CHAPTER 3

Arrays and Strings

Objectives

- One Dimensional Arrays
  - Declaring a Variable to Refer to an Array
  - Creating, Initializing, and Accessing an Array, The Arrays Class
  - One Dimensional Array (Exercises, Practice)Exercises
- Multidimensional Arrays
  - Declaring, Creating, Initializing, and Accessing a Multidimensional Array
  - Two-Dimensional Array (matrix) (Exercises, Practice)Exercises
- Strings
  - The String Class
  - The Java Regex or Regular Expression
  - StringBuffer, StringBuilder and StringTokenizer Classes
- Array of Objects
- Enum Types
- Case study

Arrays

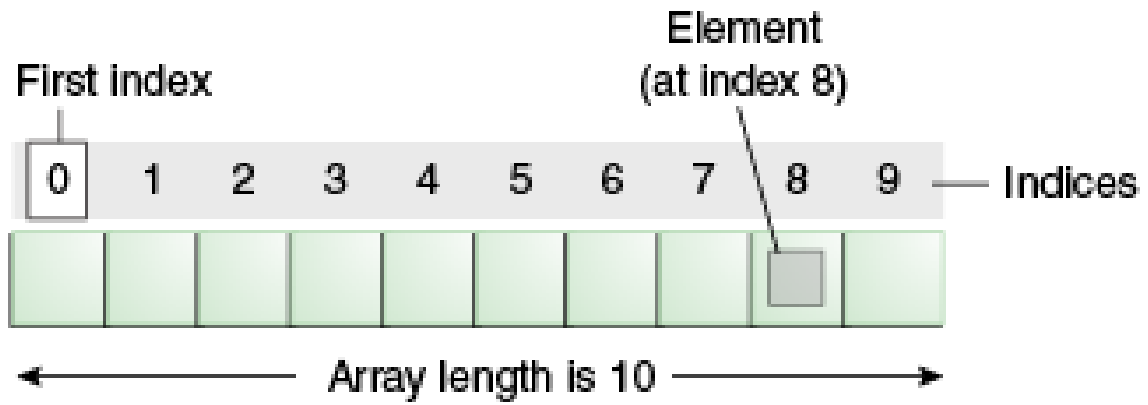
One Dimensional Arrays (Declaring, Creating, The Arrays Class)

- An array is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- Each item in an array is called an element, and each element is accessed by its numerical index.
- Declaring a Variable to Refer to an Array

```
int[] anArray;
or float anArrayOfFloats[]
```

- Creating, Initializing, and Accessing an Array

```
anArray = new int[10];
```

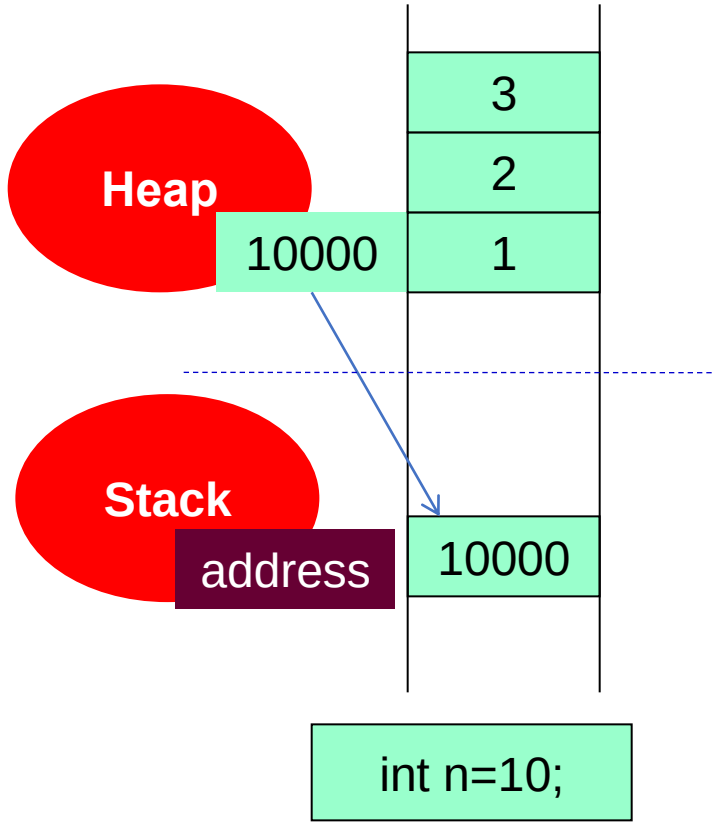


- Copying Arrays
  - Use arraycopy method from System class. (copyOf(T[] original, int newLength)

One Dimensional Arrays (Initializing)

```
int[] ar;
ar= new int[3];
ar[0]=1; ar[1]=2; ar[2]=3;
int a2[];
int[] a3 = {1,2,3,4,5};
int a4[] = {1,2,3,4,5};
```

Array is a reference variable



Example

```
public class Numbering {
    int sum(int... a){
        //int[] a
        int t=0;
        for(int x:a)
            t+=x;
        return t;
    }
    int min(int... a){
        int t=a[0];
        for(int x:a)
            if(t>x)
                t=x;
        return t;
    }
```

```
int max(int... a){
        int t=a[0];
        for(int x:a)
            if(t<x)
                t=x;
        return t;
    }
int[] sort(int... a){
  int t;
  for(int i=0;i<a.length-1;i++)
    for(int j=i+1;j<a.length;j++)
      if(a[i]>a[j]){
        t=a[i];
        a[i]=a[j];
        a[j]=t;       }
  return a;
}
```

```
int[] input(int n){
  Scanner in = new  canner(System.in);
  int[] b = new int[n];
  for(int i=0;i<n;i++){
    System.out.print("\n order "+i+": ");
    b[i]=in.nextInt();
  }
  return b;     }
void toString(int... a){
   System.out.print("\n Numbers:"+Arrays.toString(a));
}
```

Exercises

- Write a program that accepts 10 integers from the user and prints those entered numbers on the screen.

```
Scanner sc = new Scanner(System.in);
int[] a = new int[10];
for (int i = 0; i < 10; i++) {
      a[i] = sc.nextInt();
    }
for (int i = 0; i < 10; i++) {
    System.out.print(a[i] + " ");
    }
```

Code ptit(28)

- Given a sequence A[] with n elements consisting of only positive integers (no more than 10^5). Count how many times each number appears.

```
Scanner sc=new Scanner(System.in);
        int t=sc.nextInt();
        int level=1;
        while(t-->0){
                int n=sc.nextInt();
                int[] a=new int[n];
                int[] b=new int[100001];
                for(int i=0;i<n;i++){
                        a[i]=sc.nextInt();
                        b[a[i]]++;
                }
                System.out.println("Test "+level+":");
                for(int i=0;i<n;i++){
                        if(b[a[i]]!=0){
System.out.println(a[i]+" appears "+b[a[i]]+"
times");
                                b[a[i]]=0;
                        }
                }
                level++;
        }
```

Code ptit(29)

- Smallest multiplier of first positive integer (Given a natural number n. Your task is to find the smallest integer divisible by 1, 2, .., n)

```
public class Tool {
    public static long gcd(long a,long b){
        if(b==0) return a;
        return gcd(b,a%b);
     }
     public static long lcm(long a,long b){
        return a*b/gcd(a,b);
     }
}
```

```
public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
     int t=sc.nextInt();
     while(t-->0){
          int n=sc.nextInt();
          long res=1;
          for(int i=2;i<=n;i++){
               res=Tool.lcm(res,i);
          }
          System.out.println(res);
     }
    }
```

Code ptit(34)

- Given a sequence A[] consisting of N positive integers. The i-th element is said to be the equilibrium point of the sequence if the sum of the numbers on the left is equal to the sum of the numbers on its right. Your task is to be the first equilibrium of the given sequence A[]. If there is no answer, print -1.

```
Scanner sc=new Scanner(System.in);
	int t=sc.nextInt();
	while(t-->0){
		int n=sc.nextInt();
		int[] a=new int[n];
		int[] b=new int[n];
		for(int i=0;i<n;i++){
			a[i]=sc.nextInt();
			if(i==0) b[i]=a[i];
			else{
				b[i]=b[i-1]+a[i];
			}
		}
		int ans=-1;
		for(int i=1;i<n-1;i++){
			if(b[i-1]==b[n-1]-b[i]){
				ans=i+1;
				break;
			}
		}
		System.out.println(ans);
	}
```

Code ptit(39)

- Given an array of N distinct positive integers sorted in ascending order. List the missing numbers to get enough numbers between 1 and the largest number in the original sequence.

```
Scanner sc=new Scanner(System.in);
	int n=sc.nextInt();
	int[] a=new int[n];
	int[] b=new int[201];
	int res=0;
	for(int i=0;i<n;i++){
		a[i]=sc.nextInt();
		res=Math.max(res,a[i]);
		b[a[i]]++;
	}
	int check=1;
	for(int i=1;i<res;i++){
		if(b[i]==0){
			System.out.println(i);
			check=0;
		}
	}
	if(check==1){
		System.out.println("Excellent!");
	}
```

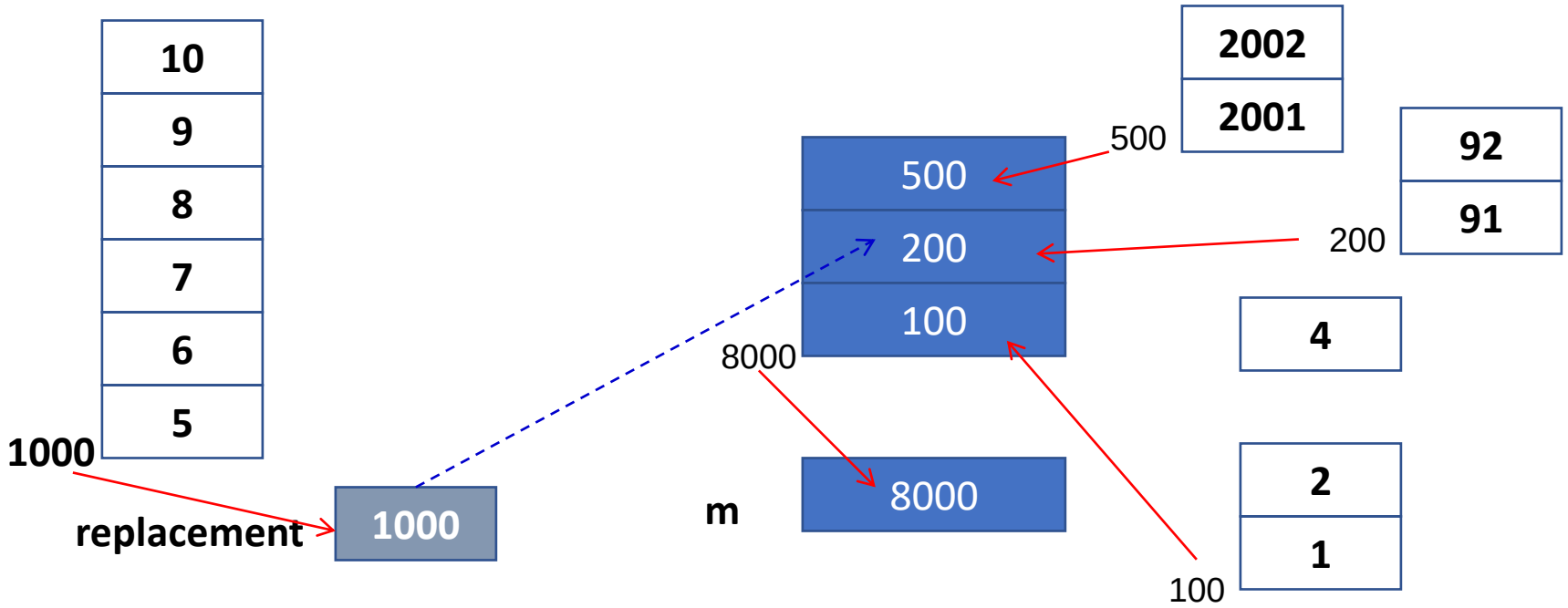
https://code.ptit.edu.vn/student/question

- From 25 to 44, from 127 to 137,

More

- Given two polynomials $P^n(x)$ and $Q^m(x)$. Write a program that does the following:
  - input 2 polynomials
  - Pn(x0) and Qm(x0)
  - Find the first derivative
  - Pn(x) + Qm(x)
  - Pn(x) - Qm(x)
  - Pn(x) / Qm(x) and residual polynomial

Multidimensional Arrays



- int[][] a; // declare a matrix
- int r=10, c=5; // number of rows, columns
- a= new int[r][c]; // memory allocate
- int m[][]= { {1,2,3,4}, {91,92}, {2001,2002}};
- int[] replacement = {5,6,7,8,9,10};
- m[1]= replacement;

- `int b[3][4];`
- `int row = b.length;`
- `int col = b[0].length;`

Example(1)

- Given a 2-dimensional array a (n rows, m columns). Write a program that accepts a from the user and calculates the sum of all elements in a.

```
public class Matrix {
    private int[][]a;

    public Matrix(int row,int col) {
        a=new int[row][col];
    }
    public void input(){
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < a.length; i++) {
                    for (int j = 0; j < a[0].length; j++) {
                        a[i][j] = sc.nextInt();
                    }
                }
    }
```

Example(2)

```
    public void out(){
        for (int i = 0; i < a.length; i++) {
                for (int j = 0; j < a[0].length; j++) {
                    System.out.print(a[i][j]+" ");
                }
                System.out.println();
            }
    }
    public int sum(){
        int answer = 0;
            for (int i = 0; i < a.length; i++) {
                for (int j = 0; j < a[0].length; j++) {
                    answer += a[i][j];
                }
            }
        return answer;
    }
}
```

to *add* elements to an *array*

```
public int[][] add(int[][] b) {
      int r = a.length;
     int c = a[0].length;
     int[][] d = new int[r][c];
     for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
          d[i][j] = a[i][j] + b[i][j];
      return d;
}
```

To multiply

```
public int[][] multiply(int[][] b) {
  int r = a.length;
  int c = a[0].length;
  int r1 = b.length;
  int c1 = b[0].length;
  if (r != c1) throw new RuntimeException("Illegal matrix
 dimensions.");
  int[][] d = new int[r][c1];
  for (int i = 0; i < r; i++)
    for (int j = 0; j < c1; j++)
      for (int k = 0; k < n1; k++)
        d[i][j] += a[i][k] * b[k][j];
   return d;
}
```

to *transpose* matrix

```
public int[][] transpose() {
   int r = a.length;
   int c = a[0].length;
   int[][] b = new int[c][r];
for (int i = 0; i < r; i++)
     for (int j = 0; j < c; j++)
       b[j][i] = a[i][j];
   return b;
}
```

Exercises: code ptit (44)

- Given a matrix A[][] with N rows and 3 columns, where the positions are binary values (0 or 1). Count how many rows there are more 1 than 0.

```
Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[][] a = new int[n][];
        int count =0;
        for (int i = 0 ; i <n;i++) {
            a[i] = new int[3];
            int c = 0;
            for (int j = 0; j <3 ;j++) {
                a[i][j] = sc.nextInt();
                if (a[i][j] ==1) c++;
            }
            if (c >= 2) count++;
        }
        System.out.println(count);
```

More

- Square matrix a
  - Input square matrix a of order n
  - Find the row, column, or diagonal with the largest sum of elements
  - Find the transpose matrix of a
  - Check if matrix is symmetric?
  - Deterministic of a
  - Find the inverse matrix of a
  - Input the column matrix B (nx1), solve the system of linear equations n hidden AX = B by Gauss method

Strings

Strings

- Java uses the **String, StringBuffer, StringBuilder and StringTokenizer** classes to encapsulate strings of characters (16-bit Unicode).

java.lang.**Object**
    java.lang.**String** (implements java.lang.CharSequence,
        java.lang.Comparable<T>, java.io.Serializable)
    java.lang.**StringBuffer** (implements java.lang.CharSequence,
        java.io.Serializable)
    java.lang.**StringBuilder** (implements
        java.lang.CharSequence, java.io.Serializable)

java.lang.String Class

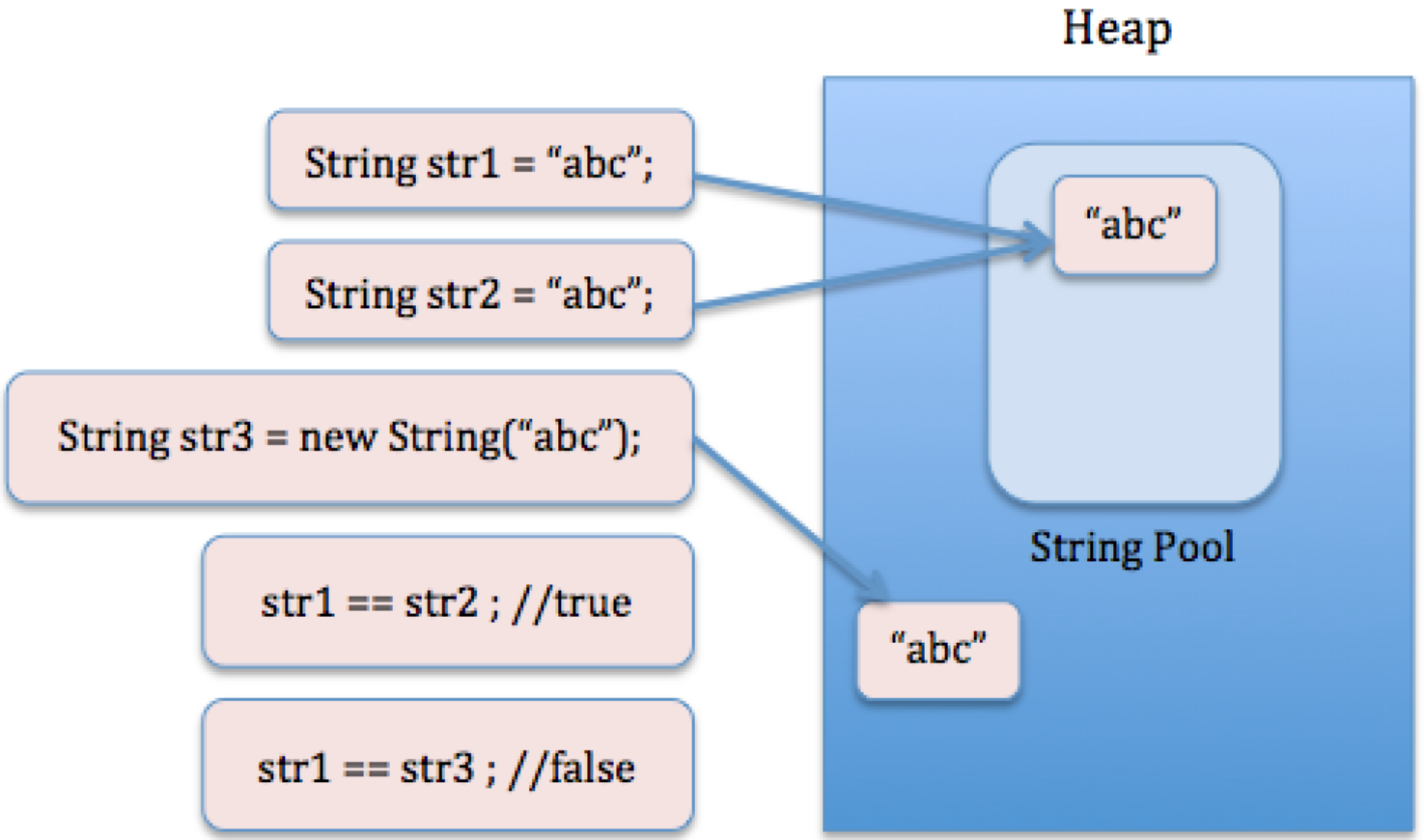
- The String class contains an immutable string (Once an instance is created, the string it contains cannot be changed) No setter is implemented
- Almost of it's methods will return a new string.

```
String a = "A String";
String b = "";
```

- Construct a string

```
String c = new String();
String d = new String("Another String");
String s2=  new String (new char[] {'a', 'b', 'c'});
String e = String.valueOf(1.23);
String f = null;
```

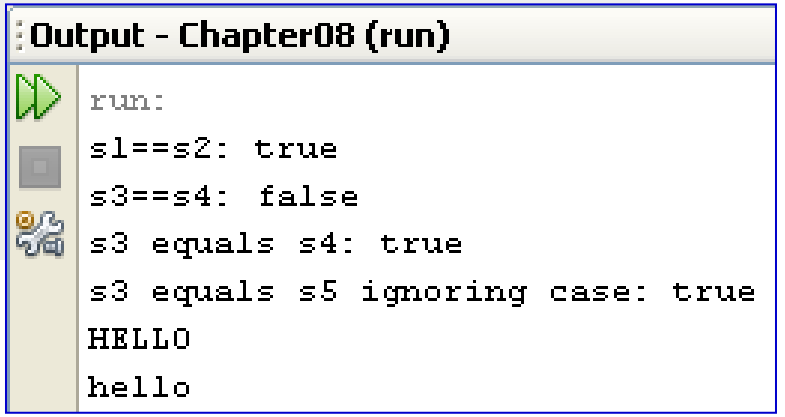
String pool (1)



String pool (2)

```
public class StringDemo {
  public static void main (String[] args)
  {  String s1="Hello"; // string pool
     String s2="Hello"; // string pool
     System.out.println("s1==s2: " + (s1==s2));
     String s3= new String("Hello");
     String s4= new String("Hello");
     System.out.println("s3==s4: " + (s3==s4));
     System.out.println("s3 equals s4: " + (s3.equals(s4)));
     String s5= new String ( new char[] { 'H', 'E', 'L', 'L', 'O' });
     System.out.println("s3 equals s5 ignoring case: " + (s3.equalsIgnoreCase(s5)));
     System.out.println(s5);
     s5= s5.toLowerCase();
     System.out.println(s5);
  }
}
```

Compare 2 strings: should use equals()



String operators

- Operator +

```
String a = "This" + " is a " + "String";
//a = “This is a String”
```

- String with print()

```
System.out.println("answer="+1 + 2 + 3);
System.out.println("answer="+(1+2+3));
System.out.println(“Number: "+1.45);
```

String methods (1)

Modifier and Type	Method and Description
char	charAt(int index)
char[]	toCharArray()
byte[]	getBytes()
int	codePointAt(int index), compareTo(String anotherString) compareToIgnoreCase(String str), hashCode(),indexOf(int ch), indexOf(…),lastIndexOf(…),length()
String	trim(),toString(),concat(String str), replace(…),replaceAll(…) replaceFirst(…),substring(…), toLowerCase(…),toUpperCase(…)
static String	copyValueOf(…), format(…), valueOf(…)
boolean	contains(CharSequence s), endsWith(String suffix), startsWith(…),isEmpty(),matches(String regex) equals(Object anObject), equalsIgnoreCase(…)
void	getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
String[]	split(String regex), split(String regex, int limit)
CharSequence	subSequence(int beginIndex, int endIndex)

String methods (2)

- compare(), concat(), equals(), split(), length(), replace(), compareTo(), substring(), …
- Example

```
String name = “Ly Lao Lo";
name.toLowerCase(); // "ly lao lo"
name.toUpperCase(); // "LY LAO LO"
“   Ly Lao Lo ".trim();// "Ly Lao Lo"
“ Ly Lao Lo".indexOf(‘L'); // 1
“ Ly Lao Lo".indexOf(“La”);
"Ly Lao Lo".length(); // 9
"Ly Lao Lo".charAt(5); // ‘o‘
"Ly Lao Lo".substring(5); // "o Lo“
"Ly Lao Lo".substring(2,5); // " La“
```

```
int compareTo(String anotherString)
int compareToIgnoreCase(String str)
String concat(String str)
boolean endsWith(String suffix)
String str = "www.tutorialspoint.com";
String endstr1 = ".com";
String endstr2 = ".org";
boolean retval1 = str.endsWith(endstr1);
boolean retval2 = str.endsWith(endstr2);
boolean equals(Object anObject)
boolean equalsIgnoreCase(String anotherString)
int indexOf(String str)
int lastIndexOf(String str)
```

```
public String[] split(String regex, int limit)
public String[] split(String regex)
1. String s1="Example   method   Split    a Line
into         Words";
  String[] st1=s1.split("\\s+");
  for(String w1:st1){
      System.out.println(w1); }
2. String s2="Who are you? Are you pretty? some have
called you cute while others just call you pretty.";
  String[] st2=s2.split("[\\.\\?\\!]");
  for(String w2:st2){
      System.out.println(w2); }
```

```
public String replaceAll(String regex, String
replacement)
public String replaceFirst(String regex, String
replacement)
“some have called you cute while others just
call you pretty”.
     replaceAll("\\s+",“ ");
2. String s3="Hanoi is famous for numerous
rivers,lakes,and mountains alongside and in the
surroundings.";      System.out.println(s3.
replaceAll("\\,\\s*", “, "));
3. "my name is khanh my name is java “.
replaceAll(“is",“was");
```

Regular Expression in Java

- The `java.util.regex` package primarily consists of three classes: `Pattern, Matcher, and PatternSyntaxException.`
  - A **Pattern** object is a compiled representation of a regular expression. The Pattern class provides no public constructors. To create a pattern, you must first invoke one of its public static compile methods, which will then return a Pattern object.
  - A **Matcher** object is the engine that interprets the pattern and performs match operations against an input string.
  - A **PatternSyntaxException** object is an unchecked exception that indicates a syntax error in a regular expression pattern.

Modifier	Description
i	Perform case-**in**sensitive matching
g	Perform a global match
gi	Perform a global case-insensitive match
^	Get a match at the beginning of a string
$	Get a match at the end of a string
[xyz]	Find any character in the specified character set
[^xyz]	Find any character not in the specified character set
\\w	Find any Alphanumeric character including the unserscore
\\d	Find any single digit
\\s	Find any single space character
?	Find **zero** or **one** occurrence of the regular expression
*	Find **zero** or **more** occurrence of the regular expression
+	Find **one** or **more** occurrence of the regular expression
()	Find the group of characters inside the parentheses & stores the matches string
X{n}	Matches any string that contains a sequence of *n X*'s
X{n,m}	Matches any string that contains a sequence of X to Y *n*'s

"^\\(?\\d{3}\\)?-?\\s*\\d{3}\\s*-?\\d{4}$"

```
public class MatchPhoneNumber {
  public static boolean isPhoneValid(String phone){
    boolean retval = false;
    String regex =
    "^\\(?\\d{3}\\)?-?\\s*\\d{3}\\s*-?\\d{4}$";
    retval = phone.matches(regex);
    if (retval)
     System.out.println("MATCH "+phone + "\n");
    return retval; }
  public static void main(String args[]) {
      isPhoneValid("(234)-  765  -8765");
      isPhoneValid("999-585-4009");
      isPhoneValid("1-585-4009");
      }      //"^[Bb]{1}\\d{2}[A-Za-z]{4}\\d{3}$";
}            //"^[NXnx]{1}\\d{3}[A-Za-z]{4}$"
```

StringBuffer, StringBuilder Classes

- Java's **StringBuffer** and **StringBuilder** classes represent strings that can be dynamically modified.
- StringBuffer is **threadsafe**.
- StringBuilder (introduced in 5.0) is not threadsafe.
- Almost of their methods are the same as methods in the String class.
- These classes do not use string pool, thus we cannot write StringBuffer t = "ABC";
- We cannot use the operator + to their objects.

Thread: Unit of code (method) is running
Multi-threading program: A program has some threads running concurrently. If 2 threads access common data, their values are not un-predictable. So, in multi-thread programming, JVM supports a mechanism in which accesses to common resources must carry out in sequence based on synchronized methods.
Threadsafe class: A class with synchronized methods.

StringBuilder

public final class **StringBuilder** extends Object
implements Serializable, CharSequence

- The StringBuilder class was introduced in 5.0. It is nearly identical to StringBuffer.
- Major difference: string builders are not threadsafe.
- If you want multiple threads to have concurrent access to a mutable string, use a string buffer.
- If your mutable string will be accessed only by a single thread, there is an advantage to using a string builder, which will generally execute faster than a string buffer.

StringBuilder - Class constructors

Constructor & Description
StringBuilder() This constructs a string builder with no characters in it and an initial capacity of 16 characters.
StringBuilder(int capacity) This constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.
StringBuilder(String str) This constructs a string builder initialized to the contents of the specified string.

StringBuilder - Class methods

StringBuilder append(String str) This method appends the specified string to this character sequence.
StringBuilder append(StringBuffer sb) This method appends the specified StringBuffer to this sequence.
char charAt(int index) This method returns the char value in this sequence at the specified index.
StringBuilder delete(int start, int end) This method removes the characters in a substring of this sequence.
StringBuilder deleteCharAt(int index) This method removes the char at the specified position in this sequence.
int indexOf(String str) This method returns the index within this string of the first occurrence of the specified substring.
int indexOf(String str, int fromIndex) This method returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

int length() This method returns the length (character count).
StringBuilder replace(int start, int end, String str) This method replaces the characters in a substring of this sequence with characters in the specified String.
StringBuilder reverse() This method causes this character sequence to be replaced by the reverse of the sequence.
String substring(int start) This method returns a new String that contains a subsequence of characters currently contained in this character sequence.
String substring(int start, int end) This method returns a new String that contains a subsequence of characters currently contained in this sequence.
String toString() This method returns a string representing the data in this sequence.
StringBuilder insert(int offset, String str) This method inserts the string into this character sequence.

Example: normal Text

```
public class WrapperDemo {
public static String normalText(String line) {
      String out = "";
      line = line.toLowerCase();
      line = line.replaceAll("\\s+", " ");
      line = line.replaceAll(" \\.", "\\.");
      line = line.replaceAll("\\.", "\\. ");
      line = line.replaceAll(" \\,", "\\,");
      line = line.replaceAll("\\,", "\\, ");
      line = line.replaceAll("\\s+", " ");
      line = line.trim();
```

```
out=line;
boolean isCap = true;
char c;
StringBuilder strb = new StringBuilder("");
for (int i = 0; i < out.length()-1; i++) {
  c = out.charAt(i);
  if (c == '.') {
    isCap = true;
  }
  if (isCap && Character.isAlphabetic(c)) {
    c = Character.toUpperCase(c);
    isCap = false;
  }
  strb.append(c);
  }
```

```
out = strb.toString();
  if (out.charAt(out.length()-1) != '.') {
     out = out + ".";
  }
  return out;
  }
public static void main(String[] args) {
   String line="We    were   both   young ,
    when   I first saw you .
     i    close my eyes and the flashback starts";
System.out.println(normalText(line));
    }
```

Reversible

```
//palindrome
public boolean reversing(long n){
   StringBuilder sn = new
StringBuilder(Long.toString(n));
      return
sn.toString().equals(sn.reverse().toString());
   }
```

The *StringBuffer - threadsafe*

public final class **StringBuffer** extends Object
implements Serializable, CharSequence

```
public class StringBufferDemo {
 public static void main(String aegs[]){
        StringBuffer sBuf= new StringBuffer ("01234567");
        System.out.println(sBuf);
        sBuf.append("ABC");
        System.out.println(sBuf);
        sBuf.insert(2, "FAT PERSON");
        System.out.println(sBuf);
        sBuf.reverse();
        System.out.println(sBuf);
 }
}
```

```
run:
01234567
01234567ABC
01FAT PERSON234567ABC
CBA765432NOSREP TAF10
```

Constructors

Constructor & Description
StringBuffer() This constructs a string buffer with no characters in it and an initial capacity of 16 characters.
StringBuffer(int capacity) This constructs a string buffer with no characters in it and the specified initial capacity.
StringBuffer(String str) This constructs a string buffer initialized to the contents of the specified string.

- `public synchronized StringBuffer append(String s)`
- `public synchronized StringBuffer insert(int offset, String s)`
- `public synchronized StringBuffer replace(int startIndex, int endIndex, String str)`
- ……..

StringTokenizer Class

- The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string.
- Constructors:
  - **StringTokenizer**(String str): creates StringTokenizer with specified string and delimeter.
  - StringTokenizer(String str, String delim): creates StringTokenizer with specified string and delimeter.

Methods of StringTokenizer class

Public method	Description
boolean hasMoreTokens()	checks if there is more tokens available.
String nextToken()	returns the next token from the StringTokenizer object.
String nextToken(String delim)	returns the next token based on the delimeter.
boolean hasMoreElements()	same as hasMoreTokens() method.
Object nextElement()	same as nextToken() but its return type is Object.
int countTokens()	returns the total number of tokens.

Example

```
import java.util.StringTokenizer;
public class Simple{
public static void main(String args[]){
  StringTokenizer st = new StringTokenizer("I
 work at HN.I am a lecturer.I love HN.","\\.");
  while(st.hasMoreTokens()) {
 System.out.println(st.nextToken());
}
}
}
```

- By default StringTokenizer breaks String

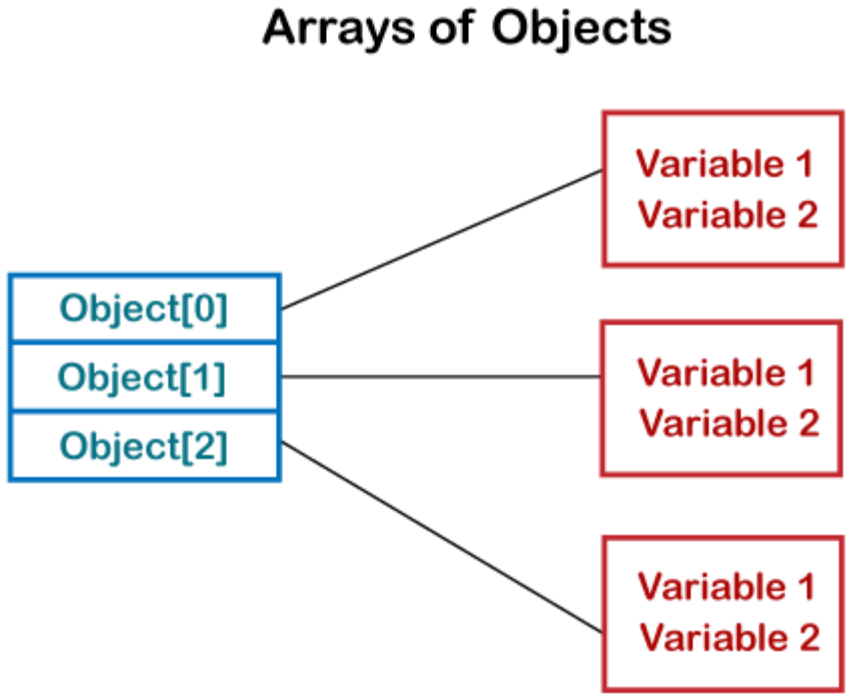
```
String str = "I am sample string and will be tokenized on
  space";
StringTokenizer dt=new StringTokenizer(str);
while (dt.hasMoreTokens()){
   System.out.println(dt.nextToken());}
```

- Multiple delimiters

```
   String s =“Who am i?Lan is my friend.I love Lan!How Lan
  love Him? of course i know!”;
   StringTokenizer mt = new StringTokenizer(s, “.?!");
   while (mt.hasMoreTokens()){
     System.out.println(mt.nextToken());
   }
```

Array of Objects

- The class is also a user-defined data type. An array that conations **class type elements** are known as an **array of objects**. It stores the reference variable of the object.



Creating an Array of Objects

- `ClassName obj[]=new ClassName[array_length];`
- `Or ClassName[] objArray;`
- `Or ClassName objeArray[];`
- `Employee emps=new Employee[20];//we have created a class named Employee.`

Problem (Array of vehicles)

- Vehicle Management System, each of the Vehicle has attributes of VIN (Vehicle Identification Number), manufacturer, manufacture year, cost, color. Write a program with the following functions:
  - Input data of vehicles
  - To view the list of the vehicles, to count the number of vehicles in the following table.
  - Searching by manufacturer, manufacture year, cost, color (exactly or approximately)
  - Searching from .. to…(manufacture year, cost)
  - Sorting by manufacturer, manufacture year, cost, color

Enum Types

- An enum type is a special data type that enables for a variable to be a set of predefined constants.
- We use enum types any time you need to represent a fixed set of named-constants (uppercase).

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY;// ; can be missed
}
```

Enum Type simple example

```
enum Season { SPRING, SUMMER, AUTUMN, WINTER  }
class Main {
    static void fun(Season x)
      {switch(x)
         {case SPRING:  System.out.println("It is spring"); break;
          case SUMMER: System.out.println("It is summer"); break;
          case AUTUMN:  System.out.println("It is autumn"); break;
          case WINTER:  System.out.println("It is winter");
}
      }
    public static void main(String[] args) {
      Season x = Season.WINTER;  fun(x);

      for(Season y: Season.values()) {
         System.out.print(y + ": "); fun(y);
       }
      System.out.println();
    }
}
```

```
It is winter
SPRING: It is spring
SUMMER: It is summer
AUTUMN: It is autumn
WINTER: It is winter
```

Enum Type with parameter constructor

```
enum Season {
    SPRING(25, 11), SUMMER(32, 13), AUTUMN(23, 10), WINTER(10, 9);
    private final int avgTemp, dayLength;
    Season(int x, int y) {
       avgTemp = x; dayLength = y;
     }
 public void display() {
   System.out.println(this + " average temperature is " + avgTemp);
   System.out.println(this + " average day's length is "+ dayLength);
  }  }
class Main {
    public static void main(String[] args) {
      Season x = Season.WINTER;
      x.display();
      System.out.println();
   }
}
```

```
WINTER average temperature is 10
WINTER average day's length is 9
```

Summary

- One Dimensional and Multiple Dimensional Arrays
- 4 String classes: Create and manipulate strings. Compares the String and StringBuilder classes…
- Regular Expression for data validation
- An array that conations objects
- And Enum Types

Case study

- Code ptit: from 25 to 64, 74,75, from 127 to 149, from 161 to 221, 232, 245, from 247 to 251, from 254 to 263, from 265 to 269, 289, 290