Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 P, NAB

Consultation: Friday 4.00 p.m. – 5.00 p.m.

BITS Pilani

Pilani Campus

Arrays

BITS Pilani
Pilani Campus

Predict the output

```
int inarr1[] = {1, 2, 3};
        int inarr2[] = {1, 2, 3};
        Object[] arr1 = {inarr1,inarr2};
        Object[] arr2 = {inarr2,inarr1};
        if (Arrays.equals(arr1, arr2))
            System.out.println("Same");
        else
            System.out.println("Not same");
```

Output: Not same

Solution: Arrays.deepEquals(arr1, arr2)

Final Arrays

```
class Test
{
    public static void main(String args[])
    {
        final int arr[] = {1, 2, 3, 4, 5};  // Note: arr is final
        for (int i = 0; i < arr.length; i++)
        {
            arr[i] = arr[i]*10;
            System.out.println(arr[i]);
        }
    }
}
```

Output:

10
20
30
40
50

Final Arrays

- Arrays are objects and object variables are always references in Java.

- When an object variable is declared as final, it means that the variable cannot be changed to refer to anything else.

Jagged Arrays

2D array with variable number of columns in each row.

```
        int arr[][] = new int[2][];
        arr[0] = new int[3];
        arr[1] = new int[2];
int count = 0;
 for (int i=0; i<arr.length; i++)
        for(int j=0; j<arr[i].length; j++)
                arr[i][j] = count++;
for (int i=0; i<arr.length; i++) {
        for (int j=0; j<arr[i].length; j++)
                System.out.print(arr[i][j] + " ");
                System.out.println();   }
```

innovate achieve lead

BITS Pilani

Pilani Campus

Array of Objects

Array of Objects

innovate achieve lead

```
class Account{
int acc;
String name;
float amount;
void insert(int acc,String name,float amt){
this.acc = acc;
this.name = name;
this.amount = amt;
}

void display(){
System.out.println(acc+" "+name+" "+amount);}
}
```

Array of Objects

innovate achieve lead

```
class TestAccount{
public static void main(String[] args){
Account[] a= new Account[3];

for(int i=0;i<3;i++)
{
 a[i]= new Account();
 a[i].insert(Integer.parseInt(args[3*i]), args[3*i+1], Float.parseFloat(args[3*i+2]));
a[i].display();
}
}}
```

Output:

111 abc 1000.0
222 bcd 2000.0
333 cde 5000.0

Strings

BITS Pilani
Pilani Campus

Unicode

- Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.

- In Unicode, character holds 2 byte, so java also uses 2 byte for characters.
    - lowest value:\u0000
    - highest value:\uFFFF

Need for Unicode

- Before Unicode, there were many language standards:
  - ASCII for the United States. (default character set)
  - ISO 8859-1 for Western European Language.
  - KOI-8 for Russian.
  - GB18030 and BIG-5 for chinese, and so on.

- Problem:
  - A particular code value corresponds to different letters in the various language standards.
  - Variable length: Some common characters are encoded as single bytes, other require two or more byte.

String creation

`String str = "abc";` is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

- Construct a string object by passing another string object.

  String str2 = new String(str);

String Constructors

- **String(byte[] byte_arr) – default character set (ASCII)**

```
byte[] b_arr = {74, 97, 118, 97};
String str =new String(b_arr);    // JAVA
```

- **String(byte[] byte_arr, Charset char_set)**

```
byte[] b_arr = {0x4a, 0x61, 0x76, 0x61};
Charset cs = Charset.forName("UTF-8");
String str = new String(b_arr, cs);
```

- ***Refer (List of character set supported by Java): https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html***

String Constructors

innovate achieve lead

- **String(byte[] byte_arr, String char_set_name)**

```
byte[] b_arr = {0x4a, 0x61, 0x76, 0x61};
String str = new String(b_arr, "UTF-8");
```

- **String(byte[] byte_arr, int start_index, int length)**
- **String(byte[] byte_arr, int start_index, int length, Charset char_set)**
- **String(byte[] byte_arr, int start_index, int length, String char_set_name)**

- **String(char[] char_arr)**
- **String(char[] char_array, int start_index, int count)**

More on Strings

- Java string is a sequence of characters. They are objects of type String.
- Once a String object is created it cannot be changed. Stings are Immutable.
- To get changeable strings use the class called StringBuffer.
- String and StringBuffer classes are declared final, so there cannot be subclasses of these classes.
- The default constructor creates an empty string.

```
String s = new String();
```

Example

```
public class StringExamples
{
    public static void main(String[] args)
    {
        String s1 = "JAVA";

        String s2 = "JAVA";

        System.out.println(s1 == s2);

         String s3 = new String("JAVA");

        System.out.println(s2 == s3);
    }
}
```

Output : true

Output : false

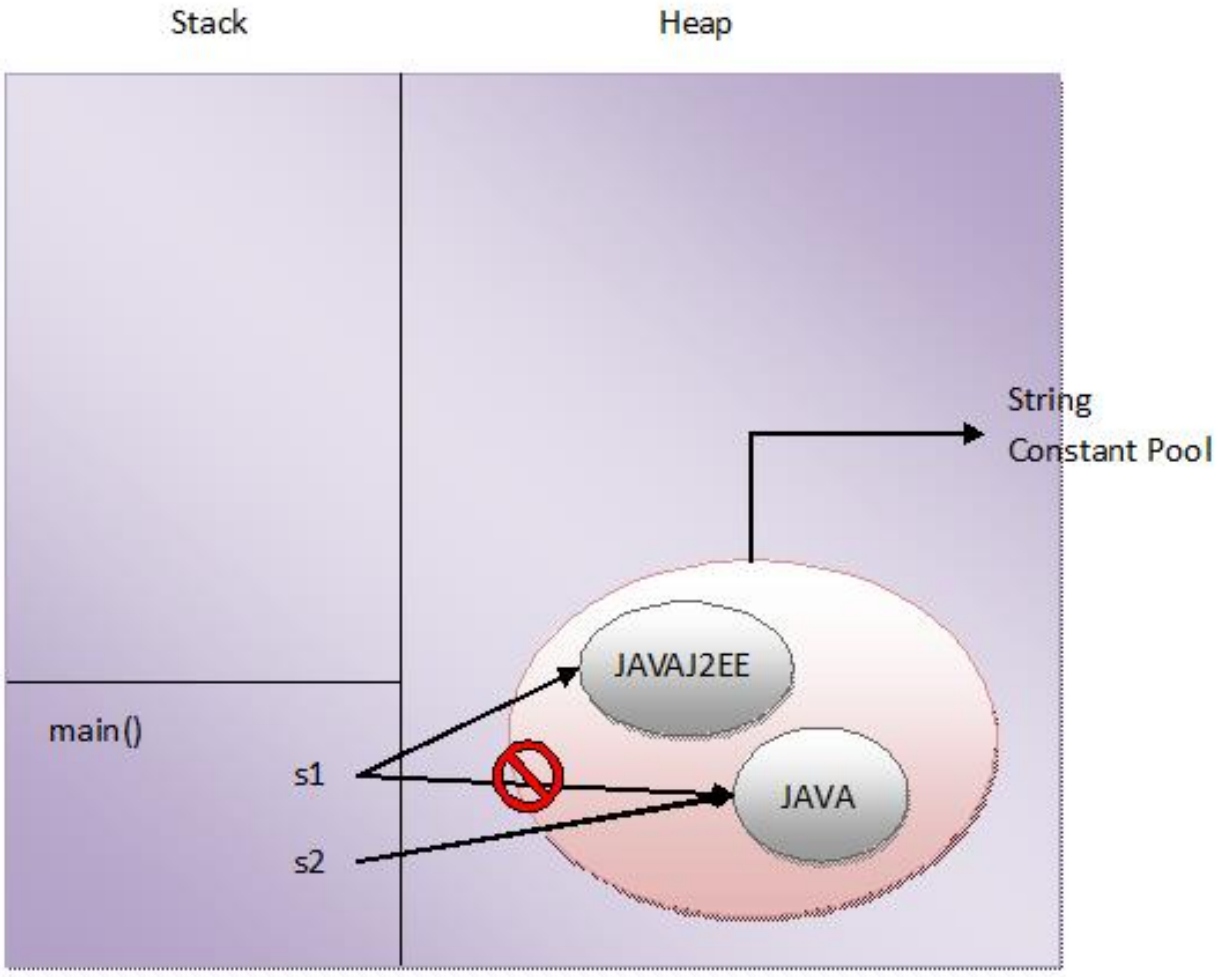
```
System.out.println(s2.equals(s3));
                                    // Output: true
```

Example

```
public class StringExamples
{
    public static void main(String[] args)
    {
        String s1 = "JAVA";
        String s2 = "JAVA";


        s1 = s1 + "J2EE";

        System.out.println(s1 == s2);

    }
}
```

Output : false

What happens in memory?



Are Strings created using new operator also immutable?

innovate achieve lead

```
public class StringExamples
{
    public static void main(String[] args)
    {
        String s1 = new String("JAVA");

        System.out.println(s1);        //Output : JAVA

        s1.concat("J2EE");

        System.out.println(s1);        //Output : JAVA
    }
}
```

Are Strings created using new operator also immutable?


```
public class StringExamples
{
    public static void main(String[] args)
    {
        String s1 = new String("JAVA");

        System.out.println(s1);         //Output : JAVA

        String s2=s1.concat("J2EE");

        System.out.println("s1: "+s1+" s2: "+s2);         //Output : s1: JAVA s2: JAVAJ2EE

    }
}
```

innovate achieve lead

BITS Pilani

Pilani Campus

String Buffer

StringBuffer Constructors

String Buffer represents growable and writable character sequences. The size grow automatically to accommodate characters and substring insert or append operations.

Constructor	Description
public StringBuffer()	create an empty StringBuffer
public StringBuffer(int capacity)	create a StringBuffer with initial room for capacity number of characters
public StringBuffer(String str)	create a StringBuffer containing the characters from str

Capacity of the String Buffer

- If the number of character increases from its current capacity, it increases the capacity by (oldCapacity*2)+2

```
StringBuffer sb = new StringBuffer("Java");
System.out.println(sb.capacity());

sb.append(" Programming");
System.out.println(sb.capacity());

sb.append("is ea");
System.out.println(sb.capacity());

StringBuffer sb1 = new StringBuffer(5);
sb1.append("Program");
System.out.println(sb1.capacity());
sb1.append(“ming");
System.out.println(sb1.capacity());
```

Output:

20
20
42
12
12

ensureCapacity of the StringBuffer


```
StringBuffer sb = new StringBuffer("Java is my favourite language");
System.out.println(sb.capacity());

sb.ensureCapacity(10);
System.out.println(sb.capacity());

sb.ensureCapacity(50);
System.out.println(sb.capacity());
```

Output:
45
45
92

String Tokenizer

String Tokenizer

- **java.util.StringTokenizer** class allows you to break a string into tokens
  - **default delimiter** set, which is " \t\n\r\f" : the space character, the tab character, the newline character, the carriage-return character, and the form-feed character.

Constructor	Description
StringTokenizer(String str)	creates StringTokenizer with specified string.
StringTokenizer(String str, String delim)	creates StringTokenizer with specified string and delimeter.
StringTokenizer(String str, String delim, boolean returnValue)	creates StringTokenizer with specified string, delimeter and returnValue. If return value is true, delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens.

Methods

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, after switching to the new delimiter.
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.

String Tokenizer - Example

innovate achieve lead

```
import java.util.StringTokenizer;
public class test{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my name is \t khan \n");
   int i=0,j;
   j = st.countTokens();
     while (st.hasMoreTokens()) {
         System.out.println(st.nextToken());
         i++;
     }
     System.out.println("i: "+i+"and j: "+j);
     System.out.println( st.countTokens());
   }

}
```

Output:

my
name
is
khan
i: 4 and j: 4
0

String Tokenizer - Example


```
import java.util.StringTokenizer;
public class Test{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my name/ is \t khan \n");
   int i=0,j;
   j = st.countTokens();
     while (st.hasMoreTokens()) {
         System.out.println(st.nextToken("/"));
         i++;
     }
     System.out.println("i: "+i+"and j: "+j);
     System.out.println( st.countTokens());
   }
}
```

Output:

```
my name
 is  khan

i: 2and j: 4
0
```

Review Questions

- StringTokenizer stuff = new StringTokenizer("abc,def,ghi");
  System.out.println(stuff.nextToken());

Output:
abc,def,ghi

- StringTokenizer stuff = new StringTokenizer("abc,def,ghi", ",");
  System.out.println(stuff.nextToken());

Output:
abc

Review Questions

- StringTokenizer stuff = new StringTokenizer("abc+def+ghi", "+", true);
  System.out.println(stuff.nextToken());
  System.out.println(stuff.nextToken());

  Output:
  abc
  +

- StringTokenizer stuff = new StringTokenizer("abc def+ghi", "+");
  System.out.println(stuff.nextToken());
  System.out.println(stuff.nextToken());

  Output:
  abc def
  ghi

Review Questions

- StringTokenizer st = **new StringTokenizer("abc+def:ghi", "+:", true);**
  while(st.hasMoreTokens()){
  System.***out.println(st.nextToken());*** }

Output:
abc
+
def
:
ghi

innovate achieve lead

BITS Pilani

Pilani Campus

Self Reading: String Methods

(Few important methods are discussed. Refer the Textbook for the remaining methods)

Length and Append

- The length() method returns the length of the string.

```
System.out.println("Hello World".length());
// prints 11
```

- The + operator is used to concatenate two or more strings.

```
String myname = "Harry"
String str = "My name is " + myname+ ".";
```

- For string concatenation the Java compiler converts an operand to a String whenever the other operand of the + is a String object.

String Methods-Character Extraction

- Characters in a string can be extracted in a number of ways.

- public char **charAt**(int index)
  - Returns the character at the specified index. An index ranges from 0 to length() - 1.

```
char ch;
ch = "Hello World".charAt(4);

String s1 = new String("Hello World");
ch=s1.charAt(4);
```

Output:
o

String Methods-Character Extraction

- **public void getChars(int start, int end, char[] destination, int destination_start)**

```
s1 = "Hello World";
char ch[]=new char[20];
s1.getChars(0, 11, ch, 0);
```

- **public byte[] getBytes()**

- **public char[] toCharArray()**

```
s1 = "Hello World";
char ch[]=s1.toCharArray();
```

String Methods

To compare two string objects

- **boolean equals(Object otherObj)**
- **boolean equalsIgnoreCase (String anotherString)**
- **int compareTo(String anotherString):** Compares two string lexicographically.

```
s1 = "World";
s2 = "Hello";
p=s1.compareTo(s2);
```

Output:
15

- This returns difference s1-s2. If :

  out < 0 // s1 comes before s2

  out = 0 // s1 and s2 are equal.

  out >0   // s1 comes after s2.

String Methods

- **int compareToIgnoreCase(String anotherString)**
  - Compares two string lexicographically, ignoring case considerations.
- **String toLowerCase()**
  - Converts all characters to lower case
- **String toUpperCase()**
  - Converts all characters to upper case
- **String trim()**
  - Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.
- **String replace (char oldChar, char newChar)**
  - Returns new string by replacing all occurrences of *oldChar* with *newChar*

String Methods

- **public boolean endsWith(String suf)**
  - Return *true* if the String has the specified suffix.

- **public boolean startsWith(String pre)**
  - Returns *true* if the String has the specified prefix

```
s1 = "Hello World";
s2 = "World";
System.out.println(s1.endsWith(s2));
```

String Methods

- **public boolean regionMatches(int start_OString, String another, int start_AString, int no_of_char)**

- **public boolean regionMatches(boolean ignore_case, int start_OString, String another, int start_AString, int no_of_char)**

```
s1 = "HellO World";
s2 = "hello";
System.out.println(s1.regionMatches(1, s2, 1,
4));
System.out.println(s1.regionMatches(true, 1, s2,
1, 4));
```

String Methods

- **String substring (int i) –** returns the substring from the i^{th} index

```
s1 = new String("Hello World");
s2=s1.substring(4);
System.out.println(s2);
```

Output:
o World

- **String substring (int i, int j):** Returns the substring from i to j-1 index.

```
s1 = new String("Hello World");
s2=s1.substring(4,7);
System.out.println(s2);
```

Output:
o W

String Methods

- **String concat(String str) –** Concatenates the string ‘str’ to the object invoking the method.

```
s1 = "Hello ";
s2 = "World";
s2=s1.concat(s2);
System.out.println("s1: :"+s1+"s2 :"+s2);
```

Output:
s1: :Hello s2 :Hello World

- **int indexOf (String s) –** returns index of the first occurrence of the specified string;
  - Returns -1 if not found

```
s1 = "World, Hello World, Hello";
s2 = "Hello";
p=s1.indexOf(s2);
```

Output:
7

String Methods

- **int indexOf (String s, int i) –** returns index of the first occurrence of the specified string, starting at the specified index

```
s1 = "World, Hello World, Hello";
s2 = "Hello";
p=s1.indexOf(s2,8);
```

Output:
20

- **int lastIndexOf(int ch):** Returns the index within the string of the last occurrence of the specified string.

```
s1 = "World, Hello World, Hello";
s2 = "Hello";
p=s1.lastIndexOf(s2);
```

Output:
20

String Methods

- **public int codePointAt(int index)**
  - returns the Unicode point of an index

```
s1 = "Hallo World";
p=s1.codePointAt(1);
```

Output:
97

- **public int codePointBefore(int index)**

- **public boolean contains(String str)**
  - Returns true if the invoking string object contains ‘str’

```
s1 = "Hello World";
s2 = “World";
System.out.println(s1.contains(s2));
```

innovate achieve lead

BITS Pilani

Pilani Campus

String Buffer Methods

String Buffer - Methods

Methods	Description
StringBuffer append(char c)	append c to the end of the StringBuffer
StringBuffer append(int i)	convert i to characters, then append them to the end of the StringBuffer
StringBuffer append(long L)	convert L to characters, then append them to the end of the StringBuffer
StringBuffer append(float f)	convert f to characters, then append them to the end of the StringBuffer
StringBuffer append(double d)	convert d to characters, then append them to the end of the StringBuffer
StringBuffer append(String s)	append the characters in s to the end of the StringBuffer
int capacity()	return the current capacity (capacity will grow as needed).
char charAt(int index)	get the character at index.
StringBuffer delete(int start, int end)	delete characters from start to end-1
StringBuffer deleteCharAt(int index)	delete the character at index

String Buffer - Methods

Methods	Description
StringBuffer insert(int index, char c)	insert character c at index (old characters move over to make room).
StringBuffer insert(int index, String st)	insert characters from st starting at position i.
StringBuffer insert(int index, int i)	convert i to characters, then insert them starting at index.
StringBuffer insert(int index, long L)	convert L to characters, then insert them starting at index.
StringBuffer insert(int index, float f)	convert f to characters, then insert them starting at index.
StringBuffer insert(int index, double d)	convert d to characters, then insert them starting at index.
int length()	return the number of characters presently in the buffer.
StringBuffer reverse()	Reverse the order of the characters.
void setCharAt(int index, char c)	set the character at index to c.
String toString()	return a String object containing the characters in the StringBuffer.