www.youtube.com/c/javaexpress

Core Java – Java Express 2020

Why JAVA?

- Java is a High level programming language and **platform independent**.
- Java is compiled and interpreted programming language.
- According to the SUN 3 billion devices run on the java language only
- Java is a object oriented programming language.

Who is introduced java?

- It is introduced by James Gosling at Sun Microsystems now which is acquired by Oracle Corporation and released in the year 1995.
- The initial name of Java is OAK.
    - In 1995, Oak was renamed to "Java" because it was already a trademark by Oak Technologies.
- The implementation languages are C and C++.
- One slogan is available for Java is WORA, Write Once Run Anywhere.
- Java is not a pure object oriented Programming language because it supports primitive data types.

How many type of applications developed in Java ?

- In Java we can develop 3 types of applications like
    - Java Standard Edition (Standalone Applications)
        - Desktop applications like Antivirus and Media Player
    - Java Enterprise Edition (Enterprise or web applications)
        - Banking Applications
    - Java Mobile Edition (Mobile Applications)
        - Mobile Applications

How many buzz words in Java ?

- Simple
- Object Oriented
- Platform Independent
- Architectural Neutral
- Portable (Platform + Architectural Neutral)
- Robust(Exceptional handling + Strong Memory management)
- Secure
- Dynamic
- Distributed
- Multithread

What are the ClassLoaders in Java ?

- Every class Loader subsystem contains 3 types of class loaders.
- **Bootstrap class loader**
    - This class loader is responsible for loading core Java API classes i.e. classes present in rt.jar.
    - Bootstrap class loader is by default available with the JVM. It is implemented in native languages like c and c++.
    - Location of rt.jar is :--jdk\jre\lib\rt.jar
- **Extension class loader**
    - It is the child of Bootstrap class loader. The class loader is responsible to load classes from extension class path.
    - Location :jdk\jre\lib\ext
- **Application Class loader:**
    - This class loader is responsible to load classes from current working directory.
- Priority of class loader is ->bootstrap->extension->application

Difference between class and Object ?

Class	**Object**
1)Class is a container which collection of variables and method and objects	1)Object is a instance of a class
2) No memory is allocated at the time of declaration	2)Sufficient memory is allocated for all the variables of class at the time of declaration.
3)One class definition should exists only once in a program	3)For one class contain multiple objects can be created.

Difference between JDK,JRE and JVM ?

- JDK is Java Development Kit it contains Development tools and JRE
- Jre is java runtime environment in order to run java files is the implementation of JVM
- JVM is abstract machine responsible is to run byte code to run. And JVM is subset of JRE

Explain the working of JVM ?

- JVM is software based virtual machine and it is runtime engine to run java based applications.
- JVM is part of JRE and JRE is part of JDK.
- JVM is responsible to load and Run java Applications. (two activities)
- 3 important components are present in JVM
    - Class Loader subsystem (to load .class file)
    - Various Memory Areas (5 memory areas)
    - Execution Engine (to run java application)
- **Heap Area :** all objects in the program are stored in heap area.
- **Stack Area:** All local and reference variables are stored in stack area.
- **Method Area**: It is permanent area. It consist of static variables, methods and class definitions.
- **Native code area**:
    - It is not part of Java, native method calls stored in the native method stack created by JVM per thread.
    - For every thread JVM will create a separate native method stack.
- **Program Counter Register**:
    - For every thread a separate pc register will be created at the time of thread creation.
    - It can hold current execution instruction.

What are the development tools available in JDK ?

- Basic tools (appletviewer, apt,jar,java,ajvac,javadoc,javah,javap,jdb)
- Security tools(keytool)
- Internationnalization tool (native2ascii)
- RMI tools(rmic,rmiregistry)
- Java Deployment tools(javafxpackager)
- Java webstart tools(javaws)
- Java TroubleShooting,Profiling,Monitoring and Management tools(jcmd,jconsole,jmc,jvisualvm)
- Java Webservice tools (schemagen,wsgen)

What are the Datatypes available in JAVA ?

- Boolean -1 bit (false)
- Char-2 byte ('\u0000')
- Byte -1 byte(zero)
- Short-2 byte
- Int-2 byte
- Long-8 byte(OL)
- Float-4 byte(0.0f)
- Double-8 byte(0.0d)

What is the default values of primitive types ?

- Boolean –false
- char –'\u0000'
- short -0
- int -0
- long -0l
- float -0.0f
- double -0.0d

Difference between instance block and static block ?

Instance block	**Static block**
1)Instance block is used for initializing instance variable or calling any instance method.	1)Static block can be used for initializing static variables or calling any static methods.
2)Instance block executes when object of class is created	2)Static block executes when class is loaded into JVM
3) Instance block executes after static block	3)static block executes before instance block
4)static and non-static variables can be used in the instance block	4)Only static related variables and methods can be executed in the static block
5)this keyword cab be used in the instance block	5)this() keyword cannot be used in static block

Difference between this and super keywords in java?

- This is used to call current class constructor ,variables and methods.
    - Super is used to call super class constructor, variables and methods.
- This() or super() should be in the first statement only in the constructor.
- Every constructor by default calls super class constructor, which is no-argument constructor of super class but if u want u can call explicitly by using this() or super().

How java makes Secure?

- JVM is an interpreter which is installed in each client machine that is updated with latest security updates by internet. When this byte codes are executed, the JVM can take care of the security. So, java is said to be more secure than other programming languages.

What are the drawbacks of Java?

- No support for low level programming
- Less features in GUI :-> Java supports GUI controls but less features. Image cannot be placed in button.
- No Control over garbage collection
- Slow performance : speed of execution is slow
- Java takes more memory space.

What is Instance variable hiding in Java ?

- if there a local variable in a method with same name as instance variable, then the local variable hides the instance variable.

```
public class InstanceVaraible {
    private int i=10;

    private void m1()
    {
//localvariable hides the instance variable
        int i=20;
        System.out.println(i);
        System.out.println(this.i);
    }
```

What is instance block in Java ?

- Instance block are used to initialize instance variables.
- IIB are executed before constructors. Whenever object is created it will execute instance blocks.

```
public class InstanceBlock {
    {
        System.out.println("InstanceBlock-1");
    }
    InstanceBlock()
    {
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
        InstanceBlock i=new InstanceBlock();
    }
    //InstanceBlock-1
    //Constructor
}
```

Execution flow

- Instance Initialization Block of super class
  Constructors of super class
- Instance Initialization Blocks of the class
  Constructors of the class

What if we want to execute some code for every object ?

- Instance block

What is static block in Java ?

- Static block is executed only once in a program ,at the time of class loading.
- Static blocks are executed before instance, constructor.

```
public class InstanceBlock {
    static{
        System.out.println("static block1");
    }
    {
        System.out.println("InstanceBlock-1");
    }
    InstanceBlock()
    {
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
        InstanceBlock i=new InstanceBlock();
    }
    //static block1
    //InstanceBlock-1
    //Constructor
```

Why Java is not pure Object Oriented Programming language ?

- Because it supports primitive types.
- Smalltalk and simula are pure Object Oriented Programming language.

How to Override equals method in Java ?

```
private int id;
public Employee(int id)
{
    this.id=id;
}
@Override
public boolean equals(Object obj)
{
    if(obj==this)
        return true;
    if(!(obj instanceof Employee))
        return false;
    Employee e=(Employee)obj;
    return this.id==e.id;
}
public static void main(String[] args) {
    Employee e1=new Employee(1);
    Employee e2=new Employee(1);
    if(e1.equals(e2))
        System.out.println("Equal");
    else
        System.out.println("Not Equal");
```

How to override toString() in Java ?

```
@Override
public String toString() {
    return "Employee [id=" + id + "]";
}
```

- If u don't override toString(),when you try to print any object it will print
- ObjectName@HashCode

How to print dynamically classes available in Object ?

```
Object obj;
obj='a';
System.out.println(obj.getClass().getName());
obj=1;
System.out.println(obj.getClass().getName());
obj="hello";
System.out.println(obj.getClass().getName());
obj=1.22;
System.out.println(obj.getClass().getName());
obj=false;
System.out.println(obj.getClass().getName());
obj=1.2f;
System.out.println(obj.getClass().getName());
obj=1.2d;
System.out.println(obj.getClass().getName());
```

```
java.lang.Character
java.lang.Integer
java.lang.String
java.lang.Double
java.lang.Boolean
java.lang.Float
java.lang.Double
```

OOPS Concepts

What is a class ?

- Class is a user defined blueprint or protype from which objects will create.
- It contain set of properties or methods that are common to all objects of one type.
- We can't create objects of abstract class and interfaces.

Can we create object of abstract class and interface ?

- No

What is a object ?

- An object consists of
  - State
    - It represents attributes/state of an object
  - Behavior
    - It represents methods of an object.
  - Identity
    - It gives unique name to an object.
- Student is a Class
  - Student s1=new Student()
  - S1 is a object
  - Student have states like name, age, number etc.

How to create Object in Java ?

- In Java all objects are dynamically allocated on heap area.
- There are 4 ways to create object in Java
  - new keyword
  - class.forName
  - Deserialization
  - Clone() method
- **New Keyword**
  - It is the general way of creating object in Java

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

- **Class.forName(String name)**
  - This is predefined class present in java.lang. package.
  - Class.forName it will load the class in java but doesn't create any object.
  - forName is a method that returns the Class object ,so we should give fully qualified name of class.
  - On class newInstance() method on Class object then it will return new instance of the class.

```
try{
    Class c=Class.forName("opps.Student");
    Student student1=(Student)c.newInstance();
}
catch(ClassNotFoundException c)
{

}
catch(InstantiationException i)
{

} catch (IllegalAccessException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

- **Clone() method**
  - Clone() method present in Object class, it creates and returns a copy object.
  - Whenever clone() method is called, JVM will create a new object and copies all contents of the previous object into it.
  - Creating a object using clone method does not invoke any constructor.
  - If u not implement cloneable interface it will throw CloneNotSupportedExceptoin

```
try {
    Student student2=(Student)student.clone();
    System.out.println(student2);
} catch (CloneNotSupportedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

- **Deserialization**
  - Deserialization is the process of reading object state from a file

```
try {
    InputStream is=new FileInputStream("abc.ser");
    ObjectInputStream ois=new ObjectInputStream(is);
    Object obj=ois.readObject();

} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Anonymous object

- Anonymous objects are the objects that are instantiated but not stored in reference variable.
- They are used for immediate method calling
- They are used for destroyed after method calling.
- They are used in different libraries like AWT when key is pressed we are simply creating anonymous object.

```
new Student(){
    public void m1()
    {
        System.out.println("m1 method called");
    }
}.m1();
```

Using newInstance() method of constructor class

- This is similar to newInstance() method of class.
- There is one more newInstance() method in java.lang.reflect.Constructor class we can use to create object.
- By using this we can call private constructors also.

```
Constructor c=ReflectionExample.class.
        getDeclaredConstructor();
c.newInstance();
```

What is Encapsulation ?

- Encapsulation means wrapping of data as a single unit to protect code from outside class.
- We can achieve encapsulation by declaring all variables as private and writing public methods in the class to set and get the values of variables.

Advantage:

- **Data Hiding:**
  - Data in a class is hidden from other classes so it is known as Data hiding.
  - The user will have no idea about inner implementation of the class. It will not visible to the user how the class storing the values in the variables.
  - Use will know about passing values to the setter method and getting initialized with that value only.
- **Increased Flexibility**
  - We can make variables of the class read only and write only.
- **Reusability**
  - It will improve the reusability and easy to change with new requirement without effecting who implemented these classes.

What is the primary benefit of Encapsulation ?

- The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code. With this Encapsulation gives maintainability, flexibility and extensibility to our code.

What is overriding rules in Java ?

- **Overriding and access modifiers**
  - Protected->public, protected
  - Default->default, protected, public
  - Public ->public
- **Final method we cannot overridden**
  - Error: final method cannot override from parent.
  - **Remove final modifier from super class**.
    - If super class defines m1() method then sub class defines final m1() then wont throw any error but vice versa it will **throw final method cannot override**.
- **Static methods can not be overridden**
  - When u define static method with same signature as a static method in base class, it is known as method hiding.
  - If u define static method in super class then sub class define instance method then it will throw **instance method m1() cannot override static method from parent**.
- **Private methods cannot be overridden**
  - When u call private method then it will throw error i.e. **the method m2() from parent is not visible.**
  - Inner class can access private members of its super class.
- **The overriding method must have same return type or sub type:**
  - In Java 5.0 onwards it is possible to have different return type for a overriding method in child class but child class return type should be sub type of parent return Type.
  - Covariant return type is based on LisKov substitution Principle.
- **Exception Rules**
  - If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but it can declare unchecked exception.
  - If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

Difference between abstraction and Encapsulation ?

Abstraction	**Encapsulation**
1)Abstraction is means hiding implementation details ,showing functionality to the user is called abstraction.	1)Encapsulation is a mechanism to wrapping data and code together into a single unit to protect data from outside world.
2)You can use abstraction using interface or abstract class	2)You can implement encapsulation using access modifiers(public, protected and private)
3)Abstraction solves the problem at Design level	3) Encapsulation solves the problem in implementation level.
4)For simplicity, abstraction means hiding implementation using abstract class and interface	4)For simplicity, encapsulation means hiding data using getters and setters

3. Abstraction lets you focus on what the object does instead of how it does it while Encapsulation

means hiding the internal details or mechanics of how an object does something.

4.For example: Outer Look of a Television, like it has a display screen and channel buttons to change channel it explains Abstraction but Inner Implementation detail of a Television how CRT and Display Screen are connect with each other using different circuits , it explains Encapsulation.

What is abstract class?

- A class that is declared as abstract is known as abstract class.
- It needs to be extended and its method implemented. It cannot be instantiated.

Can there be any abstract method without abstract class ?

- No, if there is any abstract method in a class, that class must be abstract.

Can you use abstract and final both with a method ?

- No, because abstract method need to be overridden where you can't override final method.

What is interface ?

- Interface is a blue print of a class that have static constants and abstract methods.
- It can be used to achieve fully abstraction and multiple inheritance.
- It is used to achieve loose coupling.

Can you declare an interface method static ?

- No, because by default methods in interface are abstract.
- In Java8, we can define default methods and static methods in interface also.
- You can override default method in implementation class but you can't override static methods.

Can an interface be final ? or Can you write final method in interface ?

- No, because its implementation is provided by another class.

What is marker interface ?

- An interface that have no data member and method is known as marker interface.
    - Ex: Serialization, Cloneable etc.

Can we define private and protected modifiers for variables in interfaces ?

- No, they are implicitly public, static final

When can be object reference be cast to an interface reference?

- When the object implements the referenced interface

What is a static method in Java ?

- We can define static methods in interface but we cannot override static methods in implementation class.

```
interface TestInterface1{
    static void show(){
        System.out.println("static method executed");
    }
}
public class TestClass2 implements TestInterface1 {
    public static void main(String[] args) {
        TestClass2 t=new TestClass2();
        TestInterface1.show();
    }


}
```

What is default method in interface ?

- Default methods are also known as defender methods or virtual extension methods.
- Default methods are introduced to provide backward compatibility for old interfaces so that they can have new methods without effecting existing code.
- Before Java 8, interfaces could have only abstract methods. The implementation of these methods has to be provided in a separate class. So if new method is added implementation class has to be affected .To overcome this issue they introduced default methods.

```
2  interface TestInterface{
3      public void square(int a);
4      default void show(){
5          System.out.println("Default method executed");
6      }
7  }
8  public class TestClass implements TestInterface {
9      @Override
10     public void square(int a) {
11         // TODO Auto-generated method stub
12         System.out.println(a*a);
13     }
14     public static void main(String[] args) {
15         TestClass t=new TestClass();
16         t.show();
17         t.square(45);
```

Problems Javadoc Declaration Search Console Progress Servers Error Log

<terminated> TestClass [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jul 25, 2017, 9:56:39 PM)

```
Default method executed
2025
```

How to call interfaces in multiple inheritance ?

```
package newfeatures;
interface A{
    default void m1(){
        System.out.println("m1");
    }
}
interface B{
    default void m2(){
        System.out.println("m2");
    }
}
public class InterfaceDemo implements A,B{
     void call()
    {
        A.super.m1();
        B.super.m2();
    }
    public static void main(String[] args) {
        InterfaceDemo i=new InterfaceDemo();
        i.call();
    }
}
```

Difference between abstract class and interface ?

Interface	**Abstract class**
1)Interface contain only abstract methods	1)Abstract class contain abstract and non-abstract methods
2) Interface supports multiple inheritance	2)It doesn't support multiple inheritance
3)Interface has only static and final variables	3)abstract class contain static, non-static and final variables
4)Interface can't have static methods, main method and constructor	4)abstract class can have static methods, main method and constructor
5) Interface can't provide implementation of abstract class	5)Abstract class can provide the implementation of interfaces
6)interface keyword Is used to define interface	6)abstract keyword is used to declare abstract class interface

Difference between method overloading (Early) and method overriding (late)?

Method overloading	Method overriding
1)Method overloading takes place in the same class	1)In order to implement method overriding we require inheritance
2)Compile Time polymorphism or static binding or early binding	2)Run time polymorphism or dynamic binding or late binding
3)**Condition**: Method name should be same and argument should be different	3)**Condition**: method name and argument should be same in super class and subclass
4)private, static and final methods you can override	4)we can't override static, final and private methods
5)There is no restrictions for access modifiers	5)some restrictions are available for access modifiers. we can't reduce scope 1)public ->public 2)protected ->protected public 3)default->default | public | protected

What is functional interface in Java ?

- A functional interface is an interface that contains only one abstract method. They can have only one function to exhibit.
- From Java8, lambda expressions are used to implement functional interfaces.
- Functional interfaces can have any number of default methods.
- Before Java8,we had to create anonymous inner class to implement these interfaces.

Before 1.8

```
public class TheadDemo {

    public static void main(String[] args) {

        new Thread(new Runnable(){

            public void run()
            {
                System.out.println("run method executed");
            }
        }).start();
    }
```

After 1.8

We can assign lambda expression to its functional interface object like this

```
public class TheadDemo2 {

    public static void main(String[] args) {

        new Thread(
        ()->
            {
                System.out.println("run method executed");
            }
        ).start();
    }
}
```

How to create your functional interface in Java ?

```
@FunctionalInterface
interface Demo
{
    int  calculate(int x);
}
public class CalculateDemo {
    public static void main(String[] args) {
        int a=5;
        Demo d=(int x)-> x*x ;
        int ans=d.calculate(a);
        System.out.println(ans);
    }
}
```

Which modifiers are used in nested interface ?

- Only public, default modifiers are allowed, if we use other modifiers compile time.

```
interface Map
{
    public interface Entry
    {
        public void getKey();
    }
    private interface Entry1
    {
    }
    protected interface Entry2
    {
    }
```

Strings

How to create immutable class in Java ?

- Declared class as final
- Declared all fields as final
- Provide getter methods
- Provide Parameterized constructor
- Don't provide setter methods.

```
public final class ImmutableClass {
    private final Integer id;
    public ImmutableClass(Integer id) {
        // TODO Auto-generated constructor stub
        this.id=id;
    }
    public Integer getId()
    {
        return id;
    }
    public static void main(String[] args) {
        ImmutableClass i=new ImmutableClass(10);
        System.out.println(i.getId());
    }
}
```

What is String in java ?

- String is class in java and defined in java.lang package.
- String is immutable and final in java and JVM used String pool to store all the string objects.
- It's not a primitive data type like int and long.

What are the different ways to create String Object?

- New operator
  - String str=new String("name");
- Literal
  - String str="Nani";

Why string is immutable or final in java ?

- It increases **security** because any hacker can't change its value and it's used for storing sensitive information such as username and password.
  - you can open any file in java by passing the name of the file as argument to File I/o classes.
- Ex: suppose 5 reference variables, all refers to one object "Schain".
  - If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string is immutable in java.

What is the advantage of Immutable ?

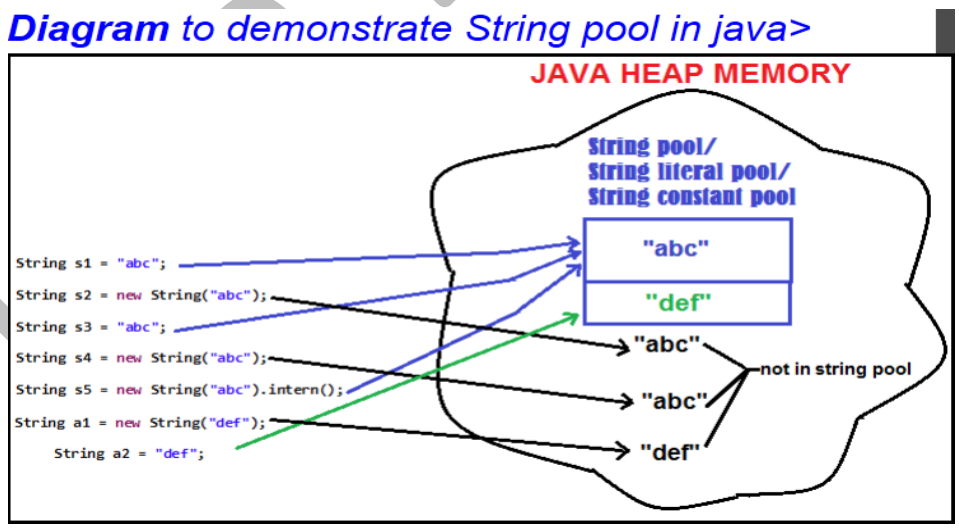
- **Security** :
  - **Java class loading mechanism works on class names passed as parameters**, then these classes are searched in class path. Imagine for a minute, Strings were mutable, then anybody could have injected its own class-loading mechanism with very little effort and destroyed or hacked in any application in a minute.
- **Performance**
- **Thread Safe**
- **Cache support**
  - Frequently we used hashcode of string in Hashset and HashMap.

What is the output for below program ?

```
String s = "Learn Share Learn Learn";
int output = s.indexOf('a',3);
System.out.println(output);// returns 8
```

What is String Pool ?

- String pool is contain pool of strings stored in Java heap memory.
- String pool helps saving lot of memory and increase performance.
- When we use double quotes to create a string, it first looks for string with same value in the string pool. if found it just returns the reference else it creates a new string in the pool and returns the reference.



How many strings created for String str=new String("cat") ?

- Either 1 or Either 2 because it will check cat is available in Stringpool.If found it will create String else it will create one string in String pool and Other string in heap memory.

What does String intern() method do ?

- When intern() method is invoked, if pool already contains a string equal to String object as determined by the equals(Object) method, then String from the pool is returned. Otherwise object is added to the pool and a reference to this string object is returned.

How do you check if two strings are equal in Java ?

- == is binary operator in java
  - It is public method of java.lang package
- == can be used for primitive types and derived types
  - Equals used for derived types.
- == you can't override this operator
  - Equals u can override in any class.
- == it will check value and location in memory
  - Equal it wil check value only

How to convert String to char and String to byte?

- String str="java ";
  - char[] ch=str.toCharArray();
  - byte[] byteArray=str.getBytes();

Can we use String in Switch case?

- In java 1.7,its possible to use string in switch case.

Difference between StringBuffer and StringBuilder ?

- StringBuffer is synchronized where as StringBuilder is not synchronized.
- String builder (Multithread environment)performance is high when compare to StringBuffer(single thread environment) because no overhead of synchronization.

Difference between length and capacity of StringBuffer ?

- Length is number of characters available and capacity is total much allocated space if u add characters it will add capacity + length.

```
StringBuffer s1=new StringBuffer("hello");
int p=s1.length();
int q=s1.capacity();
System.out.println("Hello="+p);//5
System.out.println("Capacity of string Hello="+q);//16+5=21
```

How to prove equals method is not override in StringBuffer ?

```
StringBuffer s3=new StringBuffer("hello");
StringBuffer s4=new StringBuffer("hello");
System.out.println(s3.equals(s4));//false
```

Find below program output ?

```
String s1 = "Ram";
String s2 = "Ram";
String s3 = new String(" Ram");
String s4 = new String(" Ram");
String s5 = "Shyam";

System.out.println(" Comparing strings with equals:");
System.out.println(s1.equals(s2));//T
System.out.println(s1.equals(s3));//T
System.out.println(s1.equals(s5));//F

System.out.println(" Comparing strings with ==:");
System.out.println(s1==s2);//T
System.out.println(s1==s3);//F
System.out.println(s3==s4);//F

System.out.println(" Comparing strings with compareto:");
System.out.println(s1.compareTo(s3));//0
System.out.println(s1.compareTo(s5));//-1
```

What is the difference between + operator and concat() function ?

- + operator ,one of the operands must be of the type String, else normal plus operation is executed.
- For concat(),both operands need to be strings

What is String Tokenizer ?

- Is used to break a string into tokens. This class is available in java.util package.
- Using split(Regular expression) method to break string into tokens

```
System.out.println("Using Constructor 1 - ");
StringTokenizer st1 =
     new StringTokenizer("Hello World", " ");
while (st1.hasMoreTokens())
    System.out.println(st1.nextToken());

System.out.println("Using Constructor 2 - ");
StringTokenizer st2 =
     new StringTokenizer("Hello : World", " :");
while (st2.hasMoreTokens())
    System.out.println(st2.nextToken());

System.out.println("Using Constructor 3 - ");
StringTokenizer st3 =
     new StringTokenizer("Hello:World:",":",  true);
while (st3.hasMoreTokens())
    System.out.println(st3.nextToken());
```

```
Using Constructor 1 -
Hello
World
Using Constructor 2 -
Hello
World
Using Constructor 3 -
Hello
:
World
:
```

Difference between parseInt and Valueof?

- parseInt return type is primitive type
- Valueof method return type is Object type(java.lang. Integer)

Difference between String and String Buffer ?

String	String Buffer
1)String is immutable	1)String Buffer is mutable
2)It is thread safe object	2)It is thread safe and synchronized
3)hashcode and equals methods are overdried	3)both methods are Not overrided
4)In case of performance String is slow compare to String Buffer →String is slow and consumes more memory when we concat many strings every time it creates new instance	4)StringBuffer is fast when compare to String. →It is fast and takes less memory when u concat strings.
5)It is implementing comparable and comparator interfaces	5)It is not implementing both interfaces
6)we cannot perform these operations in String like append, insert, delete operations	6)we can perform these operations in StringBuffer like append, insert, delete operations

POINTS

When to use which one :

- If a string is going to remain constant throughout the program, then use String class object because a String object is immutable.
- If a string can change (example: lots of logic and operations in the construction of the string) and will only be accessed from a single thread, using a StringBuilder is good enough.
- If a string can change, and will be accessed from multiple threads, use a StringBuffer because StringBuffer is synchronous so you have thread-safety

String Regular Expressions

Username Validation

```
String str="anbhas";
Pattern p=Pattern.compile("^[a-zA-Z]{1,6}");
Matcher m=p.matcher(str);
boolean b=m.matches();
System.out.println(b);
```

Email Validation

```
String str="a@a.com";
Pattern p=Pattern.compile("(\\w+)@(\\w+)\\.[a-z]{3}");
Matcher m=p.matcher(str);
boolean b=m.matches();
System.out.println(b);
```

Date Validation

```
private Pattern pattern;
private Matcher matcher;
private static final String IPADDRESS_PATTERN="" +
        "([0]\\d|[12]\\d?|[3][01])\\-([0]\\d|[1][0-2])\\-((19|20)\\d\\d)";

public DateValidation(){
  pattern = Pattern.compile(IPADDRESS_PATTERN);
}
public  boolean validate(final String ip){
  matcher = pattern.matcher(ip);
  return matcher.matches();
}
public static void main(String[] args)
{
    DateValidation ipadd=new DateValidation();
    System.out.println(ipadd.validate("01-12-17"));
}
```

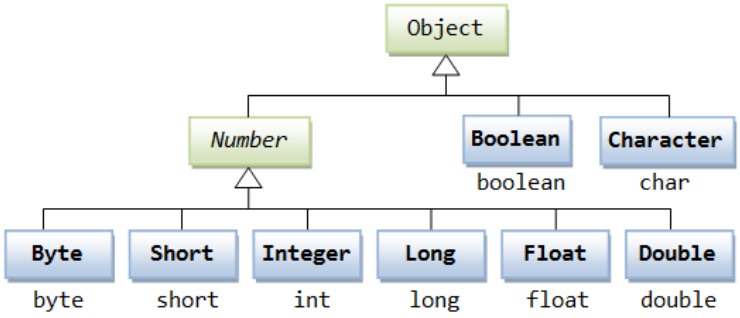
Ip Address Validation

```
private Pattern pattern;
private Matcher matcher;
private static final String IPADDRESS_PATTERN
="^([01]\\d?\\d?|[2][0-4]?\\d?|[25][0-5]?)\\." +
"([01]\\d?\\d?|[2][0-4]?\\d?|[25][0-5]?)\\."+
"([01]\\d?\\d?|[2][0-4]?\\d?|[25][0-5]?)\\." +
"([01]\\d?\\d?|[2][0-4]?\\d?|[25][0-5]?)";

public IPAddressValidator(){
  pattern = Pattern.compile(IPADDRESS_PATTERN);
}
public  boolean validate(final String ip){
  matcher = pattern.matcher(ip);
  return matcher.matches();
}
public static void main(String[] args)
{
    IPAddressValidator ipadd=new IPAddressValidator();
    System.out.println(ipadd.validate("249.249.249.249"));
}
```

Wrapper Classes

What is use of wrapper classes ?

- Wrapper classes are used to convert any datatype in to an object.
- All wrapper classes are available in java.lang package.
- There are 8 types of wrapper classes are available.



- All wrapper classes are immutable and final.

Why we use wrapper classes in Java ?

- We know that in java whenever we get input form user, it is in the form of string value so here we need to convert these string values in different different datatype (numerical or fundamental data), for this conversion, we use wrapper classes.

```
String s1="10";
int i=Integer.parseInt(s1);
System.out.println(i);
```

What is autoboxing feature in Java ?

- Converting primitive data type in to object type is called **autoboxing**.(Primitive to Wrapper)

```
int a=10;
Integer i=Integer.valueOf(a);
System.out.println(i);
```

What is unboxing feature in java ?

- Converting object type in to primitive data type is called **Unboxing**.(Wrapper to primitive).

```
Integer j=20;
int i=j;
System.out.println(i);
```

Exception Handling

What is Exception handling ?

- Exception is an event which occurs during program execution time that stops the normal flow of execution.
- Exception Handling is a mechanism to handle runtime errors such as ClassNotFound exception, SQL Exception etc.
- Exception are occurred due to two reasons.
    - Developer Mistakes
    - End User mistakes.
        - While providing inputs to the application
        - Whenever user is entered invalid data then exception occur
        - Exception is occurred when network has disconnected at the middle of the communication
- The main advantage of exception handling is to maintain the normal flow of the application.

Difference between checked exceptions and Unchecked Exception ?

Checked Exception	Unchecked Exception
1)These exceptions are checked by the compiler	1)These exceptions are not checked by the compiler
2)we can handle these exceptions by using try-catch and throws	2)These exceptions are not handled by the programmer.
3)Exception is the super class of all checked Exceptions	3)Runtime Exception is the super class of all unchecked Exceptions.

Difference between Exception and Error ?

- Exceptions are caused due to developer mistakes these are recoverable. But errors caused due to lack of system resources these are non-recoverable.
    - StackOverflowError, OutofMemoryError, hardwareFailure

Difference between throw and throws in Exception ?

Throw	Throws
1)throw keyword is used to explicitly throw an exception	1)throws keyword is used to declare an exception
2)you cannot throw multiple exceptions	2)You can declare multiple exceptions.
3)throw is used within the method	3)throws is used with the method signature
4)checked exceptions cannot be propagated	4)checked exceptions can be propagated
5)throw is followed by an instance	5)throws is followed by class

Explain about finally block ?

- Finally block is executed irrespective of try and catch.
- It is used to provide clean-up code.
    - Database connection closing
    - Streams closing
    - Object destruction
- Finally block will run after catch block or try block.
- In two cases **finally block won't be executed**
    - Whenever we are giving chance to try block then only finally block will be executed otherwise it is not executed.
    - Whenever we are using system.Exit (0) JVM will be shutdown hence rest of the code won't be executed.
    - An infinite loop would also prevent a finally being called.

What is exception Propagation ?

- Exception propagation is a way of propagating exception from method to method.

Can finally block be used without catch?

- Yes, by try block. Finally must be followed by either try or catch

Is it necessary that each try block must be followed by a catch block ?

- Not necessary. It should be followed by either a catch block or finally block

What is the base class of Error and Exception ?

- Throwable

What are the important methods available in Exception Class?

- Public String getMessage()
- Public String getLocalizedMessage()
- Public synchronized Thorwable getCause()
- Public String toString()
- Public void printStackTrace()

What is java7 ARM and catch block improvements ?

- ARM(Automatic Resource Management):
    - Most of them we use finally block just to close the resource and sometimes we forgot to close the them and get runtime Exceptions .
    - Instead of using finally block, In java 7 one feature is available ARM
    - One of the improvement is try with resources where we can create Resource in the try statement itself and use it inside the try catch block.
    - Once Execution completed, runtime environment automatically close these resources.
    - AutoCloseable interface is introduced in java7 to close the resources.
- Benefits:
    - More Readable code and easy to write
    - Number of lines of code is reduced.
    - No need of finally block just to close these resources.
    - We can open multiple resources in try-with-Resource statement separated by a semicolon.
- We can define multiple exceptions in a single catch block in Java7

Difference between NoClassDefFoundError and ClassNotFoundException ?

- NoClassDefFoundError:
    - Java Test
        - If test.class file is not available then we will get runtime exception saying NoclassDefFoundError:Test
        - It available in compile time but missed in runtime.
- ClassNotFoundException:
    - We will get error when we use Class.Forname(“”)
        - Ex: Connection database to oracle but if we miss oracle jar file in classpath it will throw ClassNotFoundException.

What is OutofMemoryError in java ?

- It is a subclass of java.lang.VirtualmachineError and its thrown by JVM when it ran out of heap memory.
- We can fix this error by providing more memory to run the java application using java options.

What are the different scenarios causing "Exception in thread main " ?

- UnsupportedClassVersionError—JDK issue
- NoClassDefFoundError –class is not available
- NoSuchMethodError: main-main method not available
- Predefined Exceptions-user mistake

What happens when exception is thrown by main method ?

- When exception is thrown by main() method, Java Runtime terminates the program and print the execution message and stack trace in system console

Can we have an empty catch block ?

- We can have empty catch block but it's the poor programming.

What is difference between final ,finally and finalize ?

- Final and finally are the keywords and finalize is a method.
- Final is used to declare a variable ,method and class
- Finally is a block it is associated with try or catch block to close the resources
- Finalize() method is executed by garbage collector before the object is destroyed. Its a great way to make sure all the global resources are closed.

What is assertion Error ?

- It is the child class of Error hence it is unchecked exception.
- Assert is java keyword used to define an assert statement.
- Assert statement is used to check expected boolean conditions
- It is mainly used for testing purpose
- Raised explicitly by the programmer or API developer to indicate that assert statement fails.
- Assert(x >10)
    - If x is not greater than 10 then we will get runtime exception saying assertion error.

Garbage Collector

What is Garbage Collector?

- The process of destroying unreferenced objects is called Garbage Collection.
- The main objective of Garbage Collector is to free heap memory by destroying unreferenced objects.
- Garbage Collection is automatic memory management feature.

How JVM can destroy unreferenced Object?

- JVM internally using Daemon thread called garbage collector to destroy all unreferenced objects.
- A daemon thread is service thread because it provides services to JVM to destroy unreferenced objects.
- It is low priority thread we cannot guarantee execution.

Can we force garbage collector?

- No, we cannot force garbage collector but we can request it by using System and Runtime.getRuntime().gc();

What is the algorithm JVM internally uses for destroying objects?

- Mark and sweep

What is the return type of finalize method ?

- Protected and it is a native method it throws Throwable exception.

www.youtube.com/c/javaexpress

What is the use of finalize method in java ?

- Finalize() method is used to perform clean up activities like Closing database connection, destroying unreferenced objects.
- Finalize method is never invoked more than once for any given object.
- In finalize method it ignores all the exceptions in the method and program will terminate normally.
- If I define one count variable ,then it will show only one time for one object.

```
public class Test {
    @Override
    protected void finalize() throws Throwable {
        // TODO Auto-generated method stub
        System.out.println("Object Garbage collected"+this);
        super.finalize();
    }
    public static void main(String[] args) {
        Test t1=new Test();
        Test t2=new Test();
        t1=null;
        System.gc();
        t2=null;
        System.gc();
    }
```

```
Problems  Javadoc  Declaration  Search  Console  Progress  Servers  Error Log
<terminated> Test (9) [Java Application] C:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (Jul 25, 2017, 6:50:57 PM)
ject Garbage collectedopps.Test@5f531aca
ject Garbage collectedopps.Test@4903f4aa
```

Difference between System.gc and Runtime.getRuntime().gc() ?

- System.gc ()
  - It is a static method.
  - It is a non-native method (i.e Code which doesn't directly interact with hardware and system resources.
  - System.gc() internally calls Runtime.getRuntime().gc
- Runtime().gc()
  - Instance method
  - Native method(A programming language directly interact with hardware and system resources.

What are the ways to make object eligible for garbage collection ?

- Object created inside a method.
- Reassigning the reference variable.
- Nullifying the reference variable
- anonymous object

Object Class

How many ways to create an Object ?

- By using new keyword
- Class.forName().newInstance();
- Class.FactoryMethod()
- By using clone()
- Object Deserialization

Difference between new operator and newInstance() method in java ?

- In general, new operator is used to create objects, but if we want to decide type of object to be created at runtime, there is no way we can use new operator. In this case, we have to use newInstance() method. Consider an example:

```
// Sample classes
class A {  int a; }
class B {  int b; }

public class Test
{
    // This method creates an instance of class whose name is
    // passed as a string 'c'.
    public static void fun(String c)  throws InstantiationException,
        IllegalAccessException, ClassNotFoundException
    {
        // Create an object of type 'c'
        Object obj = Class.forName(c).newInstance();

        // This is to print type of object created
        System.out.println("Object created for class:"
                        + obj.getClass().getName());
    }

    // Driver code that calls main()
    public static void main(String[] args) throws InstantiationException,
    IllegalAccessException, ClassNotFoundException
    {
         fun("A");
    }
}
```

When to use Factory Method in java

- When we don't know how to create the object of class(like which constructor we have to call like parameter constructor or default constructor)
    - Ex: Session Factory object.
- The purpose of factory method is creating object by calling private constructor of the class and returning instance

```
public class Factory {
    private int i;
    private Factory()
    {
        this.i=10;
    }
    public static Factory getBean()
    {
        return new Factory();
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Factory f=Factory.getBean();
        System.out.println(f);
    }

}
```

What is the use of getClass() method in Object class ?

- getClass () method return type is Class object.
- It is used to get runtime class of the object and it is used to get meta data of the class.

```
public static void main(String[] args) t
    Object obj=new String("abc");
    Class c=obj.getClass();
    System.out.println(c.getName());
}
```

How to print string class methods by using reflection ?

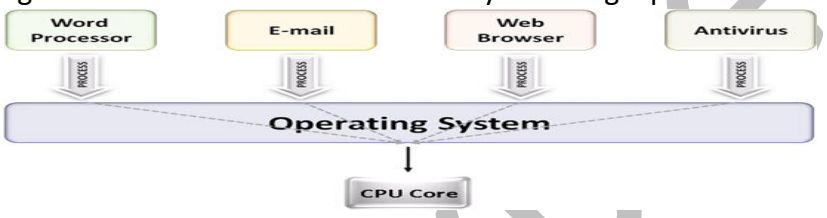
```
Object obj=new String("abc");
Class c=obj.getClass();
Method[] m=c.getMethods();
for(Method names:m)
{
    System.out.println(names);
}
System.out.println(c.getName());
```

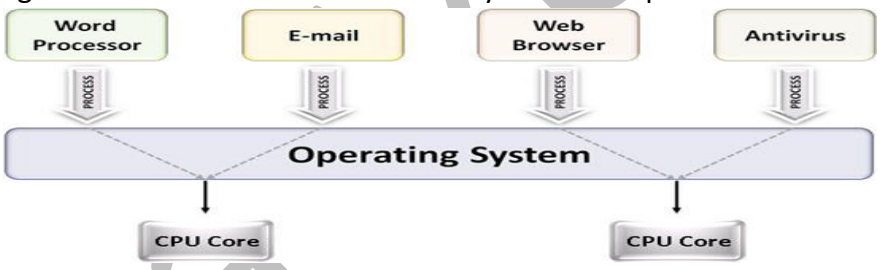
Multithreading

What is multithreading?

- The process of creating multiple threads in java stack area for executing multiple tasks concurrently in short period of time is called multithreading.
- Thread is nothing but sequential flow of execution.
- Thread is light weight sub process, a smallest unit of processing.
- Threads are independent, if exception occur in one thread, it doesn't affect other threads. It shares a common memory area.
- Executing more than one thread a time is called Multithreading.

What is multitasking and multiprocessing?

- Executing more than one task simultaneously on a single processor machine
  - 

- Executing more than one task simultaneously on a multi-processor machine
  - 

Difference between process and Thread ?

Process	Thread
1)Process have their own address space	1)Threads share the address space of the process that created it
3)Changes to the parent process does not affect the child process	3)changes to the main thread may affect the behavior of the other threads of the process
4)An executing instance of a program is called process	4)A thread is subset of the process
5)Process is controlled by the operating system	5)Threads are controlled by programmer in a program
7)Process is heavy weight process.(Process are heavily dependent on system resources)	7)Thread is used for small tasks.(Thread requires minimal amount of resources)

What are the benefits of Multithreading?

- Multiple threads are executing concurrently that improves the performance because CPU is not idle incase some thread is waiting to get some resources.
- Ex: Servlets are better in performance than CGI because Servlet support multithreading but CGI doesn't

Difference between User thread and Daemon Thread?

User thread	Daemon Thread
1) user threads are created by the programmer to perform some specific task	1)Daemon threads are mostly created by the JVM to perform some background tasks like garbage collection
2)User threads are high priority threads.	2)Daemon threads are low priority threads(main thread)
3) User threads are foreground threads	3)Daemon threads are background threads
4)JVM will not force the user threads to terminate	4)JVM will force the daemon threads to terminate if all the user threads have finished their tasks

How can we create user thread in java ?

- By using Thread class

```
ThreadDemo td=new ThreadDemo();
td.start();
```

- By using Runnable interface

```
ThreadDemo td=new ThreadDemo();
Thread t=new Thread(td);
t.start();
```

What are different states in lifecycle of Thread ?

- Life cycle stages are
  - New→when object is created that is new state
  - Ready ->when we call start() method on thread
  - Running state ->It will run method for business logic
  - Blocked /waiting ->if running thread interrupted it will go to sleeping state
  - Dead State->business logic of the project is completed means it is dead state

Life cycle stages are:-

1) **New**
2) **Ready**
3) **Running state**
4) **Blocked / waiting / non-running mode**
5) **Dead state**

New :- MyThread t=new MyThread();

Ready :- t.start()

Running state:- If thread scheduler allocates CPU for particular thread. Thread goes to running state The Thread is running state means the run() is executed.

Blocked State:-

If the running thread got interrupted of goes to sleeping state at that moment it goes to the blocked state.

- **Dead State:-**If the business logic of the project is completed means run() over thread goes dead state.

Can we call run() method of a Thread class ?

- Yes. we can call run() method of a thread class but it behave like a normal method. To actually execute it in a thread we need to start it using Thread.start() method.

How can we pause the execution of a thread for specific time ?

- We can pause the execution of thread by using sleep() method after time expiry it will run again to runnable state.

What do you understand about Thread priority?

- Every thread in java has some default priority. It may be default priority provided by JVM or customized priority provided by the programmer.
- 3 constants defined in the thread class.
  - MIN_PRIORITY=1
  - NORM_PRIORITY=5
  - MAX_PRIORITY=10
- If two threads same priority we cant expect which one is execute it depends on the Thread scheduler.

What is Thread scheduler and Time Slicing ?

- **Thread Scheduler:**
  - Thread Scheduler is part of JVM. It decides which thread is executed first and which thread is executed next.
  - Only one thread is executed at a time.
  - We can't expect exact behavior of thread scheduler it is JVM vendor dependent.
  - In Thread scheduler mainly used
    - Preemptive scheduling

- Time Slicing Scheduling

- **Preemptive Scheduling:**
  - Highest priority task is executed first after this task enters into waiting state or dead state then only another higher priority task come into existence.
- **Time Slicing Scheduling:**
  - A task is executed predefined slice of time and then return pool of ready tasks.

How can we make sure main() is the last thread to finish to java program?

- We can use Thread.join() to make sure all the threads created by the program is dead before finishing the main function.

What is join() method ?

- Join method allows one thread to wait for the completion of another thread.
- If t1 thread is executed when u call t2.join at that situation t1 must wait until completion of the t2 thread.
- Join() is used to stop the execution of the thread until completion of some other thread.
- It will throw one checked exception i.e. is Interrupted exception.

Difference between wait and sleep ?

Wait()	**Sleep()**
1)wait() method releases the lock	1)sleep() doesn't release the lock
2)this method is available in object class	2)this method is available in thread class
3)it is a non-static method	3)It is a static method
4)should be notified by calling notify() or notifyAll() method	4)after specified amount of time, sleep is completed.

Why thread communication methods wait(),notify() and notifyAll() are in Object class?

- **Two threads can communicate with each other by using wait, notify and notifyAll methods.**
- Thread class can call wait() ,notify and notifyAll all methods on any type of object so that's y it is available in Object class.
- Thread can call start() method in Thread class only not in any object

What is wait() and notify() methods ?

- **Wait():** The thread which is expecting updation is responsible to call wait() method then immediately the thread will enter into waiting state.
- **Notify():** The thread which is responsible to perform updation ,after performing updation it is responsible to call notify method then waiting thread will get notification and continue its execution with those updated items.

Why wait(), notify, notifyAll() methods have to be called from synchronized method or block?

- To call all methods on any object, thread should be owner of that object i.e thread should has lock of that object then the thread should be in synchronized area only otherwise it will throw IllegalMonitorStateException

Why Thread sleep() and yield() methods are static?

- These two method works on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state. That's why these methods are made static.

Why should we go for yield() method ?

- Yield() method causes to pause current executing thread to give the chance for waiting threads of same priority.
- If there is no waiting thread or all waiting threads have low priority then same thread can continue its execution.
- If multiple threads are waiting with same priority then we cant expect it depends on Thread Scheduler.
- It is native static method not implemented in Java.

How to achieve thread safety in java ?

- Synchronization
- Implementing concurrent lock interface using volatile keyword
- Using immutable classes and thread safe classes.

How to create daemon thread in Java?

- setDaemon(true) can be used to create daemon thread in Java. we need to call this method before calling start() method else it will throw IllegalThreadStateException.

Difference between Runnable and Callable ?

Runnable	**Callable**
1)It is introduced in 1.0 version	1)It is introduced in 1.5 version
2)It does not return anything.	2)It can return the result of parallel processing of a task.
3)we need to override run() method	3)we need to override call() method
4)It does not return any checked exception	4)It will return checked Exception
5)we can pass Runnable interface to Thread Class.	**5)There is no constructor defined in the Thread class which accepts a Callable interface. We need to use Executor Service interface in 1.5**

What is Future Object ?

- **Future is an interface it is used to hold the result of Computation for every thread object result from callable interface.**

```
for(MyCallable job:jobs)
{
    Future f=service.submit(job);
    System.out.println(f.get());
}
```

What is Thread Local ?

- Thread local class is used to create thread local variables.
- Every thread has it's own Thread local variable and they can use it's get and set() methods to get the default value or change its value local to thread.
- Thread can access this value, set the value and even we can remove the value like Servlet Context(set,get,remove).
- Thread local object maintains a separate value like userid, transaction id.

```
public class CustmerThread extends Thread{
    static Integer custId=0;
    private static ThreadLocal t1=new ThreadLocal(){
        protected Integer initialValue()
        {
            return ++custId;
        }
    };
    public void run()
    {
        System.out.println(
        Thread.currentThread().getName()+"executed with custId"+t1.get());
    }
    public CustmerThread(String name) {
        // TODO Auto-generated constructor stub
        super(name);
    }
    public static void main(String[] args) {
        CustmerThread c1=new CustmerThread("Customer1");
        CustmerThread c2=new CustmerThread("Customer2");
        CustmerThread c3=new CustmerThread("Customer3");
        c1.start();
```

What is ThreadGroup?

- ThreadGroup contain number of threads in single object. In such a way we can suspend and resume or interrupt group of thread by single method call.
- For example, imagine a program in which one set of threads is used for printing a document, another set is used to display the document on the screen, and another set saves the document to a disk file. If printing is aborted, you will want an easy way to stop all threads related to printing.

```
ThreadGroup tg=new ThreadGroup("Paren");
Thread t1=new Thread(tg,"one");
t1.start();
Thread t2=new Thread(tg,"two");
t2.start();
Thread t3=new Thread(tg,"three");
t3.start();
```

What is Thread Pool ?

- Thread pool is a container contain already created Threads is used to execute our jobs.
- Thread Pool implemented by using Executor Framework.
- Executor Service is interface implementation class of Executors.

```
public class ThreadPoolDemo extends Thread{
    String name;
    public ThreadPoolDemo(String name) {
        this.name=name;
    }
    public void run()
    {
        System.out.println(Thread.currentThread().getName()+
                "thread name "+name);
    }
    public static void main(String[] args) {
        ThreadPoolDemo[] jobs={new ThreadPoolDemo("a"),
                               new ThreadPoolDemo("b")};
        ExecutorService service=Executors.newFixedThreadPool(3);
        for(ThreadPoolDemo t:jobs)
        {
            service.submit(t);
        }
        service.shutdown();
    }
```

What is Java Thread Dump? How can we get java Thread Dump of a program?

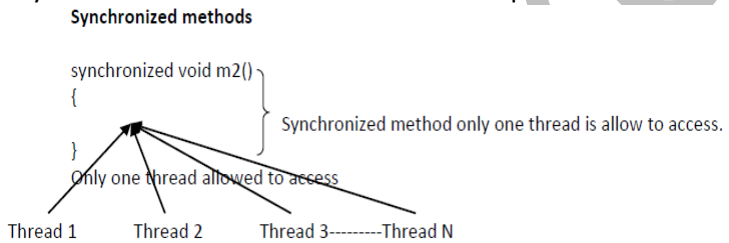
- Thread dump is list of all the threads active in JVM. It is used to analysing the deadlock situations
- There are many ways like using Profiler , kill -3 command and jstack tool (terminal based tool)comes with JDK installation.

What is synchronization ?

- The process of accessing multiple threads to modify an object in sequence is called Synchronization.

What is synchronized method?

- Synchronized modifier is applicable for method but not for classes and variables.
- IF a method or block declared as synchronized then at a time only one thread is allowed to operate on the given object.
- The main purpose of synchronized modifier is to reduce data inconsistency problems.
- The main disadvantage of synchronized is it increases the waiting time of the thread and effects performance of the system.
- Only one thread is allowed to access so the performance of the application will be reduced.



1) In the above case only one thread is allow to operate on particular method so the data inconsistency problems will be reduced.
2) Only one thread is allowed to access so the performance of the application will be reduced.
3) If we are using above approach there is no multithreading concept.

What is synchronized block ?

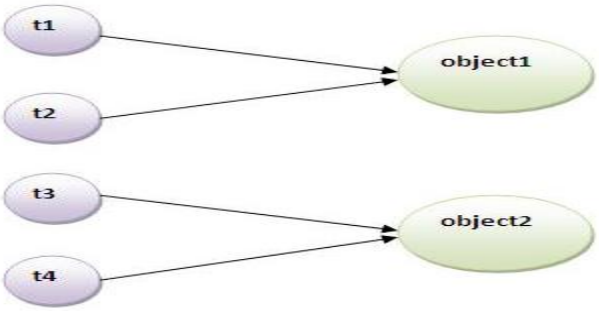
- If application method contain 100 lines but if we want to synchronized only 10 lines of code use synchronized block.
- The synchronized block contains less scope compare to method.
- If we are writing all the method code inside the synchronized blocks it will work same as the synchronized method.

What are the method available to prevent thread Execution ?

- Yield()
- Join()
- Sleep()

What is static synchronization ?

- If u make any static method as synchronized, the lock will be on the class not object.



- **Problem without synchronization:**
  - Suppose there are two objects of a shared class(e.g. Table) named object1 and object2.In case of synchronized method and synchronized block there cannot be interference between t1 and t2 or t3 and t4 because t1 and t2 both refers to a common object that have a single lock. But there can be interference between t1 and t3 or t2 and t4 because t1 acquires another lock and t3 acquires another lock. I want no interference between t1 and t3 or t2 and t4.Static synchronization solves this problem

Why use synchronization ?

- To prevent thread interference.
- To prevent consistency problem.

What is Starvation ?

- Long waiting of a thread where waiting thread never ends is called Deadlock
- Long waiting of a thread where waiting thread end at certain point is called Starvation.
- Ex: 1 crore threads are there one thread is priority is 1 and Remaining threads are priority 10 then after long waiting it will get chance to execute one thread is called starvation

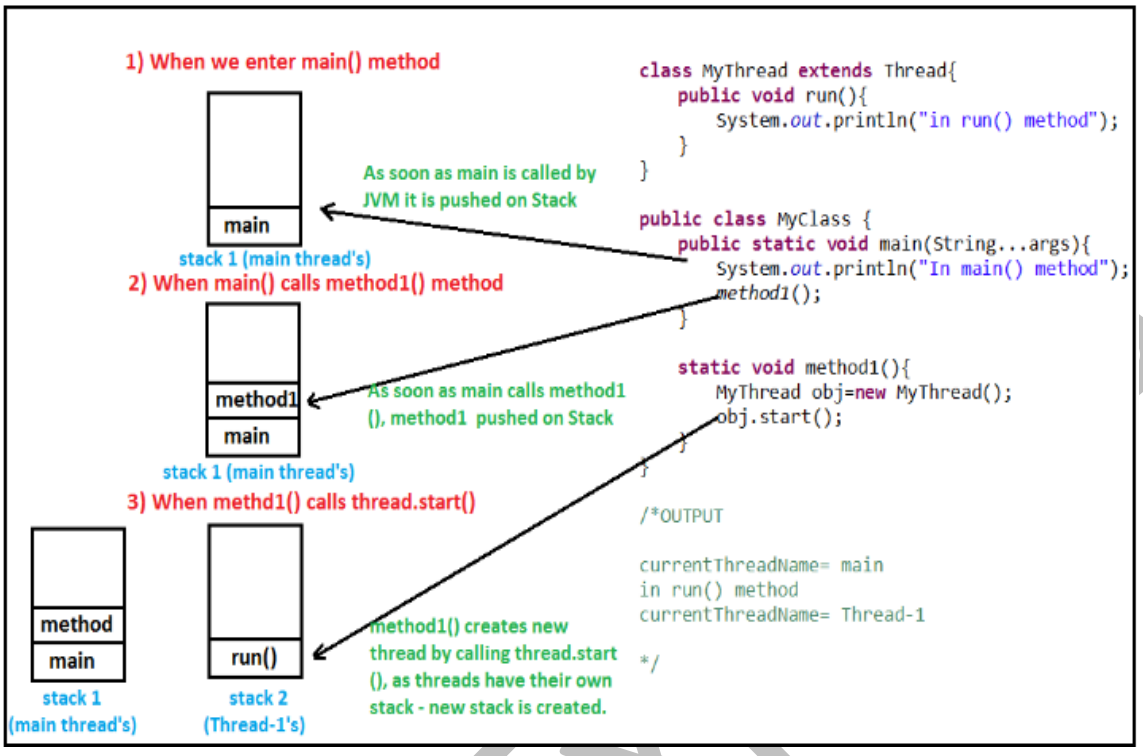
What is deadlock in Java?

- Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.
- Two threads are waiting for each other forever such type of infinite waiting is called Deadlock.
- Synchronized keyword is the only reason for deadlock situation.

How to prevent deadlock in java ?

- Mutual Exclusion->one process can use the resource at any given instant of time
- Hold and wait->process is hold and requesting other resources
- No preemption->once process allocated, they must be released by OS

Does Thread implements their own stack, if yes how ?



What is context-switching ?

- The process of switching the cpu from one process or thread to other process or thread.
- Context switching happen when we do hardware interrupt

Serialization

What is Serialization and Deserialization ?

- Serialization is the process of converting object into stream of bytes, using serialization we can store object state permanently in a destination.
- Deserialization is the process of converting stream of bytes in to original object.

What is transient variable ?

- If any variable is transient ,we cannot serialize that variable.

What is Marker Interface ?

- Serializable is a marker interface, it doesn't have any methods. It provides special identity to JVM to serialize the Object.
- If given object is not serialize then writeObject will throw unchecked exception called java.io.NotSerializable Exception.

What is the functionality of writeObject() and readObject methods ?

- writeObject() method stores object state in the file with the below information
  - its className
  - Current .class files SerialVersionUID
  - Non-transient variable names and their data type
- readObject() first read class name from serialized file and load that class using Class.forName.
  - It will compare SerialVersionUID of current loaded class with the SerialVersionUID that is stored in serialized file.
  - If both are not same it terminates deserialization process by throwing exception InvalidClassException.
  - If both values are same, it create object with the current loaded class non-static variables without using new keyword.

Difference between Serialization and Externalization ?

Serialization	**Externalization**
1)Everything take care by JVM	1)Everything take care by Programmer
2)In Serialization all objects saved to a file not part of object.	2) based on Requirement, we can save part of object or all objects to file.
3)Performance is low	3)Performance is high
4)It is suitable for if u want to save all objects to file	4)It is suitable for If u want to save part of object
5)writeObject and readObject	5)Two method available writeExternal and readExternal
6)It is the super interface.(marker)	6)Externalizable interface is sub interface of Serializable.

Program for Serialization

```
Student s1=new Student(1,"bhaskar");
Student s2=new Student(2,"nani");
OutputStream os=new FileOutputStream("student.txt");
ObjectOutput oo=new ObjectOutputStream(os);
System.out.println("Serialization process started");
oo.writeObject(s1);
oo.writeObject(s2);
oo.close();
System.out.println("Serialization process completed");
```

Program for Deserialization

```
try{
    InputStream is=new FileInputStream("student.txt");
    ObjectInput oi=new ObjectInputStream(is);
    System.out.println("DeSerialization process has started");
    Student s;
    while ((s = (Student) oi.readObject()) != null) {
        System.out.println(s);
    }
    oi.close();
}
```

How to overcome EOF file exception in Deserialization process ?

- Creating new class in Serialization
- In Deserialization loop check instance up to Count.

```
Student s1=new Student(1,"bhaskar");
Student s2=new Student(2,"nani");
OutputStream os=new FileOutputStream("student.txt");
ObjectOutput oo=new ObjectOutputStream(os);
System.out.println("Serialization process started");
oo.writeObject(s1);
oo.writeObject(s2);
oo.writeObject(new Count());
oo.close();
System.out.println("Serialization process completed");
```

Deserialization

```
InputStream is=new FileInputStream("student.txt");
ObjectInput oi=new ObjectInputStream(is);
System.out.println("DeSerialization process has started");
Object obj;
while(!((obj =  oi.readObject()) instanceof Count)){
     System.out.println(obj);
}
oi.close();
```

How to override writeExternal and readExternal in Externalizable ?

```
@Override
public void writeExternal(ObjectOutput out) throws IOException {
    // TODO Auto-generated method stub
    out.writeInt(id);
    out.writeObject(name);
}
@Override
public void readExternal(ObjectInput in) throws IOException,
        ClassNotFoundException {
    // TODO Auto-generated method stub
    this.id=in.readInt();
    this.name=(String)in.readObject();

}
```

- **In Externalization process, No argument constructor is mandatory otherwise it will throw exception.**

What is SerialVersionUID ?

- SerialVersionUID is used for version control of object.
- If we don't define, any modification is made in class, then we wont able to deserialize our class because SerialVersionUID is generate by java compiler for modified class will be different from old serialized object.
- It will throw InvalidClassException(because of SerialVersionUID mismatch).

What is Runtime Class ?

- Runtime Class is a singleton class it is used to get Runtime Information and to access system resources.

```
public static void main(String[] args) {
    Runtime r=Runtime.getRuntime();
    System.out.println("Total memory "+r.totalMemory());
    System.out.println("Free Memory "+r.freeMemory());
    System.out.println("Max Memory "+r.maxMemory());

    for(int i=0;i<10000;i++)
    {
        new MemoryTest();
    }

    System.out.println("After creating 10000 instance, Free Memory:
    System.gc();
    System.out.println("After gc(), Free Memory: "+r.freeMemory());
```

Enum Class

What is Enum ?

- The main objective of enum is to define our data types(Enumerated Data types)
- Enums are used when we know all possible values at compile time.
- Enum is used to define group of named constants.
- By Default enum constants are static final and introduced in 1.5.

Points:

- Enums are implemented by using class concept.
- Every enum constant can be represents an object of type enum.
- Every enum constant is always public, static and final.
- We can declare main() in the enum itself.
- All Enums implicitly extend java.lang.Enum class, so inheritance is not applicable for enum class.

What are the methods available in Enum class ?

- **values() method** can be used to return all values present inside enum.
- Order is important in Enums. By using **ordinal() method**, each enum constant index can be found, just like array index.
- **valueOf() method** returns the enum constant of the specified string value, if exists.

How to declare enum outside class ?

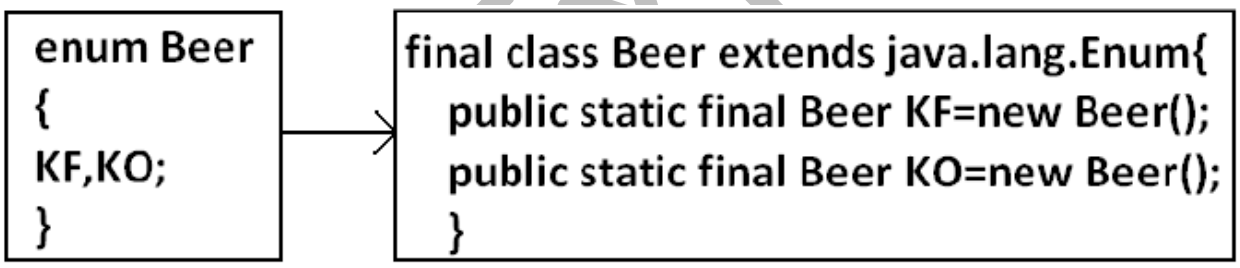
```
enum Color
{
    RED, GREEN, BLUE;
}

public class Test
{
    // Driver method
    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

How to declare enum inside class ?

```
public class Test
{
    enum Color
    {
        RED, GREEN, BLUE;
    }

    // Driver method
    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

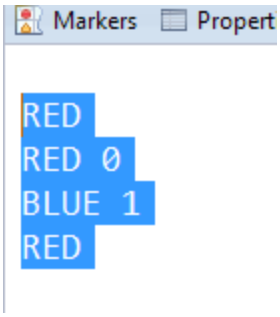
- First line contain only constants ,we cannot define methods, variables and constructors.

Internal Implementation:



Write a program all methods in enum class ?

```
Color a=Color.BLUE;
System.out.println(Color.RED);

Color[] colors=Color.values();
for(Color color:colors)
{
    System.out.println(color+" "+color.ordinal());
}
System.out.println(Color.valueOf("RED"));
```



What is the Enum Set in Java ?

- Enum set is a implementation class of Set interface.
- All elements of Enum Set instance must be of single enum type.

```
enum Days
{
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY;
}
public class EnumSetDemo {

    public static void main(String[] args) {
        EnumSet<Days> set1,set2,set3,set4;
        set1=EnumSet.of(Days.SUNDAY,Days.MONDAY);
        set2=EnumSet.complementOf(set1);
        set3=EnumSet.allOf(Days.class);
        set4=EnumSet.range(Days.MONDAY, Days.FRIDAY);
        System.out.println("of="+set1);
        System.out.println("complementof"+set2);
        System.out.println("allOf"+set3);
        System.out.println("range"+set4);
    }
}
```

```
of=[SUNDAY, MONDAY]
complementof[TUESDAY, WEDNESDAY, THURSDAY, FRIDAY]
allOf[SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRID
range[MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY]
```

What is Enum Map in Java ?

- EnumMap is the implementation class of Map interface.
- It does not allow null key and allow multiple null values.
- It maintains natural order of keys which they defined.
- It is high performance map implementation, must faster than HashMap.

```
public class EnumMapDemo {

    enum Color
    {
        RED,BLUE,GREEN
    }
    public static void main(String[] args) {
        EnumMap<Color,String> enumMap=new EnumMap<Color,String>(Color.class);
        enumMap.put(Color.RED, "Red Color");
        enumMap.put(Color.BLUE, "Red Color");
        enumMap.put(Color.GREEN, "Red Color");
        System.out.println("Size of Enum Map is "+enumMap.size());
        System.out.println(enumMap);
    }

}
```

I18N

What is I18N?

- The process of designing web applications in such a way that which provides support for various countries and various languages and various currencies automatically without performing any change in the application is called Internationalization
- We can implement i18N by using these classes
    - Locale
    - Number Format
    - Date Format
- **Locale**
    - Locale object represents a geographic location (country or language or both).
    - We can create Locale object to represent India.
    - We can create Locale Object to represent English language.
- Local is a final class and it is direct child class of Object and available in util package.

```
Locale l=Locale.getDefault();
//Locale l=new Locale("fr","fr");
System.out.println(l.getCountry());
System.out.println(l.getDisplayCountry());
System.out.println(l.getLanguage());
System.out.println(l.getDisplayLanguage());
System.out.println(l.getISO3Language());
System.out.println(l.getISO3Country());
```

- **Number Format**:
    - Various locations follow various styles to represent a java number
    - Ex:double d=”123456.789
        - In:1,23,456.789
        - Us:123,456.789
    - We can use Number Format class to format a java number according to a particular locale
    - Number Format it is available in java.text package and it is abstract class.

```
static void printNumber(Locale l)
{
    double d=123.456;
    NumberFormat format=NumberFormat.getNumberInstance(l);
    String number=format.format(d);
    System.out.println(number+" "+l.getDisplayCountry());
}
public static void main(String[] args) {
    printNumber(Locale.JAPAN);
    printNumber(Locale.CHINESE);
    printNumber(Locale.FRANCE);
}
```

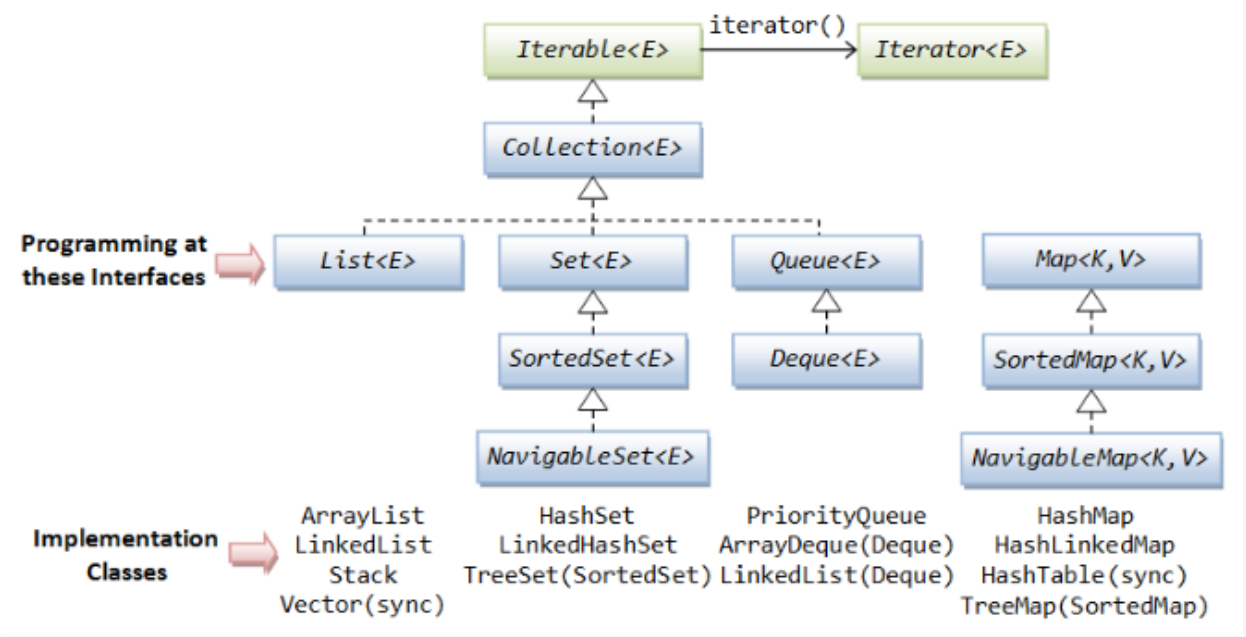
```
123.456 Japan
123.456
123,456 France
```

Date Format:

```
static void printDate(Locale l)
{
    DateFormat d=DateFormat.getDateInstance(DateFormat.DEFAULT,l);
    Date curr=new Date();
    String date=d.format(curr);
    System.out.println(date+" "+l.getDisplayCountry());
}
public static void main(String[] args) {
    printDate(Locale.UK);
    printDate(Locale.CHINA);
    printDate(Locale.GERMAN);
    printDate(Locale.JAPAN);
     printDate(Locale.FRANCE);
}
```

```
24-May-2017 United
2017-5-24 China
24.05.2017
2017/05/24 Japan
24 mai 2017 France
```

Collections



What is collection Framework ?List out some benefits of Collections Framework ?

- **Collection Framework** contain many classes and interfaces to collect and organize group of objects.
    - All the operations we can perform using collection framework like sorting, searching and manipulation etc.
    - A collection is group of individual objects as a single unit.
- **Benefits:**
    - Reduced Development effort
    - Code quality is enhanced
    - Reusability
    - No need to write code

What is the benefit of Generics in Collection framework ?

- Before any collection could store any type of object so there is a chance of incompatible types in collection.
- Java 1.5 generics are introduced, all collection classes and interfaces are heavily used generics.
- Generics provide type safety at compilation time only.
- It avoids classCastException at runtime because If u use generics you will get the error at compilation.

What are the interfaces available in Collection Framework?

- List
- Set
- Queue
- Deque
- SortedSet
- NavigableSet
- SortedMap
- NavigableMap

Why Map interface doesn't extend Collection interface ?

- Map and its implementations are part of collection framework but Collection interface does not contain Map
- Because they are incompatible, Collection has a method add(Object).Map can not have such method because it needs key-value pair,keySet and valueSet .
- So Collection does not have such views. Due to this Collection interface was not used in Map interface, it was build in separate hierarchy.

What is difference between fail-fast and fail safe ?

Fail-fast	**Fail-safe**
1)By the design all collection classes in java.util package are fail fast	1)Collection classes in java.util.Concurrent are fail safe.(ConcurrentHashMap and CopyOnWriteArrayList)
2)Fail-fast throw ConcurrentModificationException	2)It never throw ConcurrentModificationException

Which collection classes provide random access of it's elements?

- ArrayList
- Vector
- HashTable

Which collection classes are thread safe ?

- Vector
- Stack
- HashTable
- Properties

What are concurrent Collection classes ?

- CopyOnWriteArrayList
- CopyOnWriteArraySet
- ConcurrentHashMap
- ConcurrentSkipListSet

Difference between collection and collections

Collection	Collections
1.collection is the root interface in the hierarchy of java Collection framework.	1.Collections is a class
2.collection interface provide normal functionality of data structure to List, Set and Queue	2. Collections is a utility class available in java.util.package. It consists of only static methods which are used to operate on objects of type collection. Reverse, min, max, sort, shuffle, synchronized Collection

Difference between Iterator and ListIterator:

Iterator	ListIterator
1.Iterator traverse the elements in forward direction only.	1.ListIterator can traverse the elements in forward and reverse direction
2.Iterator cursor can be used in List, Set, Queue	2.ListIterator cursor can be used in List only
3.Iterator can perform read, remove operations	3.ListIterator can perform read, remove and update operations
4) hasNext, next methods are available and it is implemented using Iterator design pattern	4)ListIterator inherit from iterator interface with extra functionalities like hasPrevious(), previous(), nextIndex(),previousIndex,add and set methods are available.

How many types of exceptions will throw when use iterator.remove ?

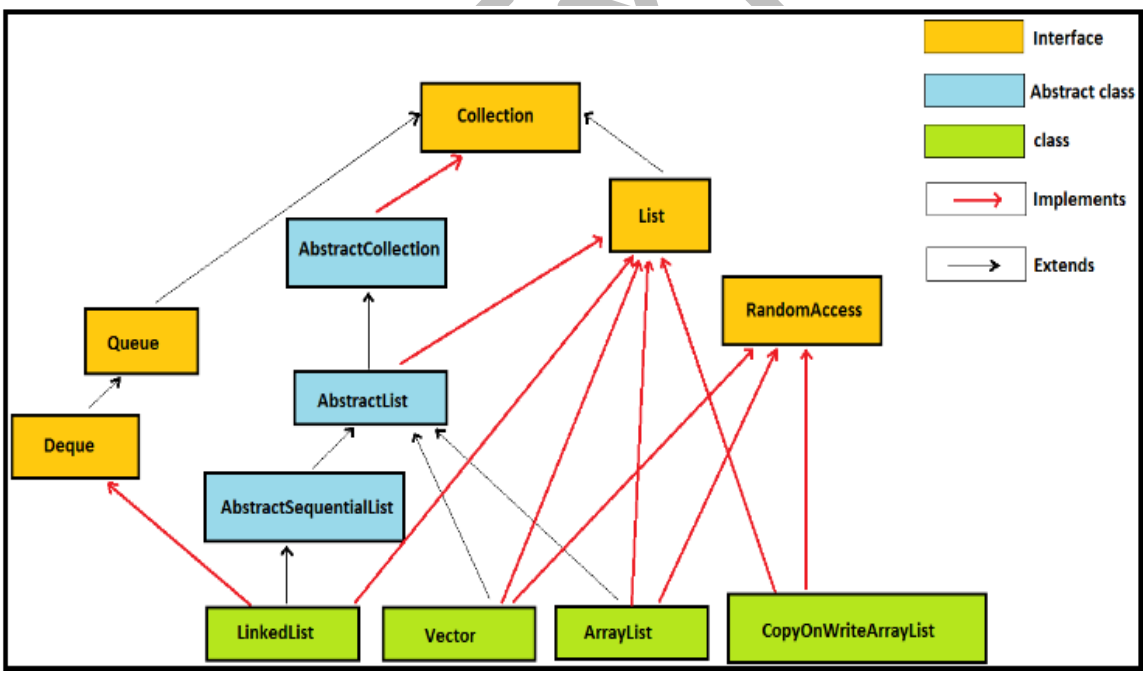
- **UnsupportedOperationException**
  - If remove operation is not supported then it will throw this exception
- **IllegalStateException**

```
while(it.hasNext())
{
    it.remove();
    System.out.print(it.next()+",");
}
```

Difference between Iterator and Enumeration:

Iterator	Enumeration
1.Iterator can traverse legacy and non-legacy classes	1.Enumeration can traverse only on legacy classes
2.Iterator is fail fast	2.Enumeration is fail safe
3.Iterator is slower than Enumeration	3.Enumeration is fast than Iterator
4.We can perform read, remove operations	4.We can perform only read operation
5) hasNext(),next	5) hasMoreElements(), nextElement()

List Hierarchy:



How to remove duplicate from ArrayList ?

```
for(int i=0;i<al.size();i++)
{
    for(int j=1;j<al.size();j++)
    {
        if(al.get(i)==al.get(j))
        {
            al.remove(j);
        }
    }
}
```

What is the difference between List<?> and List<Object> in Java ?

- List<?> ->we can store unknow type.->List<Integer>,List<String>
- List<Object> ->we can store list of any type including string and Integer also.

How size of ArrayList increases automatically? Could you please write a code

```
public int ensureCapacity(){
    int newcapacity=(elements.length*3)/2+1;
    elements=Arrays.copyOf(elements, newcapacity);
    return newcapacity;
}
```

How to convert ArrayList to array and vice versa?

```
int[] arr={10,20,20};
//1->ArrayList al=new ArrayList(Arrays.asList(arr));
//2->al.toArray();
```

How can we sort list of objects ?

- To sort array of objects we will use Arrays.sort() method.
- If we need to sort collection of object we will use Collections.sort().

What is the default capacity of ArrayList and Vector and LinkedList ?

- 10 for ArrayList and Vector and LinkedList is 0;

Difference between for-Each and Iterator In Java ?

- In for-each loop we can't modify the collection ,on the other hand we can modify the collection using iterator i.e. we can remove data.

How to traverse elements by using lambda expressions ?

```
ArrayList al=new ArrayList();
al.add(1);
al.add("a");
al.add("a");
al.add("b");
al.add("b");
al.add("b");
al.add("b");

al.forEach(e  -> System.out.println(e) );
```

Difference between List and Set:

List	Set
1)List contains ordered collection of data	1)Set contain unordered collection of data
2)Insertion order is preserved	2)Insertion order is not preserved.
3)Duplicate objects are allowed	3)Duplicate objects are not allowed
4)List allow any number of null values	4)Set allow null value atmost once
5)List implements ArrayList, LinkedList and Vector	5)Set implements HashSet, TreeSet, LinkedHashSet
6)List contain one legacy class Vector	6)Set does not contain any legacy classes
7)In List Enumeration, Iterator and ListIterator used to traverse the elements.	7)In set only Iterator is used to traverse the elements.

Difference between Array List and Linked List:

ArrayList	LinkedList
1)ArrayList internally uses **dynamic** array	1)LinkedList internally uses **double** linked list
2)Manipulation with array list is **slow** because it internally uses array. If any element is removed from the array, all bits are shifted in memory.	2)Manipulation is **faster** than arraylist because it internally uses double linked list and no bit shifting required in memory.
3)Arraylist is better for **storing** and **accessing** elements	3)Linked list is better for **manipulating** data
4)Arraylist is act as a **list** because it implements List interface.	4)Linked list act as a **List** and **queue** both implements List and Deque interfaces.
5)Initial Capacity of array list is 10	5)Initial Capacity is zero
6)For fast retrieving array list implements Random-access interface(marker interface)	6)LinkedList does not implement Random-access
7)ArrayList is index based structure	7)LinkedList is data structure contain group of nodes which together represent a sequence
8)ArrayList extends Abstract List which provides implementation to list interface	8)LinkedList extends AbstractSequentialList

What is the difference between the size and capacity of a Vector?

- Number of elements actually stored in the vector, while capacity is the maximum number of elements it can store at a given instance of time.

How do you sort any ArrayList of user defined objects?

- Collections.sort(List, Comparator)

Difference between Array List and Vector:

ArrayList	Vector
1)ArrayList is introduced in 1.2 version	1)Vector is introduced in 1.0 version
2)It is not a legacy class	2)It is legacy class
3)Arraylist methods are non-synchronized	3)Vector methods are synchronized
4)For arraylist we are using iterator and list iterator cursors and fail fast	4)For vector we are using Enumeration cursor and it is fail safe
5)Arraylist increments 50% size of array if number of element exceeds from its capacity	5)Vector increments 100% of size it will double the array size.
6)Arraylist is fast than vector because it is not synchronized	6)Vector is slow because it is synchronized

How to get synchronized version of ArrayList ?

- By default array list is non-synchronized but we can get synchronized version of ArrayList by using Collection class method is synchronized List() method.
    - Collections.synchrnonizedList(arraylist);
    - Collections.synchrnonizedSet(Set s);
    - Collections.synchrnonizedMap(Map m);

How to make collection read only?

- Collections.unmodifiableList(arraylist)

What is UnsupportedOperationException?

- If u try to add any element to collection read only then it will throw this exception.

Difference between poll() and remove() method of Queue interface ?

- Poll() – retrieve the element and remove the element of head
- Remove()—it will remove last element of the queue.

What is the default capacity of LinkedList?

- LinkedList does not need initial value ,as because it does not depend on boundary values.

How to retrieve elements for range ?

- Al.subList(2,7);

If element is not found what u will get output ?

- Al.indexOf(“Hello”) ->-1

How to copy elements from vector to array ?

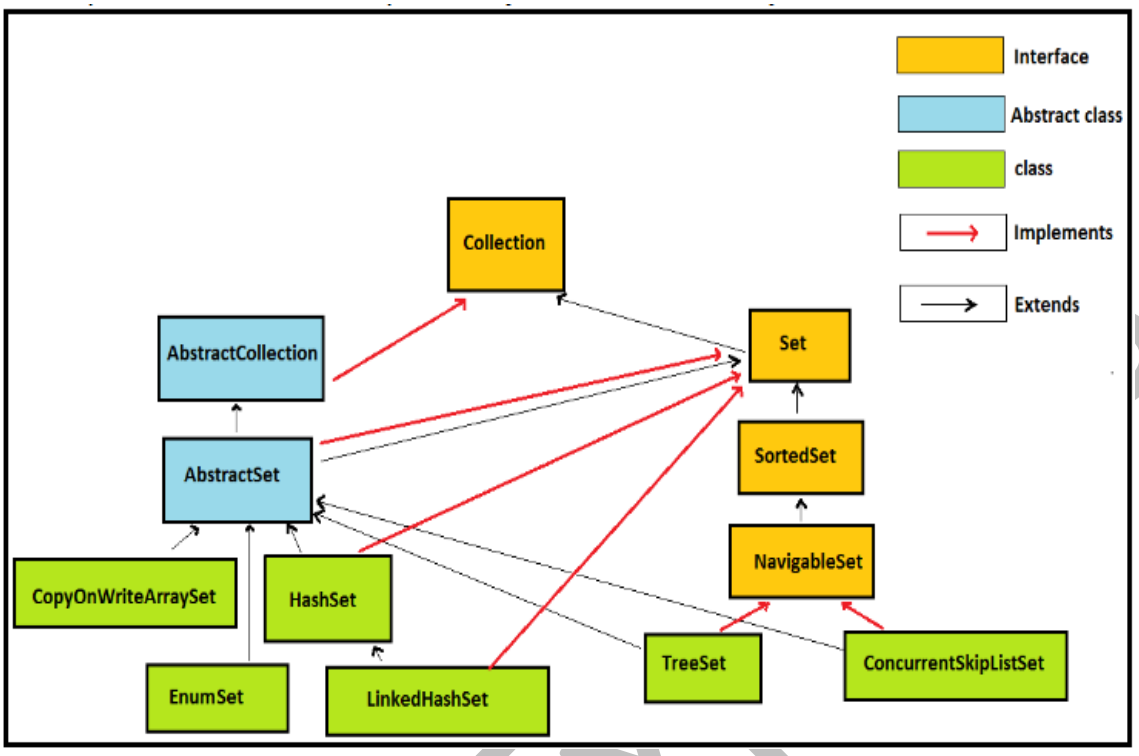
- v.copyInto(Array name);

What is the output for below program ?

```
Stack s=new Stack();
s.pop();
System.out.println(s);
```

java.util.EmptyStackException

Set Hierarchy:



Difference between Hashset, LinkedHashSet and Tree Set ?

Property	HashSet	LinkedHashSet	TreeSet
Insertion	Does not maintain	Maintained	Natural Sorting order
Null Elements	One null Value	One null Value	No null key
Data Structure	HashMap	LinkedHashMap	TreeMap
Version	1.2	1.4	1.2
Implements	Set	Set	SortedSet, NavigableSet

In which collection object heterogenous objects are not allowed?

- TreeSet

```
TreeSet ts = new TreeSet();

ts.add("sree");
ts.add(1);// Integer.valueOf(1).compareTo((Integer)("sree"));
```

How to override hashCode() and equals() method in Hashset using String parameter?

```
public int hashCode()
{
    int result=1;
    result=31*result+id;
    result=31*result+name.hashCode();
    return result;
}

public boolean equals(Object obj)
{
    if(obj==null)
        return false;
    if(obj==this)
        return true;
    if(!(obj instanceof Employee))
        return false;
    Employee e=(Employee)obj;
    return this.name.equals(name) && this.id==id;
}
```

What is the importance of hashCode() and equals() methods ?

- **hashCode**() method is used to determine ,where specified key will be stored. Since different keys may produce same hashvalue .
- **equals** method is used to determine whether specified key actually exists in the collection or not it is very cruicial in the collection framework.

What is SortedSet interface ?

- SortedtedSet interface is extend Set interface, it must implement Comprable interface like TreeSet.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    SortedSet<Integer> sites = new TreeSet<>();
    sites.add(1);
    sites.add(2);
    sites.add(3);
    sites.add(4);
    sites.add(5);
    System.out.println("Sorted Set: " + sites);
    System.out.println("First: " + sites.first());
    System.out.println("Last: " + sites.last());
    SortedSet<Integer> beforeQuiz = sites.headSet(3);
    System.out.println("headSet(3) "+beforeQuiz);
    SortedSet<Integer> afterCode = sites.tailSet(3);
    System.out.println("tailSet 3 "+afterCode);
    SortedSet<Integer> betweenCodeAndQuiz =
                                sites.subSet(3,7);
    System.out.println("subset (3,7) "+betweenCodeAndQuiz);
```

```
Sorted Set: [1, 2, 3, 4, 5]
First: 1
Last: 5
headSet(3) [1, 2]
tailSet 3 [3, 4, 5]
subset (3,7) [3, 4, 5]
```

What is NavigableSet ?

- NavigableSet interface is extend Set interface, it must implement Comprable interface like TreeSet.
- **They provided navigation methods.**

What is the difference between lower,floor,higher,ceiling in NavigableSet ?

```
NavigableSet<Integer> ns = new TreeSet<>();
ns.add(0);
ns.add(1);
ns.add(2);
ns.add(3);
ns.add(4);
ns.add(5);
ns.add(6);
NavigableSet<Integer> reverseNs = ns.descendingSet();
System.out.println("Reverse order: " + reverseNs);
NavigableSet<Integer> threeOrMore = ns.tailSet(3,true);
System.out.println("3 or  more:  " + threeOrMore);
System.out.println("lower(3): " + ns.lower(3));
System.out.println("floor(3): " + ns.floor(3));
System.out.println("higher(3): " + ns.higher(3));
System.out.println("ceiling(3): " + ns.ceiling(3));
```

OUTPUT

```
Reverse order: [6, 5, 4, 3, 2, 1, 0]
3 or  more:  [3, 4, 5, 6]
lower(3): 2
floor(3): 3
higher(3): 4
ceiling(3): 3
pollFirst(): 0
Navigable Set:  [1, 2, 3, 4, 5, 6]
pollLast(): 6
Navigable Set:  [1, 2, 3, 4, 5]
```

Comparable and Comparator Interfaces

Difference between Comparable and Comparator:

Comparable	Comparator
1.It is available in java.lang	1.It is available in java.util
2.It is used for one sort of sequence	2.It is used for multiple sort of sequence
3.It provide only one method compareTo()	3.It provide one method compare()
4.If we implement comparable, actual class is modified	4.Actual class is not modified

Comparable Interface :

```
@Override
public int compareTo(Country country) {

    return this.countryName.
            compareTo(country.getCountryName());
}

 Collections.sort(list);
```

```
@Override
public int compareTo(Country country) {
    // TODO Auto-generated method stub
    return (this.countryId<country.countryId)?-1:
        (this.countryId>country.countryId)?1:0;
}
```

Comparator Interface:

```
public class CountrySortByIdComparator
            implements Comparator<Country>{

    @Override
    public int compare(Country c1, Country c2) {
        return c1.getCountryId()-c2.getCountryId();
    }

}
```

```
Collections.sort(list, new CountrySortByIdComparator());
```

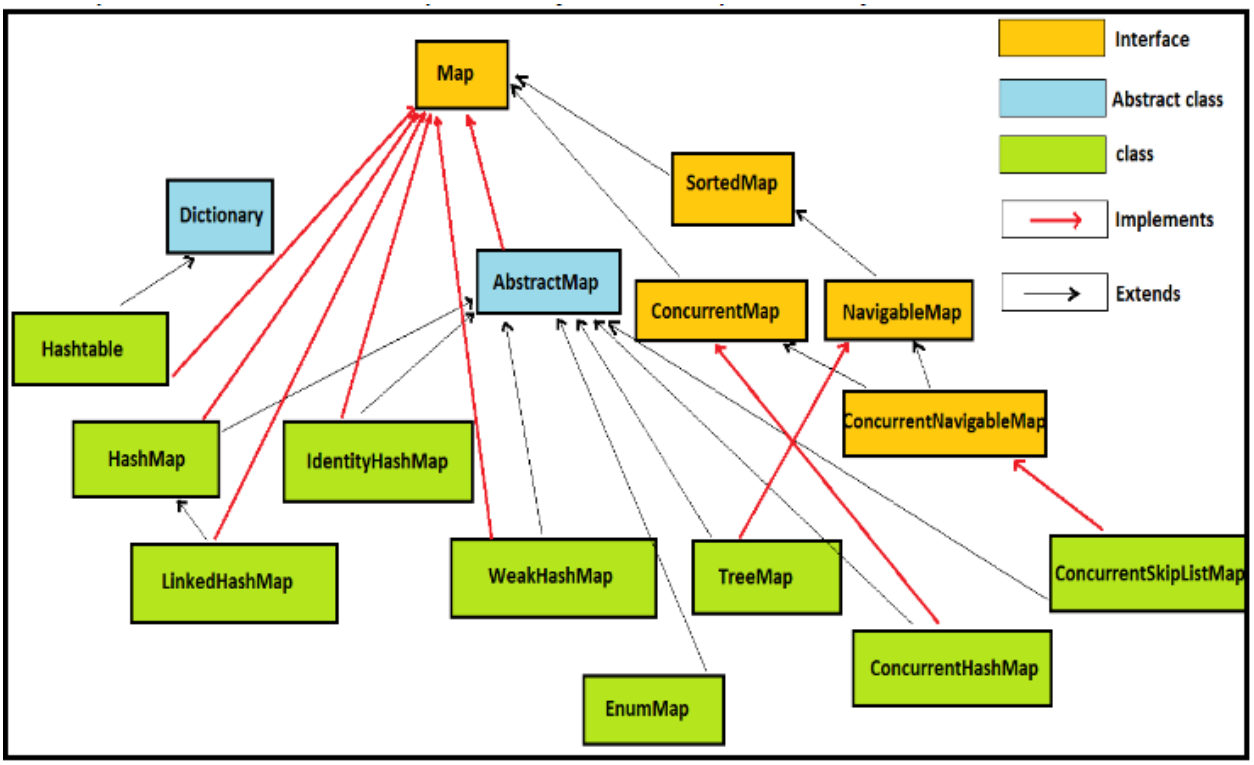
By using Anonymous Class

```
Collections.sort(list,new Comparator<Country>(){

    @Override
    public int compare(Country o1, Country o2) {
        // TODO Auto-generated method stub
        return o1.getCountryName().compareTo(o2.getCountryName());
    }


});
```

How to perform comparator using lambda expressions ?

```
List<Student> list=new ArrayList<Student>();
list.add(new Student("hcl",501));
list.add(new Student("tcs",502));
list.add(new Student("wipro",503));
list.add(new Student("ibm",504));

Collections.sort(list,(s1,s2)->
        s1.getName().compareTo(s2.getName()));
list.forEach((s)->System.out.println(s));
list.forEach(System.out::println);
```

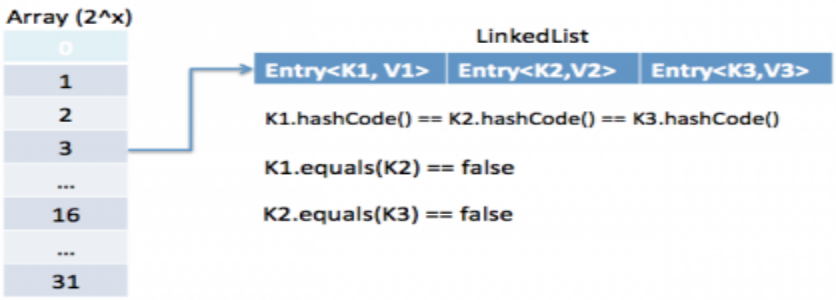
Map Hierarchy



How HashMap works in java ?

- HashMap stores key-value pair in Map & Map.Entry is static nested class implementation.
- HashMap works on hashing algorithm and uses hashCode() and equals() method in put and get methods.
- HashMap uses Key hashCode() is used to find out the index to store the key-value pair.
- The Map.Entry is stored in the LinkedList, so if there are already existing entry, it uses equals() method to check if the passed key already exists.
- If entry already exists then it overwrites the value else it creates a new entry and store this key-value Entry.
- When we call get method by passing Key :Again it uses the hashCode() to find the index in the array and then use equals() method to find the correct

```
public boolean equals(Object obj) {
  //null instanceof Object will always return false
    CustomerID c=(CustomerID)obj;
  if (!(obj instanceof CustomerID))
    return false;
  if (obj == this)
    return true;
  return  this.crmID == c.crmID &&
          this.nameSpace == c.nameSpace;
}

public int hashCode() {
  int result = 0;
  result = (int)(crmID/12) + nameSpace;
  return result;
}
```



Difference between HashMap and ConcurrentHashMap ?

HashMap	ConcurrentHashMap
1)HashMap is not synchronized	1)ConcurrentHashMap is synchronized
2)Relatively performance is high because threads are not required to wait operate in HashMap.	2)Relatively performance is low because threads are required to wait operate in ConcurrentHashMap.
2)HashMap can be synchronized by using synchronized Map	2)ConcurrentHashMap is by default synchronized
3)It allow only one null key	3)It does not allow any null keys
4)HashMap is faster than ConcurrentHashMap	4)It is slow it is used in single thread environment.
5)It is available in util package	6)It is available in Concurrent package
6)1.2 version	7)1.5 version

What are different Collection views provided by Map interface?

- Set<K> keySet()
- Collection<V> values()
- Set<Map.Entry<K,V>> entrySet()

Explain about ConcurrentHashMap ?

1)Underlying data structure is HashTable

2)It allows Concurrent read and thread safe update operations.

3)To perform read operation they wont require any lock but to perform update operation it requires a lock but it is lock of only a particular part of map(Segment lock/Bucket Level Lock).

4)Concurrent update achieved by internally dividing into smaller parts which is defined as Concurrency Level.

5)The default initial capacity of Concurrency level is 16.

6)We can perform any number of read operations but 16 update operations at a time by default.

7)Null is not allowed for keys and values.

8)while one thread is iterating, other thread can perform update operation it does not throw any exception.

Difference between HashMap and TreeMap:

HashMap	**TreeMap**
1.HashMap maintains no order	1.TreeMap maintains ascending order
2.Underlying data structure is HashTable	2.Underlying data structure is Redblacktree
3.Peformance of HashMap is higher than TreeMap	3.TreeMap performance is slow compared to HashMap because minimum execution of two method calls to create a tree structure and to print natural order

Differences between HashMap,HashTable,LinkedHashMap and TreeMap

Property	HashMap	HashTable	LinkedHashMap	TreeMap
Insertion Order	No order	No Order	Order preserved	Natural Sorting
Performance	Fast compare to HashTable	Slow when compare to HashMap	It must be used when maintain inserting order	It is used when we need sorting
Null Keys	One null key	No null key	No Null key	No Null key
Implementation	Map	Map	Map	NavigableMap, SortedMap
Version	1.2	1.0	1.4	1.2
Complexity of put and get methods	O(1)	O(1)	O(1)	O(log(n))

What is load factor ?

- default capacity of HashMap is 16 and load factory is 0.75%f.
- Value =HashMap(16) * 0.75 =12
- At 12^{th} key HashMap size will increase now value is 16+12=32.

What is Properties in Java ?

- Properties object contain key and value pair both as string.
- Java.util. Properties is the sub class of HashTable.
- It can be used to get the property value based on the property key.
- Properties class provide methods to get data from properties file and store data into properties file.
- If any information is changed from the properties files, recompilation is not required for java class.
- It is used to store information which is to be changed frequently.

```
public static void main(String[] args)throws Exception{
    FileReader reader=new FileReader("db.properties");

    Properties p=new Properties();
    p.load(reader);

    System.out.println(p.getProperty("user"));
    System.out.println(p.getProperty("password"));
}
}
```

What is the need of concurrent collections ?

- Existing collection classes are not thread safe i.e data inconsistency problems
- Already existing collection class are thread safe but one thread is allowed to operate .
    - Vector,Hashtable,synchronizedMap,synchronizedSet,synchronizedList
- While one thread is iterating other thread cannot modify the underlying the collection object, if modify it will throw ConcurrentModificationException.

Difference between Traditional and Concurrent collections ?

- Concurrent collections are always thread safe.
- Performance is more because of locking mechanisms.
- While one thread interacting with collection other thread can modify the collection in same manner.
- Important classes are
    - ConcurrentHashMap
    - CopyOnWriteArrayList
    - CopyOnWriteArraySet

What are the method available in ConcurrentHashMap?

- putIfAbsent (Object key, Object Value)
    - If key is present, it wont add otherwise it will add key
- Remove(Object key, Value value)
    - If key and value is matched then only key will remove from map.
- Replace(Object key, Object oldValue,Object newValue)
    - If old value and key is same then only value is replaced.

What is CopyOnWriteArrayList?

- CopyOnWriteArrayList is thread safe version of ArrayList.
- For every write operation a separate clone copy will be created.
- Later original object and clone copy will be sync by JVM.
- It is suitable for when more number of read operations and less number of update operations.
- While iterating elements if we try to remove elements using iterator it will throw Unsupported Operation Exception.
- Methods:
    - Boolean addIfAbsent(Object o)
    - Boolean addAllAbsent(Object o)

```
CopyOnWriteArrayList c1=new CopyOnWriteArrayList();
c1.add("A");
c1.add("C");
Iterator itr=c1.iterator();
while(itr.hasNext())
{
    String value=(String)itr.next();
    if(value.equals("C"))
        itr.remove();
}
System.out.println(c1);
```

```
Exception in thread "main"
java.lang.UnsupportedOperationException
```

Difference between ArrayList and CopyOnWriteArrayList ?

ArrayList	CopyOnWriteArrayList
1)1.2version	1)1.5 version
2)Fail fast	2)Fail safe
3)It is not thread safe	3)It is thread safe but every update operation will be performed on separate cloned copy
4)we can perform remove operation while using iterator	4)we cannot perform remove operation while iterating.

What is CopyOnWriteArraySet ?

- It is internally using CopyOnWriteArrayList.
- Insertion order is preserved.

Java 8 Features

Why use lambda expressions ?

- To provide the implementation of Functional interface
- Less coding.
- It is the replacement of java inner anonymous class.
- It is treated as function so compiler does not create .class file.

What is functional interface ?

- An interface which has only one abstract method is called functional interface.

How to call Runnable interface by using lambda expressions ?

```
public static void main(String[] args) {
    Runnable r=()->
            System.out.println("run method is executed");
    Thread t=new Thread(r);
    t.start();
}
```

How to pass zero argument in interface using lambda expressions ?

```
@FunctionalInterface
public interface MyInterface1 {
    void m1();
}
```

```
MyInterface1 myinterface=()->
        System.out.println("zero argument");
myinterface.m1();
```

How to pass one argument in interface using lambda expressions ?

```
@FunctionalInterface
public interface MyInterface2 {
    void m1(String msg);
}
```

```
MyInterface2 myInterface2=(name)->
        System.out.println(name);
myInterface2.m1("bhaskar");
```

How to handle return type in interface using lambda expressions ?

```
public interface MyInterface3 {
    int add(int a,int b);
}
```

```
MyInterface3 myInterface3=(a,b)->a+b;
System.out.println(myInterface3.add(10, 20));
```

How to use forEach loop in ArrayList ?

```
List<Student> list=new ArrayList<Student>();
list.add(new Student("hcl",501));
list.add(new Student("tcs",502));
list.add(new Student("wipro",503));
list.add(new Student("ibm",504));
list.forEach((student)->System.out.println(student));
        /*Student [name=hcl, id=501]
        Student [name=tcs, id=502]
        Student [name=wipro, id=503]
        Student [name=ibm, id=504]*/
```

How to use forEach loop in HashMap ?

```
Map<String,Student> m=new HashMap<String,Student>();
m.put("a", new Student("hcl",121));
m.put("b", new Student("tcs",122));
m.put("c", new Student("ibm",123));
m.put("d", new Student("wipro",124));

m.forEach((k,v)->System.out.println(k+" "+v));
```

How to print Array elements using method reference ?

```
List<Student> list=new ArrayList<Student>();
list.add(new Student("hcl",501));
list.add(new Student("tcs",502));
list.add(new Student("wipro",503));
list.add(new Student("ibm",504));
//list.forEach((student)->System.out.println(student));
list.forEach(System.out::println);
```

How to perform comparator using lambda expressions ?

```
List<Student> list=new ArrayList<Student>();
list.add(new Student("hcl",501));
list.add(new Student("tcs",502));
list.add(new Student("wipro",503));
list.add(new Student("ibm",504));

Collections.sort(list,(s1,s2)->
        s1.getName().compareTo(s2.getName()));
list.forEach((s)->System.out.println(s));
list.forEach(System.out::println);
```

How to call method reference in Thread ?

```
    ExecutorService execute=Executors.newSingleThreadExecutor();

    //Runnable r=()->System.out.println("Run method");
    Runnable r1=MethodInference::myRun;
    execute.submit(r1);

}
public static void myRun()
{
    System.out.println("Run method executed");
}
```