*Advance JAVA*

1. Difference between ArrayList & Array.

Ans:

|  |  |
| --- | --- |
| Array | ArrayLIst |
| 1. An array is basic functionality provided by Java | 1. ArrayList is part of collection framework in Java |
| 1. Array is a fixed size data structure | 1. ArrayList is not a fixed size data structure |
| 1. Array can contain both primitive data | 1. ArrayList only supports object entries, not the primitive data types |
| 1. In array, it depends whether the arrays is of primitive type or object type. In case of primitive types, actual values are contiguous locations, but in case of objects, allocation is similar to ArrayList. | 1. ArrayList are always references to objects at different memory locations |
| V.  import java.util.ArrayList;    class Test{    public static void main(String args[]){          /\* Normal Array. \*/          int[] arr = new int[2];          arr[0] = 1;          arr[1] = 2;          System.out.println(arr[0]);      }  }  // Output:1 | V.  import java.util.Arrays;    class Test{  public static void main(String args[]){  ArrayList<Integer> arrL = new ArrayList<Integer>(2);            // Add elements to ArrayList          arrL.add(1);          arrL.add(2);            // Access elements of ArrayList          System.out.println(arrL.get(0));   }  } // Output:1 |

1. Explain 'String class is immutable'.

Ans: String objects are immutable which 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. Because java uses the concept of string literal. Suppose there are 5 reference variables, all refers 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.

**Example:** String class is immutable

**class** Testimmutablestring1{

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

     String s="Sachin";

   s = s.concat(" Tendulkar");

   System.out.println(s);

}

}

1. Life Cycle of thread.

Ans:

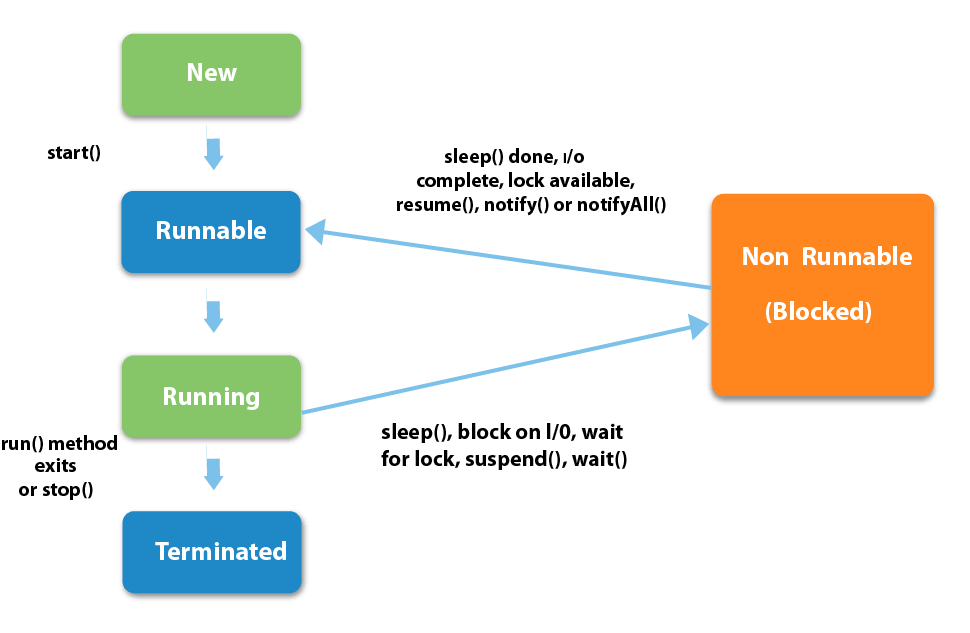
1) New: The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

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



1. Write a program to Create a thread.

Ans: Java Thread Example by extending Thread class:

**class** Multi **extends** Thread{

**public** **void** run(){

System.out.println ("thread is running...");

}

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

Multi t1=**new** Multi();

t1.start();

 }

} //Output: thread is running…

1. Different content of wrapper class.

Ans: The **wrapper class in Java** provides the mechanism *to convert primitive into object and object into primitive*.  A**utoboxing** and **unboxing** feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

Example:

// Java program to demonstrate Wrapping and UnWrapping

// in Java Classes

class WrappingUnwrapping

{

    public static void main(String args[])

    {

        byte a = 1; //  byte data type

        Byte byteobj = new Byte(a);    // wrapping around Byte object

        int b = 10;          // int data type

        Integer intobj = new Integer(b); //wrapping around Integer object

        float c = 18.6f; // float data type

        Float floatobj = new Float(c);  // wrapping around Float object

        double d = 250.5; // double data type

        Double doubleobj = new Double(d); // Wrapping around Double object

        char e='a';  // char data type

        Character charobj=e; // wrapping around Character object

        //  printing the values from objects

        System.out.println("Values of Wrapper objects (printing as objects)");

        System.out.println("Byte object byteobj:  " + byteobj);

        System.out.println("Integer object intobj:  " + intobj);

        System.out.println("Float object floatobj:  " + floatobj);

        System.out.println("Double object doubleobj:  " + doubleobj);

        System.out.println("Character object charobj:  " + charobj);

        // objects to data types (retrieving data types from objects)

        // unwrapping objects to primitive data types

        byte bv = byteobj;

        int iv = intobj;

        float fv = floatobj;

        double dv = doubleobj;

        char cv = charobj;

        // printing the values from data types

        System.out.println("Unwrapped values (printing as data types)");

        System.out.println("byte value, bv: " + bv);

        System.out.println("int value, iv: " + iv);

        System.out.println("float value, fv: " + fv);

        System.out.println("double value, dv: " + dv);

        System.out.println("char value, cv: " + cv);

    }

}

Outputs:

Values of Wrapper objects (printing as objects)

Byte object byteobj: 1

Integer object intobj: 10

Float object floatobj: 18.6

Double object doubleobj: 250.5

Character object charobj: a

Unwrapped values (printing as data types)

byte value, bv: 1

int value, iv: 10

float value, fv: 18.6

double value, dv: 250.5

char value, cv: a

😁😁 😘😘💕💕💕 BEST of LUCK 😘😘💕💕💕😁😁