***Dt : 18/11/2022***

***\*imp***

***The following are the list of objects generated from CoreJava:***

***1.User defined class objects***

***2.String Objects***

***3.WrapperClass Objects***

***4.Array Objects***

***5.Collection<E> Objects***

***6.Map<K,V> Objects***

***7.Enum<E> Objects***

***Complete List of Objects:***

***1.User defined class objects***

***2.String Objects***

***(a)String class Objects***

***(b)StringBuffer class Objects***

***(c)StringBuilder class Objects***

***3.WrapperClass Objects***

***(a)Byte Objects***

***(b)Short Objects***

***(c)Integer Objects***

***(d)Long Objects***

***(e)Float Objects***

***(d)Double Objects***

***(e)Character Objects***

***(f)Boolean Objects***

***4.Array Objects***

***(a)Array holding User defined class Objects***

***(b)Array holding String Objects***

***(c)Array holding WrapperClass objects***

***(d)Array holding Array Objects(Jagged Array)***

***(e)Array holding Dis-Similer Objects(Object Array)***

***5.Collection<E> Objects***

***(a)Set<E>***

***(i)HashSet<E>***

***(ii)LinkedHashSet<E>***

***(iii)TreeSet<E>***

***(b)List<E>***

***(i)ArrayList<E>***

***(ii)LinkedList<E>***

***(iii)Vector<E>***

***=>Stack<E>***

***(c)Queue<E>***

***=>PriorityQueue<E>***

***(d)Deque<E>***

***(i)ArrayDeque<E>***

***(ii)LinkedList<E>***

***6.Map<K,V> Objects***

***(a)HashMap<K,V>***

***(b)LinkedHashMap<K,V>***

***(c)TreeMap<K,V>***

***(d)Hashtable<K,V>***

***7.Enum<E> Objects***

***=======================================================================***

***faq:***

***wt is the diff b/w***

***(a)Container Objects***

***(b)Utility Objects***

***(c)Cursor Objects***

***(a)Container Objects:***

***=>The objects which hold data are known as Container Objects.***

***(b)Utility Objects:***

***=>The Objects which perform operations on other objects are known as Utility***

***Objects.***

***EX:***

***Scanner***

***StringTokenizer***

***StringJoiner***

***Arrays***

***Collections***

***(c)Cursor Objects:***

***=>The Objects which are used to retrieve elements from Collection objects are***

***known as Cursor Objects.***

***Ex:***

***Iterator<E>***

***ListIterator<E>***

***Enumeration<E>***

***Spliterator<E>***

***====================================================================***

***faq:***

***wt is the diff b/w***

***(i)Collection<E>***

***(ii)Collections***

***(i)Collection<E>:***

***=>Collection<E> is an interface and which is root of Java Collection<E> Framework.***

***(ii)Collections:***

***=>"Collections" is a class and which provide some utility methods***

***Ex:***

***sort()***

***binarySearch()***

***=========================================================================***

***\*Imp***

***MultiThreading in Java:***

***define Application?***

***=>set-of-programs collected together to perform defined action is known as***

***application.***

***define process?***

***=>According to Java,the application under execution is a process.***

***defien Task?***

***=>The part of process is known as Task***

***Note:***

***=>According to Java,each program in application is a Task***

***define Multi-Tasking?***

***=>Executing multiple tasks simultaneously is known as Multi-Tasking.***

***(Simultaneously means at-a-time but not parallel)***

***Note:***

***=>In the process of executing Multiple-Tasks simultaneously,only some part of***

***task is executed and the part of task is known as Thread.***

***define Thread?***

***=>The part of Task is known as Thread.***

***=>Thread is a LightWeight and Background process.***

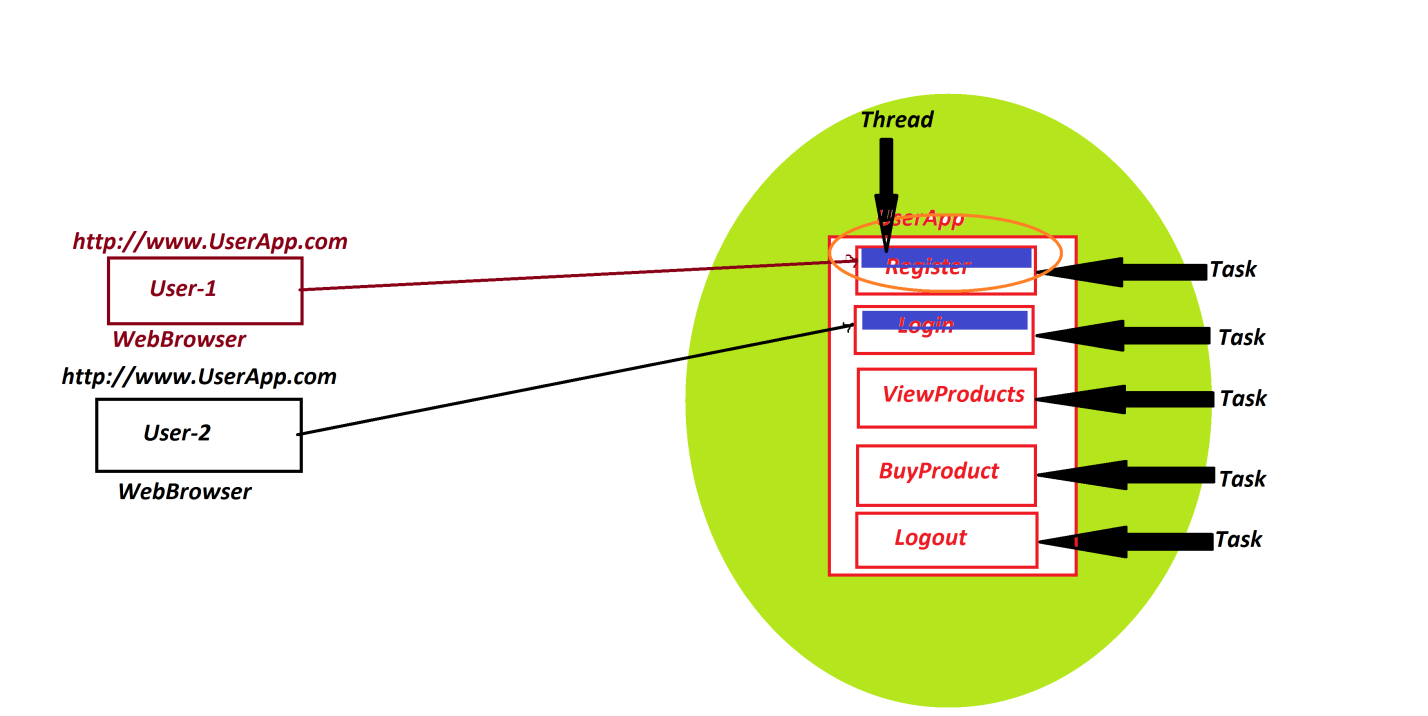
***LightWeight process - means consumes less execution time***

***Background process - means there is no separate identification.***

***define Multi-Threading?***

***=>Executing multiple threads Simultaneously is known as Multi-Threading.***

***Diagram:***

******

***====================================================================***

***\*imp***

***Creating and Executing thread:***

***step-1 : The user defined class must be implemented from "java.lang.Runnable"***

***interface.***

***structure of Runnable:***

***public interface java.lang.Runnable***

***{***

***public abstract void run();***

***}***

***step-2 : The implemented class must construct body for run() method and which is***

***holding program-logic***

***step-3 : Create object for user defined implementation class***

***step-4 : Create object for pre-defined "Thread" class,while object creation pass***

***the reference of User defined implementation class object as parameter.***

***step-5 : Execute run() method using start() method.***

***===========================================================================***