What is synchronization ?

Synchronization in Java is the capability to control the access of multiple threads to any shared resource.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

What is java synchronized method?

If you declare any method as synchronized, it is known as synchronized method.

Which lock used to Synchronized method?

Synchronized method is used to lock an object for any shared resource.

What is Synchronized block?

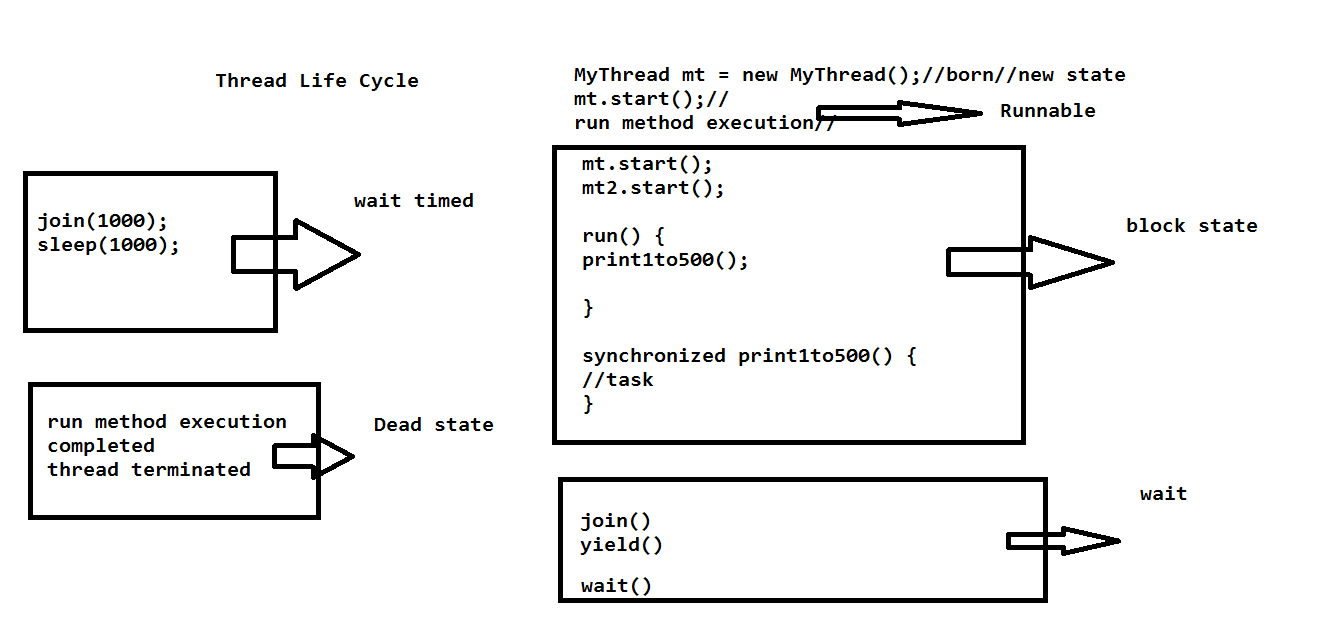
Synchronized block can be used to perform synchronization on any specific resource of the method.

Which lock used to Synchronized block?

Synchronized block is used to lock an object for any shared resource.

What is class lock?

If you make any static method as synchronized, the lock will be on the class.it known as class lock.



What is StringBuffer ?

Java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

What is StringBuilder?

Java StringBuilder class is used to create mutable (modifiable) String. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.

What is difference between String Buffer nd String Builder?

StringBuffer :

1. String Buffer is synchronized i.e. thread safe. It means two threads can't call the methods of String Buffer simultaneously
2. String Buffer is less efficient than String Builder.
3. String Buffer was introduced in Java 1.0

StringBuilder :

1. StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2. StringBuilder is more efficient than StringBuffer.
3. StringBuilder was introduced in Java 1.5

SERIALIZATION

What is serialization ?

If you want to store the object in a file we can go for a Serialization.

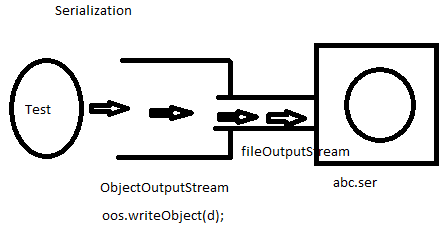
or

If you want to convert your object state in to the byte stream Than we can go for serialization.

or

If you want to transfor your object in to the network we can go for serialization.

Ex:



What is marker interface ?

Marker interface nothing but a it doesn’t contains any method but it contains some special behaviour.it tell to the compiler it as special bhhaviour.

Ex:

1.Cloneable: if you implements the cloneable interface so it will support the clone the object.

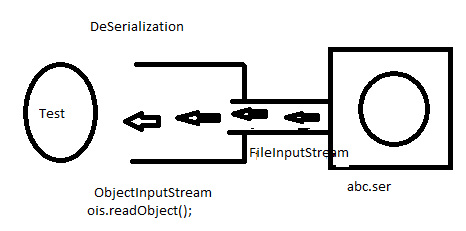
2.Serializable: If you use the serialization interface so it will support the serialization a object.

DeSerialization:

What is deserialization ?

The process of reading state of an object from a file is called Deserialization.

Ex:



What is Transient Keyword ?

Transient is the modifier applicable only for variable.

At time of serialization JVM ignores the original value of transient variable and save default value to the file.

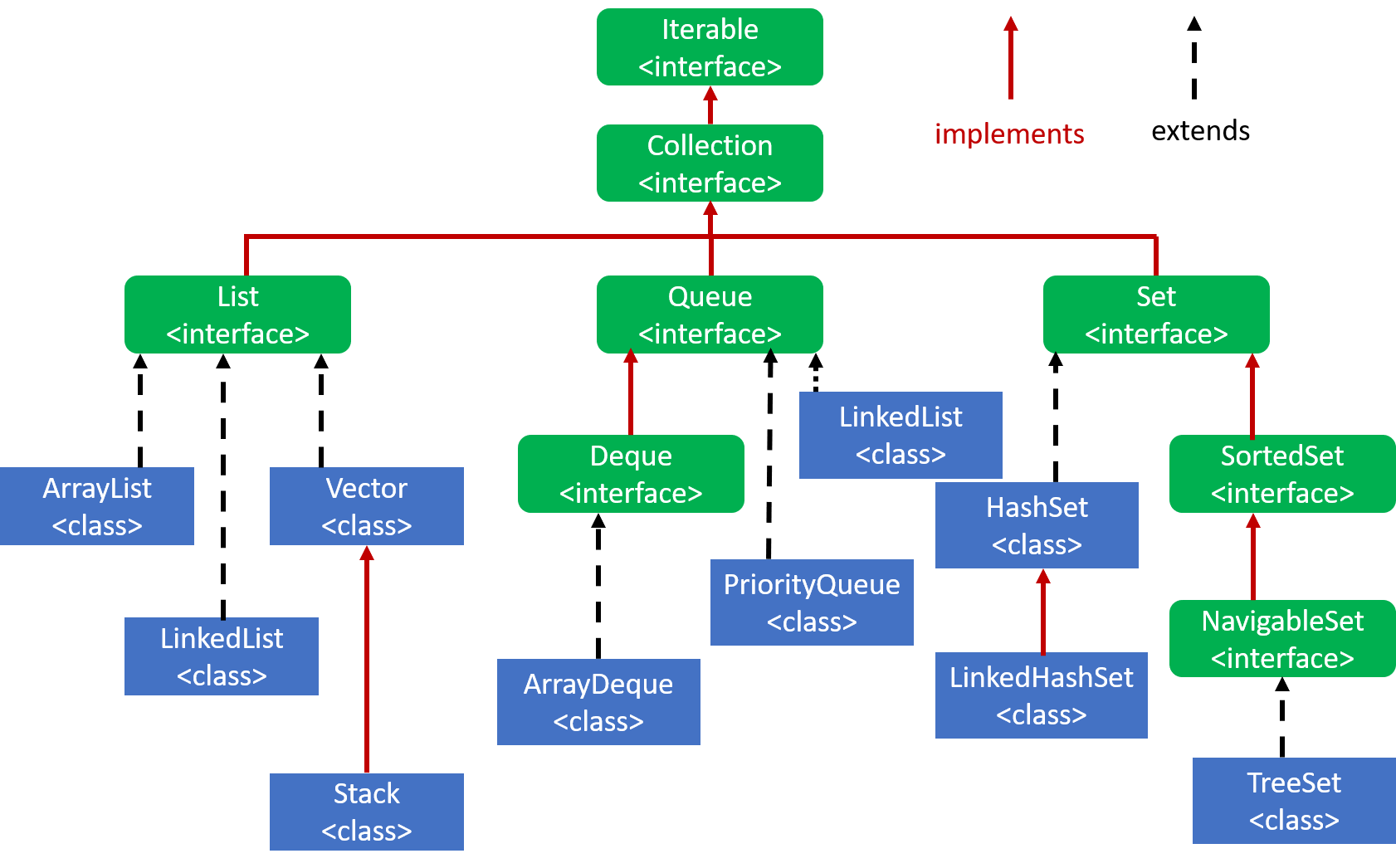
Collections

What is collections ?

If you want to store multiple objects in a single object we can go for a collections.

Or

If you want to store multiple objects in a single unit we can go for a collections.

****

What is List interface?

If you want the store the values and preserved the insertion order and allows the duplicates then we should go for list.

What is queue interface ?

If you want to follow the filo order we can go for a queue.

What is set interface ?

If you don’t want to allow the duplicates and don’t want to follow the preserved insertion order we can go for a set.

ArrayList:

The ArrayList in Java can have the duplicate elements also. It implements the List interface so we can use all the methods of List interface here. The ArrayList maintains the insertion order internally.

1.Java ArrayList class is non [synchronized](https://www.javatpoint.com/synchronization-in-java).

List list = new ArrayList();

list.add(“srinu”);

list.add(10);

list.add(25);

list.add(10);

list.add(‘M’);

System.out.println(list);//[srinu,10,25,10,M]

List.remove(2);

System.out.println(list);//[srinu,25,10,M]

What is the contract between equals and hashcode method ?

1. If two objects are equal, then they must have the same hash code.
2. If two objects have the same hash code, they may or may not be equal.

What is LinkedList ?

LinkedList class uses a doubly linked list to store the elements. It provides a linked-list data structure. It inherits the AbstractList class and implements List and Deque interfaces.

1.Java LinkedList class can contain duplicate elements.

2.Java LinkedList class maintains insertion order.

3.Java LinkedList class is non synchronized.

Vector:

1.The underlying data structure is resizable array or growable array.

2.Insertion order is preserved.

3.Duplicate objects are allowded.

4.Hetrogeneous objects are allowed.

5.null insertion is possible.

6.Every method present inside vector is synchronized and hence vector object is thread safe.

7.vector is the best choice if our frequent operation is retrieval.

8.vector is synchronized.

HashSet:

What is hashset ?

If you don’t want to allow the duplicates and don’t want to follow the preserved insertion order we can go for a hashset.

package collections;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

public class HashSetExamples {

public static void main(String[] args) {

Set<String> hashset = new HashSet<>();

System.out.println(hashset.add("srinu"));

System.out.println(hashset.add("srinu"));

System.out.println(hashset);

hashset.add("gopi");

hashset.add("anil");

hashset.add("venki");

System.out.println(hashset);

Iterator <String> itr = hashset.iterator();

while(itr.hasNext()) {

System.out.println(itr.next());

}

Set<Batch2> batch2 = new HashSet<>();

Batch2 b1 = new Batch2("sai","pavan","nari");

Batch2 b2 = new Batch2("sai","pavan","nari");

batch2.add(b1);

batch2.add(b2);

System.out.println(batch2);

Set<Integer>hashset1 = new HashSet<>();

hashset1.add(5);

hashset1.add(2);

hashset1.add(1);

hashset1.add(3);

hashset1.add(4);

hashset1.add(6);

System.out.println(hashset1);

}

}

LinkedHashSet:

What is LinkedHashset ?

If you don’t want to allow the duplicates and follow the preserved insertion order we can go for a LinkedHashSet.

**package** collections;

**import** java.util.ArrayList;

**import** java.util.HashSet;

**import** java.util.LinkedHashSet;

**import** java.util.List;

**import** java.util.Set;

**public** **class** LinkedHashSetExamples {

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

//Set<String>names = new LinkedHashSet<>();

List<String> names = **new** ArrayList<>();

names.add("srinu");

names.add("gopi");

names.add("anil");

names.add("venki");

names.add("nari");

names.add("srinu");

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

Set<String>uniqueNames = **new** HashSet<>(names);

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

Set<String>uniqueNames1 = **new** LinkedHashSet<>(names);

System.***out***.println(uniqueNames1);

Set<User> user = **new** LinkedHashSet<>();

User ur1 = **new** User(01,"srinu");

User ur2 = **new** User(02,"srinu");

User ur3 = **new** User(01,"srinu");

user.add(ur1);

user.add(ur2);

// user.add(ur3);

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

}

}

TreeSet:

What is TreeSet ?

If you don’t want to allow the duplicates and follow the natural order we can go for a TreeSet.

1. null insertion is not possible.

package collections;

import java.util.HashSet;

import java.util.LinkedHashSet;

import java.util.Set;

import java.util.TreeSet;

public class TreeSetExample {

public static void main(String[] args) {

//Set<String> names = new TreeSet<>();

Set<String> names = new HashSet<>();

//Set<String> names = new LinkedHashSet<>();

names.add("srinu");

names.add("anil");

names.add("venki");

names.add("gopi");

names.add("naresh");

names.add(null);

System.out.println(names);

Set<Integer> numbers = new TreeSet<>();

numbers.add(60);

numbers.add(50);

numbers.add(20);

numbers.add(40);

numbers.add(10);

numbers.add(30);

System.out.println(numbers);

Set<Customers> customers = new TreeSet<>();

Customers cst1 = new Customers(01,"srinu");

Customers cst2 = new Customers(02,"pavan");

Customers cst3 = new Customers(03,"gopi");

Customers cst4 = new Customers(04,"anil");

customers.add(cst1);

customers.add(cst2);

customers.add(cst3);

customers.add(cst4);

System.out.println(customers);

}

}

Map

What is map ?

If you want to store a values as a key value pairs then we can go for map.

1.hashmap

2.linkedhashmap

3.hash table

4.treemap

HashMap

JDBC

1. We have had to jar file into our application

2. Driver Register

3. Create connection

4. Write SQL queries

5. Get the Result

JAR = java archive

**package** com.srinu.jdbc;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** JDBCApp1 {

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

**try** {

Class.forName("com.mysql.jdbc.Driver");//driver register

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/batch2", "root", "root");

Statement stmt = con.createStatement();

stmt.executeUpdate("create table student (s\_id Integer, s\_name varchar(20),s\_age Integer)");

**int** count = stmt.executeUpdate("insert into student values(1,"+"'"+"srinu"+"'"+",25)");

**if**(count != 0) System.**out**.println("table created");

//driver register

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}

**DataUpdate and Delete**

**package** com.srinu.jdbc;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** JDBCapp2 {

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

**int** s\_id = 5;

**try** {

Class.*forName*("com.mysql.jdbc.Driver");// driver register

Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/batch2", "root", "root");

Statement stmt = con.createStatement();

// String updateQuary = "update student set s\_name ="+"'"+"venki"+"'"+"where s\_id ="+s\_id;

String deleteQuary = "delete from student where s\_id="+s\_id;

// System.out.println(updateQuary);

// int count = stmt.executeUpdate(updateQuary);

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

**int** count = stmt.executeUpdate(deleteQuary);

**if**(count !=0) System.***out***.println("Record deleted");

//if(count !=0) System.out.println("Record updated");

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}

**package** com.srinu.jdbc;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.util.Scanner;

**public** **class** JDBCApp3 {

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

Scanner scr = **new** Scanner(System.***in***);

System.***out***.println("Enter Id");

**int** s\_id = scr.nextInt();

System.***out***.println("Enter Name");

String s\_name = scr.next();

System.***out***.println("Enter age");

**int** s\_age = scr.nextInt();

System.***out***.println(s\_id + " "+s\_name+" "+ s\_age);

**try** {

Class.*forName*("com.mysql.jdbc.Driver");//driver register

Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/batch2","root","root");

Statement stmt = con.createStatement();

String insertQuary = "insert into student values ("+s\_id+","+"'"+s\_name+"'"+","+s\_age+")";

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

**int** count = stmt.executeUpdate(insertQuary);

**if**(count != 0) System.***out***.println("record updated");

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}