Collections

List:

* List is a child interface of collection interface
* List can allow duplicates values.
* Insertion ordered will be preserved
* Allow null values
* We can access elements randomly.
* Stored elements index based
* Random access using get(index)

Queue :

* Queue interface will take only homogenous elemets
* Queue is a child interface of collection in interface
* Queue allow duplicate values
  + Priority queue : allows only hogemous elements
  + Arrayqueue allow both homogenous and heterogenous.
* Null values are allowed
  + Arraydeque will allow null values
  + Priority queue will allow null values
* Queue do not store the elements in index based
* Random access is not possible

Set

* Set is a interface will hold only homogenous elements
  + Hash set linked hashset will allow both homogenous sand hetrogenous elements.
* Set is a child interface of collection interface
* Not allow duplicates
* Insertion order will not be preserved
  + Linkedhashset will preserve the insertion order
  + Treeset will sort the elements
* Set may or may not allow duplicates values
  + Hashset and linkedhashset allow null values
  + Treeset will not allow duplicates
* Set can stored in the key value format
  + So random access in not possible

Multasking:

Executing multiple tasks simultaneously is called multitasking.

Example:

Suppose that a programmer write code in text editor, listening songs in music player, downloading file in the browser. these all are executing sequentially without disturbing other, independent execution takes place, here each and every task is treating as separate process and perform its operation.

There are 2 types of multi-tasking:

* Process based
* Thread based

Process based:

* This is under OS level, programmer no need more about this.
* Executing several tasks simultaneously, where each process is separate independent process.
* Best suitable for OS level
* **multiple independent processes**

Thread based:

* Executing several threads simultaneously, where each thread is a separate, independent part of the same program, is called thread-based multitasking.
* Best suitable for the programmer level.
* Suppose there is a 1000-line code; if it runs independently, it is a single-threaded program that takes time. To achieve better performance and fast execution of the program, we need to divide it into threads, and these threads are executing simultaneously.
* **1 program ----- multiple independent parts called threads**

Advantages of multithreading

Performance of system is improved

Complete a task in less time (response time reduced)

To achieve multitasking java provides built-in support with libraries ,such as

* By implements runnable interface (best approach)
* By extends thread class

1. **By extends thread class**

package multithreading;

public class ThreadClass

{

public static void main(String[] args)

{

Mythread m=new Mythread(); //thread instanceasition

m.start(); //starting a thread

for(int i=11;i<=20;i++)

{

System.***out***.println("main thread");

}

}

}

class Mythread extends Thread

{

public void run()

{

for(int i=1;i<=10;i++) // job of the therad

System.***out***.println("child thread");

}

}

The output of the multithreading is never predicted because it varies each and every time of the executions.

Different compilers follows different logarithms to execute the threads

Some compilers follows fifo ,lifo ,round robin etc.