KtuQbank

MODULE-4

STRING

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

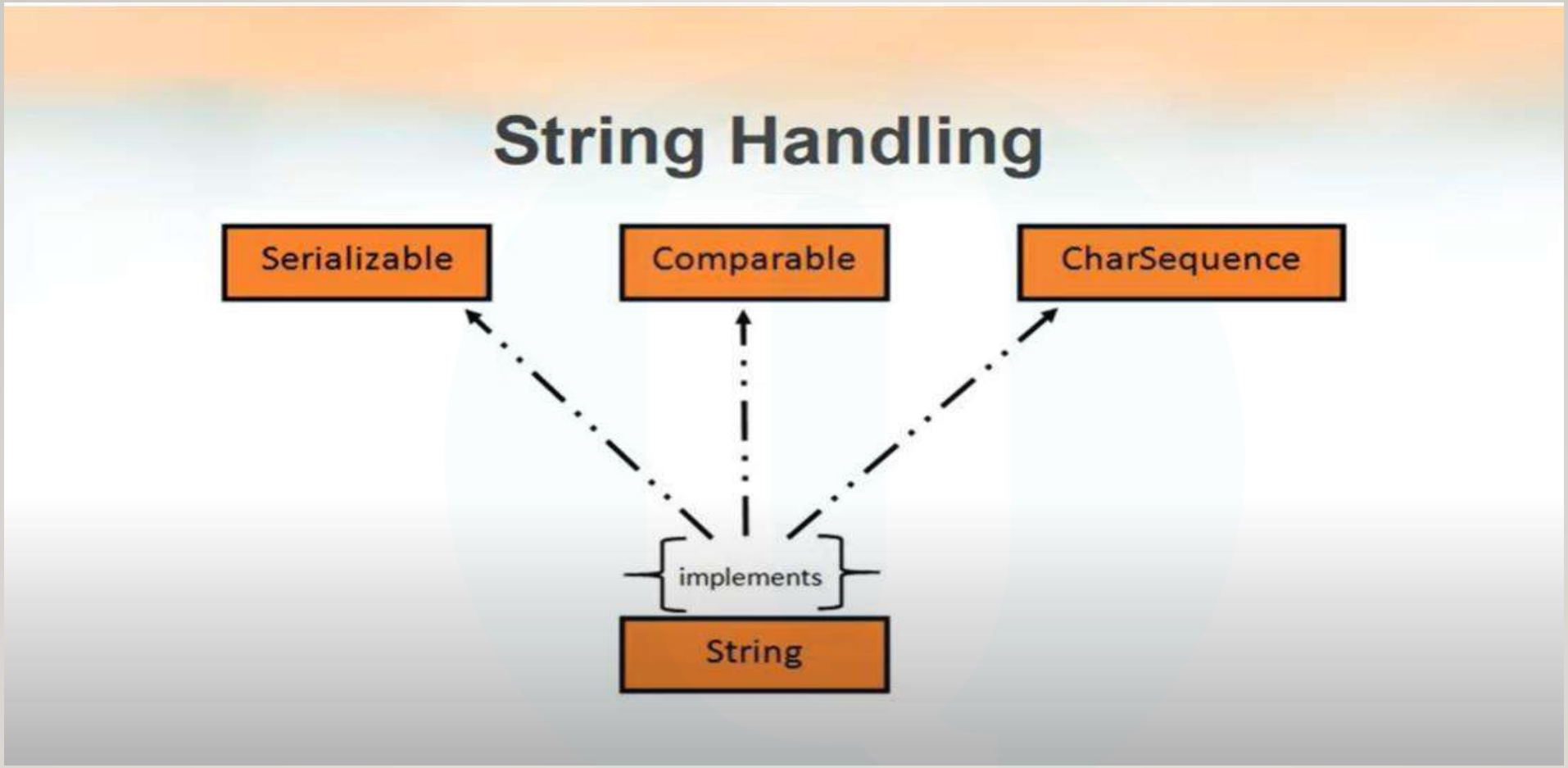
KtuQbank

String Handling

- String is basically **an object** that represents sequence of char values.
  - for e.g. "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.
- An array of characters works same as Java string.
- String is represented by **String class** which is located into **java.lang** package
- In java, **every string** that we create is actually **an object of type String**.
- In java, string is an immutable object which means it is constant and cannot be changed once it has been created.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank


String Handling
Serializable
Comparable
CharSequence
implements
String

KtuQbank

String Handling

Creating a String

- There are two ways to create a String in Java

1. String literal

1. a simple string enclosed in double quotes " ". A string literal is treated as a String object.

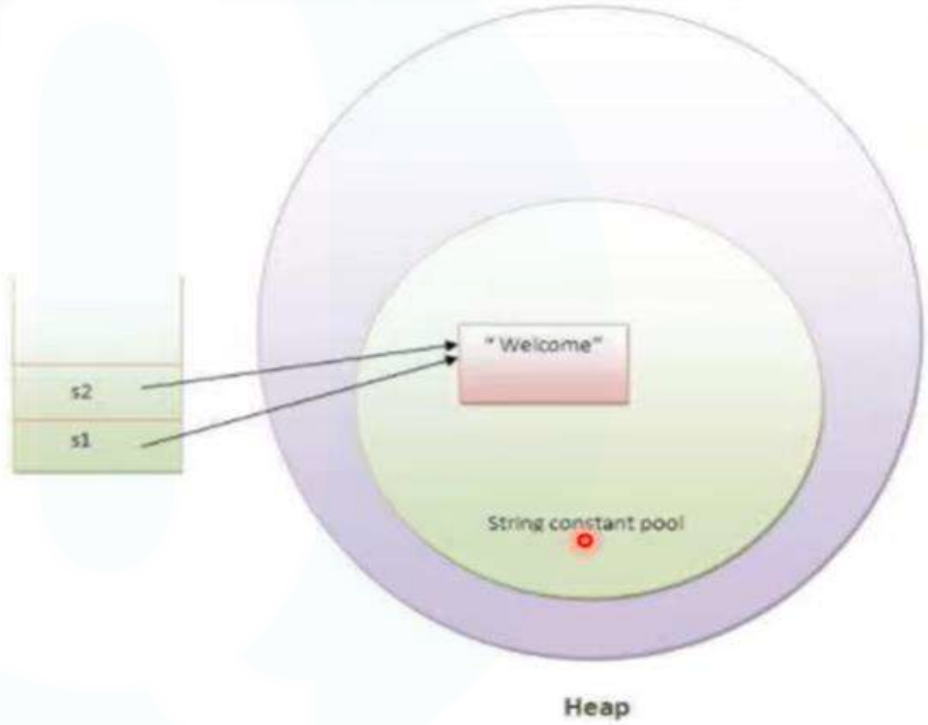
```
String str1 = "Welcome";
String str2 = "Welcome";
```

2. Each time you create a string literal, the JVM checks the "string constant pool" first
3. If the string already exists in the pool, a reference to the pooled instance is returned
4. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.
5. String objects are stored in a special memory area known as the **"string constant pool"**

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

String Handling



KtuQbank

String Handling

Creating a String

- There are two ways to create a String in Java

2. Using new keyword

1. String s=new String("Welcome");//creates two objects and one reference variable
2. JVM will create a new string object in normal (non-pool) heap memory,
3. and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

EXAMPLE PROGRAM

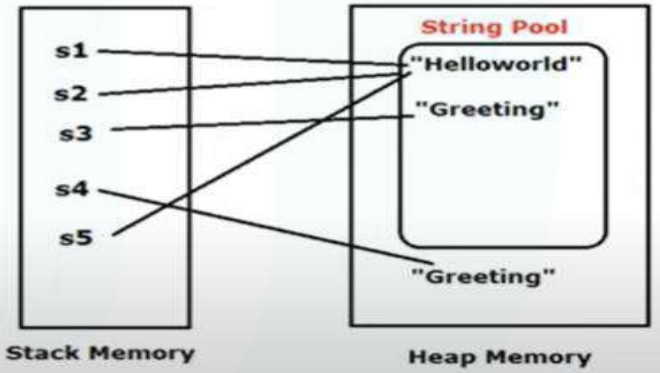
```
Class StringExample{
Public static void main(String arg[])
{
String s1="Java";
char ch[]={'s', 't', };
String s2=new String(char ch[]);
String s3=new String("example");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

- Output
- Java
- St
- example

KtuQbank

String Handling

```
ing s1 = "Helloworld";
ing s2 = "Helloworld";
ing s3 = "Greeting";
ing s4 = new String("Greeting");
ing s5 = "Helloworld";
```



For More Visit : KtuQbank.com | Fair Use Policy
KtuQbank

KtuQbank

String Constructors

- The most commonly used constructors of String class are as follows:

1. **String str = new String();**
    1. It will create a string object in the heap area with no value.
2. **String str = new String("string literal");**
    1. creates string object on heap for given string literal.
    2. String s2 = new String("Hello Java");
3. **String str = new String(char ch[]);**
    1. create a string object and stores the array of characters in it
    2. char chars[ ] = { 'a', 'b', 'c', 'd' }; String s3 = new String(chars);

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Constructors

- The most commonly used constructors of String class are as follows:

4. String(char chars[ ], int startIndex, int count);

1. create and initializes a string object with a subrange of a character array
2. startIndex specifies the index at which the subrange begin
3. count specifies the number of characters to be copied

```
char chars[ ] = { 'w', 'i', 'n', 'd', 'o', 'w', 's' };
String str = new String(chars, 2, 3);
```

The object str contains the address of the value "ndo" stored in the heap area because the starting index is 2 and the total number of characters to be copied is 3

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

String Constructors

```
package stringPrograms;
public class Windows
{
public static void main(String[] args)
{
 char chars[] = { 'w', 'i', 'n', 'd', 'o', 'w', 's' };
 String s = new String(chars, 0,4);
 System.out.println(s);
 }
}
```

KtuQbank

String Constructors

- The most commonly used constructors of String class are as follows:

5. String(byte byteArr[ ]) ;

1. constructs a new string object by decoding the given array of bytes (i.e, by decoding ASCII values into the characters) according to the system's default character set.

```
package stringPrograms;
public class ByteArray
{
public static void main(String[] args)
{
 byte b[] = { 97, 98, 99, 100 }; // Range of bytes: -128 to 127. These byte
values will be converted into corresponding characters.
 String s = new String(b);
 System.out.println(s);
 }
}
```

Output:

```
abcd
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Constructors

- The most commonly used constructors of String class are as follows:

6. String(byte byteArr[ ], int startIndex, int count) ;

1. creates a new string object by decoding the ASCII values using the system's default character set.

```
package stringPrograms;
public class ByteArray
{
public static void main(String[] args)
{
 byte b[] = { 65, 66, 67, 68, 69, 70 }; // Range of bytes: -128 to 127.
 String s = new String(b, 2, 4); // CDEF
 System.out.println(s);
 }
}
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Length

- The **java string length()** method gives length of the string.
- It returns count of total number of characters.

```
import java.io.*;
public class Test {

   public static void main(String args[]) {
      String Str1 = new String("Welcome to Tutorialspoint.com");
      String Str2 = new String("Tutorials" );

      System.out.print("String Length :" );
      System.out.println(Str1.length());

      System.out.print("String Length :" );
      System.out.println(Str2.length());
   }
}
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Length

- The **java string length()** method gives length of the string.
- It returns count of total number of characters.

```
import java.io.*;
public class Test {

   public static void main(String args[]) {
      String Str1 = new String("Welcome to Tutorialspoint.com");
      String Str2 = new String("Tutorials" );

      System.out.print("String Length :" );
      System.out.println(Str1.length());

      System.out.print("String Length :" );
      System.out.println(Str2.length());
   }
}
```

Output

```
String Length :29
String Length :9
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Character Extraction

- There are several ways by which characters can be extracted from String class object.
- String is treated as an object in Java so we can't directly access the characters that comprise a string.
- For doing this String class provides various predefined methods.
- charAt()
- getChars()
- getBytes()
- toCharArray()

KtuQbank

Character Extraction

charAt(int index)

- To extract a single character from a String, you can refer directly to an individual character via the charAt( ) method.

```
String str = "Welcome to string handling guide";
char ch1 = str.charAt(0);
char ch2 = str.charAt(5);
char ch3 = str.charAt(11);
char ch4 = str.charAt(20);
System.out.println("Character at 0 index is: " + ch1);
System.out.println("Character at 5th index is: " + ch2);
System.out.println("Character at 11th index is: " + ch3);
System.out.println("Character at 20th index is: " + ch4);
```

Output:

```
Character at 0 index is: W
Character at 5th index is: m
Character at 11th index is: s
Character at 20th index is: n
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Character Extraction

getChars()

- If you need to extract more than one character at a time, you can use the getChars( ) method.

Syntax

```
void getChars(int stringStart, int stringEnd, char arr[], int arrStart)
```

- **stringStart** and **stringEnd** is the starting and ending **index** of the substring.
- **arr** is the character array that will contain the substring. It will contain the characters starting from stringStart to stringEnd-1.
- **arrStart** is the index inside arr at which substring will be copied.
- The arr array should be large enough to store the substring.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Character Extraction

```
class temp
{
public static void main(String...s)
{
String str="Hello World";
char ch[]=new char[4];
str.getChars(1,5,ch,0);
System.out.println(ch);
}
}
```

Output

ello

Character Extraction

getBytes()

- getBytes() extract characters from String object and then convert the characters in a byte array.

Syntax

```
byte [] getBytes()
```

Example

```
String str="Hello";
byte b[]=str.getBytes();
```

KtuQbank

Character Extraction

toCharArray()

- It is an alternative of getChars() method.
- toCharArray() convert all the characters in a String object into an array of characters.
- It is the best and easiest way to convert string to character array.

Syntax

```
char [] toCharArray()
```

```
class temp
{
public static void main(String...s)
{
String str="Hello World";
char ch[]=str.toCharArray();
System.out.println(ch);
}
}
```

Output

Hello World

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Comparison

- There are several ways to compare string in java:
- By equals() method
- By equalsIgnoreCase() method
- By = = operator
- By compareTo() method
- By compareToIgnoreCase() method

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

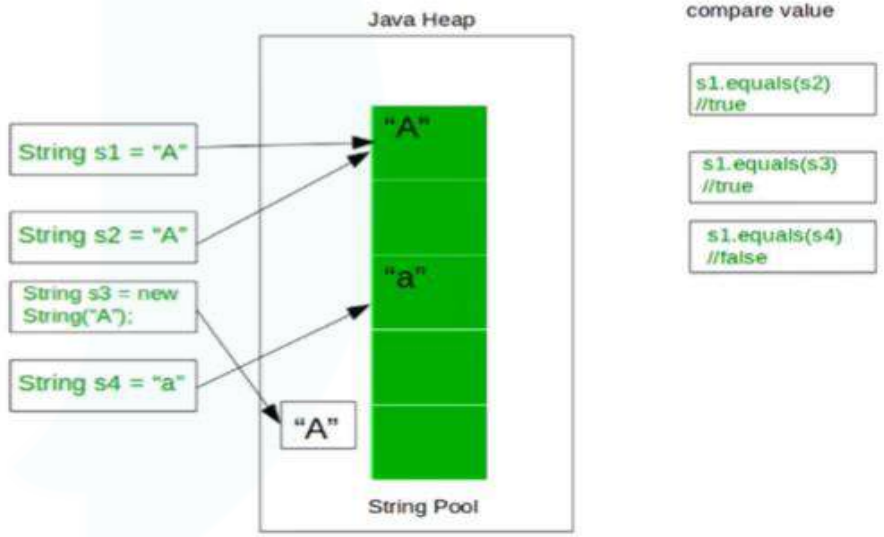
String Comparison

By equals() & equalsIgnoreCase() method

- compares the original content of the string. It compares values of string for equality

```
class Teststringcomparison2{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="SACHIN";

   System.out.println(s1.equals(s2));//false
   System.out.println(s1.equalsIgnoreCase(s2));//true
 }
}
```



KtuQbank

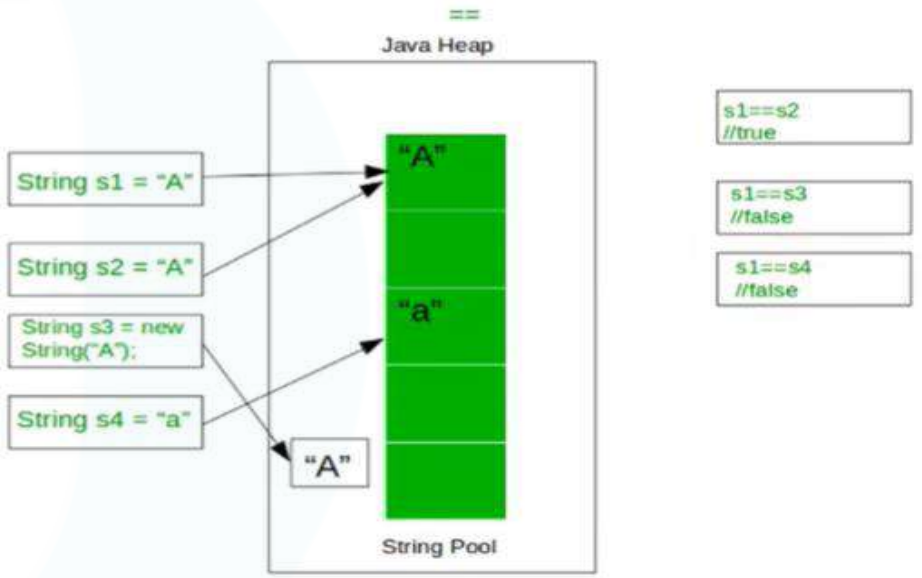
KtuQbank

String Comparison

By == operator

- compares references not values

```
class Teststringcomparison3{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="Sachin";
   String s3=new String("Sachin");
   System.out.println(s1==s2);//true (because both refer to same instance)
   System.out.println(s1==s3);//false(because s3 refers to instance created in nonpool)
 }
}
```



For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

String Comparison

By compareTo() method

- compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two string variables. If:

- s1 == s2 :0
- s1 > s2 :positive value
- s1 < s2 :negative value

```
class Teststringcomparison4{
 public static void main(String args[]){
  String s1="Sachin";
  String s2="Sachin";
  String s3="Ratan";
  System.out.println(s1.compareTo(s2));//0
  System.out.println(s1.compareTo(s3));//1(because s1>s3)
  System.out.println(s3.compareTo(s1));//-1(because s3 < s1 )
 }
}
```

```
Output:0
       1
       -1
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Searching Strings

- The String class provides 2 methods for searching a string. They are :

1. **indexOf()** : Searches for the first occurrence of a character or substring.
2. **lastIndexOf()** : Searches for the last occurrence of a character or substring.

- These methods return the starting index of character or a substring on a successful search else they return -1.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Searching Strings

- There are many overloaded forms of these methods:

Syntax	Explanation
int indexOf(char ch)	This method will return the index of the first occurrence of a character variable `ch` in the invoked string.
int lastIndexOf(char ch)	This method will return the index of the last occurrence of a character variable `ch` in the invoked string.
int indexOf(String st)	This method will return the index of the first occurrence of a `substring st` in the invoked string.
int lastIndexOf(String st)	This method will return the index of the last occurrence of a `substring st` in the invoked string.
int indexOf(char ch, int startIndex) ***int indexOf(String st, int startIndex)***	Here `startIndex` specifies the starting point of search. The search runs from `startIndex`to end of the String.
int lastIndexOf(char ch, int startIndex) ***int lastIndexOf(String st, int startIndex)***	Here `startIndex` specifies the starting point of search. The search runs from `startIndex` to zero.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Searching Strings

```
class StringSearchDemo
{
    public static void main(String arg[])
    {
        String str = "The Sun rises in the east and sets in the west.";
        System.out.println("Length of str : " + str.length());
        System.out.println("indexOf(i) : " + str.indexOf('i')); // LINE A
        System.out.println("lastIndexOf(i) : " + str.lastIndexOf('i')); // LINE B
        System.out.println("indexOf(st) : " + str.indexOf("st")); // LINE C
        System.out.println("lastIndexOf(st) : " + str.lastIndexOf("st")); // LINE D
        System.out.println("indexOf(e, 2) : " + str.indexOf('e', 2)); // LINE E
        System.out.println("lastIndexOf(e, 46) : " + str.lastIndexOf('e', 46)); // LINE F
        System.out.println("indexOf(st, 2) : " + str.indexOf("st", 2)); // LINE G
        System.out.println("lastIndexOf(st, 46) : " + str.lastIndexOf("st", 46)); // LINE H
    }
}
```

OUTPUT

```
Length of str : 47
indexOf(i) : 9
lastIndexOf(i) : 35
indexOf(st) : 23
lastIndexOf(st) : 44
indexOf(e, 2) : 2
lastIndexOf(e, 46) : 43
indexOf(st, 2) : 23
lastIndexOf(st, 46) : 44
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Modifying Strings

Java String replace()

- The java string replace() method returns a string replacing all the old char or CharSequence to new char or CharSequence.
- **There are two type of replace methods in java string.**

```
public String replace(char oldChar, char newChar)
and
public String replace(CharSequence target, CharSequence replacement)
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Modifying Strings

```
public class ReplaceExample1{
public static void main(String args[]){
String s1="javatpoint is a very good website";
String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'
System.out.println(replaceString);
}}
```

```
jevetpoint is e very good website
```

```
public class ReplaceExample2{
public static void main(String args[]){
String s1="my name is khan my name is java";
String replaceString=s1.replace("is","was");//replaces all occurrences of "is" to "was"
System.out.println(replaceString);
}}
```

```
my name was khan my name was java
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Modifying Strings

Java String concat()

- The java string concat() method combines specified string at the end of this string.
- It returns combined string.

```
public class ConcatExample{
public static void main(String args[]){
String s1="java string";
s1.concat("is immutable");
System.out.println(s1);
s1=s1.concat(" is immutable so assign it explicitly");
System.out.println(s1);
}}
```

```
java string
java string is immutable so assign it explicitly
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Modifying Strings

Java String substring()

- The java string substring() method returns a part of the string.
- We pass begin index and end index number position in the java substring method where start index is inclusive and end index is exclusive.
- There are two types of substring methods in java string.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Modifying Strings

Java String substring()

```
public String substring(int startIndex)
and
public String substring(int startIndex, int endIndex)
```

```
public class SubstringExample{
public static void main(String args[]){
String s1="javatpoint";
System.out.println(s1.substring(2,4));//returns va
System.out.println(s1.substring(2));//returns vatpoint
}}
```

KtuQbank

Modifying Strings

Other String methods to refer

1. split()
2. trim()
3. contains()
4. startsWith()
5. endsWith()

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

SEARCHING STRINGS

- The String class provides two methods that allow you to search a string for a specified
- character or substring:
- • indexOf( ) Searches for the first occurrence of a character or substring.
- • lastIndexOf( ) Searches for the last occurrence of a character or substring.
- These two methods are overloaded in several different ways. In all cases, the methods return the index at which the character or substring was found, or –1 on failure.
- To search for the first occurrence of a character, use
- int indexOf(int ch)

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

- To search for the last occurrence of a character, use
- int lastIndexOf(int ch)
- Here, ch is the character being sought.
- To search for the first or last occurrence of a substring, use
- int indexOf(String str)
- int lastIndexOf(String str)
- Here, str specifies the substring.

For More Visit : KtuQbank.com | Fair Use Policy

- You can specify a starting point for the search using these forms:
- int indexOf(int ch, int startIndex)
- int lastIndexOf(int ch, int startIndex)
- int indexOf(String str, int startIndex)
- int lastIndexOf(String str, int startIndex)
- Here, startIndex specifies the index at which point the search begins.
- For indexOf( ), the search runs from startIndex to the end of the string. For lastIndexOf( ), the search runs from startIndex to zero.

```
// Demonstrate indexOf() and lastIndexOf().
class indexOfDemo {
  public static void main(String args[]) {
    String s = "Now is the time for all good men " +
               "to come to the aid of their country.";

    System.out.println(s);
    System.out.println("indexOf(t) = " +
                       s.indexOf('t'));
    System.out.println("lastIndexOf(t) = " +
                       s.lastIndexOf('t'));
    System.out.println("indexOf(the) = " +
                       s.indexOf("the"));
    System.out.println("lastIndexOf(the) = " +
                       s.lastIndexOf("the"));
    System.out.println("indexOf(t, 10) = " +
                       s.indexOf('t', 10));
    System.out.println("lastIndexOf(t, 60) = " +
                       s.lastIndexOf('t', 60));
    System.out.println("indexOf(the, 10) = " +
                       s.indexOf("the", 10));
    System.out.println("lastIndexOf(the, 60) = " +
                       s.lastIndexOf("the", 60));
  }
}
```

OUTPUT

- Here is the output of this program:
- Now is the time for all good men to come to the aid of their country.
- indexOf(t) = 7
- lastIndexOf(t) = 65
- indexOf(the) = 7
- lastIndexOf(the) = 55
- indexOf(t, 10) = 11
- lastIndexOf(t, 60) = 55
- indexOf(the, 10) = 44
- lastIndexOf(the, 60) = 55

For More Visit : KtuQbank.com | Fair Use Policy

MODIFYING A STRING

- concat( )
- You can concatenate two strings using concat( ), shown here:
- String concat(String str)
- This method creates a new object that contains the invoking string with the contents of str appended to the end.
- concat( ) performs the same function as +.

For More Visit : KtuQbank.com | Fair Use Policy

EXAMPLE“:

- String s1 = "one";
- String s2 = s1.concat("two");
- puts the string "onetwo" into s2.
- It generates the same result as the following sequence:
- String s1 = "one";
- String s2 = s1 + "two";

For More Visit : KtuQbank.com | Fair Use Policy

- replace( )
- The replace( ) method replaces all occurrences of one character in the invoking string with another character. It has the following general form:
- String replace(char original, char replacement)

Here, original specifies the character to be replaced by the character specified by replacement.

For More Visit : KtuQbank.com | Fair Use Policy

EXAMPLE

- String s = "Hello".
- replace('l', 'w');

- puts the string "Hewwo" into s.

For More Visit : KtuQbank.com | Fair Use Policy

- trim( )
- The trim( ) method returns a copy of the invoking string from which any leading and
- trailing whitespace has been removed. It has this general form:
- String trim( )
- Here is an example:
- String s = " Hello World ".
- trim();
- This puts the string "Hello World" into s

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

valueOf()

- converts different types of values into string.
- By the help of string valueOf() method, you can convert
  - int to string, long to string,
  - boolean to string, character to string,
  - float to string
  - double to string,
  - object to string
  - char array to string.

```
public static String valueOf(boolean b)
public static String valueOf(char c)
public static String valueOf(char[] c)
public static String valueOf(int i)
public static String valueOf(long l)
public static String valueOf(float f)
public static String valueOf(double d)
public static String valueOf(Object o)
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

valueOf()

```
public class StringValueOfExample{
public static void main(String args[]){
int value=30;
String s1=String.valueOf(value);
System.out.println(s1+10);//concatenating string with 10
}}
```

```
public class StringValueOfExample2 {
    public static void main(String[] args) {
        // Boolean to String
        boolean bol = true;
        boolean bol2 = false;
        String s1 = String.valueOf(bol);
        String s2 = String.valueOf(bol2);
        System.out.println(s1);
        System.out.println(s2);
    }
}
```

```
char ch1 = 'A';
char ch2 = 'B';
String s1 = String.valueOf(ch1);
String s2 = String.valueOf(ch2);
System.out.println(s1);
System.out.println(s2);
```

```
float f  = 10.05f;
double d = 10.02;
String s1 = String.valueOf(f);
String s2 = String.valueOf(d);
System.out.println(s1);
System.out.println(s2);
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

StringBuffer vs String

No.	String	StringBuffer
1)	String class is immutable.	StringBuffer class is mutable.
2)	String is slow and consumes more memory when you concat too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when you cancat strings.
3)	String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.	StringBuffer class doesn't override the equals() method of Object class.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

StringBuffer vs String

- In java, string objects are immutable.
- Immutable simply means unmodifiable or unchangeable.
- Once string object is created its data or state can't be changed but a new string object is created.

```
class Testimmutablestring{
 public static void main(String args[]){
  String s="Sachin";
  s.concat(" Tendulkar");//concat() method appends the string at the end
  System.out.println(s);//will print Sachin because strings are immutable objects
 }
}
```

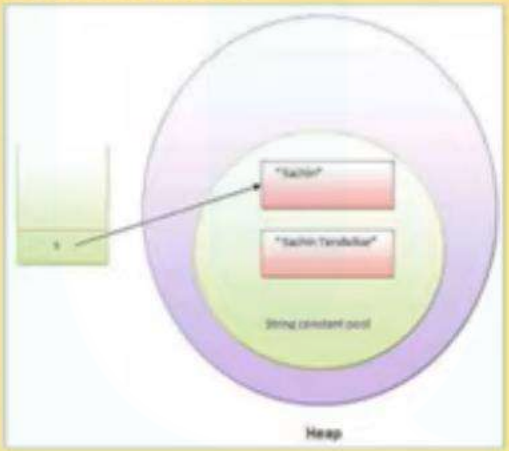
Output Sachin

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

StringBuffer vs String

It can be understood by the diagram given below. Here Sachin is not changed but a new object is created with sachintendulkar. That is why string is known as immutable.



KtuQbank

StringBuffer vs String

- As you can see in the figure that two objects are created but s reference variable still refers to "Sachin" not to "Sachin Tendulkar".
- But if we explicitely assign it to the reference variable, it will refer to "Sachin Tendulkar" object. For example:

```
class Testimmutablestring1{
 public static void main(String args[]){
  String s="Sachin";
  s=s.concat(" Tendulkar");
  System.out.println(s);
 }
}
```

Output

Sachin Tendulkar

In such case, s points to the "Sachin Tendulkar". Please notice that still sachin object is not modified.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

StringBuffer vs String

❖Why string objects are immutable in java

- Because java uses the concept of string literal.
- Suppose there are 5 reference variables,all referes to one object "sachin".
- If one reference variable changes the value of the object, it will be affected to all the reference variables.
- That is why string objects are immutable in java.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

StringBuffer vs String

- Java StringBuffer class is used to create mutable (modifiable) string.
- The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

StringBuffer append() method

```
class StringBufferExample{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello ");
sb.append("Java");//now original string is changed
System.out.println(sb);//prints Hello Java
}
}
```

StringBuffer insert() method

The insert() method inserts the given string with this string at the given position.

```
class StringBufferExample2{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello ");
sb.insert(1,"Java");//now original string is changed
System.out.println(sb);//prints HJavaello
}
}
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

StringBuffer vs String

❖StringBuffer replace() method

➢The replace() method replaces the given string from the specified beginIndex and endIndex.

```
class StringBufferExample3{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello");
sb.replace(1,3,"Java");
System.out.println(sb);//prints HJavalo
}
}
```

❖StringBuffer delete() method

➢The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

```
class StringBufferExample4{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello");
sb.delete(1,3);
System.out.println(sb);//prints Hlo
}
}
```

❖ StringBuffer reverse() method

➢The reverse() method of StringBuffer class reverses the current string.

```
class StringBufferExample5{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Hello");
sb.reverse();
System.out.println(sb);//prints olleH
}
}
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Collections overview

- The Collections in Java is a framework that provides an architecture to store and manipulate the group of objects.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection means a single unit of objects.
- Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Collections overview

What is Collections in Java

- A Collection represents a single unit of objects, i.e., **a group**.

What is a framework in Java

- It provides **readymade architecture**.
- It represents a **set of classes and interfaces**.
- It is optional.

What is Collection framework

- The Collection framework represents a unified architecture for storing and manipulating a group of objects. It **has Interfaces and its implementations**, i.e., classes & Algorithm
- The **java.util package** contains all the classes and interfaces for the Collection framework.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank


Collections overview
interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

KtuQbank

Collection Interface

- The **Collection interface** is the interface which is implemented by all the classes in the collection framework.
- It declares the methods that every collection will have.
- Collection interface builds the foundation on which the collection framework depends.
- Some of the methods of Collection interface are:

  Boolean add (Object obj), Boolean addAll (Collection c), void clear(), etc.

  which are implemented by all the subclasses of Collection interface.

For More Visit : KtuQbank.com | Fair Use Policy

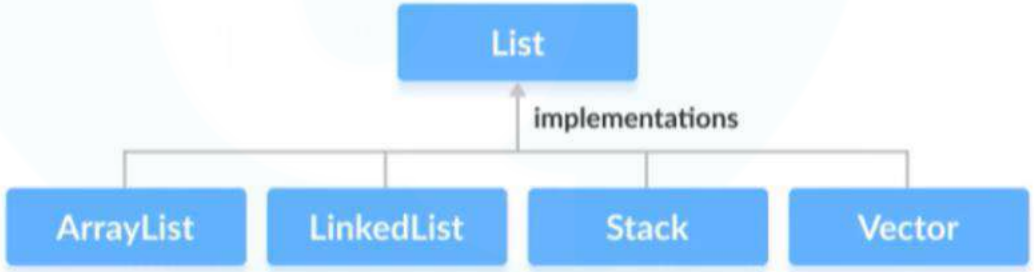
KtuQbank

KtuQbank

List Interface

- The List interface extends Collection and declares the behavior of a collection that stores a sequence of elements.
- The List interface is found in the java.util package
- Since List is an interface, we cannot create objects from it.
- In order to use functionalities of the List interface, we can use these classes:

1. ArrayList
2. LinkedList
3. Vector
4. Stack



For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

List Interface

- Elements can be inserted or accessed by their position in the list using a zero based index
- A list may contain duplicate elements.
- In addition to the methods defined by Collection, List defines some of its own
- List in Java provides the facility to maintain the ordered collection

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

List Interface

Java List Methods

- `add()` - adds an element to a list
- `addAll()` - adds all elements of one list to another
- `get()` - helps to randomly access elements from lists
- `iterator()` - returns iterator object that can be used to sequentially access elements of lists
- `set()` - changes elements of lists

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

List Interface

Java List Methods

- `remove()` - removes an element from the list
- `removeAll()` - removes all the elements from the list
- `clear()` - removes all the elements from the list (more efficient than `removeAll()`)
- `size()` - returns the length of lists
- `toArray()` - converts a list into an array
- `contains()` - returns `true` if a list contains specified element

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

List Interface

How to create List

- The ArrayList and LinkedList classes provide the implementation of List interface

```
//Creating a List of type String using ArrayList
List<String> list=new ArrayList<String>();

//Creating a List of type Integer using ArrayList
List<Integer> list=new ArrayList<Integer>();

//Creating a List of type Book using ArrayList
List<Book> list=new ArrayList<Book>();

//Creating a List of type String using LinkedList
List<String> list=new LinkedList<String>();
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

List Interface

How to create List

- The ArrayList and LinkedList classes provide the implementation of List interface
- In short, you can create the List of any type.
- The ArrayList<T> and LinkedList<T> classes are used to specify the type. Here,T denotes the type .

```
//Creating a List of type String using ArrayList
List<String> list=new ArrayList<String>();

//Creating a List of type Integer using ArrayList
List<Integer> list=new ArrayList<Integer>();

//Creating a List of type Book using ArrayList
List<Book> list=new ArrayList<Book>();

//Creating a List of type String using LinkedList
List<String> list=new LinkedList<String>();
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

```java
import java.util.*;
public class ListExample2{
 public static void main(String args[]){
 //Creating a List
 List<String> list=new ArrayList<String>();
 //Adding elements in the List
 list.add("Mango");
 list.add("Apple");
 list.add("Banana");
 list.add("Grapes");
 //accessing the element
 System.out.println("Returning element: "+list.get(1));
 //changing the element
 list.set(1,"Dates");
 //Iterating the List element using for-each loop
 for(String fruit:list)
  System.out.println(fruit);

 }
}
```

```
Returning element: Apple
Mango
Dates
Banana
Grapes
```

KtuQbank

List Interface

```
import java.util.*;
public class ListExample1{
public static void main(String args[]){
 //Creating a List
 List<String> list=new ArrayList<String>();
 //Adding elements in the List
 list.add("Mango");
 list.add("Apple");
 list.add("Banana");
 list.add("Grapes");
 //Iterating the List element using for-each loop
 for(String fruit:list)
  System.out.println(fruit);

}
}
```

```
Mango
Apple
Banana
Grapes
```

```
import java.util.*;
public class ArrayToListExample{
public static void main(String args[]){
//Creating Array
String[] array={"Java","Python","PHP","C++"};
System.out.println("Printing Array: "+Arrays.toString(array));
//Converting Array to List
List<String> list=new ArrayList<String>();
for(String lang:array){
list.add(lang);
}
System.out.println("Printing List: "+list);

}
}
```

```
Printing Array: [Java, Python, PHP, C++]
Printing List: [Java, Python, PHP, C++]
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

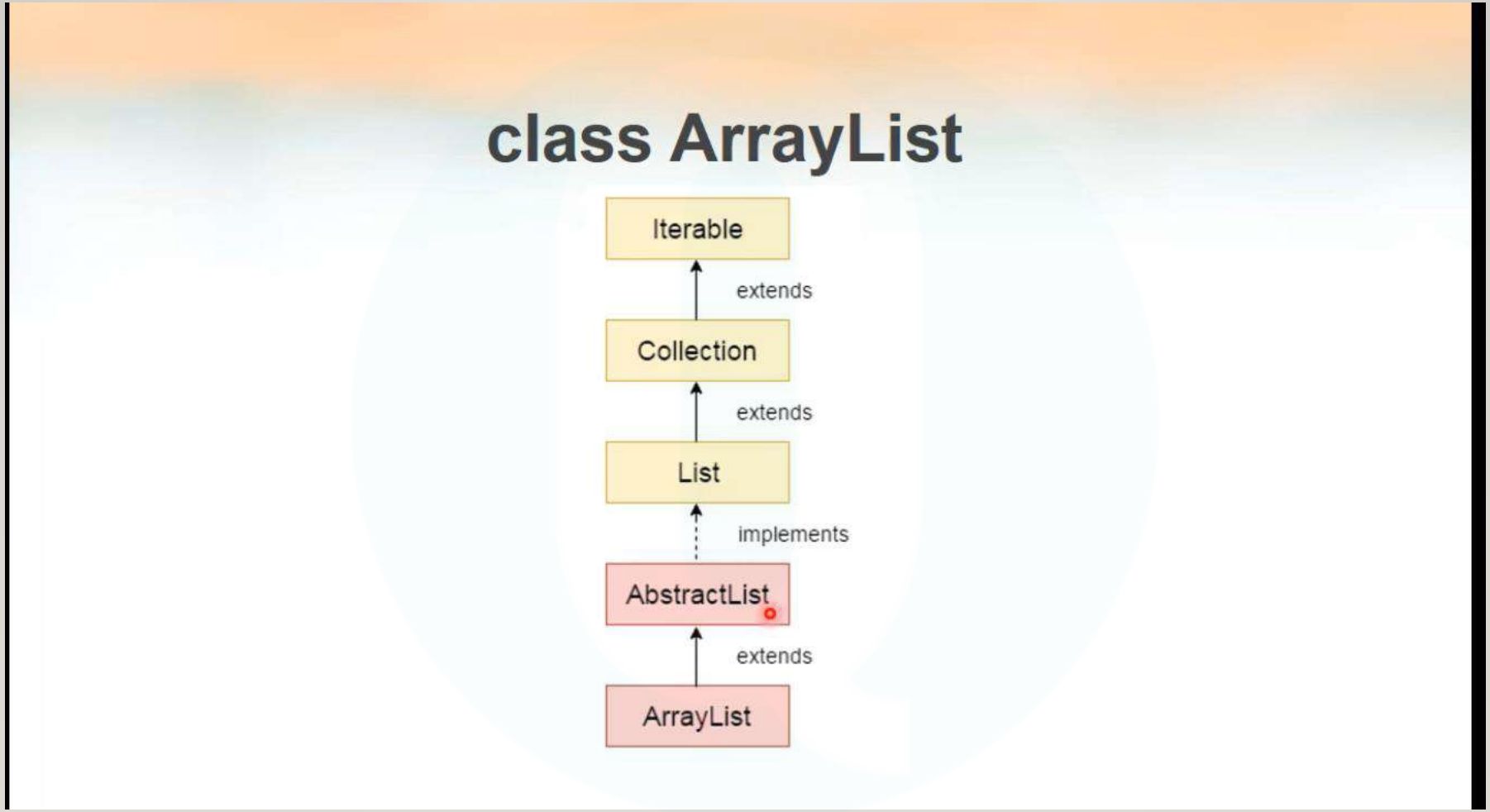
KtuQbank

class ArrayList

- Java ArrayList class uses a dynamic array for storing the elements.
- It is like an array, but there is no size limit. We can add or remove elements anytime.
- It is found in the java.util package.
- The ArrayList in Java can have the duplicate elements also.
- It implements the List interface so we can use all the methods of List interface here.
- The ArrayList maintains the insertion order internally.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank


class ArrayList
Iterable
extends
Collection
extends
List
implements
AbstractList
extends
ArrayList

KtuQbank

KtuQbank

class ArrayList

- The important points about Java ArrayList class are:

1. Java ArrayList class can contain duplicate elements.
2. Java ArrayList class maintains insertion order.
3. Java ArrayList allows random access because array works at the index basis.
4. In ArrayList, manipulation is little bit slower than the LinkedList in Java because a lot of shifting needs to occur if any element is removed from the array list.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

class ArrayList

Constructors of ArrayList

Constructor	Description
ArrayList()	It is used to build an empty array list.
ArrayList(Collection<? extends E> c)	It is used to build an array list that is initialized with the elements of the collection c.
ArrayList(int capacity)	It is used to build an array list that has the specified initial capacity.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

class ArrayList

void add(int index, E element)	It is used to insert the specified element at the specified position in a list.
boolean add(E e)	It is used to append the specified element at the end of a list.
void clear()	It is used to remove all of the elements from this list.
E get(int index)	It is used to fetch the element from the particular position of the list.
boolean isEmpty()	It returns true if the list is empty, otherwise false.
E remove(int index)	It is used to remove the element present at the specified position in the list.
E set(int index, E element)	It is used to replace the specified element in the list, present at the specified position.
void sort(Comparator<? super E> c)	It is used to sort the elements of the list on the basis of specified comparator.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

class ArrayList

Java ArrayList Example

```
import java.util.*;
 public class ArrayListExample1{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
     list.add("Mango");//Adding object in arraylist
     list.add("Apple");
     list.add("Banana");
     list.add("Grapes");
     //Printing the arraylist object
     System.out.println(list);
 }
}
```

Output:

```
[Mango, Apple, Banana, Grapes]
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

class ArrayList

Iterating ArrayList using For-each loop

```
import java.util.*;
public class ArrayListExample3{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
  list.add("Mango");//Adding object in arraylist
  list.add("Apple");
  list.add("Banana");
  list.add("Grapes");
  //Traversing list through for-each loop
  for(String fruit:list)
    System.out.println(fruit);

 }
}
```

```
Mango
Apple
Banana
Grapes
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Iterator Interface

- Iterator Interface is used to traverse a list in forward direction, enabling you to remove or modify the elements of the collection.
- Each of the collection classes provides an iterator( ) method that returns an iterator to the start of the collection.
- By using this iterator object, you can access each element in the collection, one element at a time

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Iterator Interface

- We use the following steps to access a collection of elements using the Iterator.

  - **Step - 1:** Create an object of the Iterator by calling `collection.itertor( )` method.
  - **Step - 2:** Use the method `hasNext( )` to access to check does the collection has the next element. (Use a loop).
  - **Step - 3:** Use the method `next( )` to access each element from the collection. (use inside the loop).

> The `Iterator` allows us to move only forward direction.
>
> The `Iterator` does not support the replacement and addition of new elements.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Iterator Interface

The Methods Declared by Iterator

Sr.No.	Method & Description
1	**boolean hasNext( )** Returns true if there are more elements. Otherwise, returns false.
2	**Object next( )** Returns the next element. Throws NoSuchElementException if there is not a next element.
3	**void remove( )** Removes the current element. Throws IllegalStateException if an attempt is made to call remove( ) that is not preceded by a call to next( ).

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Iterator Interface

- For collections that implement List, you can also obtain an iterator by calling listIterator( ).
- A list iterator gives you the ability to access the collection in either the forward or backward direction and lets you modify an element.
- Otherwise, ListIterator is used just like Iterator.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Iterator Interface

The Methods Declared by ListIterator

Sr.No.	Method & Description
1	**void add(Object obj)** Inserts obj into the list in front of the element that will be returned by the next call to next( ).
2	**boolean hasNext( )** Returns true if there is a next element. Otherwise, returns false.
3	**boolean hasPrevious( )** Returns true if there is a previous element. Otherwise, returns false.
4	**Object next( )** Returns the next element. A NoSuchElementException is thrown if there is not a next element.
5	**int nextIndex( )** Returns the index of the next element. If there is not a next element, returns the size of the list.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Iterator Interface

6	**Object previous( )** Returns the previous element. A NoSuchElementException is thrown if there is not a previous element.
7	**int previousIndex( )** Returns the index of the previous element. If there is not a previous element, returns -1.
8	**void remove( )** Removes the current element from the list. An IllegalStateException is thrown if remove( ) is called before next( ) or previous( ) is invoked.
9	**void set(Object obj)** Assigns obj to the current element. This is the element last returned by a call to either next( ) or previous( ).

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Iterator Interface

```
import java.util.*;
class Demo
{
  public static void main(String[] args)
  {
    ArrayList< String> ar = new ArrayList< String>();
    ar.add("ab");
    ar.add("bc");
    ar.add("cd");
    ar.add("de");
    Iterator it = ar.iterator();    //Declaring Iterator
    while(it.hasNext())
    {
      System.out.print(it.next()+" ");
    }
  }
}
```

OUTPUT:

```
ab bc cd de
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Handling

- Changing the state of an object is known as an event.
- For example, click on button, dragging mouse etc.
- The java.awt.event package provides many event classes and Listener interfaces for event handling.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Handling

Components of Event Handling

- Event handling has **three main components**,

1. **Events** : An event is a change in state of an object.
2. **Events Source** : Event source is an object that generates an event.
3. **Listeners** : A listener is an object that listens to the event.
    1. A listener gets notified when an event occurs.
    2. Event listeners must define the methods to process the notification they are interested to receive.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Handling

How Events are handled?

- A source generates an Event and send it to one or more listeners registered with the source.
- Once event is received by the listener, they process the event and then return.
- Events are supported by a number of Java packages, like java.util, java.awt and java.awt.event.

Steps to handle events:

1. Implement appropriate interface in the class.
2. Register the component with the listener.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Handling

- Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.
- This mechanism have the code which is known as event handler that is executed when an event occurs.
- Java Uses the Delegation Event Model to handle the events.

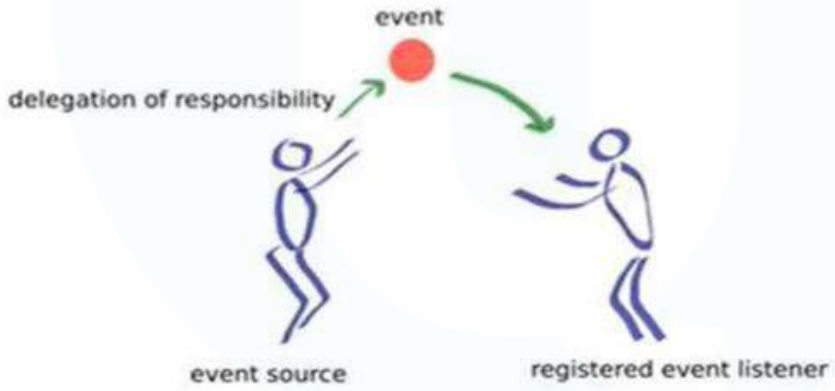
For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

- Its concept is quite simple: a source generates an event and sends it to one or more listeners
- In this scheme, the listener simply waits until it receives an event.
- Once an event is received, the listener processes the event and then returns.



For More Visit : KtuQbank.com | Fair Use Policy
KtuQbank

KtuQbank

Delegation Event Handling

- The advantage - the application logic that processes events is cleanly separated from the user interface logic that generates those events.
- In the delegation event model, listeners must register with a source in order to receive an event notification.
- This provides an important benefit: notifications are sent only to listeners that want to receive them.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

- The Delegation Event Model has the following key participants namely:
- **Source** –
  - The source is an object on which event occurs.
  - Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.
- **Listener** –
  - It is also known as event handler.
  - Listener is responsible for generating response to an event.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

- **Steps involved in event handling**

1. The User clicks the button and the event is generated.
2. Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
3. Event object is forwarded to the method of registered listener class.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

- A source must register listeners in order for the listeners to receive notifications about a specific type of event.
- Each type of event has its own registration method. Here is the general form:

  public void addTypeListener (TypeListener el)

- Here, ***Type*** is the **name of the event**, and ***el*** is a **reference to the event listener**.
- When an event occurs, **all registered listeners are notified** and receive a copy of the event object.
- This is known as **multicasting** the event. In all cases, notifications are sent only to listeners that register to receive them.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

Event classes and interface

Event Classes	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Steps to handle events:

- Implement appropriate interface in the class.
- Register the component with the listener.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

Sources of Events

SOURCES OF EVENT

Event Source	Description
Button	Generates action events when the button is pressed.
Check box	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scroll bar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

Listener Interfaces

Interface	Description
ActionListener	Defines one method to receive action events.
AdjustmentListener	Defines one method to receive adjustment events.
ComponentListener	Defines four methods to recognize when a component is hidden, moved, resized, or shown.
ContainerListener	Defines two methods to recognize when a component is added to or removed from a container.
FocusListener	Defines two methods to recognize when a component gains or loses keyboard focus.
ItemListener	Defines one method to recognize when the state of an item changes.
KeyListener	Defines three methods to recognize when a key is pressed, released, or typed.
MouseListener	Defines five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
MouseMotionListener	Defines two methods to recognize when the mouse is dragged or moved.
MouseWheelListener	Defines one method to recognize when the mouse wheel is moved.
TextListener	Defines one method to recognize when a text value changes.
WindowFocusListener	Defines two methods to recognize when a window gains or loses input focus.
WindowListener	Defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Delegation Event Handling

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener{
TextField tf;
AEvent(){
 //create components
 tf=new TextField();
 tf.setBounds(60,50,170,20);
 Button b=new Button("click me");
 b.setBounds(100,120,80,30);
 //register listener
 b.addActionListener(this);//passing current instance
 //add components and set size, layout and visibility
 add(b);add(tf);
 setSize(300,300);
 setLayout(null);
 setVisible(true);
 }
 public void actionPerformed(ActionEvent e){
 tf.setText("Welcome");
 }
 public static void main(String args[]){
 new AEvent();
 }
 }
```

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- The classes that represent events are at the core of Java's event handling mechanism.
- At the root of the Java event class hierarchy is EventObject, which is in java.util.
- It is the superclass for all events

1	**Object getSource()** The object on which the Event initially occurred.
2	**String toString()** Returns a String representation of this EventObject.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **AWTEvent Class**
- It is the root event class for all AWT events.
- This class is defined in java.awt package.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **ActionEvent class**
- ActionEvent is generated when button is clicked or the item of a list is double clicked.
  - static int ALT_MASK -- The alt modifier.
  - static int CTRL_MASK -- The control modifier.
  - static int META_MASK -- The meta modifier.
  - static int SHIFT_MASK -- The shift modifier.

java.lang.String getActionCommand()

Returns the command string associated with this action.

int getModifiers()

Returns the modifier keys held down during this action event.

long getWhen()

Returns the timestamp of when this event occurred.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **KeyEvent class**
- On entering the character the Key event is generated.
- There are three types of key events which are represented by the integer constants.
- These key events are following
  - KEY_PRESSED
  - KEY_RELASED
  - KEY_TYPED

char getKeyChar()

Returns the character associated with the key in this event.

static String getKeyText(int keyCode)

Returns a String describing the keyCode, such as "HOME", "F1" or "A".

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **MouseEvent class**
- This event indicates a mouse action occurred in a component.
  - a mouse button is pressed
  - a mouse button is released
  - a mouse button is clicked (pressed and released)
  - a mouse cursor enters the unobscured part of component's geometry
  - a mouse cursor exits the unobscured part of component's geometry
  - a mouse is moved
  - the mouse is dragged

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **MouseEvent class**

int getButton()

Returns which, if any, of the mouse buttons has changed state.

int getClickCount()

Returns the number of mouse clicks associated with this event.

int getX()

Returns the horizontal x position of the event relative to the source component.

int getY()

Returns the vertical y position of the event relative to the source component.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Classes

- **MouseMotionEvent class**
- indicates a mouse action occurred in a component.

void mouseDragged(MouseEvent e)

Invoked when a mouse button is pressed on a component and then dragged.

void mouseMoved(MouseEvent e)

Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Sources

Event Source	Description
Button	Generates action events when the button is pressed.
Check box	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scroll bar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Registration Methods

For registering the component with the Listener, many classes provide the registration methods. For example:

- **Button**
  - public void addActionListener(ActionListener a){}
- **MenuItem**
  - public void addActionListener(ActionListener a){}
- **TextField**
  - public void addActionListener(ActionListener a){}
  - public void addTextListener(TextListener a){}
- **TextArea**
  - public void addTextListener(TextListener a){}
- **Checkbox**
  - public void addItemListener(ItemListener a){}
- **Choice**
  - public void addItemListener(ItemListener a){}

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Listener Interfaces

- **ActionListener Interface**
- The class which processes the ActionEvent should implement this interface.
- The object of that class must be registered with a component.
- The object can be registered using the addActionListener() method.
- When the action event occurs, that object's actionPerformed method is invoked.

Interface methods

S.N.	Method & Description
1	**void actionPerformed(ActionEvent e)** Invoked when an action occurs.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Listener Interfaces

- **ItemListener Interface**
- The class which processes the ItemEvent should implement this interface.
- The object of that class must be registered with a component.
- The object can be registered using the addItemListener() method.
- When the action event occurs, that object's itemStateChanged method is invoked.

Interface methods

S.N.	Method & Description
1	**void itemStateChanged(ItemEvent e)** Invoked when an item has been selected or deselected by the user.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Listener Interfaces

- **KeyListener Interface**
- The class which processes the KeyEvent should implement this interface.
- The object of that class must be registered with a component.
- The object can be registered using the addKeyListener() method.

1	**void keyPressed(KeyEvent e)** Invoked when a key has been pressed.
2	**void keyReleased(KeyEvent e)** Invoked when a key has been released.
3	**void keyTyped(KeyEvent e)** Invoked when a key has been typed.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Listener Interfaces

- **MouseListener Interface**
- The class which processes the MouseEvent should implement this interface.
- The object of that class must be registered with a component.
- The object can be registered using the addMouseListener() method.

void mouseClicked(MouseEvent e)

Invoked when the mouse button has been clicked (pressed and released) on a component.

void mouseEntered(MouseEvent e)

Invoked when the mouse enters a component.

void mouseExited(MouseEvent e)

Invoked when the mouse exits a component.

void mousePressed(MouseEvent e)

Invoked when a mouse button has been pressed on a component.

void mouseReleased(MouseEvent e)

Invoked when a mouse button has been released on a component.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Event Listener Interfaces

- **MouseMotionListener Interface**
- used for receiving mouse motion events on a component.
- The class that process mouse motion events needs to implements this interface.

void mouseDragged(MouseEvent e)

Invoked when a mouse button is pressed on a component and then dragged.

void mouseMoved(MouseEvent e)

Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

- Java is a multi-threaded programming language which means we can develop multi-threaded program using Java.
- A multi-threaded program contains two or more parts that can run concurrently
- Each of those part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.
- Each part of such program is called a thread. So, threads are lightweight processes within a process.
- Threads allows a program to operate more efficiently by doing multiple things at the same time

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

- Multiprocessing and multithreading, both are used to achieve multitasking.
- Multitasking is a process of executing multiple tasks simultaneously.
- We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

1. Process-based Multitasking (Multiprocessing)
2. Thread-based Multitasking (Multithreading)

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

Process-based Multitasking (Multiprocessing)

- Each process has an address in memory. In other words, each process allocates a separate memory area.
- A process is heavyweight.
- Cost of communication between the process is high.
- Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

Thread-based Multitasking (Multithreading)

1. Threads share the same address space.
2. A thread is lightweight.
3. Cost of communication between the thread is low.

- Therefore, we use multithreading than multiprocessing because
  - threads use a shared memory area.
  - They don't allocate separate memory area so saves memory, and
  - context-switching between the threads takes less time than process.
- Each of the threads can run in parallel. Java Multithreading is mostly used in games, animation, etc.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

What is Thread in java

- A thread is a lightweight subprocess, the smallest unit of processing.
- It is a separate path of execution.
- Threads are independent.
- If there occurs exception in one thread, it doesn't affect other threads.
- It uses a shared memory area.

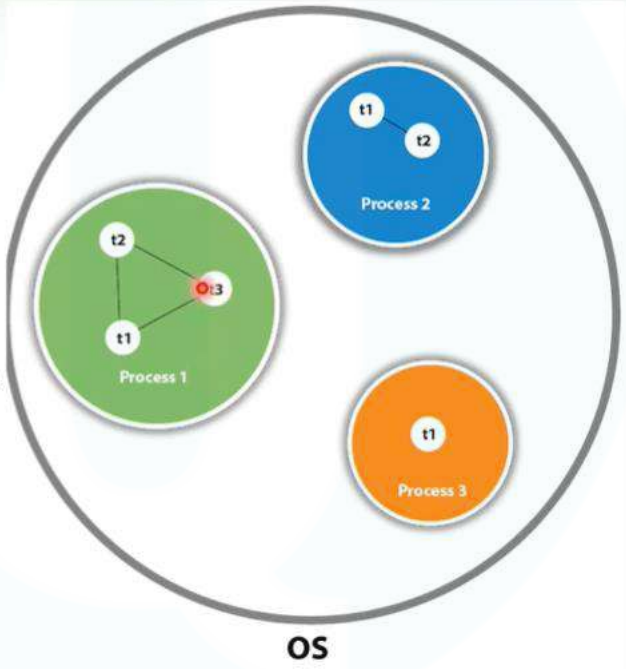
For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Threads

- As shown in the figure, a thread is executed inside the process.
- There is context-switching between the threads
- There can be multiple processes inside the OS, & one process can have multiple threads.



For More Visit : KtuQbank.com | Fair Use Policy
KtuQbank

KtuQbank

Threads

Advantages of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.

2) You can perform many operations together, so it saves time.

3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Life Cycle of Threads

Java Thread class

- Java provides Thread class to achieve thread programming.
- Thread class provides constructors and methods to create and perform operations on a thread.
- Thread class extends Object class and implements Runnable interface.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

KtuQbank

Life Cycle of Threads

Java Thread class

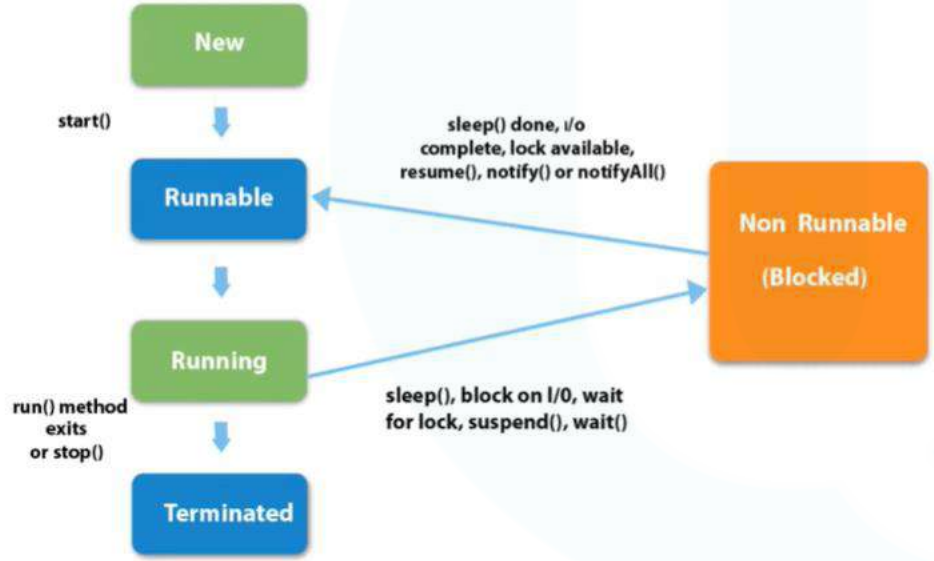
1. void start() : It is used to start the execution of the thread.
2. void run() : It is used to do an action for a thread.
3. static void sleep() : It sleeps a thread for the specified amount of time.
4. int getPriority() : It returns the priority of the thread.
5. void setPriority() : It changes the priority of the thread.
6. void stop() : It is used to stop the thread.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

Life Cycle of Threads

- A thread goes through various stages in its life cycle.



KtuQbank

Life Cycle of Threads

1) New

- The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

- The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

- The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

- This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

- A thread is in terminated or dead state when its run() method exits.

For More Visit : KtuQbank.com | Fair Use Policy

KtuQbank

THREAD

For More Visit : KtuQbank.com | Fair Use Policy