PROGRAMMING

Lecture 8

Sushil Paudel


Doctors: Googling stuff online does not make you a doctor.
Programmers:

PREVIOUS TOPIC

- Encapsulation
- Array
- Class Diagram

TODAY'S TOPIC

- Static methods
- Wrapper Classes
- String class
- Array List

STATIC METHODS

- If you apply static keyword with any method, it is known as static method.
  - A static method belongs to the class rather than the object of a class.
  - A static method can be invoked without the need for creating an instance of a class.
  - A static method can access static data member and can change the value of it.

STATIC METHODS

```
public class Utils{

    public static int add(int a, int b){
        return a + b;
    }

    public int subtract(int a, int b){
        return a - b;
    }

    public static void main(String[] args){
        Utils util = new Utils();
        int subtractResult = util.subtract(100, 20);
        System.out.println("Subtract Result: " + subtractResult);

        int addResult = add(10, 20);
        System.out.println("Add Result: " + addResult);

    }
}
```

STATIC METHODS

```
public class UtilsTest
{

    public static void main(String[] args){
        Utils util = new Utils();
        int subtractResult = util.subtract(100, 20);
        System.out.println("Subtract Result: " + subtractResult);

        int addResult = Utils.add(10, 20);
        System.out.println("Add Result: " + addResult);

    }
}
```

WRAPPER CLASSES

- Normally, when we work with Numbers, we use primitive data types such as byte, int, long, double, etc. Example:

```
int i = 5000;

float f = 13.65f;

double d =13.3;
```

- However, in development, we come across situations where we need to use objects instead of primitive data types. In order to achieve this, Java provides wrapper classes.

WRAPPER CLASSES

- Converting primitive data types into object is called boxing, and this is taken care by the compiler.
- And the Wrapper object will be converted back to a primitive data type, and this process is called unboxing.

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

CONVERT STRING TO NUMBER

```
String number = "10";
int result = Integer.parseInt(number);
System.out.println(result);
```

FOR DOUBLE AND FLOAT

```
String number = "10.1";

double result = Double.parseDouble(number);

float result = Float.parseFloat(number);
```

CONVERT NUMBER TO STRING

```
int number = 782;

String result = String.valueOf(number);
```

STRING CLASS

- The java.lang.String class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.
- Here is the list of few methods by String class –

FEW STRING METHODS

1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length
3	String substring(int beginIndex)	returns substring for given begin index.
4	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.

EXAMPLE

```
class StringExample{
        public static void main(String args[]){

                String s1=”Ram";
                String s2=”Ram";
                String s3=new String(”Ram");
                String s4=”Shyam";

                System.out.println(s1.equals(s2));//true
                System.out.println(s1.equals(s3));//true
                System.out.println(s1.equals(s4));//false
        }
}
```

SUBSTRING

- A part of string is called substring. In other words, substring is a subset of another string. In case of substring startIndex is inclusive and endIndex is exclusive.
- You can get substring from the given string object by one of the two methods:
  - **public String substring(int startIndex)**: This method returns new String object containing the substring of the given string from specified startIndex (inclusive).
  - **public String substring(int startIndex, int endIndex)**: This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

STRING

```
String s="hello";

System.out.println(s.substring(0,2));//he
```

```
String s=”ICP";

System.out.println(s.length());//3
```

```
String s=”Informatics";

System.out.println(s.charAt(0));//I

System.out.println(s.charAt(3));//o
```

STRING EXAMPLE

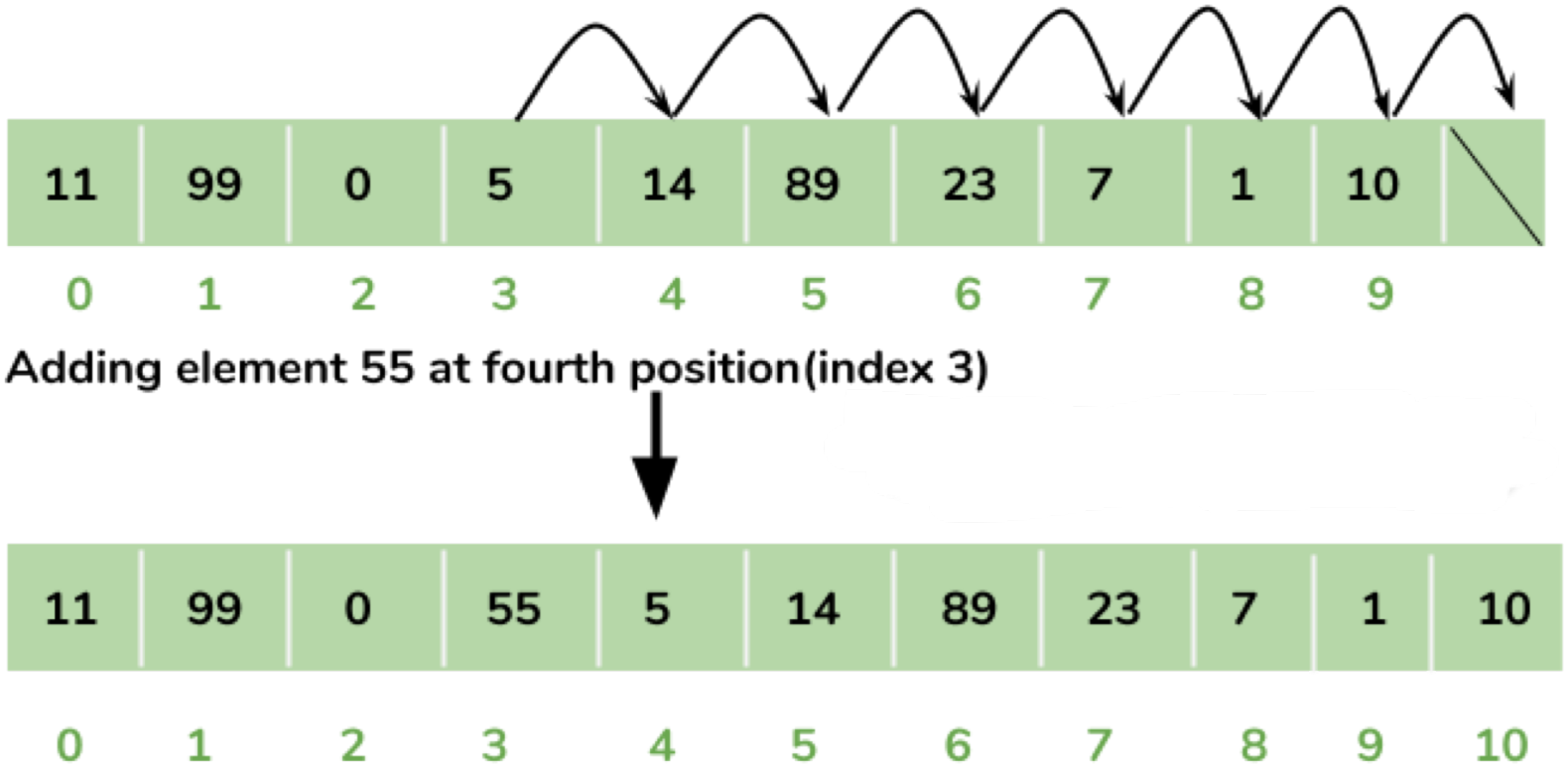
```
String s=”Ram";

System.out.println(s.toUpperCase());//RAM

System.out.println(s.toLowerCase());//ram

System.out.println(s);//Ram(no change in original)
```

ARRAYLIST

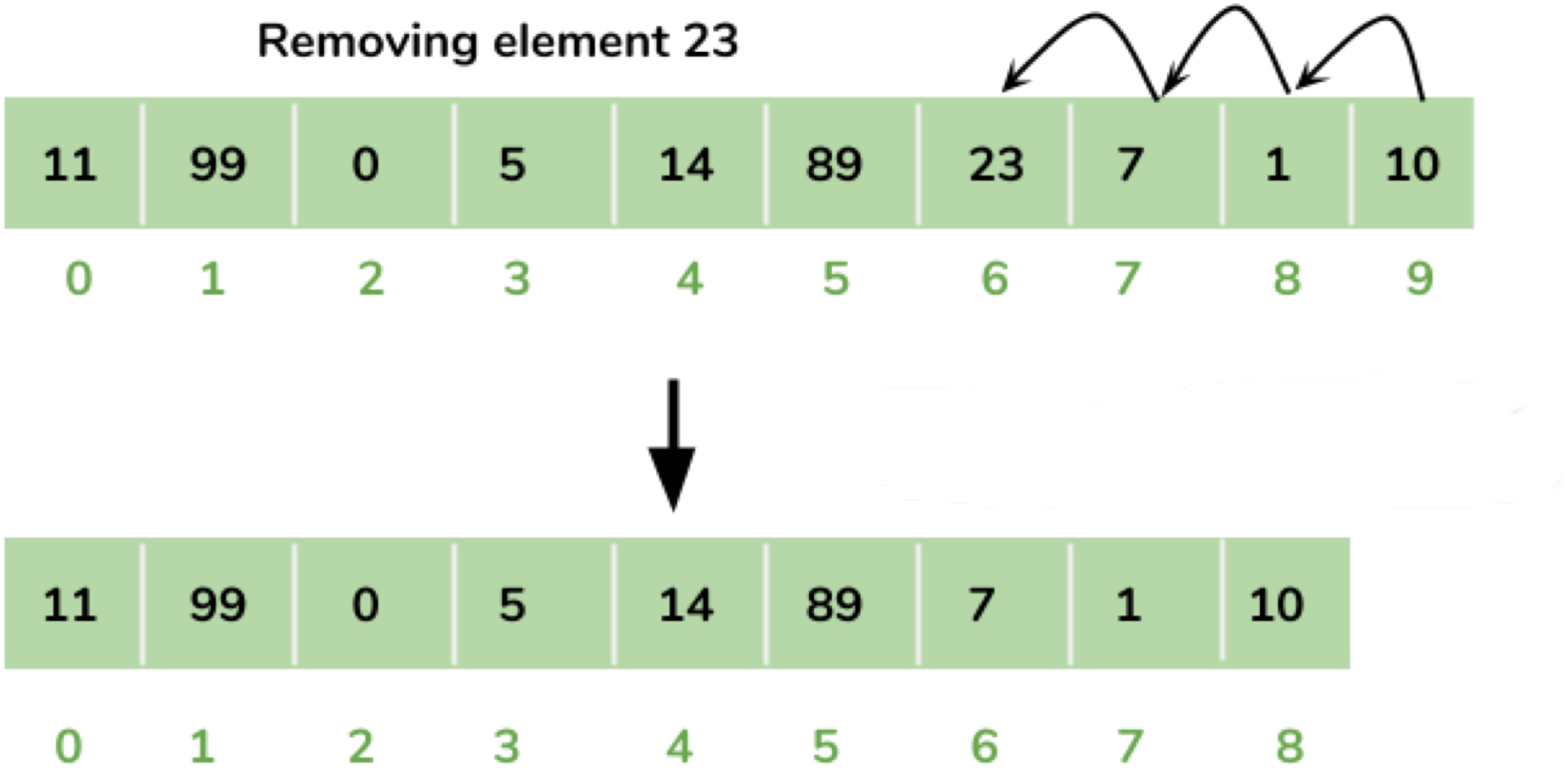
ArrayList in Java is used to store dynamically sized collection of elements. Contrary to Arrays that are fixed in size, an ArrayList grows its size automatically when new elements are added to it. Following are few key points to note about ArrayList in Java –

- An ArrayList is a re-sizable array, also called a dynamic array.
- Just like arrays, it allows you to retrieve the elements by their index.
- Java ArrayList is an ordered collection. It maintains the insertion order of the elements.
- You cannot create an ArrayList of primitive types like int, char etc. You need to use boxed types like Integer, Character, Boolean etc.
- ArrayList has to be imported from **java.util.ArrayList** package.

ADDING ELEMENT IN ARRAYLIST



REMOVING ELEMENTS FROM ARRAYLIST



HOW TO CREATE ARRAYLIST?

- We can create an ArrayList by writing a simple statement like this:

```
ArrayList<String> list=new ArrayList<String>();
```

- Similarly we can create ArrayList that accepts Integer elements.

```
ArrayList<Integer> list=new ArrayList<Integer>();
```

ADDING ELEMENTS IN ARRAYLIST

- We add elements to an ArrayList by using add() method, this method has couple of variations, which we can use based on the requirement. For example: If we want to add the element at the end of the List then simply do it like this:

```
alist.add("Steve"); //This will add "Steve" at the end of List
```

- To add the element at the specified location in ArrayList, we can specify the index in

```
alist.add(2, "Steve"); //This will add "Steve" at third position
```

EXAMPLE

```
import java.util.ArrayList;

class JavaExample{
   public static void main(String args[]){
      ArrayList<String> alist=new ArrayList<String>();
      alist.add("Steve");
      alist.add("Tim");
      alist.add("Lucy");
      alist.add("Pat");
      alist.add("Angela");
      alist.add("Tom");

      //displaying elements
      System.out.println(alist);

      //Adding "Steve" at the fourth position
      alist.add(3, "Steve");

      //displaying elements
      System.out.println(alist);
   }
}
```

OUTPUT

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Steve, Tim, Lucy, Steve, Pat, Angela, Tom]

REMOVE ELEMENTS FROM ARRAYLIST

- We use remove() method to remove elements from an ArrayList, Same as add() method, this method also has few variations.

- Example -

REMOVE ELEMENTS FROM ARRAYLIST

```
import java.util.ArrayList;
class JavaExample{
   public static void main(String args[]){
      ArrayList<String> alist=new ArrayList<String>();
      alist.add("Steve");
      alist.add("Tim");
      alist.add("Lucy");
      alist.add("Pat");
      alist.add("Angela");
      alist.add("Tom");

      System.out.println(alist); //displaying elements

      alist.remove("Steve"); //Removing "Steve"
      alist.remove("Angela"); //Removing "Angela"

      System.out.println(alist); //displaying elements

      //Removing 3rd element
      alist.remove(2);

      System.out.println(alist); //displaying elements
   }
}
```

OUTPUT

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Tim, Lucy, Pat, Tom]

[Tim, Lucy, Tom]

ITERATING ARRAYLIST

- In the above examples, we have displayed the ArrayList elements just by referring the ArrayList instance.
- Now, we will learn to display the elements is by using an advanced for loop (for each) like this.

EXAMPLE

```
import java.util.*;

class JavaExample{

  public static void main(String args[]){
     ArrayList<String> alist=new ArrayList<String>();
     alist.add(”Ram");
     alist.add(”Laxman");
     alist.add(”Sita");
     alist.add(”Bharat");
     alist.add(”Rawan");

     //iterating ArrayList
     for(String element:alist)
        System.out.println(element);
     }
}
```

OUTPUT

Ram

Laxman

Sita

Bharat

Rawan

METHODS OF ARRAYLIST

In the above example we have used methods such as add() and remove(). However there are number of methods available which can be used directly using object of ArrayList class. Let's discuss few important methods of ArrayList class.

1) **add(Object o)**: This method adds an object o to the arraylist.

```
list.add("hello");
```

This statement would add a string hello in the arraylist at last position.

ARRAYLIST METHODS

2) add(int index, Object o): It adds the object o to the array list at the given index.

```
list.add(2, "bye");
```

It will add the string bye to the 2nd index (3rd position as the array list starts with index 0) of array list.

ARRAYLIST METHODS

3) **remove(Object o)**: Removes the object o from the ArrayList.

```
list.remove(”Ram");
```

This statement will remove the string “Chaitanya” from the ArrayList.

ARRAYLIST METHODS

4) **remove(int index)**: Removes element from a given index.

```
list.remove(3);
```

It would remove the element of index 3 (4th element of the list – List starts with o).

ARRAYLIST METHODS

4) **set(int index)**: Replaces the element at the specified index.

```
list.set(3, "Ram");
```

It would replace the value at index 3 by Ram.

ARRAYLIST METHODS

6) **int indexOf(Object o)**: Gives the index of the object o. If the element is not found in the list then this method returns the value -1.

```
int pos = list.indexOf("Tom");
```

- This would give the index (position) of the string Tom in the list.

ARRAYLIST METHODS

- **Object get(int index)**: It returns the object of list which is present at the specified index.

```
String str= list.get(2);
```

- Function get would return the string stored at 3rd position (index 2) and would be assigned to the string "str".
- We have stored the returned value in string variable because in our example we have defined the ArrayList is of String type.
- If you are having integer array list then the returned value should be stored in an integer variable.

STRING CLASS

8) **int size()**: It gives the size of the ArrayList – Number of elements of the list.

```
int numberofitems = list.size();
```

STRING CLASS

9) **boolean contains(Object o)**: It checks whether the given object o is present in the array list if its there then it returns true else it returns false.

```
list.contains("Steve");
```

It would return true if the string “Steve” is present in the list else we would get false.

STRING CLASS

10) **clear():** It is used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList.

```
list.clear();
```

EXAMPLE

```
import java.util.ArrayList;

public class ArrayListPractice
{
    public static void main(String[] args) {
        ArrayList<String> cars = new ArrayList<String>();
        cars.add("Volvo");
        cars.add("BMW");
        cars.add("Ford");
        cars.add("Mazda");

        System.out.println(cars);

        // Another way of printing
        for(String c: cars){
            System.out.println(c);
        }

        System.out.println("Contains Tata? "+cars.contains("Tata"));

        cars.remove("Mazda");
        System.out.println("After removing Mazda: "+cars);
        cars.remove(0);

        System.out.println("After removing 0 index: "+cars);
    }
}
```

OUTPUT

[Volvo, BMW, Ford, Mazda]

Volvo

BMW

Ford

Mazda

Contains Tata? false

After removing Mazda: [Volvo, BMW, Ford]

After removing 0 index: [BMW, Ford]

THANK YOU!

Any questions?