***Dt: 14/9/2022***

***Case-3 : Creating Object using 'java.lang.StringBuffer(String)'***

***syntax:***

***StringBuffer sb = new StringBuffer("NIT");***

***=>In this syntax the StringBuffer object is created with the***

***default capacity equal to '16 + length of String' passsed as***

***parameter while object creation.***

***Ex : DemoBuffer3.java***

***package maccess;***

***import java.util.\*;***

***public class DemoBuffer3 {***

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

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

***StringBuffer sb = new StringBuffer("NIT");***

***System.out.println("data : "+sb.toString());***

***System.out.println("default capacity : "+sb.capacity());***

***System.out.println("length : "+sb.length());***

***System.out.println("=====appending data====");***

***System.out.println("Enter the data:");***

***sb.append(s.nextLine());//Adding data to buffer***

***System.out.println("data : "+sb.toString());***

***System.out.println("capacity : "+sb.capacity());***

***System.out.println("length : "+sb.length());***

***s.close();***

***}***

***}***

***o/p:***

***data : NIT***

***default capacity : 19***

***length : 3***

***=====appending data====***

***Enter the data:***

***java programming***

***data : NIT java programming***

***capacity : 40***

***length : 20***

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

***Case-4 :Creating object using 'java.lang.StringBuffer(CharSequence)'***

***syntax:***

***StringBuffer sb1 = new StringBuffer("NIT");***

***StringBuffer sb2 = new StringBuffer(sb1);***

***=>This syntax is used to interlink two StringBuffer objects,which***

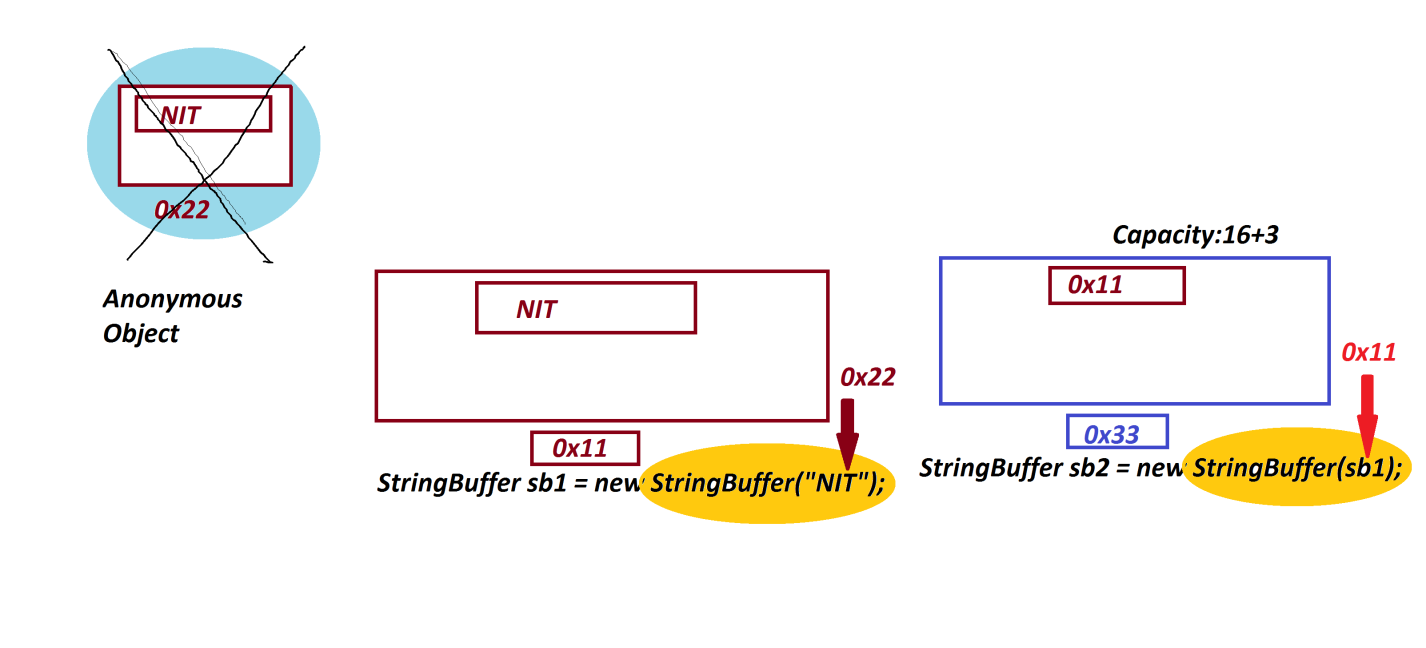
***means one StringBuffer object will hold the reference of another***

***StringBuffer object.***

***=>The default capacity of sb2 is eqaul to '16 + length of String***

***available in sb1'.***

***Diagram:***

******

***Ex : DemoBuffer4.java***

***package maccess;***

***public class DemoBuffer4 {***

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

***StringBuffer sb1 = new StringBuffer("NIT");***

***StringBuffer sb2 = new StringBuffer(sb1);***

***System.out.println("data : "+sb2.toString());***

***System.out.println("default capacity : "+sb2.capacity());***

***System.out.println("length : "+sb2.length());***

***sb2.append("java Programming language");***

***System.out.println("data : "+sb2.toString());***

***System.out.println("capacity : "+sb2.capacity());***

***System.out.println("length : "+sb2.length());***

***}***

***}***

***o/p:***

***data : NIT***

***default capacity : 19***

***length : 3***

***data : NITjava Programming language***

***capacity : 40***

***length : 28***

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

***Note:***

***=>StringBuffer class is Synchronized and Thread-safe class.***

***faq:***

***define synchronized class?***

***=>The class which is holding synchronized methods is known as***

***synchronized class.***

***faq:***

***define synchronized methods?***

***=>The methods which are declared with 'synchronized' keyword are***

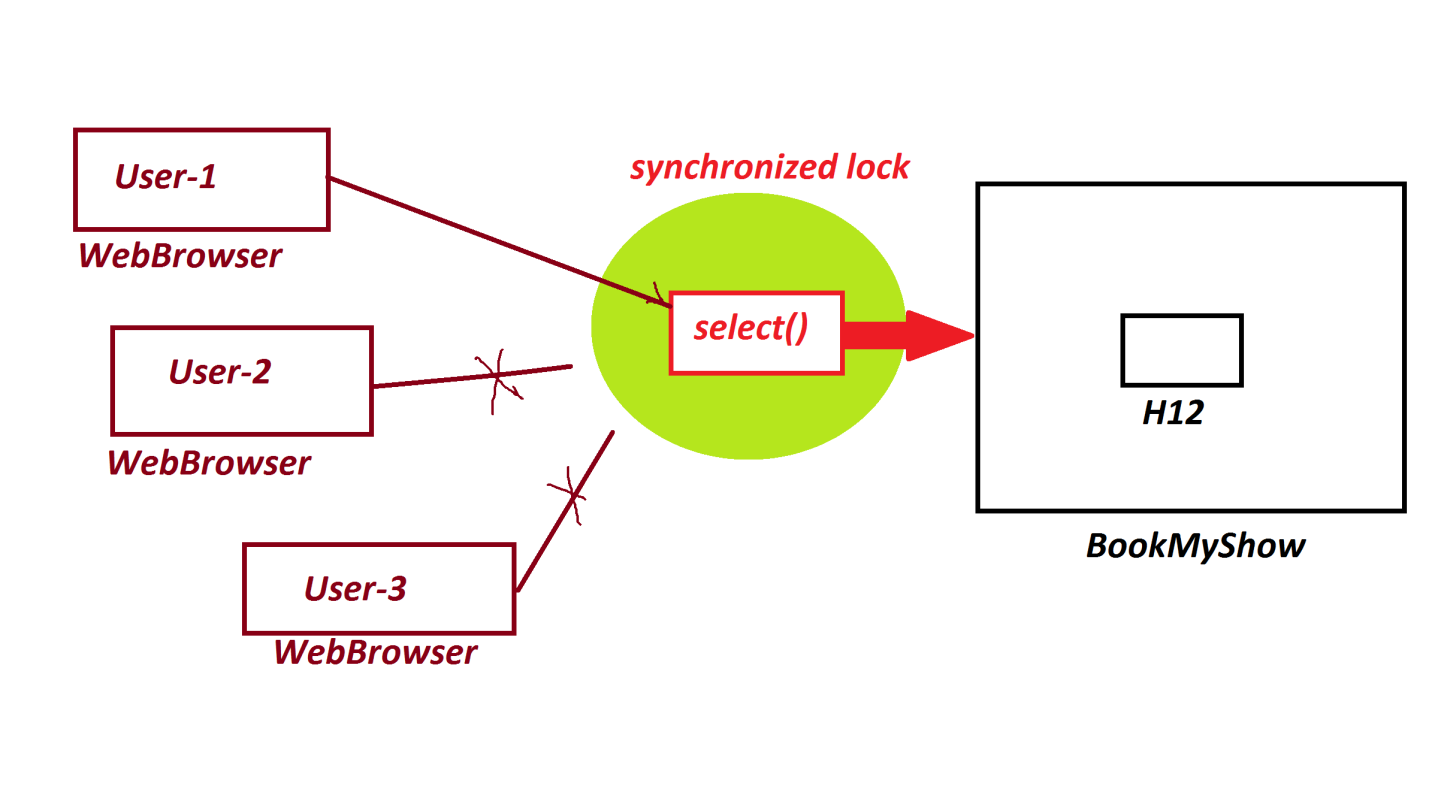
***known as synchronized methods.***

***faq:***

***wt is the advantage of synchronized methods?***

***=>These synchronized methods will be under 'synchronized lock'***

***and these methods can be used by one user at-a-time.***

******

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

***3.java.lang.StringBuilder class:***

***=>The objects which are created using 'java.lang.StringBuilder'***

***class are also "Mutable Objects".***

***=>The following are four constructors from 'StringBuilder':***

***public java.lang.StringBuilder();***

***public java.lang.StringBuilder(int);***

***public java.lang.StringBuilder(java.lang.String);***

***public java.lang.StringBuilder(java.lang.CharSequence);***

***=>StringBuilder is having same internal behaviour like***

***StringBuffer.***

***Note:***

***=>StringBuilder is Non-synchronized class,which means holding***

***Non-synchronized methods.***

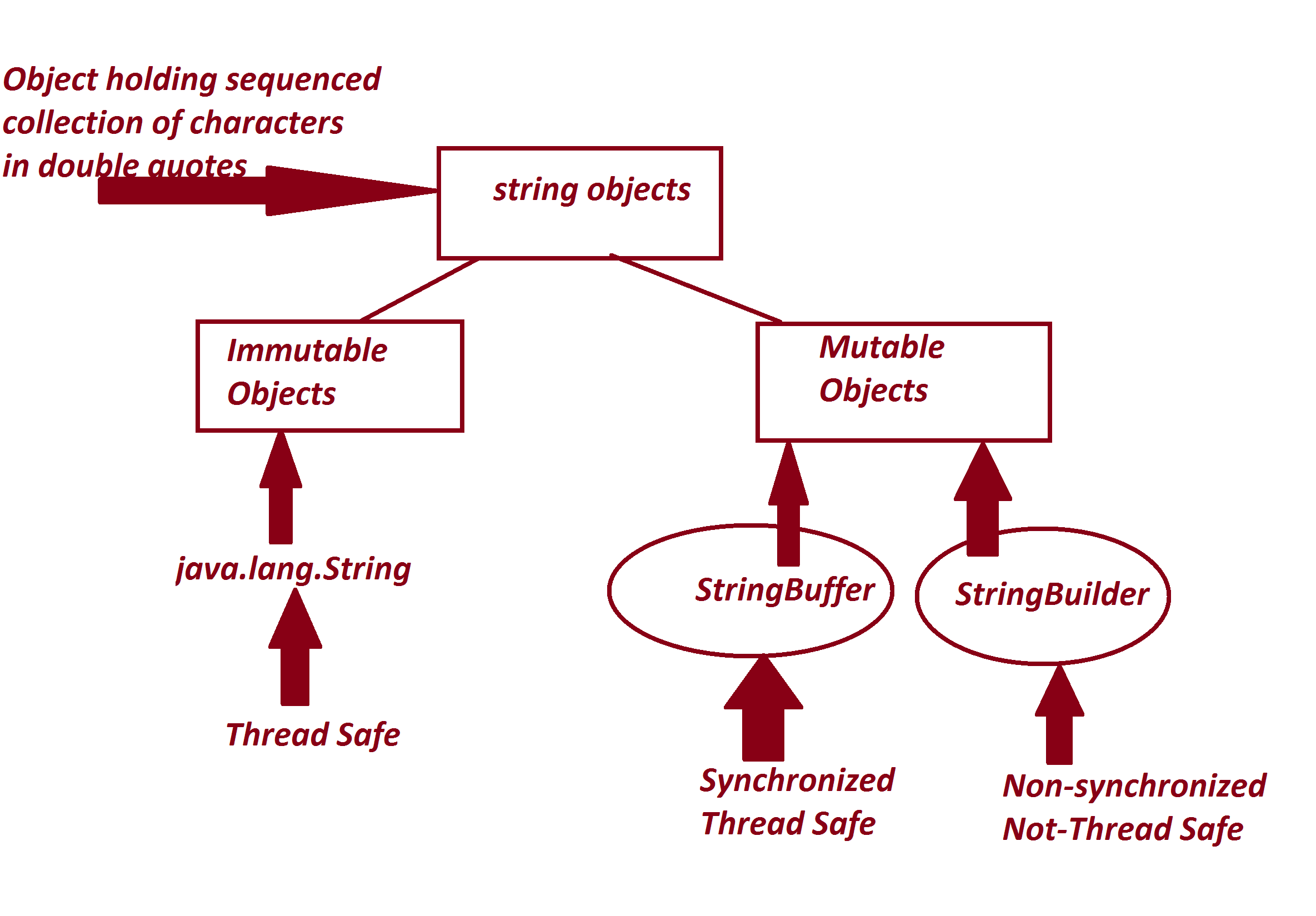
***=>StringBuilder is not Thread-safe,because of this reason it is***

***used in StandAlone applications.***

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

***Dt : 15/9/2022***

***Summary of Objects:***

******

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

***\*imp***

***=>The following some important Utility classes used on Strings:***

***(a)StringTokenizer class***

***(b)StringJoiner class***

***(a)StringTokenizer class:***

***=>StringTokenizer class is from java.util package and which is***

***used to break the given string into piecies(Tokens) based on the***

***delimiter.***

***=>The following are some important methods from StringTokenizer:***

***(i)hasMoreTokens()***

***(ii)nextToken()***

***(iii)countTokens()***

***(i)hasMoreTokens():***

***=>hasMoreTokens() method will check the token is available or***

***not,and generate boolean result.***

***Method Signature:***

***public boolean hasMoreTokens();***

***(ii)nextToken():***

***=>nextToken() method will retrieve and delete the token from***

***the StringTokenizer object.***

***Method Signature:***

***public java.lang.String nextToken();***

***(iii)countTokens():***

***=>countTokens() method is used to count the number of tokens***

***from the StringTokenizer object.***

***Method Signature:***

***public int countTokens();***

***syntax:***

***StringTokenizer ob = new StringTokenizer(str\_var,"delimiter");***

***Ex : DemoTokenizer.java***

**package** maccess;

**import** java.util.\*;

**public** **class** DemoTokenizer {

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

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

System.***out***.println("Enter the String:");

String str = s.nextLine();

System.***out***.println("Enter the delimiter(break specification):");

String dm = s.nextLine();

StringTokenizer ob = **new** StringTokenizer(str,dm);//ConCall

System.***out***.println("Count of tokens:"+ob.countTokens());

System.***out***.println("====Display tokens====");

**while**(ob.hasMoreTokens())

{

String tk=ob.nextToken();

System.***out***.println("Token : "+tk.toString());

System.***out***.println("Count of tokens:"+ob.countTokens());

}//end of loop

s.close();

}

}

***o/p:***

***Enter the String:***

***java program***

***Enter the delimiter(break specification):***

***a***

***Count of tokens:4***

***====Display tokens====***

***Token : j***

***Count of tokens:3***

***Token : v***

***Count of tokens:2***

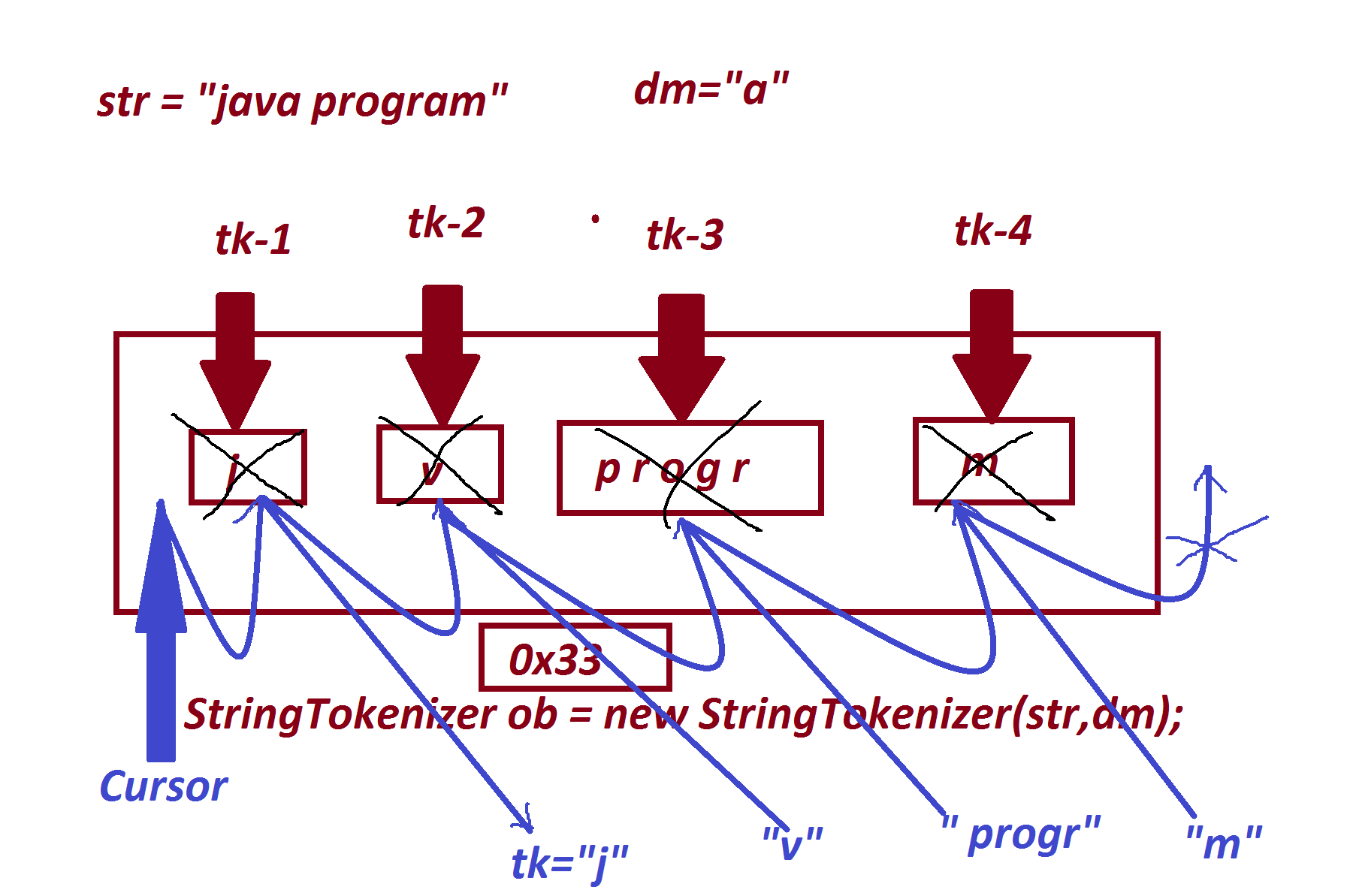
***Token : progr***

***Count of tokens:1***

***Token : m***

***Count of tokens:0***

***Diagram:***

******

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

***Assignment:***

***Wap to read an String and break into tokens when the delimiter is***

***Vowel?***

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

***(b)StringJoiner class:(Java8 - Version)***

***=>StringJoiner class is from java.util package introduced by***

***Java8 version and which is used to join the Strings based on***

***delimiter.***

***=>The following are some important methods of StringJoiner:***

***public java.util.StringJoiner setEmptyValue***

***(java.lang.CharSequence);***

***public java.lang.String toString();***

***public java.util.StringJoiner add(java.lang.CharSequence);***

***public java.util.StringJoiner merge(java.util.StringJoiner);***

***public int length();***

***syntax:***

***StringJoiner sj = new StringJoiner("delimiter");***