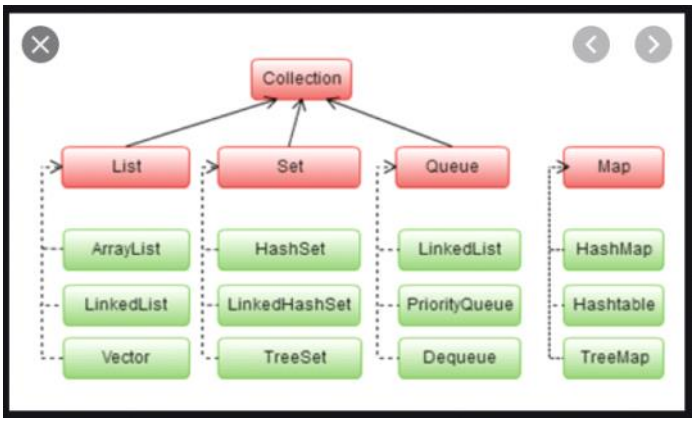
- **Arrays are fixed in size i.e. once we created an array with some size then there is no increasing or decreasing its size based on the requirements.**
- **Arrays can hold only homogeneous data elements.**
- **Array concept is not implemented based on some standard data structure. Hence ready-made methods are not available for the requirement.**
- **Array Size cannot be increased dynamically. This is the Drawback of Array.**
- **Add, remove, Traverse over the array should be done manually.**
- **There are no predefined methods of an array.**
- **Array Size can only be integer.**

- **These are some drawbacks of having the array and hence collection framework is introduced.**

Collections

01 July 2020 09:02





Collection framework is used to store the group of objects
ArrayList extends AbstractList which implements List interface
If you print any collection it will print values present in the collection because it has overridden the toString() method
Where as Array cannot print

Array List:
==========

- **Array List accepts Duplicates**
- **Defauly capacity of AL is 10**
- **What is the data structure used by ArrayList is Array (Object Array)**
- **There is a variable present in Arraylist class and it creates the Array list with this capacity when we create the object or when we call the constructor**
- **Array List allows duplicates**
- **Array list is not synchronized.**
- **Array List is Ordered**

What happens inside add method ?
AL internally uses the Object array (Object[])
First it checks for the capacity
If is reached then it will increment the capacity by 50% on old capacity
Later it does the copy of the elements

If the capacity is not reached then directly it will add the elements

What internally happens when remove(2) is called

The argument that the remove method accepts is index.
First AL checks whether the index is present or not, if not present then throws Array index out of bound exception if not get the element present in the index and then it removes from that position and re arrange the AL and returns the removed element

How to convert Array to AL
Using ->iteration or Arrays.asList();
After performing the Arrays.asList we cannot add any new element because it is not creating the AL.
To solve this
List l= new ArrayList(Arrays.asList());
This statement will call the single arguemnet constructor of AL which accepts collection object

Vector
==========
In AL increase capacity by 50 % of the existing capacity where Vector will increment the capacity by 100%
Vector is synchronized and AL is not synchronized.
Vector is also Ordered and allows duplicates

Custom ArrayList
==========

1. **Create the class**
2. **Have the variables for default value, index**
3. **Define the array, because AL internally uses Array List**
4. **Create the Default constructor and initialize the array**
5. **Create Add method**
6. **In Add method check the capacity by comparing index and the default capacity**
7. **If index reached to default capacity then increment the size by 50% and do array copy**
8. **Else Add the element and increment the index.**
9. **Create another class and instead of using the AL use custom arraylist**
10. **Try to work on the remove functionality.**

HashMap
========

How does hashmap works internally

1. **Default hashmap is of size 16 and the loading factor is 0.75**
2. **What is loading factor ?**

   The Load factor is a measure that decides when to increase the HashMap capacity to maintain the get() and put() operation complexity of O(1). The default load factor of HashMap is 0.75f (75% of the map size)

   We insert the first element, the current load factor will be 1/16 = 0.0625. Check is 0.0625 > 0.75 ? The answer is No, therefore we don't increase

the capacity.

Next we insert the second element, the current load factor will be 2/16 = 0.125. Check is 0.125 > 0.75 ? The answer is No, therefore we don't increase the capacity.

Similarly, for 3rd element, load factor = 3/16 = 0.1875 is not greater than 0.75, No change in the capacity.

4th element, load factor = 4/16 = 0.25 is not greater than 0.75, No change in the capacity.

5th element, load factor = 5/16 = 0.3125 is not greater than 0.75, No change in the capacity.

6th element, load factor = 6/16 = 0.375 is not greater than 0.75, No change in the capacity.

7th element, load factor = 7/16 = 0.4375 is not greater than 0.75, No change in the capacity.

8th element, load factor = 8/16 = 0.5 is not greater than 0.75, No change in the capacity.

9th element, load factor = 9/16 = 0.5625 is not greater than 0.75, No change in the capacity.

10th element, load factor = 10/16 = 0.625 is not greater than 0.75, No change in the capacity.

11th element, load factor = 11/16 = 0.6875 is not greater than 0.75, No change in the capacity.

12th element, load factor = 12/16 = 0.75 is equal to 0.75, still No change in the capacity.

13th element, load factor = 13/16 = 0.8125 is greater than 0.75, at the insertion of the 13th element we double the capacity.

Now the capacity is 32

3. **Whenever we tried to add element to hashmap that is map.put**
   Internal implementation of PUT method
4. **First find out hash of the key**
5. **Then divide it by the 16 buckets which gives the index**
6. **Now at this index store the element in the form of Node**
7. **Node will contain hash, key , value and address of next node**
8. **If it happens that index of 2 keys are same then it is referred as collision**
9. **In case collision hash map follows linked list approach**
10. **At index first node will contain the address next node**
11. **What happens when you try to retrieve the element-> first system finds the hash of the key then find out index by dividing the hash /16**
    Now it compares the hash at the index. If there are multiple nodes present then it checks hash of each and every node until it is matched
    Once it is matched it checks the key. If key also matches the retrives the value and returns it
12. **If there are multiple nodes a given index this process may introduce performance issue**
13. **To get rid of this we go for Tree in hash map from 1.8**
14. **Difference between hashmap and linked hashmap is**
    Linked hashmap is ordered -> it will give the element in the insertion order
    Whereas hash map doesn't guarantee the order
15. **Try putting user defined employee object and see if those are equal.**

Set

==========

1. **Set internally uses the Map structure**

2. **Where the map gets initialized in (hashset constructor)**
3. **Default capacity is 16 and loading factor 0.75**
4. **When the set is using the map internally, then what is the value for all the keys (OBJECT)--you can find final Object PRESENT=new Object();**
5. **Find why set doesn't allow duplicates, because it follows map structure and map doesn't allow duplicates**
6. **Question is why map doesn't allow duplicate keys**
7. **Hashset or hashmap doesn’t guarantee the order**
8. **Where linked hashmap and linked hash set guarantee the insertion order why because it uses double linked list approach and each node contain the address of next hence insertion order is preserved.**

Concurrent Hash Map

==========

1. **Map, Set and List whenever we try to iterate and trying add the element then these collection will throw the concurrent modification exception.**

```
l.add(10);
l.add(25);
Iterator it=l.iterator();
While(it.hasNext()){
Sysout(it.next());
l.add(45)
}
```

2. **Here in the above program, whenever we add the element to collection internally java uses the modcount variable and it gets incremented whenever we try to perform the add or remove operation so when we try to call next() method (look at next() method in ArrayList or any other collection) it checks for the earlier mod count and expected mod count, if both are not matching it throws the concurrent modification exception**

To resolve this we go for concurrent Hash Map

1. **In Concurrent Hash Map lock acquires at segment level and it uses the Reentrant lock**
2. **While calling next(), it iterates over new collection not on the original collection hence concurrent modification exception is not thrown**

CopyOnWriteArrayList

==========

1. **In order to have the concurrency in the list we go for Copy on write ArrayList**
2. **It has the lock in add and remove etc**
3. **It uses the reentrant lock**
4. **It doesn't throw concurrent modiofication exception because the iteration will happen on new array and not on the original array**
5. **Same currencny in set will be achieved with CopyOnWriteSet.**

Java 8 Notes

07 July 2020 10:36

1. **Lamda enables the functional programming. We can send functionality as method argument or code as data**
2. **Interface in Java8 enables the default and static methods**
3. **What is the advantage of having Default methods in java 8?**
   It provides the backward compatibility
4. **How to call default method of interface**
   InterfaceName.super.MethodName()
   Test.super.display()
5. **Can we override the default method of interface ? Yes we can override**
6. **Java 8 enabled to have static methods so can this static method be overridden ?**
   No because static methods are depending class level. In the class if you try add a method of static with same name as static method that is present in interface then it will be treated as redefining not overriding
7. **How to call the interface static method ?**
   InterfaceName.methodName()
   Test.Addition();
8. **How it is different from calling the static method of a class**
   ClassName.methodName for class
   InterfaceName.methodName for interface
9. **Advantage of having static methods in interface ?**
   To perform utility conditions like null check etc
10. **Every lamda corresponds to Interface and that interface should have only one abstract method**
11. **If there are more abstract methods then Java 8 compiler doesn't know which abstract method it should refer.**

Hence we should allow only one abstract method per interface. In order to facilitate this feature the corresponding interface should be marked as @FunctionalInterface.

12. **Java 8 interface can have any number of abstract, default and static methods but If we try to add the abstract method in a functional interface then compiler will throw the error.**
13. **Creating the interface for each lamda is overhead, To resolve this Java 8 has provided the package called Functions under java.util.**
14. **Different interfaces present in Java 8 are**
    Predicate
    Consumer
    Supplier
    Function
15. **With this predefined interfaces we no need to create the new interfaces and we can make use of this during programming.**

They are merely a way of organizing utility methods in more convenient fashion. For example, a common use of static methods in an interface is for static factory methods

Stream

==========

A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result

Terminal operator ->Collect or for each
Intermediate operator ->filter, distinct

Finding or printing employee names whose salary is >500
Finding or printing employee names whose salary is >500

and whose name starts with "k"
Finding the duplicates
Sorting -> try this out

Cloning

08 July 2020 10:40

- **Cloning means creating the new object in an easy way.**
- **If I want to create the new copy of object then I can create new object using cloning or create the new object like (A a= new A()) and then copy all the elements to the new objects manually. This is the tedious process so to overcome this we go for cloning**

How can we do the clone.

- **Clone is the method present in the Object class so all other classes can directly access this clone method using it's object**
  (Object class is the super class for all the classes)
- **Clone is the Protected method.**
- **So without doing anything just try below**
  Employee e= new Employee()
  e.clone()
- **Run the program with the above statements and see the output.**

```
Exception in thread "main" java.lang.CloneNotSupportedException:
com.ashokit.fullstack.cloning.Test
        at java.lang.Object.clone(Native Method)
        at com.ashokit.fullstack.cloning.Test.main(Test.java:7)
```

- **Output is error because in order to perform cloning in Java, we have to use Cloneable interface**
- **Whichever the object that we want to clone, corresponding class should implement the cloneable interface.**
- **Cloneable interface doesn't have any methods(abstract and nonabstract).**
- **An interface which doesn't have the abstract and nonabstract methods is called as Marker interface.**
- **After implementing the cloneable interface in the above**

example for the employee class please try again.

- **It will call the object class clone method and provides the new object**
- **To know whether the original object and cloned object are different then check the hash code and equals**
- **Every class that implements cloneable interface should also override the Object class clone method should call super.clone() to obtain the cloned object reference.**
- **The above behavior is called as Shallow cloning**
- **By default Java follows Shallow cloning**
- **We have to override clone method because,**
  For example, I am trying to clone Test class object and in the same class I have written the main method then without overriding clone method also I can get the cloned object
  But if Test is one class and Demo is another class, so in Demo we have the main method in this main method I am creating the Test object then on the test object I cannot see the clone method, reason for this is clone method is protected and other package sub class only we can access
  Object class is present in Java.lang package
  And my class is present in separate package so if I create class then it becomes the sub class to object class so in order to see the method in test class we should override.

The class must also implement java.lang.Cloneable interface whose object clone we want to create otherwise it will throw CloneNotSupportedException when clone method is called on that class's object.

Syntax:

```
protected Object clone() throws
CloneNotSupportedException
```

Shallow Cloning:

1. **By default Java follows Shallow cloning.**
2. **In Shallow cloning, clone will create the separate object.**
3. **If the object has primitive values then if you change those values in one object then change will not reflect in another object.**
4. **If the Object is referring to other object internally like**
   Class Employee{
   Department d;
   }
   Then ,Whenever any change happen to department object in cloned copy then the same gets reflected in main (other) object, reason for this is both objects will point to the same location

Deep Cloning:

Whenever change happens in one object then the same will not be reflected in other object, reason for this is both objects will point to the different location.

To do this we have to implement the clonable interface and call the clone method of Object class

In the clone method instead of calling return super.clone() method create the separate object and store the data like below

```
Employee e= new Employee();
Department d1= new Department()
d.depId=e.d.depId
e.empId=empId;
e.d=d1;
Return e;
```

Comparator and Comparable

06 July 2020 21:28

Comparable	Comparator
Present in Java.lang package	Present in Java.util package
We should override compareTo() method	We should override compare Method

Example is : all wrapper class implement Comparable and this will have the natural sorting order (Ex: Integer wrapper class implements Comparable and provides the implementation for compareTo and it has the logic for increasing order)
l.add(4,"ABC");
l.add(0,"Sg");
l.add(1,"Str");
 	
Collections.Sort(l);
It will print output in increasing order
	If at all you need to implement the different order some logic then you should implement the Comparator.
	l.add(4,"ABC");
	l.add(0,"Sg");
	l.add(1,"Str");
	
	Collections.Sort(l,new ComparatorExample());

Overriding Equals and Hash code

06 July 2020 21:24

Why to override hashcode and equals method

1. **By default object class hash code and equals will be called**
2. **Equals method of Object class uses the shallow comparison that means it compares the memory location rather than values**
3. **If we override, equals and hash code then the corresponding methods gets called and perform the operation.**
   For Example:
   Map<Customer,Integer> m= new HashMap<Customer,Integer>();
   Customer c=new Customer("Karthik",456); //first is name and second parameter is id
   Customer c1=new Customer("Karthik",456);
   m.put(c,789);
   m.put(c1,789);
   System.out.println(m);
   Now the output or the map contains 2 elements because internally hash map calls the equals and hash code method of Object class. Since 'C' and 'C1' are different objects, memory gets allocated at different positions in JVM and hash code method return different hash codes for each key and equals also doesn't match because of shallow comparison hence 2 elements will be added to hash map.

   But if the interviewer is asking for the requirement like since both the customer name and id are same then we should not allow to add to hashmap.

In order to achieve this requirement, we must override the equals and hashcode method and
In hash code method you can return, hash as id like below

```
@Override
    public int hashCode() {

        return this.id;
    }

@Override
    public boolean equals(Object obj) {
        Customer c=(Customer)obj;

        if(this.getName().equalsIgnoreCase(c.getName())
        ) {
            return true;
        }else {
            return false;
        }
    }
```

Now hashcode returns same number that is id (since id is same for both objects)
And then we should check once the hashcode is same for both objects is the key same so for that we override the equals method and now in this user defined equals method we are comparing the name. since the name is same for both objects it returns true.
Equals and hashcode both are returning true hence c key gets overridden. Output will be

{Customer@7b=567}

Collections.sort()

06 July 2020 21:35

1. **Collections.sort() method accepts the list as a parameter. Below is the signature of sort method.**
2. **public static <T extends Comparable<? super T>> void sort(List<T> list)**

From the above signature we can understand that sort is static method and accepts the List parameter it can be ArrayList or Linked List etc.

And List is of type 'T' and 'T' can be Integer or String or custom class. And that T should implement the Comparable (T extends Comparable<? super T)
Hence when we call collections.sort on List of Integer, it automatically calls the compareTo method of Integer class.
Hence List<Integer> l = new ArrayList<Integer>();
l.add(23);
l.add(01);
l.add(24);

Collections.sort(l)--> here l is list and it is of type Integer so when we call sort method on this list internally Integer class compareTo gets called and natural sorting order will be performed.

If at all you want to implement the user defined comparator follow the below steps

1. **Create the class and implement the comparator interface**
2. **Override the compare method**
3. **And write the logic for compare method**
4. **And call collections.sort like below**
   Collections.sort(l, new ComparatorExample()); -> This automatically calls the user defined class compare method of ComparatorExample class does the operation.

Core Java Interview Questions

1. How to compile the Java file ?
2. How to run the java file ?
3. How to pass run time arguments while running ?
4. What happens when the java code is compiled ?
5. Can we compile the java class without writing any code ?
6. Can I compile the java program without main method ?
7. Can I run the Java program without main method ?
8. Can we interchange the static and public in main method ?
9. What does value of arg array if we don't pass any value while running the program ?
10. How to pass run time arguments to the program ?
11. What are the OOP principles ?
12. Is the java platform independent or dependent ?
13. What is the purpose of constructor ?
14. Can a constructor be overload ?
15. Can a constructor be overridden ? If not why ?
16. What is the difference between constructor and method ?
17. What does the interface contain(method and variables) before java 8 ?
18. Can I compile the interface ?
19. What is the hierarchy of constructor ?
20. What is the output of the below program ?

```
class A {
	A(){
		System.out.println("A class No Arg constructor");
	}
	A(int i){
		System.out.println("A class single arg constructor");
	}
}
class B extends A{
	B(){
		System.out.println("B class No Arg constructor");
	}
	B(int i){

		System.out.println("B class single arg constructor");
	}
```

```
        public static void main(String[] args) {
                A a= new A();
                A a1= new A(10);
                B b= new B();
                B b1= new B(15);
                A a2= new B();
                A a3= new B(20);
        }
}
```

21. **What is the default super keyword present in child class ?**
22. **What are the difference between abstract class and interface ?**
23. **Can a main method be over loaded ?**
24. **Can a main method be over ridden ?**
25. **A a= new B () -> what kind of methods can be accessed ? (List l = new ArrayList())**
26. **How the constructor gets executed when there is a instance block is present ?**
27. **Why we cannot create the object to abstract class ?**
28. **What is the difference between abstraction and encapsulation ?**
29. **What are the differences between over loading and over riding**
30. **What is the difference between default and protected access modifiers ?**
31. **What is blank final variable ?**
32. **What kind of methods and variables present in interface ?**
33. **What happens if I make the method in interface as final or private ?**
34. **What happens when I make the constructor of a class as Private ?**
35. **Can the private method or variable be inherited and if not why ?**
36. **Why Java is called platform in dependent and why JVM is platform dependent ?**
37. **How does JVM will handle the exception ?**
38. **What is the design pattern used by JVM to handle the exception ?**
39. **What is Thread?**

40. How many ways we can create the thread Object ?
41. What are the different states in the Thread ?
42. What happens when we call start on dead thread ? What exception ?
43. Can we call start method on a thread twice ?
44. Where the thread flow of execution start ?
45. How to set priority of a thread ?
46. How can we set a name to Thread ?
47. What is join method of a thread ?
48. What is the difference between sleep and wait ?
49. What is inter thread communication ?
50. Why wait notify and notify all are present in Object class ?
51. Where the thread flow of execution thread ?
52. Why to override run method ?
53. What is Synchronization ?
54. What is Object lock and what is class Lock ?
55. T1 is working on M1 synchronized method on A object
    Can the T2 execute M2 synchronized method on A object ?
56. What is the disadvantage of general way of creating the thread ?
57. What kind of data can we make thread safe using Class lock ?
58. What is Thread pool ?
59. How many thread pools are present in java?
60. How to create the thread pool ?
61. How many threads will be created in Single thread pool ?
62. How many threads will be created in cached thread pool ?
63. Difference between fixed and cached thread pool ?
64. How the thread will take the work in thread pool ?
65. Why are we going for callable interface ?
66. What is the abstract method present in callable ?
67. Is thread platform dependent or platform independent ?

68. How to know the current thread name ?
69. What are the different memories are available in Heap ?
70. What is Number format exception, Stack over flow exception ?
71. What is the difference between class not found class def not found ?
72. Can we write multiple finally blocks ?
73. Can we catch the parent exception first and then child exception ?
74. How to stop executing the finally block
75. In case of exception will the finally block gets executed ?
76. If I have the return statement in try and in finally then which return is given the preference ?
77. If I have the return statement in try and not in finally then which return is given the preference ?
78. What is eden memory ?
79. How to call garbage collection ?
80. What are the object class method ?
81. Which method gets executed when the object is garbage collected ?
82. How many times does the finalize method gets executed per object ?
83. How to make the object eligible for garbage collection ?
84. What are the different steps present in garbage collection ?
85. What happens when the exception occurred in finalize method ?
86. Is the finalize method protected or public ?
87. How to set minimum and maximum heap size ?
88. What are the steps involved in clonning an object ?
89. What is the difference between shallow cloning and deep cloning ?
90. What is marker interface ?

91. **What is the difference between collection and collections ?**
92. **What is the parent in collection framework ?**
93. **What do we go for collection framework ?**
94. **What is the internal implementation of Array List add method, get method and remove method ?**
95. **What is the internal data structure used by Array list ?**
96. **What are the differences between Array List and Vector ?**
97. **What is the internal data structure used by Hash Map ?**
98. **What is the internal implementation of Hash map put method, get and remove method ?**
99. **What happens when the collection objects is printed using sysout ?**
100. **Why does hash set doesn't allow duplicates ?**
101. **What happens if I pass same key in hash map ?**
102. **How many keys can be null in hash map**
103. **How can I make array list, or hash set synchronized ?**
104. **How to make the hash map and hash set object ordered ?**
105. **What is concurrent modification exception ?**
106. **How does concurrent modification exception arises ?**
107. **How to resolve the concurrent modification exception in Array list, Hash Set and Hash Map ?**
108. **What is the output of the below program ?**
     This is called as Widening

```
public class Test1 {
	public static void main(String[] args) {
		Test1 t= new Test1();
		float z=10;
		t.display(z);
	}
	void display(int i){
		System.out.println("int arg of display method");
	}
	void display(long i){
		System.out.println("long arg of display method");
	}
	void display(double i){
		System.out.println("double arg of display method");
	}
}
```

109. **How to convert String into integer ?**

110. What is the output of the below program ?

```
import com.ashokit.fullstack.oop.Employe;
public class Test1 {
	public static void main(String[] args) {
		Test1 t= new Test1();
		int z=10;
		t.display(z);
	}
	void display(byte i){
		System.out.println("byte arg of display method");
	}
	void display(Short i){
		System.out.println("Short arg of display method");
	}
}
```

111. What is the internal data structure used by Hash Set ?