PROGRAMMAZIONE II

E

INGEGNERIA DEL SOFTWARE

A.A. 2017/2018

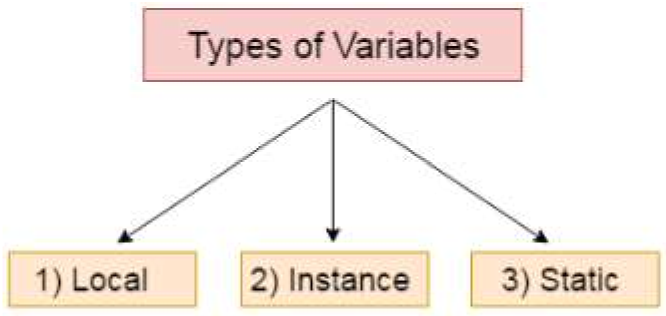
Enrico Martini

Main

Tutte le chiamate al main valide sono:

```
1. public static void main(String[] args)
2. public static void main(String []args)
3. public static void main(String args[])
4. public static void main(String... args)
5. static public void main(String[] args)
6. public static final void main(String[] args)
7. final public static void main(String[] args)
8. final strictfp public static void main(String[] args)
```

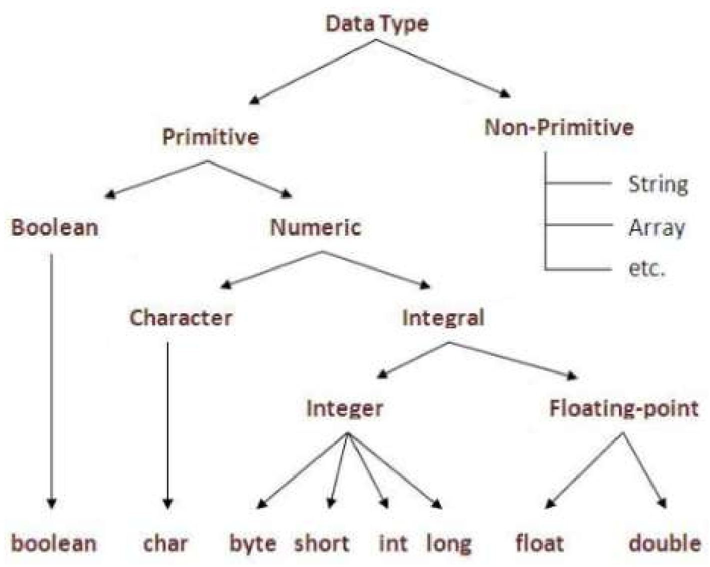
Variabili



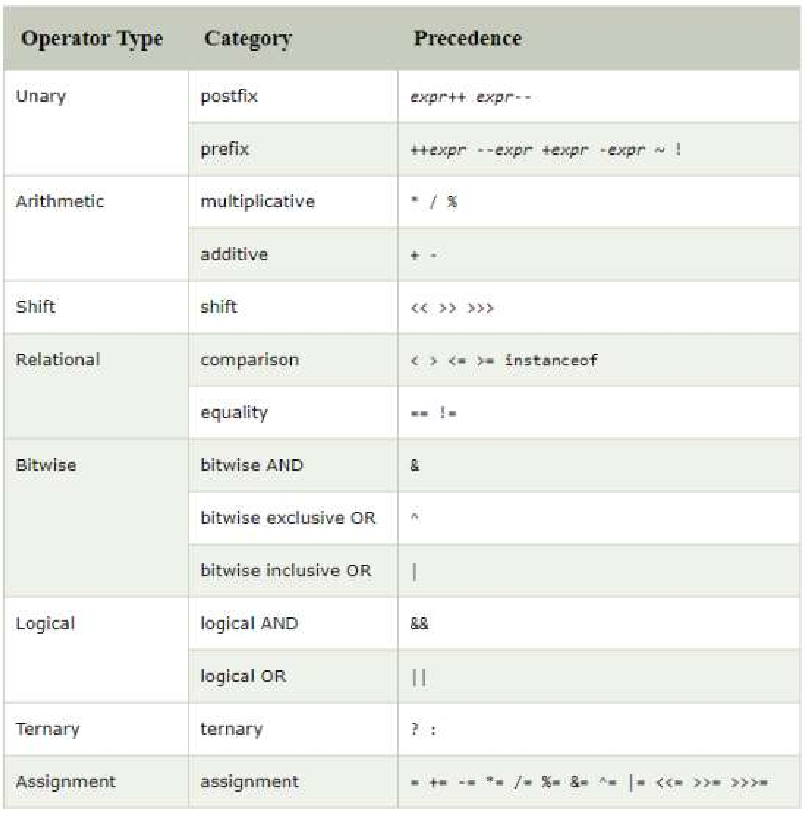
Modificatori:

1) <u>Local</u>: variabile dichiarata dentro ad un metodo (non possono essere *public*);
2) <u>Instance</u>: variabile dichiarata dentro la classe ma fuori dal metodo
3) <u>Static</u>: variabile dichiarata come statica dentro la classe ma fuori dal metodo.

Tipi di dati



Operatori in Java



Operator Type	Category	Precedence
Unary	postfix	`expr++ expr--`
	prefix	`++expr --expr +expr -expr ~ !`
Arithmetic	multiplicative	`* / %`
	additive	`+ -`
Shift	shift	`<< >> >>>`
Relational	comparison	`< > <= >= instanceof`
	equality	`== !=`
Bitwise	bitwise AND	`&`
	bitwise exclusive OR	`^`
	bitwise inclusive OR	`|`
Logical	logical AND	`&&`
	logical OR	`||`
Ternary	ternary	`? :`
Assignment	assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Operatore terziario – (a *condition* b) ? c : d;

```java
class OperatorExample{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}}
```

Costrutti

Costrutto IF-else-if

```
if(condition1){
//code
}else if(condition2){
//code
}
else{
//code
}
```

Switch

```
switch(expression){
case value1:
        //code
        break;
case value2:
        //code
        break;
default:
        //code
}
```

For

```
for(initialization;condition;incr/decr){
//code to be executed
}
```

For-each

Usato per attraversare array o collezioni di java, più semplice di un comune for.

```
for(Type var:array){
//code to be executed
}
```

While

```
while(condition){
//code to be executed
}
```

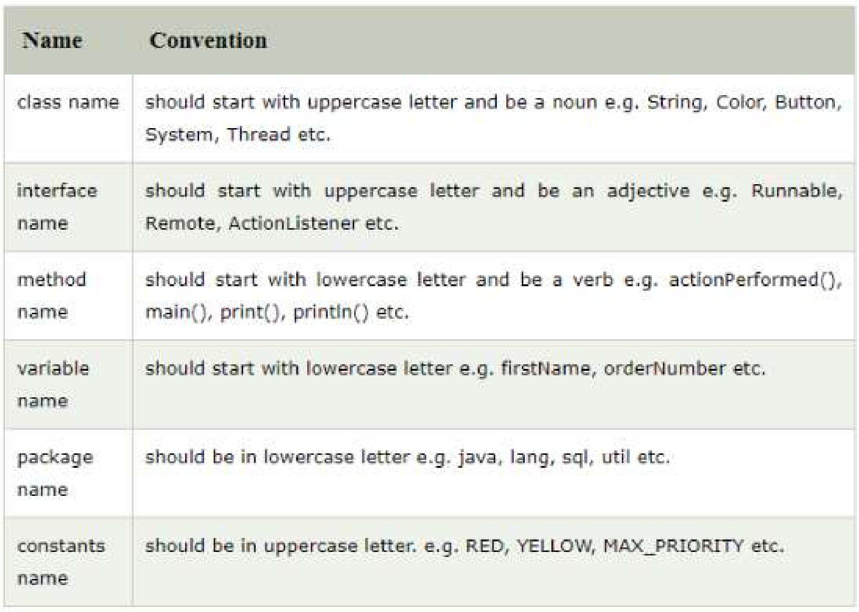
Do-while

```
do{
//code to be executed
}while(condition);
```

Uscire dal Loop

*Per uscire dal loop basta inserire "*break;*".*

Convenzioni sui nomi in java



Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Sintassi di una classe

```
class class_name{
    field;
    method;
}
```

IL METODO COSTRUTTORE

1) Ha lo stesso nome della classe
2) Non ha tipo di ritorno
3) Chiamato automaticamente ogni volta che è istanziato un oggetto
4) Presente in ogni classe
5) Serve a inizializzare le variabili d'istanza
6) È possibile avere più di un costruttore per classe, a patto che il numero di variabili in input differisca.

CASTING

Esiste la "*promotion*" automatica in caso di espressioni, nell'ordine:

- Se uno degli operandi è double, l'altro sarà convertito in double;
- Se uno degli operandi è float, l'altro sarà convertito in float;
- Se uno degli operandi è long, l'altro sarà convertito in long;
- Entrambi gli operandi saranno convertiti in int.

N.B: una variabile *final* è una costante!

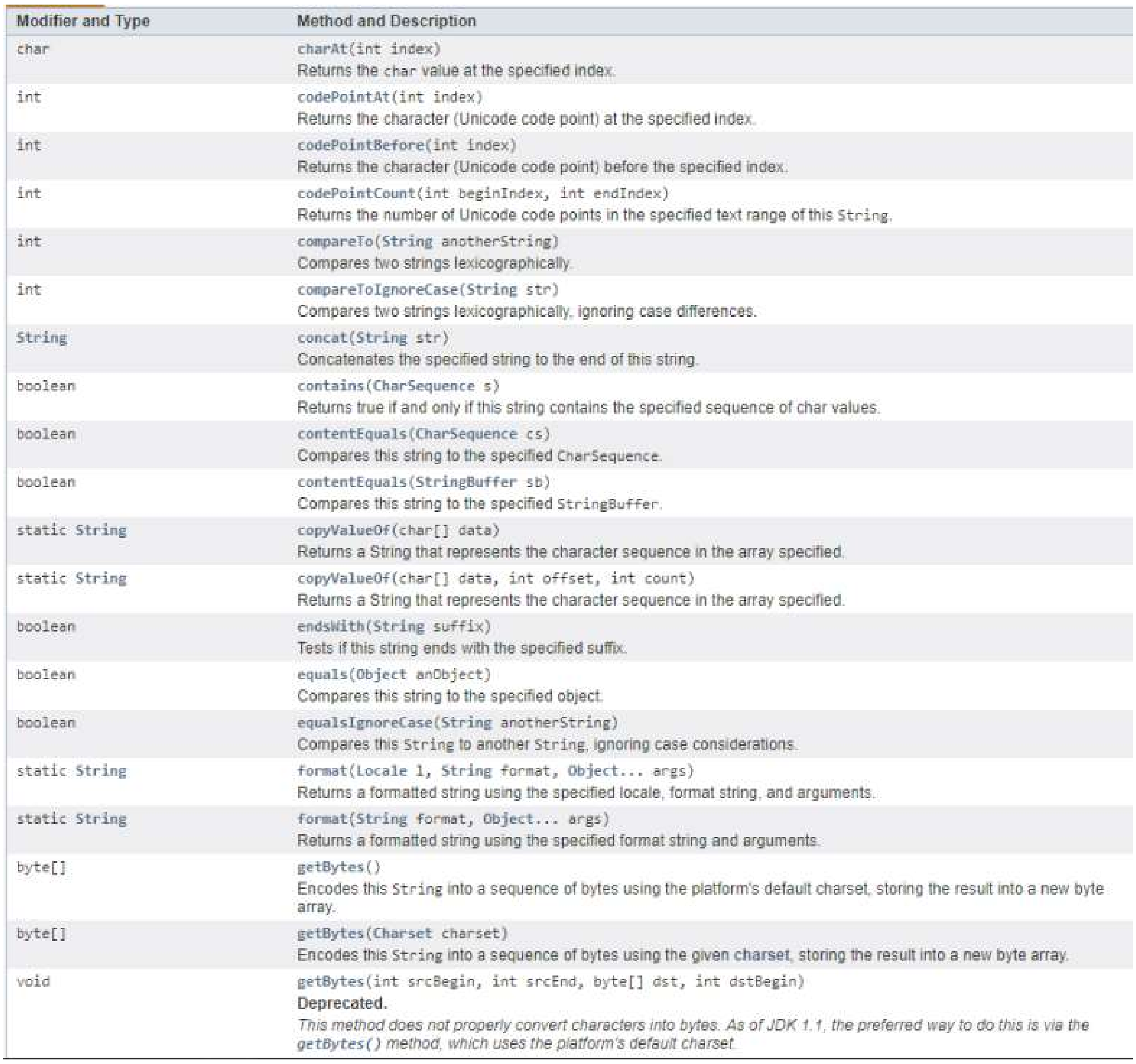
Una variabile *static* è condivisa da tutte le istanze della classe.

LA CLASSE *STRING*

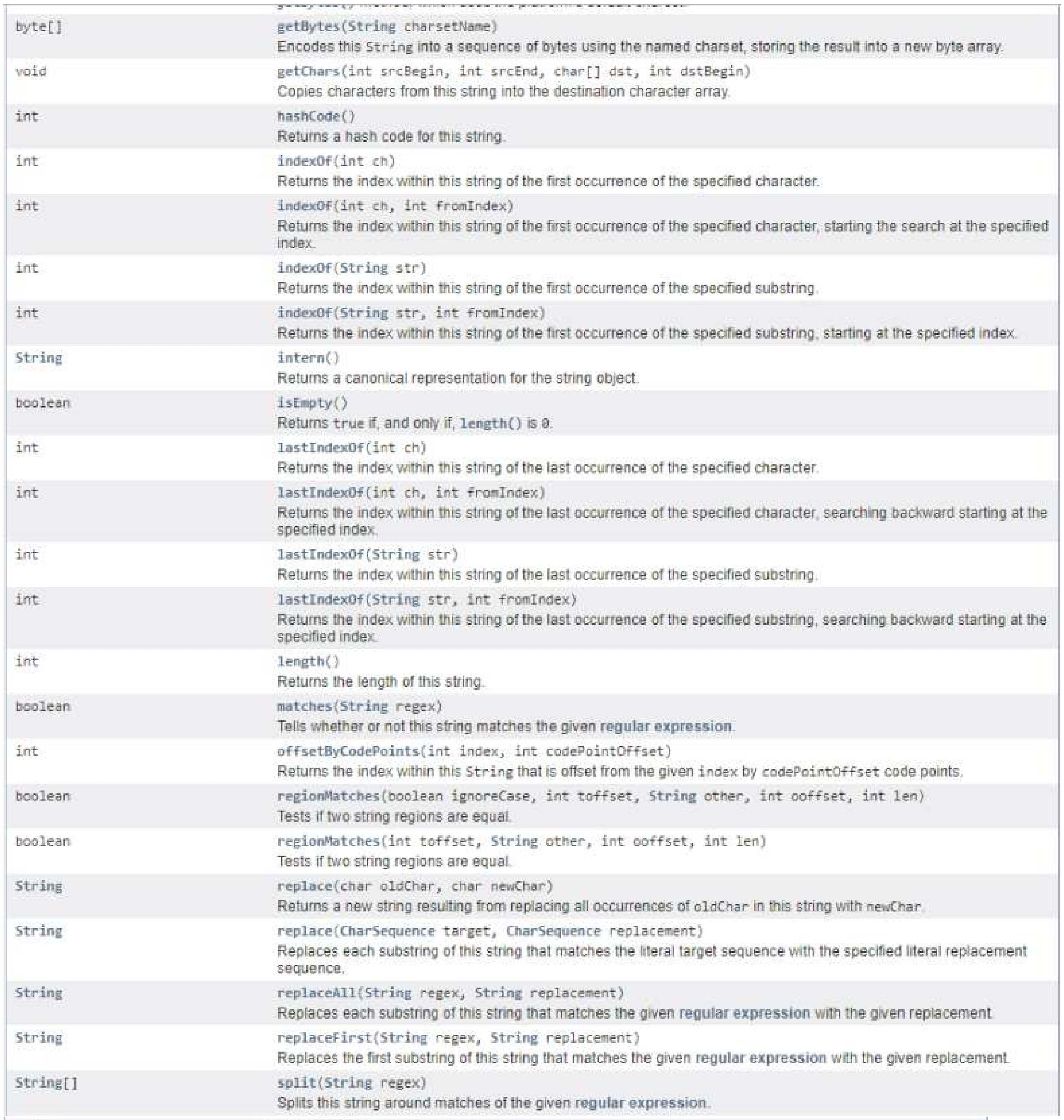
Inizializzazione:

```
String name = new String ("Mario Rossi");
String name = "Mario Rossi";
```

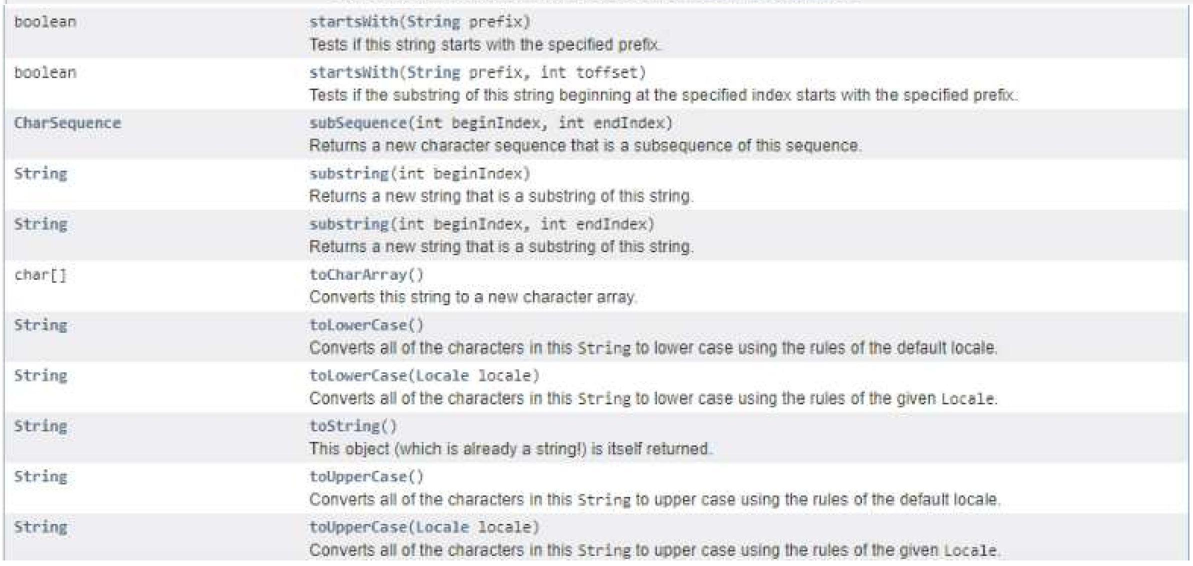
Metodi di String:



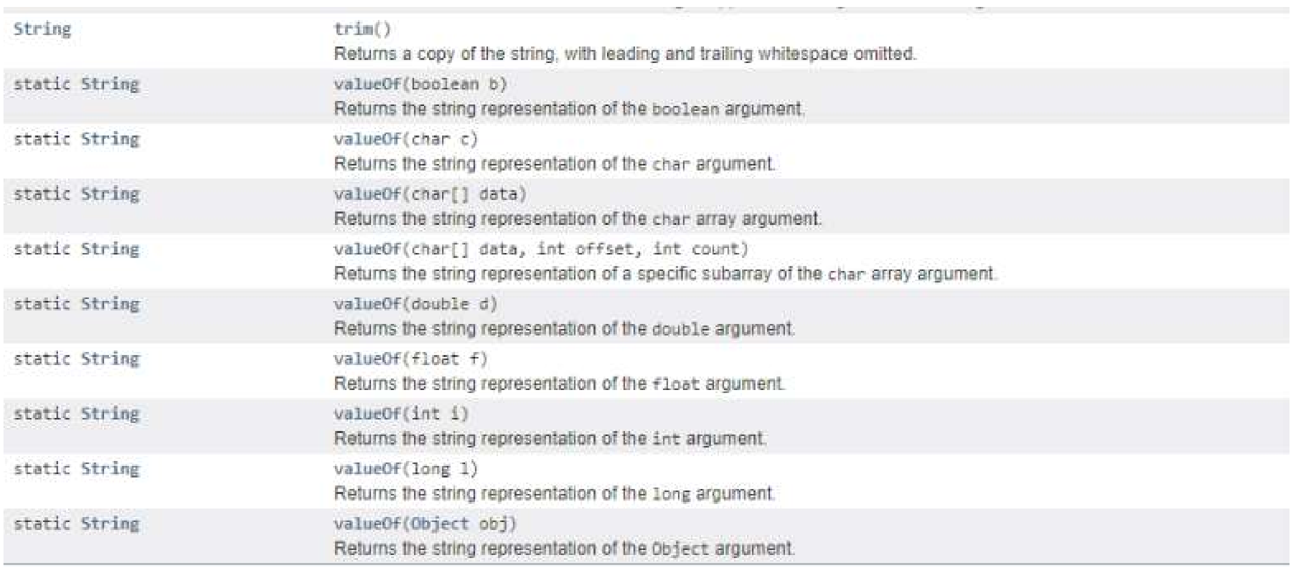
Modifier and Type	Method and Description
`char`	`charAt(int index)` Returns the `char` value at the specified index.
`int`	`codePointAt(int index)` Returns the character (Unicode code point) at the specified index.
`int`	`codePointBefore(int index)` Returns the character (Unicode code point) before the specified index.
`int`	`codePointCount(int beginIndex, int endIndex)` Returns the number of Unicode code points in the specified text range of this `String`.
`int`	`compareTo(String anotherString)` Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case differences.
`String`	`concat(String str)` Concatenates the specified string to the end of this string.
`boolean`	`contains(CharSequence s)` Returns true if and only if this string contains the specified sequence of char values.
`boolean`	`contentEquals(CharSequence cs)` Compares this string to the specified `CharSequence`.
`boolean`	`contentEquals(StringBuffer sb)` Compares this string to the specified `StringBuffer`.
`static String`	`copyValueOf(char[] data)` Returns a String that represents the character sequence in the array specified.
`static String`	`copyValueOf(char[] data, int offset, int count)` Returns a String that represents the character sequence in the array specified.
`boolean`	`endsWith(String suffix)` Tests if this string ends with the specified suffix.
`boolean`	`equals(Object anObject)` Compares this string to the specified object.
`boolean`	`equalsIgnoreCase(String anotherString)` Compares this `String` to another `String`, ignoring case considerations.
`static String`	`format(Locale l, String format, Object... args)` Returns a formatted string using the specified locale, format string, and arguments.
`static String`	`format(String format, Object... args)` Returns a formatted string using the specified format string and arguments.
`byte[]`	`getBytes()` Encodes this `String` into a sequence of bytes using the platform's default charset, storing the result into a new byte array.
`byte[]`	`getBytes(Charset charset)` Encodes this `String` into a sequence of bytes using the given `charset`, storing the result into a new byte array.
`void`	`getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)` **Deprecated.** *This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the `getBytes()` method, which uses the platform's default charset.*



byte[]	`getBytes(String charsetName)` Encodes this `String` into a sequence of bytes using the named charset, storing the result into a new byte array.
void	`getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)` Copies characters from this string into the destination character array.
int	`hashCode()` Returns a hash code for this string.
int	`indexOf(int ch)` Returns the index within this string of the first occurrence of the specified character.
int	`indexOf(int ch, int fromIndex)` Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
int	`indexOf(String str)` Returns the index within this string of the first occurrence of the specified substring.
int	`indexOf(String str, int fromIndex)` Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
String	`intern()` Returns a canonical representation for the string object.
boolean	`isEmpty()` Returns `true` if, and only if, `length()` is `0`.
int	`lastIndexOf(int ch)` Returns the index within this string of the last occurrence of the specified character.
int	`lastIndexOf(int ch, int fromIndex)` Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
int	`lastIndexOf(String str)` Returns the index within this string of the last occurrence of the specified substring.
int	`lastIndexOf(String str, int fromIndex)` Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
int	`length()` Returns the length of this string.
boolean	`matches(String regex)` Tells whether or not this string matches the given regular expression.
int	`offsetByCodePoints(int index, int codePointOffset)` Returns the index within this `String` that is offset from the given `index` by `codePointOffset` code points.
boolean	`regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)` Tests if two string regions are equal.
boolean	`regionMatches(int toffset, String other, int ooffset, int len)` Tests if two string regions are equal.
String	`replace(char oldChar, char newChar)` Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.
String	`replace(CharSequence target, CharSequence replacement)` Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
String	`replaceAll(String regex, String replacement)` Replaces each substring of this string that matches the given regular expression with the given replacement.
String	`replaceFirst(String regex, String replacement)` Replaces the first substring of this string that matches the given regular expression with the given replacement.
String[]	`split(String regex)` Splits this string around matches of the given regular expression.



boolean	`startsWith(String prefix)` Tests if this string starts with the specified prefix.
boolean	`startsWith(String prefix, int toffset)` Tests if the substring of this string beginning at the specified index starts with the specified prefix.
CharSequence	`subSequence(int beginIndex, int endIndex)` Returns a new character sequence that is a subsequence of this sequence.
String	`substring(int beginIndex)` Returns a new string that is a substring of this string.
String	`substring(int beginIndex, int endIndex)` Returns a new string that is a substring of this string.
char[]	`toCharArray()` Converts this string to a new character array.
String	`toLowerCase()` Converts all of the characters in this `String` to lower case using the rules of the default locale.
String	`toLowerCase(Locale locale)` Converts all of the characters in this `String` to lower case using the rules of the given `Locale`.
String	`toString()` This object (which is already a string!) is itself returned.
String	`toUpperCase()` Converts all of the characters in this `String` to upper case using the rules of the default locale.
String	`toUpperCase(Locale locale)` Converts all of the characters in this `String` to upper case using the rules of the given `Locale`.

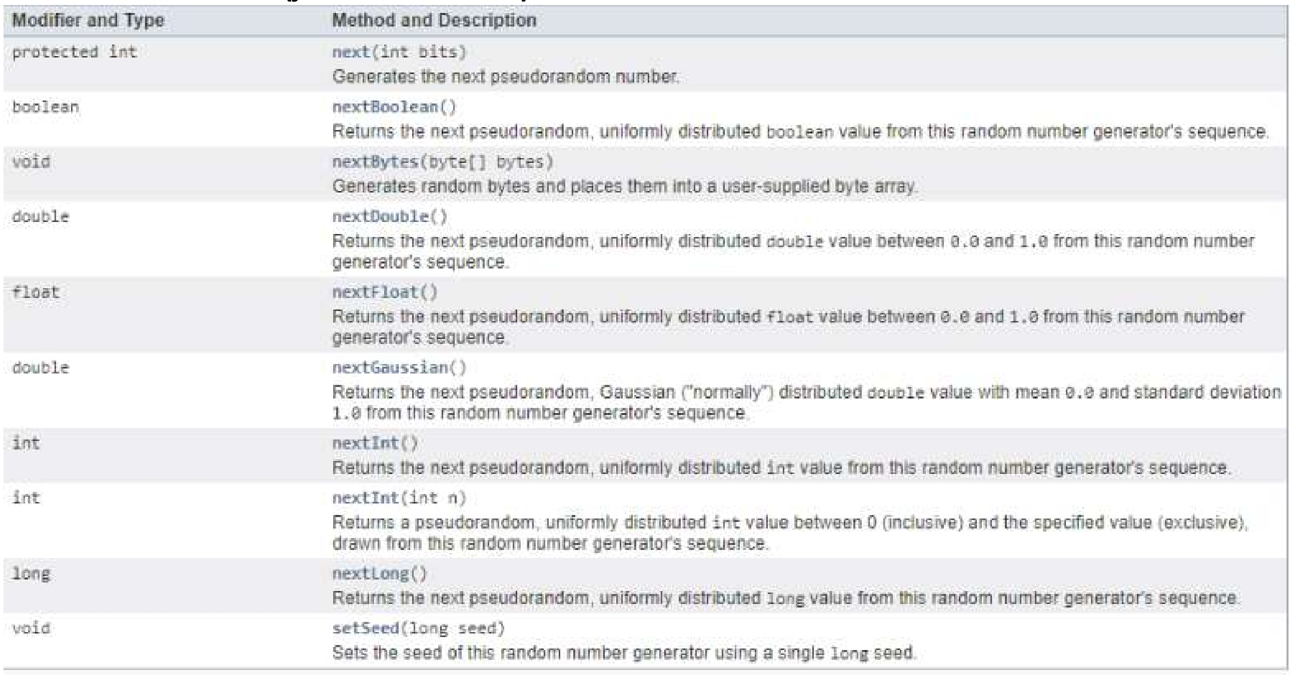


String	trim() Returns a copy of the string, with leading and trailing whitespace omitted.
static String	valueOf(boolean b) Returns the string representation of the boolean argument.
static String	valueOf(char c) Returns the string representation of the char argument.
static String	valueOf(char[] data) Returns the string representation of the char array argument.
static String	valueOf(char[] data, int offset, int count) Returns the string representation of a specific subarray of the char array argument.
static String	valueOf(double d) Returns the string representation of the double argument.
static String	valueOf(float f) Returns the string representation of the float argument.
static String	valueOf(int i) Returns the string representation of the int argument.
static String	valueOf(long l) Returns the string representation of the long argument.
static String	valueOf(Object obj) Returns the string representation of the Object argument.

LA CLASSE SCANNER (*java.util.Scanner*)

```
      import java.util.Scanner;
      …
      Scanner tastiera = new Scanner(System.in);
      String s = tastiera.nextLine();
```

LA CLASSE RANDOM (java.util.Random)



Modifier and Type	Method and Description
protected int	next(int bits) Generates the next pseudorandom number.
boolean	nextBoolean() Returns the next pseudorandom, uniformly distributed boolean value from this random number generator's sequence.
void	nextBytes(byte[] bytes) Generates random bytes and places them into a user-supplied byte array.
double	nextDouble() Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.
float	nextFloat() Returns the next pseudorandom, uniformly distributed float value between 0.0 and 1.0 from this random number generator's sequence.
double	nextGaussian() Returns the next pseudorandom, Gaussian ("normally") distributed double value with mean 0.0 and standard deviation 1.0 from this random number generator's sequence.
int	nextInt() Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
int	nextInt(int n) Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.
long	nextLong() Returns the next pseudorandom, uniformly distributed long value from this random number generator's sequence.
void	setSeed(long seed) Sets the seed of this random number generator using a single long seed.

```
      import java.util.Random;
      …
      private static Random random = new Random();
      …
      int x = ran.nextInt();
```

RIDEFINIRE IL METODO EQUALS

```
	public boolean equals(Object other) {
		return (other instanceof ThisClasse) && (this.value == ((ThisClasse)other).value);
	}
```

VARARGS

Utile per un numero non definito di caratteri. C'è un solo argomento del genere per metodo e deve essere sempre come ultimo.

```
Public void somma(int …interi){
…
}
```

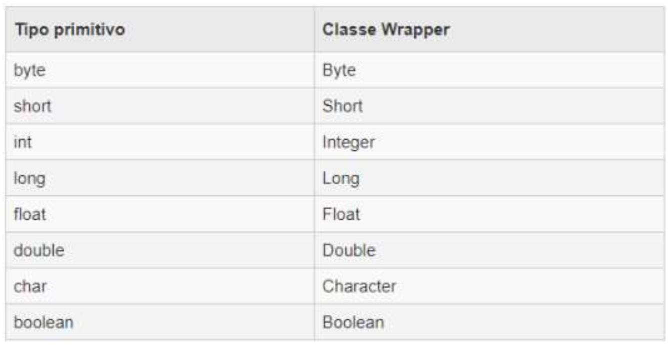
ITERABLE

```
	public Iterator<Date> iterator() {
		return new Iterator<Date>() {
			private int offset;

			public boolean hasNext() {
				return isLeapYear() ? offset <= 365 : offset <= 364;
			}

			public Date next() {
				return new Date(offset++);
			}
		};
	}
```

GESTIONE ECCEZIONI

```
public abstract class MyException extends Exception {
	protected MyException(String message) {
		super(message);
	}
}
```

CLASSI WRAPPER

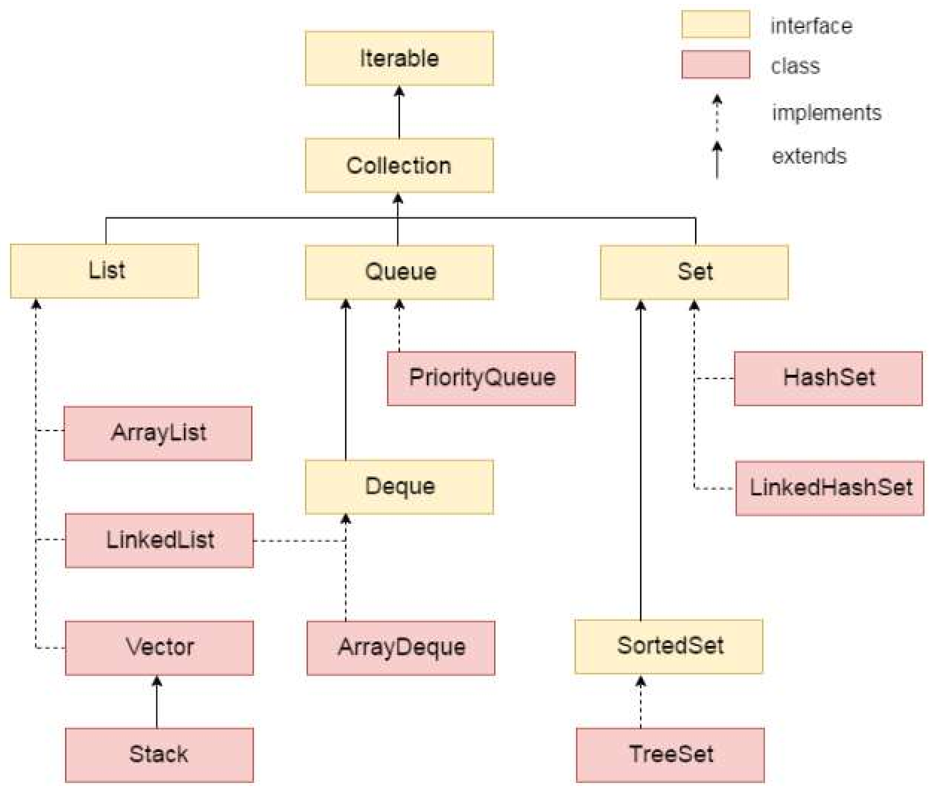


Tipo primitivo	Classe Wrapper
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

CLASSI ANONIME

```
BaseTitle englishTitle= new TitledName() {

    @Override
    public String femaleTitle(String name) {
        return "Ms "+name;
    }

    @Override
    public String maleTitle(String name) {
        return "Mister "+name;
    }
};
```

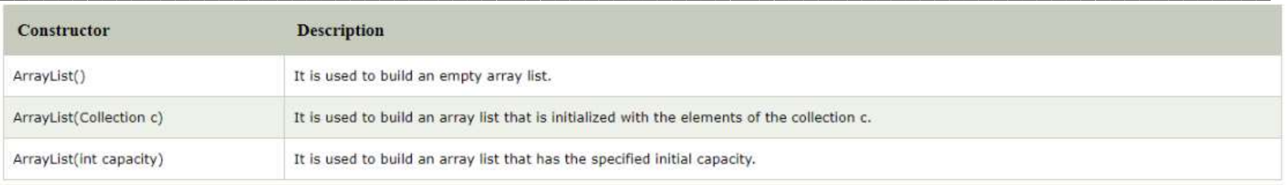
COLLEZIONI



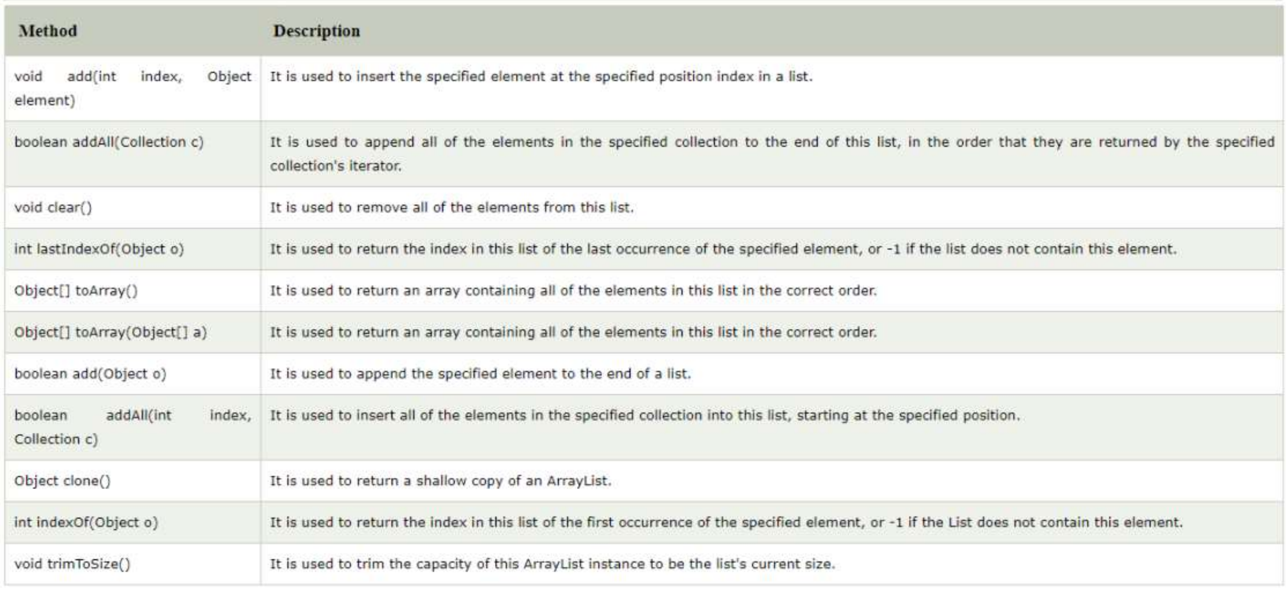
Method	Description
public boolean add(Object element)	is used to insert an element in this collection.
public boolean addAll(Collection c)	is used to insert the specified collection elements in the invoking collection.
public boolean remove(Object element)	is used to delete an element from this collection.
public boolean removeAll(Collection c)	is used to delete all the elements of specified collection from the invoking collection.
public boolean retainAll(Collection c)	is used to delete all the elements of invoking collection except the specified collection.
public int size()	return the total number of elements in the collection.
public void clear()	removes the total no of element from the collection.
public boolean contains(Object element)	is used to search an element.
public boolean containsAll(Collection c)	is used to search the specified collection in this collection.
public Iterator iterator()	returns an iterator.
public Object[] toArray()	converts collection into array.
public boolean isEmpty()	checks if collection is empty.
public boolean equals(Object element)	matches two collection.
public int hashCode()	returns the hashcode number for collection.

ARRAYLIST

```
import java.util.ArrayList;
import java.util.List;
…
      List<Class> name = new ArrayList<>();
```

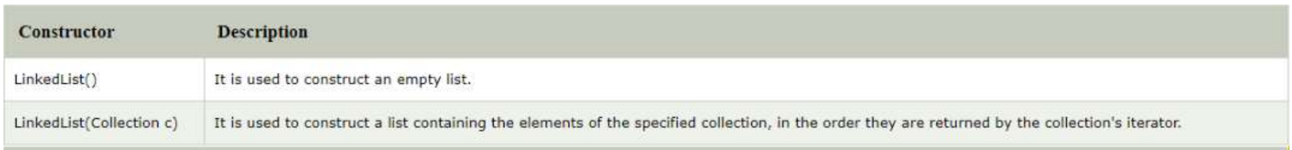


Constructor	Description
ArrayList()	It is used to build an empty array list.
ArrayList(Collection c)	It is used to build an array list that is initialized with the elements of the collection c.
ArrayList(int capacity)	It is used to build an array list that has the specified initial capacity.

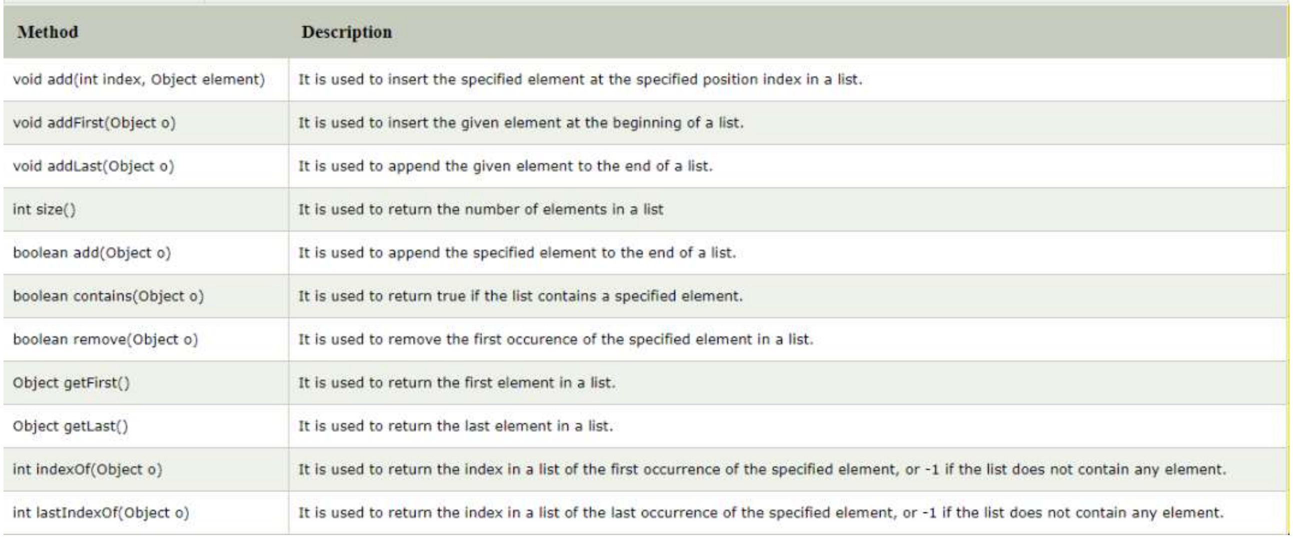


Method	Description
void add(int index, Object element)	It is used to insert the specified element at the specified position index in a list.
boolean addAll(Collection c)	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
void clear()	It is used to remove all of the elements from this list.
int lastIndexOf(Object o)	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
Object[] toArray()	It is used to return an array containing all of the elements in this list in the correct order.
Object[] toArray(Object[] a)	It is used to return an array containing all of the elements in this list in the correct order.
boolean add(Object o)	It is used to append the specified element to the end of a list.
boolean addAll(int index, Collection c)	It is used to insert all of the elements in the specified collection into this list, starting at the specified position.
Object clone()	It is used to return a shallow copy of an ArrayList.
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
void trimToSize()	It is used to trim the capacity of this ArrayList instance to be the list's current size.

LINKEDLIST



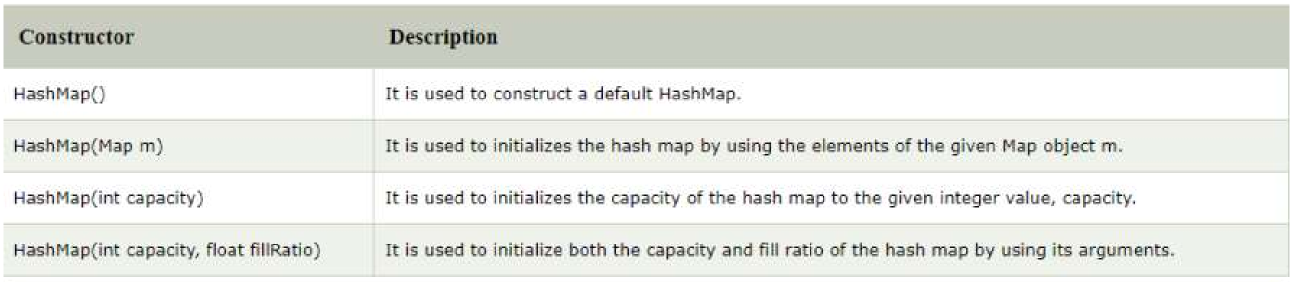
Constructor	Description
LinkedList()	It is used to construct an empty list.
LinkedList(Collection c)	It is used to construct a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.



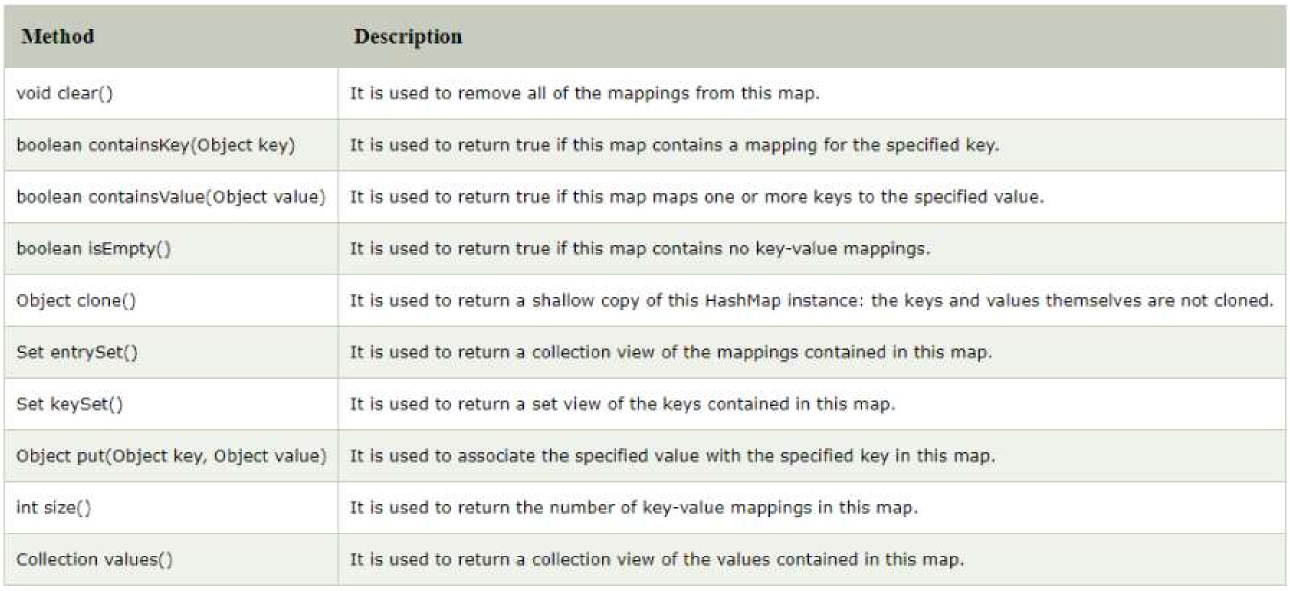
Method	Description
void add(int index, Object element)	It is used to insert the specified element at the specified position index in a list.
void addFirst(Object o)	It is used to insert the given element at the beginning of a list.
void addLast(Object o)	It is used to append the given element to the end of a list.
int size()	It is used to return the number of elements in a list
boolean add(Object o)	It is used to append the specified element to the end of a list.
boolean contains(Object o)	It is used to return true if the list contains a specified element.
boolean remove(Object o)	It is used to remove the first occurence of the specified element in a list.
Object getFirst()	It is used to return the first element in a list.
Object getLast()	It is used to return the last element in a list.
int indexOf(Object o)	It is used to return the index in a list of the first occurrence of the specified element, or -1 if the list does not contain any element.
int lastIndexOf(Object o)	It is used to return the index in a list of the last occurrence of the specified element, or -1 if the list does not contain any element.

HASHMAP

```
import java.util.Map;
import java.util.HashMap;
…
private final Map<Class1, Class2> nome_variabile = new HashMap<>();
```



Constructor	Description
HashMap()	It is used to construct a default HashMap.
HashMap(Map m)	It is used to initializes the hash map by using the elements of the given Map object m.
HashMap(int capacity)	It is used to initializes the capacity of the hash map to the given integer value, capacity.
HashMap(int capacity, float fillRatio)	It is used to initialize both the capacity and fill ratio of the hash map by using its arguments.



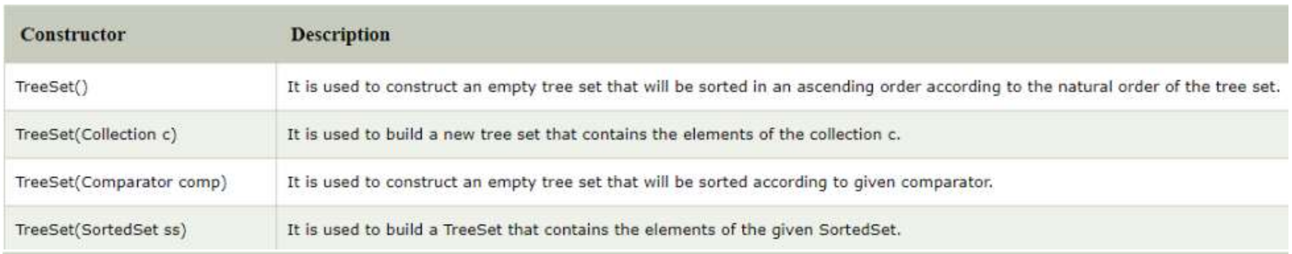
Method	Description
void clear()	It is used to remove all of the mappings from this map.
boolean containsKey(Object key)	It is used to return true if this map contains a mapping for the specified key.
boolean containsValue(Object value)	It is used to return true if this map maps one or more keys to the specified value.
boolean isEmpty()	It is used to return true if this map contains no key-value mappings.
Object clone()	It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
Set entrySet()	It is used to return a collection view of the mappings contained in this map.
Set keySet()	It is used to return a set view of the keys contained in this map.
Object put(Object key, Object value)	It is used to associate the specified value with the specified key in this map.
int size()	It is used to return the number of key-value mappings in this map.
Collection values()	It is used to return a collection view of the values contained in this map.

TREEMAP

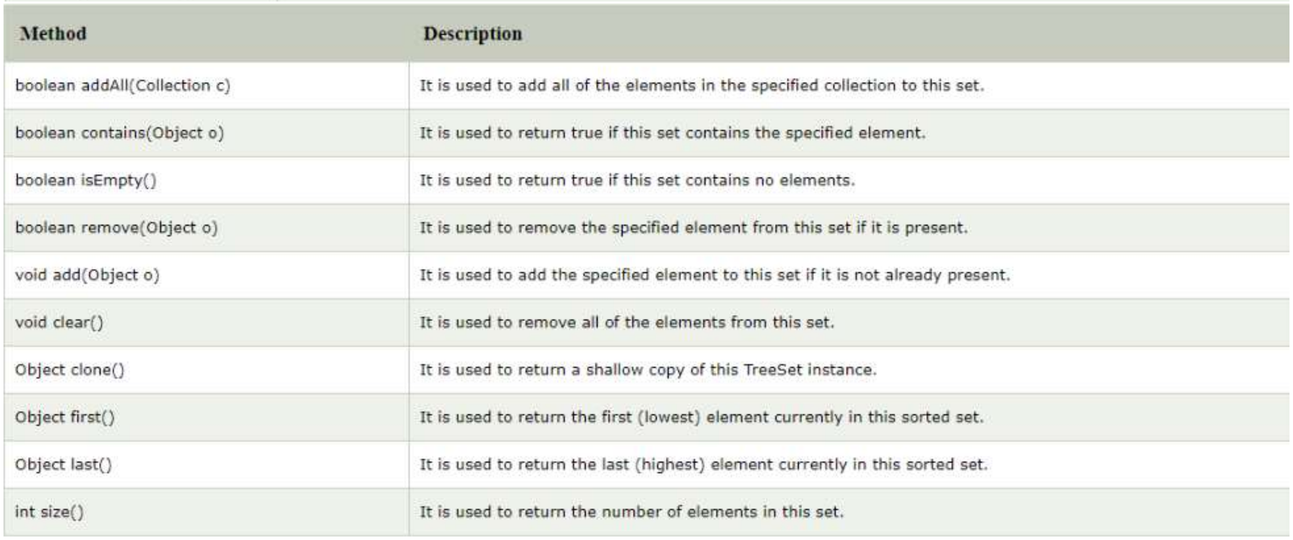
```
import java.util.SortedMap;

import java.util.TreeMap;

…

private SortedMap<Class1, Class2> nome_variabile = new TreeMap<>();
```

TREESET

```
import java.util.SortedSet;

import java.util.TreeSet;

…

       SortedSet<String> result = new TreeSet<>();
```



Constructor	Description
TreeSet()	It is used to construct an empty tree set that will be sorted in an ascending order according to the natural order of the tree set.
TreeSet(Collection c)	It is used to build a new tree set that contains the elements of the collection c.
TreeSet(Comparator comp)	It is used to construct an empty tree set that will be sorted according to given comparator.
TreeSet(SortedSet ss)	It is used to build a TreeSet that contains the elements of the given SortedSet.



Method	Description
boolean addAll(Collection c)	It is used to add all of the elements in the specified collection to this set.
boolean contains(Object o)	It is used to return true if this set contains the specified element.
boolean isEmpty()	It is used to return true if this set contains no elements.
boolean remove(Object o)	It is used to remove the specified element from this set if it is present.
void add(Object o)	It is used to add the specified element to this set if it is not already present.
void clear()	It is used to remove all of the elements from this set.
Object clone()	It is used to return a shallow copy of this TreeSet instance.
Object first()	It is used to return the first (lowest) element currently in this sorted set.
Object last()	It is used to return the last (highest) element currently in this sorted set.
int size()	It is used to return the number of elements in this set.