Session 05:

JAVA Arrays and Strings

# Describe ArrayList and accessing values from an ArrayList

# Describe String and StringBuilder classes

# Explain command line arguments

# Describe Wrapper classes, autoboxing, and unboxing

# Describe ArrayList and accessing values from an ArrayList \

## Java ArrayList

import java.util.ArrayList; // import the ArrayList class

ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object

## Add Items

import java.util.ArrayList;

public class Main {

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);

}

}

## Access an Item

cars.get(0);

## Change an Item

cars.set(0, "Opel");

## Remove an Item

cars.remove(0);

cars.clear();

## ArrayList Size

cars.size();

## Loop Through an ArrayList

public class Main {

public static void main(String[] args) {

ArrayList<String> cars = new ArrayList<String>();

cars.add("Volvo");

cars.add("BMW");

cars.add("Ford");

cars.add("Mazda");

for (int i = 0; i < cars.size(); i++) {

System.out.println(cars.get(i));

}

}

}

Foroech

public class Main {

public static void main(String[] args) {

ArrayList<String> cars = new ArrayList<String>();

cars.add("Volvo");

cars.add("BMW");

cars.add("Ford");

cars.add("Mazda");

for (String i : cars) {

System.out.println(i);

}

}

}

### Create an ArrayList to store numbers (add elements of type Integer):

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Integer> myNumbers = new ArrayList<Integer>();

myNumbers.add(10);

myNumbers.add(15);

myNumbers.add(20);

myNumbers.add(25);

for (int i : myNumbers) {

System.out.println(i);

}

}

}

## Sort an ArrayList

import java.util.ArrayList;

import java.util.Collections; // Import the Collections class

public class Main {

public static void main(String[] args) {

ArrayList<String> cars = new ArrayList<String>();

cars.add("Volvo");

cars.add("BMW");

cars.add("Ford");

cars.add("Mazda");

Collections.sort(cars); // Sort cars

for (String i : cars) {

System.out.println(i);

}

}

}

### Example

#### Sort an ArrayList of Integers:

import java.util.ArrayList;

import java.util.Collections; // Import the Collections class

public class Main {

public static void main(String[] args) {

ArrayList<Integer> myNumbers = new ArrayList<Integer>();

myNumbers.add(33);

myNumbers.add(15);

myNumbers.add(20);

myNumbers.add(34);

myNumbers.add(8);

myNumbers.add(12);

Collections.sort(myNumbers); // Sort myNumbers

for (int i : myNumbers) {

System.out.println(i);

}

}

}

# Describe String and StringBuilder classes

## Java String length() Method

* The length() method is used to find the length of a string. For example, String str = ”Hello”; System.out.println(str.length()); // output: 5

## Java String charAt() Method

Returns the character at the specified index (position)

public class Main {

public static void main(String[] args) {

String myStr = "Hello";

char result = myStr.charAt(0);

System.out.println(result);

}

}

## Java String concat() Method

Appends a string to the end of another string

public class Main {

public static void main(String[] args) {

String firstName = "John ";

String lastName = "Doe";

System.out.println(firstName.concat(lastName));

}

}

## Java String compareTo() Method

Compares two strings lexicographically

public class Main {

public static void main(String[] args) {

String myStr1 = "Hello";

String myStr2 = "Hello";

System.out.println(myStr1.compareTo(myStr2)); // Returns 0 because they are equal

}

}

## Java String indexOf() Method

The indexOf() method returns the position of the first occurrence of specified character(s) in a string.

public class Main {

public static void main(String[] args) {

String myStr = "Hello planet earth, you are a great planet.";

System.out.println(myStr.indexOf("planet"));

}

}

## Java String lastIndexOf() Method

The lastIndexOf() method returns the position of the last occurrence of specified character(s) in a string.

public class Main {

public static void main(String[] args) {

String myStr = "Hello planet earth, you are a great planet.";

System.out.println(myStr.lastIndexOf("planet"));

}

}

## Java String replace() Method

The replace() method searches a string for a specified character, and returns a new string where the specified character(s) are replaced.

public class Main {

public static void main(String[] args) {

String myStr = "Hello";

System.out.println(myStr.replace('l', 'p'));

}

}

## Java String substring() Method

* The substring() method is used to retrieve a part of a string, that is, substring from the given string.
* For example,

public class Main {

public static void main(String[] args) {

System.out.println(str.substring(2,5)); // output: ‘llo’

}

}

## Java String toString() Method

* The toString() method is used to return a String object.
* It is used to convert values of other data types into strings. For example,

public class Main {

public static void main(String[] args) {

Integer length = 5;

System.out.println(length.toString()); // output: 5

}

}

## Java String trim() Method

The trim() method removes whitespace from both ends of a string.

public class Main {

public static void main(String[] args) {

String myStr = " Hello World! ";

System.out.println(myStr);

System.out.println(myStr.trim());

}

}

# Explain command line arguments

### While running a class **Demo**, you can specify command line arguments as

### java Demo arg1 arg2 arg3 …

#### Command Line Arguments can be used to specify configuration information while launching your application.

#### There is no restriction on the number of java command line arguments.You can specify any number of arguments

#### Information is passed as Strings.

#### They are captured into the String args of your main method

**Example:**

**Step 1)** Copy the following code into an editor.

class Demo{

public static void main(String args[]){

String[] b={"apple","orange"};

System.out.println("Argument one = "+b[0]);

System.out.println("Argument two = "+b[1]);

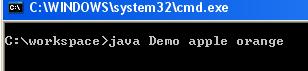
}

}

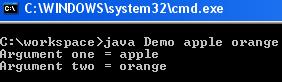
**Step 2)** Save & Compile the code

**Step 3)** Run the code as **java Demo apple orange.**

10 Most Common Interview Questions and Answers ????



**Step 4)** You must get an output as below.



# Describe Wrapper classes, autoboxing, and unboxing

## Wrapper classes

### For example, the following methods are used to get the value associated with the corresponding wrapper object: intValue(),  byteValue(), shortValue(), longValue(), floatValue(), doubleValue(), charValue(), booleanValue().

public class Main {

public static void main(String[] args) {

Integer myInt = 5;

Double myDouble = 5.99;

Character myChar = 'A';

System.out.println(myInt.intValue());

System.out.println(myDouble.doubleValue());

System.out.println(myChar.charValue());

}

}

## Autoboxing and unboxing

### The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So java programmer doesn't need to write the conversion code.

### **Simple Example of Autoboxing in java:**

**class** BoxingExample1{

**public** **static** **void** main(String args[]){

**int** a=50;

Integer a2=**new** Integer(a);//Boxing

Integer a3=5;//Boxing

        System.out.println(a2+" "+a3);

 }

}

**class** UnboxingExample2{

**public** **static** **void** main(String args[]){

    Integer i=**new** Integer(50);

**if**(i<100){            //unboxing internally

        System.out.println(i);

        }

 }

}